ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Μηχανική Μάθηση για την Εκτίμηση της Ποιότητας της Ομιλίας με Συνδυασμό Πληροφορίας Ήχου και Κειμένου

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σοφία Ελευθερίου

Εξωτερικός Επιβλέπων: Θεόδωρος Γιαννακόπουλος
Β΄ Ερευνητής ΕΚΕΦΕ Δημόκριτος

Επιβλέπων Ε.Μ.Π.: Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2021

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Μηχανική Μάθηση για την Εκτίμηση της Ποιότητας της Ομιλίας με Συνδυασμό Πληροφορίας Ήχου και Κειμένου

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σοφία Ελευθερίου

Εξωτερικός Επιβλέπων: Θεόδωρος Γιαννακόπουλος
Β΄ Ερευνητής ΕΚΕΦΕ Δημόκριτος

Επιβλέπων Ε.Μ.Π.: Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 8η Ιουλίου.

.....	.....	.....
Στέφανος Κόλλιας	Ανδρέας-Γεώργιος	Γιώργος Στάμου
Καθηγητής	Σταφυλοπάτης	Αναπληρωτής Καθηγητής
Ε.Μ.Π.	Καθηγητής	Ε.Μ.Π.
	Ε.Μ.Π.	

Αθήνα, Ιούλιος 2021

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF INFORMATION AND COMPUTER TECHNOLOGY

Machine Learning for Assessing the Quality of Speech with a Combination of Audio and Text Information

DIPLOMA THESIS

Sofia Eleftheriou

External Supervisor: Theodoros Giannakopoulos
Principal Researcher NCSR Demokritos

NTUA Supervisor: Stefanos Kollias
Professor at the Electrical and Computer
Engineering Department of NTUA

Athens, July 2021

.....
Σοφία Ελευθερίου
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Το θέμα της παρούσας διπλωματικής εργασίας είναι η αυτόματη αξιολόγηση της ποιότητας της ομιλίας μέσω τεχνικών μηχανική μάθησης. Για τον σκοπό αυτό γίνεται ανάλυση τόσο του ηχητικού σήματος της ομιλίας, όσο και του κειμένου της. Οι δυο αυτές προσεγγίσεις αποδίδουν διαφορετικού είδους πληροφορίες, οι οποίες στην συνέχεια μπορούν να χρησιμοποιηθούν αυτοτελώς ή και να συνδυαστούν ώστε να επιτευχθεί ο τελικός στόχος της αξιολόγησης.

Η ποιότητα της ομιλίας είναι μια υποκειμενική γνώμη, βασισμένη στην αίσθηση του ακροατή πάνω στην ομιλία που άκουσε. Επομένως, η αντικειμενική αξιολόγηση της ποιότητας της ομιλίας αποτελεί μία πρόκληση, ιδίως όταν δεν υπάρχει καθαρή αναφορά (που ονομάζεται επίσης μη παρεμβατική ή μεμονωμένη αξιολόγηση ποιότητας ομιλίας). Η ανάγκη για «χρυσή» αναφορά περιορίζει σημαντικά τη δυνατότητα εφαρμογής τέτοιων εργαλείων αξιολόγησης σε σενάρια του πραγματικού κόσμου. Ωστόσο, τα ανθρώπινα όντα μπορούν εύκολα να αξιολογήσουν την ποιότητα της ομιλίας χωρίς καμία αναφορά. Με άλλα λόγια, η ανθρώπινη αντίληψη ακρόασης μπορεί να αντιμετωπιστεί ως λειτουργία χαρτογράφησης για να αντιστοιχίσει οποιαδήποτε ομιλία σε αντίστοιχο δείκτη ποιότητας.

Για την εξαγωγή των χαρακτηριστικών χρησιμοποιούνται τόσο χαρακτηριστικά υψηλού επιπέδου, όπως ο αριθμός παύσεων, οι λέξεις ανά δευτερόλεπτο, η μέση διάρκεια σιωπής κ.α., όσο και χαρακτηριστικά που προκύπτουν από μοντέλα-ταξινομητές τμημάτων ήχου ή κειμένου, οι οποίοι προβλέπουν ετικέτες όπως το συναίσθημα, το σθένος και την διέγερση. Στην συνέχεια, λαμβάνοντας τον μέσο όρο των προβλέψεων αυτών για όλα τα τμήματα, καταλήγουμε σε χαρακτηριστικά που αφορούν τον συνολικό ήχο ή κείμενο (δηλαδή την συνολική ομιλία). Συνδυάζοντας τα παραπάνω χαρακτηριστικά από ήχο και κείμενο, χρησιμοποιούμε τελικούς ταξινομητές επιπέδου εγγραφής, οι οποίοι αξιολογούν την ομιλία σε διαφορετικούς άξονες (εκφραστικότητα, ευκολία παρακολούθησης και διασκέδαση).

Για τους ταξινομητές τμημάτων χρησιμοποιήθησαν ανοιχτού τύπου σύνολα δεδομένων, ενώ για τους τελικούς ταξινομητές εγγραφής, ακολουθήθηκε διαδικασία συλλογής και επισημείωσης δεδομένων, καθώς επίσης και συνάθροισης/συμφωνίας των επισημειώσεων.

Λέξεις Κλειδιά:

Αυτόματη αξιολόγηση, ποιότητα ομιλίας, τεχνικές μηχανικής μάθησης, ηχητικό σήμα, κείμενο, δείκτης ποιότητας, εξαγωγή χαρακτηριστικών, χαρακτηριστικά υψηλού επιπέδου, αριθμός παύσεων, λέξεις ανά δευτερόλεπτο, μέση διάρκεια σιωπής, μοντέλα-ταξινομητές τμημάτων, συναίσθημα, σθένος, διέγερση, ταξινομητές επιπέδου εγγραφής, συλλογή και επισημείωση δεδομένων, συνάθροιση/συμφωνία επισημειώσεων.

Abstract

The subject of this dissertation is the automatic evaluation of the quality of speech through machine learning techniques. For this purpose, both the audio signal of the speech and its text are analyzed. These two approaches provide different types of information, which can then be used independently or combined to achieve the final goal of the evaluation.

The quality of speech is a subjective opinion, based on the listener's sense of the speech he heard. Therefore, objective assessment of speech quality is a challenge, especially when there is no clear reporting (also called non-invasive or individual speech quality assessment). The need for "golden" reporting significantly limits the applicability of such evaluation tools to real-world scenarios. However, human beings can easily evaluate the quality of speech without any reference. In other words, human listening perception can be treated as a mapping function to match any speech to a corresponding quality index.

High-level features such as number of pauses, words per second, average silence duration, etc. are used to extract the features, as well as features derived from audio or text segment classifier models that provide labels, such as emotion, valence and arousal. Then, taking the average of these predictions of all the segments, we come to characteristics related to the total sound or text (ie the total speech). Combining the above features from audio and text, we use final recording level classifiers, which evaluate speech in different axes (expressiveness, ease of following and enjoyment).

For the segment classifiers, open-source data sets were used, while for the final recording level classifiers, a data collection and annotation procedure was followed, as well as an aggregation/agreement of the annotations.

Key Words:

Automatic evaluation, speech quality, machine learning techniques, audio signal, text, quality index, high Level features, extract features, number of pauses, words per second, average silence duration, segment models, emotion, valence, arousal, record level classifiers, data collection and annotation, aggregation / agreement of annotations.

Ευχαριστίες

Η παρούσα διπλωματική εκπονήθηκε στο Εργαστήριο Υπολογιστικής Ευφυίας (Computational Intelligence Laboratory - CIL) του Ινστιτούτου Πληροφορικής και Τηλεπικοινωνιών του Εθνικού Κέντρου Έρευνας Φυσικών Επιστημών (ΕΚΕΦΕ) "Δημόκριτος".

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Κόλλια, για την ευκαιρία που μου έδωσε, να εκπονήσω τη διπλωματική μου εργασία, σε ένα θέμα τόσο ενδιαφέρον και δημιουργικό.

Επίσης ευχαριστώ ιδιαίτερα τον Β' ερευνητή του ΕΚΕΦΕ Δημόκριτος κ. Γιαννακόπουλο για την εξαιρετική συνεργασία και τη συνεχή καθοδήγηση κατά την ανάπτυξη της εργασίας αυτής.

Από καρδίας θα ήθελα να ευχαριστήσω πολύ τους φίλους και τις φίλες μου για την ουσιαστική συμπαράσταση, όλα αυτά τα χρόνια.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξη και την εμπιστοσύνη της που με βοήθησαν να πραγματοποιήσω τους στόχους και τα όνειρά μου, καθώς επίσης και να ξεπεράσω κάθε εμπόδιο. Τους ευχαριστώ που είναι πάντα δίπλα μου και με στηρίζουν σε κάθε μου βήμα.

Contents

List of Figures

List of Tables

Ελληνικό Κείμενο

1 Εισαγωγή

Σε αυτό το κεφάλαιο, υπογραμμίζεται η σημασία και η σημαντικότητα της ομιλίας. Συζητείται το κίνητρο για τη διπλωματική εργασία. Επιπλέον, παρουσιάζεται ο στόχος και το περίγραμμα των διαφόρων κεφαλαίων αυτής της διατριβής.

1.1 Ομιλία

Η δημόσια ομιλία (ονομάζεται επίσης και ρητορική) είναι η ομιλία "πρόσωπο με πρόσωπο" παρουσία ζωντανού κοινού. Ωστόσο, λόγω της εξέλιξης της δημόσιας ομιλίας, θεωρείται σύγχρονα ως οποιαδήποτε μορφή ομιλίας (επίσημα και ανεπίσημα) μεταξύ κοινού και ομιλητή. Παραδοσιακά, η δημόσια ομιλία θεωρείται μέρος της τέχνης της πειθούς. Η πράξη μπορεί να επιτύχει συγκεκριμένους σκοπούς, συμπεριλαμβανομένης της ενημέρωσης, της πειθούς και της ψυχαγωγίας. Επιπλέον, μπορούν να χρησιμοποιηθούν διαφορετικές μέθοδοι, δομές και κανόνες ανάλογα με την κατάσταση της ομιλίας.

Έτσι, όταν μιλάμε για δημόσια ομιλία, δεν αναφερόμαστε μόνο στην παραδοσιακή μορφή μεταξύ κοινού και ομιλητή, αλλά και στη γενική καθημερινή ανθρώπινη ομιλία που περιλαμβάνει 2 ή περισσότερους συνομιλητές.

Η δημόσια ομιλία αναπτύχθηκε στη Ρώμη και την Ελλάδα. Διακεκριμένοι στοχαστές από αυτά τα εδάφη επηρέασαν την ανάπτυξη και την εξελικτική ιστορία της. Επί του παρόντος, η τεχνολογία συνεχίζει να μεταμορφώνει την τέχνη αυτή μέσω της νέας διαθέσιμης τεχνολογίας, όπως τηλεδιάσκεψη, παρουσιάσεις πολυμέσων και άλλες μη παραδοσιακές μορφές. Η γνώση του πότε η ομιλία είναι πιο αποτελεσματική και πώς γίνεται σωστά είναι το κλειδί για την κατανόηση της σημασίας της.

Η ομιλία για επαγγελματικές και εμπορικές εκδηλώσεις γίνεται συχνά από επαγγελματίες. Αυτοί οι ομιλητές μπορούν να συναφθούν ανεξάρτητα, μέσω εκπροσώπησης από ένα γραφείο ομιλητών ή με άλλα μέσα. Η ομιλία παίζει μεγάλο ρόλο στον επαγγελματικό κόσμο. Στην πραγματικότητα, πιστεύεται ότι το 70 τοις εκατό όλων των θέσεων εργασίας περιλαμβάνει κάποια μορφή αυτής [1].

Έτσι η προφορική επικοινωνία αναγνωρίζεται ως μία από τις πιο πολύτιμες δεξιότητες για επιτυχία στον εργασιακό χώρο (cf. Kyllonen, 2012) και στο σχολείο. Η ομιλία είναι ευρέως γνωστή ως η πιο φοβισμένη μορφή προφορικής επικοινωνίας (Pull, 2012) αλλά συχνά παραβλέπεται στην εκπαιδευτική αξιολόγηση. Μια επιτυχημένη ομιλία διανέμεται σε πολλούς άξονες - π.χ. το περιεχόμενο της ομιλίας, τη φωνή και τον τονισμό, το συναίσθημα του ομιλιτή, το σθένος, τον ρυθμό , τις παύσεις αλλά και τις χειρονομίες χεριών και τις στάσεις του σώματος.

Ενώ οι περισσότερες τρέχουσες πληροφορίες για την αξιολόγηση της απόδοσης ομιλίας αφορούν τόσο λεκτικές όσο και μη λεκτικές πτυχές, σχεδόν όλες οι υπάρχουσες αξιολογήσεις στην πράξη απαιτούν ανθρώπινη βαθμολογία (Ward, 2013: Schreiber, Paul and Shibley, 2012: Carlson and Smith-Howell, 1995) [2].

1.2 Κίνητρο για τη Συγγραφή της Διπλωματικής Εργασίας

Ως νέοι ηλεκτρολόγοι μηχανικοί και μηχανικοί υπολογιστών, είμαστε υποχρεωμένοι να παρακολουθούμε τις τεχνολογικές τάσεις που συμβαίνουν στον τομέα μας και να εκπαιδευόμαστε για να είμαστε σε θέση να συμβάλλουμε στις υπερσύγχρονες λύσεις.

Η μηχανική μάθηση είναι ένας μεγάλος τομέας, αρκετά ανερχόμενος στη σύγχρονη τεχνολογική εποχή, ο οποίος δεν θα μπορούσε να ταιριάζει περισσότερο στον τομέα σπουδών μας.

Η επιτακτική ανάγκη να χρησιμοποιήσουμε την ομιλία μας στην καθημερινή μας ζωή, είναι αυτή που απαιτεί την αξιολόγησή της και τη βελτίωσή της. Αυτή η αξιολόγηση είναι ευκολότερη και ταχύτερη εάν γίνεται από ένα μηχάνημα και όχι από το ίδιο το άτομο. Έτσι, η αξιολόγηση της ποιότητας του λόγου μέσω της μηχανικής μάθησης είναι μονόδρομος.

Η ομιλία υπάρχει παντού και ο τρόπος που μιλάμε είναι εξίσου σημαντικός με το τι λέμε. Έτσι, η ανάλυση της ομιλίας και μάλιστα η πολυτροπική ανάλυσή της (multimodal speech analytics), αποτελεί μια σημαντική διεργασία που εμπίπτει σε διαφόρων ειδών προβλήματα, όχι μόνο στην εκτίμηση της ποιότητάς της, όπως είναι για παράδειγμα ο εντοπισμός μαθησιακών δυσκολιών (δυσλεξία, αυτισμός), η ανάκτηση σημάτων από τηλεφωνικά κέντρα κ.α.

Οι προαναφερθέντες λόγοι παρακίνησαν την έρευνα και τη σύνταξη αυτής της διπλωματικής εργασίας. Είναι χρήσιμο για τους νέους επαγγελματίες να εξοικειωθούν με έννοιες που φαίνεται να είναι το μέλλον στους τομείς της τεχνολογίας, του προγραμματισμού και του αυτοματισμού.

1.3 Στόχος της Διπλωματικής Εργασίας

Ο στόχος αυτής της διπλωματικής είναι πρώτα απ᾽ όλα η εξοικείωση και η εις βάθος μελέτη τόσο της μηχανικής μάθησης όσο και των τεχνικών βαθιάς μάθησης, η χρήση αυτών των τεχνικών σε πραγματικά ζητήματα και δεδομένα και η εμφάνιση ωφέλιμων αποτελεσμάτων. Επιπλέον, αυτή η διπλωματική στοχεύει να επισημάνει λύσεις που θα μπορούσαν να χρησιμοποιηθούν για την επίλυση άλλων προβλημάτων τα οποία απαιτούν παρόμοιες τεχνικές, δηλαδή άλλων εφαρμογών επεξεργασίας σήματος ομιλίας.

Πιο συγκεκριμένα στην διπλωματική εργασία αυτή, διερευνάται το ζήτημα της αξιολόγησης της ομιλίας, έτσι ώστε το μηχάνημα να κατανοεί πότε ένας ομιλητής έχει καλές δεξιότητες ομιλίας και πότε υστερεί και έχει περιθώρια βελτίωσης. Καθίσταται φανερή η σημαντικότητα της αξιολόγησης της ποιότητας της ομιλίας, μια διαδικασία η οποία είθισται να διεξάγεται από ανθρώπους και της οποίας η αυτοματοποίηση μέσω μεθόδων μηχανικής μάθησης είναι απαραίτητη.

Είναι βέβαια σημαντικό να αναφέρουμε πως η ανάλυση του σήματος ομιλίας, όπως είναι για παράδειγμα η αναγνώριση του συναισθήματος της ομιλίας, είναι ένα δύσκολο έργο, καθώς η ομιλία είναι άμεσα εξαρτώμενη από πολλούς παράγοντες, όπως οι συνθήκες ηχογράφησης, το περιεχόμενο, τα δεδομένα και ο τρόπος με τον οποίο διεξήχθη η επισημείωση αυτών, οι ομιλητές, το γένος/φύλο κ.α. Οι εξαρτήσεις αυτές καλούνται να αντιμετωπιστούν στην παρούσα διπλωματική εργασία.

Οι τεχνικές ή αλλιώς το σύστημα που αναπτύσσεται στην παρούσα διατριβή, θα μπορούσε να εφαρμοστεί και σε άλλα ζητήματα ανάλυσης της ομιλίας, εκτός από την αξιολόγηση της ποιότητάς της, όπως για παράδειγμα στον εντοπισμός του αυτισμού, της

δυσλεξίας ή άλλων γλωσσικών διαταραχών, στην εξαγωγή συμπερασμάτων και νοημάτων από την ομιλία κ.α.

1.4 Δομή του Ελληνικού Κειμένου της Διπλωματικής Εργασίας

Αυτή η διπλωματική εργασία οργανώνεται ως εξής. Στο **κεφάλαιο 2** δίνεται ένα λεπτομερές θεωρητικό υπόβαθρο απαραίτητο για την κατανόηση του πεδίου της τεχνητής νοημοσύνης και της μηχανικής μάθησης, καθώς και των μεθόδων που θα χρησιμοποιηθούν. Στο **κεφάλαιο 3** θα αναλυθεί η αρχιτεκτονική του συστήματος, τόσο από πλευράς ήχου όσο και κειμένου, καθώς και ο συνδυασμός αυτών. Στο **κεφάλαιο 4** περιγράφεται η ανάλυση σε επίπεδο τμήματος τόσο για τον ήχο όσο και για το κειμένου, οι οποίες περιλαμβάνουν την τμηματοποίηση, την εξαγωγή χαρακτηριστικών και την εκπαίδευση των μοντέλων επιπέδου τμήματος. Στο **κεφάλαιο 5**, αναλύεται το επίπεδο εγγραφής, τόσο στο κομμάτι του ήχου όσο και του κειμένου, το οποίο περιλαμβάνει τα χαρακτηριστικά επιπέδου εγγραφής. Στο **κεφάλαιο 6**, περιγράφεται η διαδικασία συλλογής και επισημείωσης δεδομένων. Στο **κεφάλαιο 7**, παρατίθονται τα τελικά πειράματα για τα μοντέλα επιπέδου εγγραφής και τα αποτελέσματά τους. Τέλος, στο **κεφάλαιο 8** παρουσιάζονται τα συμπεράσματα της παρούσας διπλωματικής εργασίας.

2 Θεωρητικό υπόβαθρο

Σε αυτό το κεφάλαιο, θα πραγματοποιηθεί ένα εκτεταμένο θεωρητικό υπόβαθρο για την τεχνητή νοημοσύνη και τη μηχανική μάθηση. Στην ουσία, αυτό θα παρέχει ένα χρήσιμο εργαλείο για τον αναγνώστη να κατανοήσει τις μεθόδους που χρησιμοποιούνται σε αυτήν τη διπλωματική εργασία.

2.1 Τεχνητή Νοημοσύνη

2.1.1 Ορισμός

Η Τεχνητή Νοημοσύνη (AI) είναι ένα πεδίο της τεχνολογίας πληροφοριών που σχεδιάζει και αναπτύσσει συστήματα τα οποία μιμούνται την ανθρώπινη συμπεριφορά. Η τεχνητή νοημοσύνη είναι η ικανότητα των μηχανών να σκέφτονται σαν τον ανθρώπινο εγκέφαλο. Η διαφορά μεταξύ του τελευταίου και του πρώτου είναι ότι ο ανθρώπινος εγκέφαλος εμπεριέχει συνείδηση και συναισθήματα. Συνήθως, ο όρος «τεχνητή νοημοσύνη» χρησιμοποιείται για να περιγράψει μηχανές (ή υπολογιστές), οι οποίοι μιμούνται «γνωστικές» λειτουργίες που οι άνθρωποι συνδέουν με το ανθρώπινο μυαλό, όπως «μάθηση» και «επίλυση προβλημάτων». Οι μηχανές κατανοούν το περιβάλλον τους και επιλύουν προβλήματα, ενεργώντας για την επίτευξη ενός συγκεκριμένου στόχου.

Η Τεχνητή Νοημοσύνη εξελίσσεται συνεχώς για να ωφελήσει πολλές διαφορετικές βιομηχανίες. Οι μηχανές είναι ενσύρματες χρησιμοποιώντας μια διεπιστημονική προσέγγιση βασισμένη στα μαθηματικά, την επιστήμη των υπολογιστών, τη γλωσσολογία, την ψυχολογία και πολλά άλλα.

Η εικόνα που έχουν οι περισσότεροι άνθρωποι στο μυαλό τους όταν ακούνε τεχνητή νοημοσύνη είναι ρομπότ που μοιάζουν με ανθρώπους σε σχήμα και κατασκευή. Αλλά η πραγματικότητα απέχει πολύ από αυτό.

Ο όρος τεχνητή νοημοσύνη δεν αναφέρεται απαραίτητα σε ρομπότ αλλά σε μηχανές που μαθαίνουν να μιμούνται την ανθρώπινη νοημοσύνη και να εκτελούν εργασίες.

2.1.2 Εφαρμογές

Η τεχνητή νοημοσύνη μπορεί να εφαρμοστεί σε πολλούς διαφορετικούς τομείς όπως στη βιομηχανία υγειονομικής περίθαλψης για δοσολογία φαρμάκων και για χειρουργικές επεμβάσεις.

Ένα άλλο τρανταχτό παράδειγμα της χρήσης της τεχνητής νοημοσύνης είναι οι υπολογιστές που παίζουν σκάκι και τα αυτοκινούμενα αυτοκίνητα. Κάθε ένα από αυτά τα μηχανήματα πρέπει να διενεργεί λαμβάνοντας υπόψη τις συνέπειες των ενεργειών του προκειμένου να επιτευχθεί το επιθυμητό αποτέλεσμα. Για παράδειγμα, το επιθυμητό αποτέλεσμα στο σκάκι είναι να κερδίσει το παιχνίδι, ενώ σε αυτοκίνητα πρέπει να αποφεύγεται οποιαδήποτε σύγκρουση και πρέπει να τηρούνται οι απαραίτητοι κανονισμοί (όπως ο κωδικός οδικής κυκλοφορίας), με απώτερο στόχο το μηχάνημα να φτάσει στον προορισμό του.

Η τεχνητή νοημοσύνη έχει επίσης εφαρμογές στον χρηματοπιστωτικό κλάδο, όπου χρησιμοποιείται για τον εντοπισμό και την επισήμανση της δραστηριότητας στον τραπεζικό και χρηματοοικονομικό τομέα, όπως ασυνήθιστη χρήση χρεωστικής κάρτας και μεγάλες καταθέσεις λογαριασμού - όλα αυτά βοηθούν το τμήμα απάτης μιας τράπεζας. Οι εφαρμογές για την τεχνητή νοημοσύνη χρησιμοποιούνται επίσης για τον εξορθολογισμό και τη διευκόλυνση των συναλλαγών. Αυτό γίνεται διευκολύνοντας την εκτίμηση της προσφοράς, της ζήτησης και της τιμολόγησης των κινητών αξιών.

2.1.3 Κατηγοριοποίηση

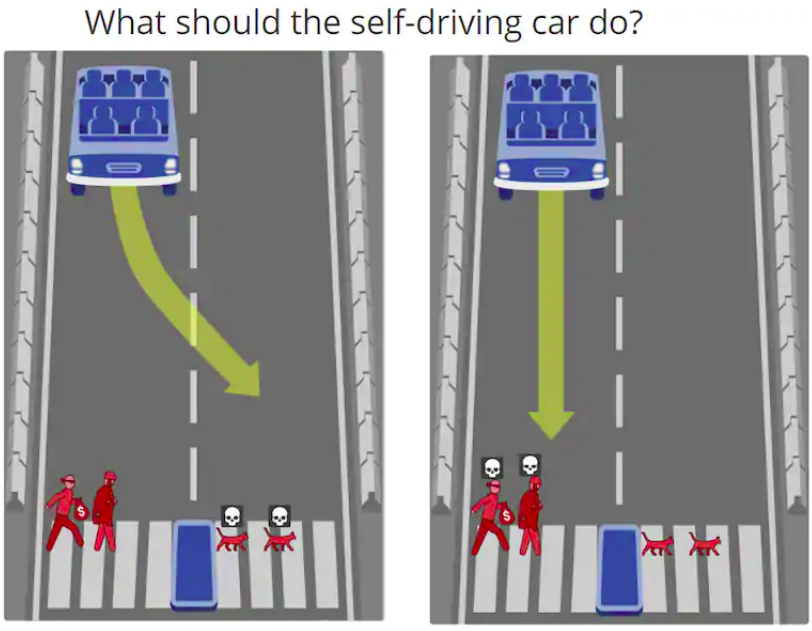
Η τεχνητή νοημοσύνη χωρίζεται σε δύο διαφορετικές κατηγορίες: αδύναμη και ισχυρή. Η αδύναμη τεχνητή νοημοσύνη αφορά συστήματα προγραμματισμένα για να εκτελούν μια συγκεκριμένη λειτουργία. Τέτοια συστήματα είναι τα βιντεοπαιχνίδια όπως το παράδειγμα του σκακιού που προαναφέρθηκε, οι προσωπικοί-ηλεκτρονικοί βοηθοί που τους ρωτάς μια ερώτηση και σου απαντάνε,το λογισμικό ανάλυσης εικόνας, οι μηχανές αναζήτησης, τα συστήματα αναγνώρισης προσώπου και ομιλίας κ.α.

Η ισχυρή τεχνητή νοημοσύνη αφορά συστήματα που εκτελούν λειτουργίες οι οποίες θεωρούνται ανθρώπινες. Οι λειτουργίες αυτές είναι πιο δύσκολες και περίπλοκες. Αυτά τα συστήματα είναι προγραμματισμένα να χειρίζονται καταστάσεις και να επιλύουν προβλήματα χωρίς την ανθρώπινη παρέμβαση. Τέτοια συστήματα είναι τα ρομπότ, τα αυτόνομα αυτοκίνητα, τα τηλεκατευθυνόμενα αεροσκάφη (drones), το διαδίκτυο των πραγμάτων (Internet of Things) κ.α.

2.1.4 Ηθικά Ζητήματα

Μία κοινή σκέψη σε ότι αφορά την τεχνητή νοημοσύνη είναι ότι οι μηχανές θα αναπτυχθούν σε τέτοιο βαθμό που οι άνθρωποι να μην μπορούν να ακολουθήσουν και πως θα επανασχεδιάζονται μόνες τους με εκθετικό ρυθμό.

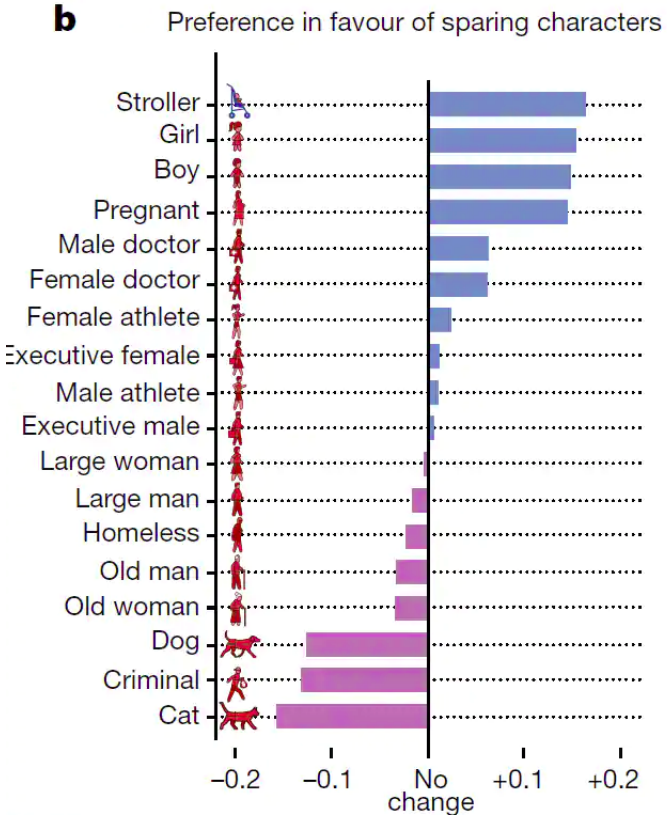
Ένα άλλο ζήτημα είναι ότι οι μηχανές μπορούν να παραβιάσουν την ιδιωτικότητα των ανθρώπων, όπως επίσης και το πόσα και ποια δικαιώματα μπορεί να έχει μια μηχανή. Σε αυτήν την περίπτωση αναφερόμαστε σε ηθικά ζητήματα. Παραδείγματος χάριν, στην περίπτωση του αυτοκινούμενου αυτοκινήτου, εάν το αμάξι βρεθεί μπροστά σε ένα δίλημμα όπου πρέπει να αποφασίσει ποιος πρέπει να ζήσει και ποιός πρέπει να πεθάνει, ποια θα ήταν η επιλογή του; Φανταστείτε αυτό το σενάριο: τα φρένα αποτυγχάνουν σε ένα αυτο-οδηγούμενο αυτοκίνητο καθώς οδηγεί προς μια πολυσύχναστη διάβαση πεζών. Ένας άστεγος και ένας εγκληματίας περνούν μπροστά από το αυτοκίνητο. Δύο γάτες βρίσκονται στην αντίθετη λωρίδα. Πρέπει το αυτοκίνητο να περιστραφεί για να σκοτώσει τις γάτες ή να συγκρουστεί με τα δύο άτομα;



Σχήμα 2.1: Παράδειγμα ηθικού ζητήματος τεχνητής νοημοσύνης. Τι πρέπει ένα αυτοκινούμενο αυτοκίνητο να κάνει μπροστά σε ένα δίλημμα;

Με σκοπό να ερευνηθεί η γνώμη των ανθρώπων σε τέτοιου είδους ηθικά ζητήματα που μπορεί να αντιμετωπίσει η τεχνητή νοημοσύνη, δημιουργήθηκε μια διαδικτυακή πλατφόρμα, η "Moral Machine", από την ομάδα κλιμάκωσης συνεργασίας του Iyad Rahwan, στο Ινστιτούτο Τεχνολογίας της Μασαχουσέτης, η οποία δημιουργεί ηθικά διλήμματα και συλλέγει πληροφορίες σχετικά με τις αποφάσεις που λαμβάνουν οι άνθρωποι μεταξύ δύο καταστρεπτικών αποτελεσμάτων. Η πλατφόρμα [4] είναι η ιδέα των Iyad Rahwan και κοινωνικών ψυχολόγων Azim Shariff και Jean-François Bonnefon, που συνέλαβαν την ιδέα σχετικά με την ηθική των αυτοκινούμενων αυτοκινήτων. Οι βασικοί συντελεστές στη δημιουργία της πλατφόρμας ήταν οι μεταπτυχιακοί φοιτητές του MIT Media Lab, Edmond Awad και Sohan Dsouza [5].

Τα πειράματα σκέψης που έθεσε ο ιστότοπος Moral Machine έγιναν πασίγνωστα, με το εικονογραφικό κουίζ τους που συμμετείχαν αρκετά εκατομμύρια άνθρωποι σε 233 χώρες ή περιοχές. Η μελέτη, η οποία περιελάμβανε 40 εκατομμύρια απαντήσεις σε διαφορετικά διλήμματα, παρέχει ένα συναρπαστικό στιγμιότυπο της παγκόσμιας κοινής γνώμης, καθώς η εποχή των αυτοκινούμενων αυτοκινήτων είναι ένα όραμα μελλοντικής ευκολίας.Η μελέτη, που δημοσιεύθηκε, εντόπισε μερικές προτιμήσεις που ήταν οι πιο ισχυρές: Οι άνθρωποι επιλέγουν να σώσουν ανθρώπους από τα κατοικίδια ζώα, να σώσουν τα πολλά από τα λίγα και να σώσουν παιδιά και έγκυες γυναίκες από ηλικιωμένους. Ωστόσο, βρήκε επίσης άλλες προτιμήσεις για την αποφυγή γυναικών έναντι ανδρών, αθλητών έναντι παχύσαρκων ατόμων και ατόμων υψηλότερης κοινωνικής θέσης, όπως στελέχη, αντί για άστεγους ή εγκληματίες. Υπήρχαν επίσης πολιτιστικές διαφορές στον βαθμό, για παράδειγμα, ότι οι άνθρωποι θα προτιμούσαν να σώσουν νεότερους από τους ηλικιωμένους σε ένα σύμπλεγμα κυρίως ασιατικών χωρών[7].

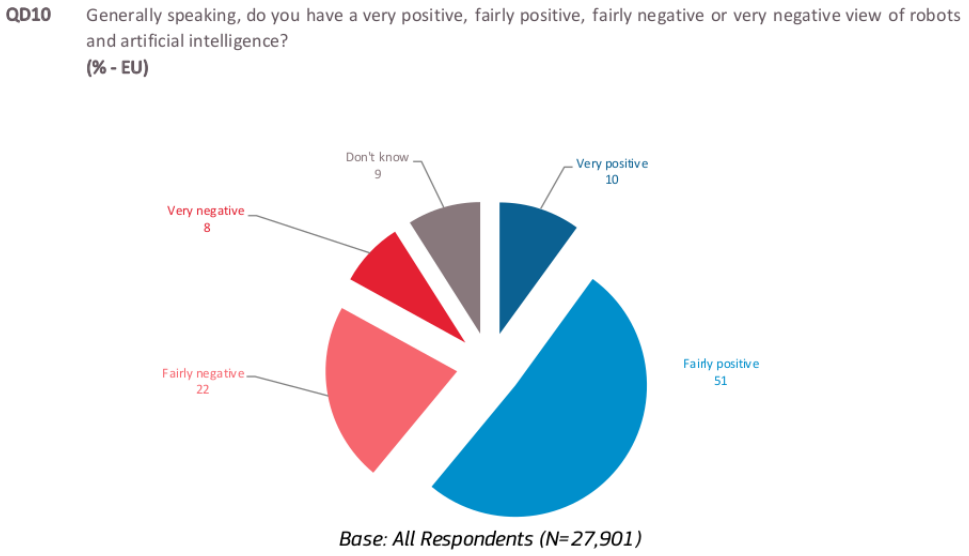


Σχήμα 2.2: Αποτελέσματα μελέτης της πλατφόρμας "Moral Machine".

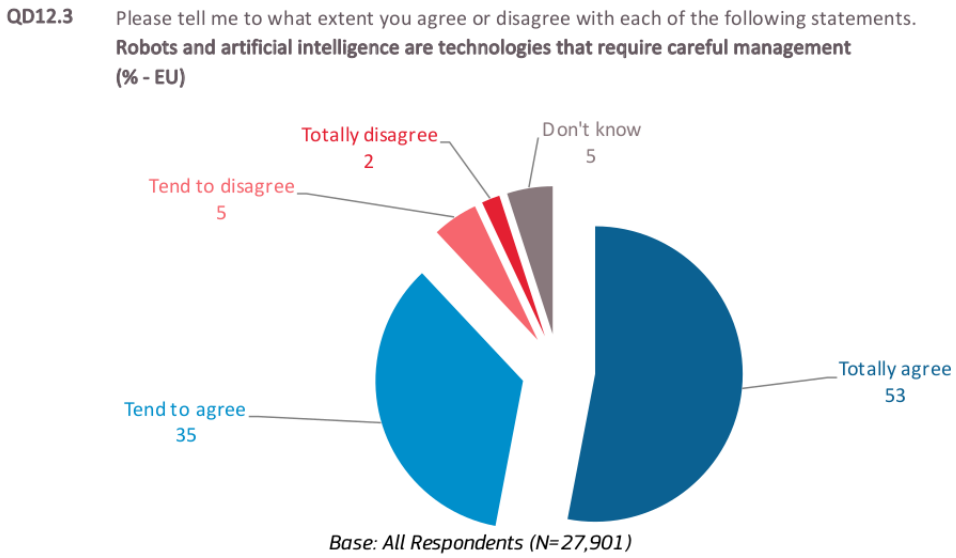
Ενα άλλο αμφιλεγόμενο ζήτημα είναι ο φόβος πως η τεχνητή νοημοσύνη μπορεί να οδηγήσει σε μεγάλα επίπεδα ανεργείας καθώς οι μηχανές θα καταλάβουν εργασίες που

γινόντουσαν ενίοτε απο ανθρώπινα χέρια. Όπως για παράδειγμα τα αυτοκινούμενα αυτοκίνητα μπορεί να αφαιρέσουν την ανάγκη για ταξί και προγράμματα μεριδίων αυτοκινήτων, ενώ οι κατασκευαστές μπορούν εύκολα να αντικαταστήσουν την ανθρώπινη εργασία με μηχανήματα, καθιστώντας τις δεξιότητες των ανθρώπων πιο ξεπερασμένες.

Εδώ θα ήταν σημαντικό να σημειώσουμε ότι σε έρευνα που διεξήχθη το 2017 από την Ευρωπαϊκή επιτροπή και την γενική διεύθυνση δικτύων επικοινωνιών, περιεχομένου και τεχνολογίας,ως προς τις στάσεις απέναντι στον αντίκτυπο της ψηφιοποίησης και του αυτοματισμού στην καθημερινή ζωή, φαίνεται το 61 τοις εκατό των ευρωπαίων να διάκειται θετικά στην τεχνητή νοημοσύνη και στα ρομπότ, ενώ το 88 τοις εκατό θεωρεί ότι οι τεχνολογίες τεχνητής νοημοσύνης απαιτούν προσεκτική διαχείρηση (Ευροβαρόμετρο 2017,EE28). [6]



Σχήμα 2.3: Πάνω από έξι στους δέκα (61%) έχουν θετική γνώμη για τα ρομπότ και την τεχνητή νοημοσύνη.



Σχήμα 2.4: Σχεδόν εννέα στους δέκα (88%) συμφωνούν ότι τα ρομπότ και η τεχνητή νοημοσύνη είναι τεχνολογίες που απαιτούν προσεκτική διαχείριση.

2.1.5 Πεδία Έρευνας

Η τεχνητή νοημοσύνη έχει διάφορα πεδία έρευνας όπως η επεξεργασία φυσικής γλώσσας (NTLK), η ρομποτική (Robotics), η όραση υπολογιστών (Computer Vision) κ.α.

Τα πεδία τα οποία θα χρησιμοποιηθούν στην παρούσα διπλωματική είναι η μηχανική μάθηση (Machine Learning) και η βαθυά μηχανική μάθηση (Deep Learning). Τα επόμενα υποκεφάλαια αναλύουν και εξηγούν αυτά τα πεδία.

2.2 Μηχανική Μάθηση

2.2.1 Ορισμός

Η μηχανική μάθηση θεωρείται κομμάτι της τεχνητής νοημοσύνης. Ανήκει στο πεδίο της επιστήμης υπολογιστών και έχει να κάνει με αλγορίθμους εκμάθησης οι οποίοι εκπαιδεύονται σε δείγματα δεδομένων, δημιουργώντας μοντέλα που μπορούν να κάνουν προβλέψεις ή να πάρουν αποφάσεις από μόνα τους. Αυτά τα στατιστικά μοντέλα θεωρείται ότι χρησιμοποιούνται κάπως για την επίλυση ενός πρακτικού προβλήματος.

Το 1959, ο Arthur Samuel, πρωτοπόρος στον τομέα της μηχανικής μάθησης (ML) την καθόρισε ως «πεδίο σπουδών που δίνει στους υπολογιστές τη δυνατότητα να μάθουν χωρίς να προγραμματίζονται ρητά».

Η μηχανική μάθηση μπορεί να οριστεί ως υπολογιστικές μέθοδοι που χρησιμοποιούν την εμπειρία για τη βελτίωση της απόδοσης ή για να κάνουν ακριβείς προβλέψεις. Σε αυτήν την περίπτωση, η εμπειρία αναφέρεται σε προηγούμενες πληροφορίες ή δεδομένα που είναι διαθέσιμα σε εμάς, τα οποία έχουν επισημανθεί και κατηγοριοποιηθεί. Όπως με κάθε υπολογιστική άσκηση, η ποιότητα και η ποσότητα των δεδομένων θα είναι καθοριστικής σημασίας για την ακρίβεια των προβλέψεων που θα γίνουν.

Αν δούμε λίγο πιο προσεκτικά, η μηχανική μάθηση μοιάζει πολύ με τη στατιστική μοντελοποίηση. Στη στατιστική μοντελοποίηση, συλλέγουμε δεδομένα, επαληθεύουμε ότι είναι καθαρά - με άλλα λόγια, ότι έχουμε ολοκληρώσει, διορθώσει ή διαγράψει τυχόν ελλιπή, λανθασμένα ή άσχετα μέρη των δεδομένων - και στη συνέχεια χρησιμοποιούμε αυτό το καθαρό σύνολο δεδομένων για να δοκιμάσουμε υποθέσεις και να κάνουμε προβλέψεις. Η ιδέα πίσω από τη στατιστική μοντελοποίηση είναι η προσπάθεια αναπαράστασης σύνθετων ζητημάτων με σχετικά γενικευμένους όρους, δηλαδή όρους που εξηγούν τα περισσότερα συμβάντα που μελετήθηκαν. Ως εκ τούτου, προγραμματίζουμε τον αλγόριθμο για την εκτέλεση συγκεκριμένων λειτουργιών με βάση τα δεδομένα που υποβάλλουμε. Με άλλα λόγια, ο αλγόριθμος είναι στατικός. Χρειάζεται προγραμματιστής για να του πει τι να κάνει όταν τροφοδοτείται με δεδομένα.

Παρακάτω παρουσιάζονται πέντε διαφορετικοί ορισμοί του όρου, από αξιόπιστες πηγές:

- «Η μηχανική μάθηση είναι η πρακτική της χρήσης αλγορίθμων για να αναλυσεις δεδομένα, να μάθεις από αυτά και, στη συνέχεια, να πάρεις μια αποφαση ή να κάνεις μια πρόβλεψη για κάτι στον κόσμο.» - Nvidia [8]

- «Η μηχανική μάθηση είναι η επιστήμη του να κάνεις τους υπολογιστές να ενεργούν χωρίς να είναι προγραμματισμένοι ρητά». - Στάνφορντ [9]

- «Η μηχανική μάθηση βασίζεται σε αλγόριθμους που μπορούν να μάθουν από δεδομένα χωρίς να βασίζονται σε προγραμματισμό βάσει κανόνων.» - McKinsey & Co [10]

- «Οι αλγόριθμοι μηχανικής μάθησης μπορούν να καταλάβουν πώς να εκτελούν σημαντικές εργασίες γενικεύοντας από παραδείγματα.» - Πανεπιστήμιο της Ουάσιγκτον [11]

- «Το πεδίο της Μηχανικής Μάθησης επιδιώκει να απαντήσει στην ερώτηση« Πώς μπορούμε να κατασκευάσουμε συστήματα υπολογιστών που βελτιώνονται αυτόματα με την εμπειρία και ποιοι είναι οι θεμελιώδεις νόμοι που διέπουν όλες τις μαθησιακές διαδικασίες; » - Πανεπιστήμιο Carnegie Mellon [12]

2.2.2 Κατηγορίες Μάθησης

Η μηχανική μάθηση χωρίζεται σε τέσσερις κατηγορίες: **Επιβλεπόμενη**, **Μη-επιβλεπόμενη**, **Ημι-επιβλεπόμενη** και **Ενισχυτική**.

1. **Επιβλεπόμενη Μάθηση**
   Στην επιβλεπόμενη μάθηση ή αλλιώς επιτηρούμενη μάθηση, το σύνολο των δεδομένων που χρησιμοποιεί ο αλγόριθμος είναι μια συλλογή από δείγματα που είναι επισημασμένα με κάποια ετικέτα (label) $\{(x_i, y_i)\}_{i=1}^{N}$. Κάθε στοιχείο x_i ανάμεσα στα N, ονομάζεται διάνυσμα χαρακτηριστικών. Το διάνυσμα χαρακτηριστικών έχει διαστάσεις $j = 1, ..., D$, όπου κάθε j αντικατοπτρίζει ένα χαρακτηριστικό. Παραδείγματος χάριν, εάν το δείγμα x_1 αφορά έναν άνθρωπο τότε το $x_1^{(1)}$ θα μπορούσε να είναι το ύψος του, το $x_1^{(2)}$ η ηλικία του, το $x_1^{(3)}$ το φύλο του κ.ο.κ. Κάθε δείγμα x_i έχει το ίδιο είδος χαρακτηριστικού στην θέση j με τα υπόλοιπα δείγματα. Δηλαδή ο δεύτερος άνθρωπος x_2, πάλι θα έιχε στην θέση $x_2^{(1)}$ το ύψος του. Η ετικέτα y_i μπορεί να είναι είτε ένα στοιχείο που ανήκει σε ένα πεπερασμένο σύνολο κλάσεων $\{1, 2, ..., C\}$, ή πραγματικός αριθμός, ή μία πιο σύνθετη δομή, όπως ένα διάνυσμα, ένας πίνακας, ένα δέντρο ή ένα γράφημα. Η ετικέτα μπορεί να οριστεί ως μια κατηγορία στην οποία ανήκει το συγκεκριμένο δείγμα. Αν για παράδειγμα τα δείγματά μας ήταν e-mails τότε το κάθε δείγμα θα μπορούσε να έχει μία ετικέτα που ανήκει στο σύνολο κλάσεων {ανεπιθύμητη αλληλογραφία, μη-ανεπιθύμητη αλληλογραφία}.
   Για τις δύο πρώτες περιπτώσεις ετικετών (a. πεπερασμένο σύνολο κλάσεων και b. πραγματικός αριθμός) έχουμε αντίστοιχα προβλήματα **ταξινόμησης (classification)** και προβλήματα **παλινδρόμησης (Regression)**.

   (a) **Ταξινόμηση (classification)**
   Στα προβλήματα ταξινόμησης, οι τιμές των ετικετών ανήκουν σε ένα πεπερασμένο σύνολο κλάσεων, οι ετικέτες δηλαδή εκφράζουν ένα ποιοτικό χαρακτηριστικό. Ο αλγόριθμος έχει στόχο να ταξινομήσει τα δείγματα της εισόδου στην κατηγορία που ανήκουν. Για παράδειγμα, ένα πρόβλημα ταξινόμησης είναι και αυτό που αναφέρθηκε παραπάνω ως προς την ταξινόμηση των e-mails σε ανεπιθύμητα και μη.

   (b) **Παλινδρόμηση (Regression)**
   Στα προβλήματα παλινδρόμησης, οι τιμές των ετικετών παίρνουν συνεχείς τιμές, οι ετικέτες δηλαδή εκφράζουν ένα ποσοτικό χαρακτηριστικό. Ο αλγόριθμος σε αυτή την περίπτωση έχει ως στόχο να προβλέψει μια τιμή για τα δείγματα της εισόδου. Για παράδειγμα, ένα πρόβλημα παλινδρόμησης είναι η πρόβλεψη της ηλικίας ενός ανθρώπου.

2. **Μη-επιβλεπόμενη Μάθηση**
   Στην μη-επιβλεπόμενη μάθηση ή αλλιώς μη-επιτηρούμενη μάθηση, το σύνολο δεδομένων που χρησιμοποιεί ο αλγόριθμος είναι δείγματα τα οποία δεν είναι επισημειωμένα με κάποια ετικέτα (label) $\{x_i\}_{i=1}^{N}$. Σε αυτήν την περίπτωση, για κάθε διάνυσμα χαρακτηριστικών x_i, ο αλγόριθμος παράγει στην έξοδό του, είτε ένα νέο διάνυσμα χαρακτηριστικών, είτε μια τιμή. Παραδείγματος χάριν, το νέο διάνυσμα χαρακτηριστικών θα μπορούσε να είναι λιγότερων διαστάσεων απ᾽ αυτό της εισόδου έτσι ώστε ο αλγόριθμος να ικανοποιήσει τον σκοπό της μείωσης διαστάσεων. Αντίστοιχα, η

τιμή εξόδου θα μπορούσε να είναι ένας πραγματικός αριθμός που λειτουργεί ως ένα αναγνωριστικό του εκάστοτε δείγματος-διανύσματος εισόδου και δείχνει πως αυτό διαφέρει απ' τα υπόλοιπα δείγματα του συνόλου.

3. **Ημι-επιβλεπόμενη Μάθηση**
   Στην ημι-επιβλεπόμενη μάθηση ή αλλιώς ημι-επιτηρούμενη μάθηση, το σύνολο δεδομένων που χρησιμοποιεί ο αλγόριθμος περιέχει τόσο επισημειούμενα δείγματα με ετικέτες όσο και δείγματα χωρίς ετικέτες $(x_i, y_i)_{i=1}^{N}$, $x_{i\,i=N+1}^{K}$. Συνήθως τα δείγματα χωρίς ετικέτα υπερέχουν σε αριθμό, των δειγμάτων με ετικέτα. Ο στόχος του αλγορίθμου σε αυτήν την περίπτωση είναι ίδιος με τον στόχο της επιβλεπόμενης μάθησης. Η ελπίδα εδώ είναι ότι η χρήση πολλών παραδειγμάτων χωρίς ετικέτα μπορεί να βοηθήσει τον αλγόριθμο μάθησης να παράγει ένα καλύτερο μοντέλο.

4. **Ενισχυτική Μάθηση**
   Στην ενισχυτική μάθηση, το σύνολο των δεδομένων που χρησιμοποιείται δεν είναι επισημασμένο. Η μηχανή «ζει» σε ένα περιβάλλον και είναι ικανή να αντιληφθεί την κατάσταση αυτού του περιβάλλοντος ως ένα διάνυσμα χαρακτηριστικών. Η μηχανή μπορεί να ανταπεξέλθει σε κάθε κατάσταση (για κάθε διάνυσμα εισόδου). Η κάθε δράση της μηχανής μπορεί να έχει ως αποτέλεσμα διαφορετικές ανταμοιβές ή να φέρει την μηχανή σε μια νέα κατάσταση. Ο στόχος της μηχανής είναι να δημιουργήσει μια συνάρτηση-μοντέλο, η οποία δεδομένης μιας κατάστασης, θα επιλέγει μια ενέργεια που συμβάλλει στην μέγιστη αναμενόμενη μέση ανταμοιβή. Πιο συγκεκριμένα, ένας πράκτορας (agent) ανταμοίβεται ή τιμωρείται για τις κινήσεις (actions) που επιλέγει να κάνει σε ένα περιβάλλον (environment) προσπαθώντας να μεγιστοποιήσει μια αθροιστική μεταβλητή (reward). Παραδείγματα χρήσης τέτοιου είδους μοντέλων είναι τα αυτοκινούμενα αυτοκίνητα και τα προγράμματα που λειτουργούν ως παίκτες σκακιού. Σε αυτές τις περιπτώσεις μπορούμε να παρατηρήσουμε πως η μηχανή αναπροσαρμόζεται στο περιβάλλον της σύμφωνα με τα νέα δεδομένα/καταστάσεις που λαμβάνει (εμπόδια,δρόμοι και φανάρια για τα αυτοκίνητα ή νέες κινήσεις αντιπάλου για το σκάκι), έχοντας ως στόχο την μακροπρόθεσμη ανταμοιβή (να φτάσει στον προορισμό του ή να νικήσει στο σκάκι).

Στην παρούσα διπλωματική εργασία η κατηγορία μηχανικής μάθησης που χρησιμοποιείται είναι η επιβλεπόμενη μάθηση και πιο συγκεκριμένα η ταξινόμηση, καθώς στόχος είναι η ταξινόμηση-κατηγοριοποίηση των ομιλιών(δειγμάτων) σε κακής ή καλής ποιότητας.

2.2.3 Θεμελιώδεις Αλγόριθμοι

2.2.3.1 Linear Regression

Η γραμμική παλινδρόμηση (linear regression), είναι ένας γνωστός αλγόριθμος παλινδρόμησης ο οποίος λαμβάνει ως είσοδο μια συλλογή από επισημειωμένα δείγματα $\{x_i\}_{i=1}^{N}$, τα οποία είναι διανύσματα χαρακτηριστικών D-διαστάσεων και προβλέπει στην έξοδο γραμμικούς συνδιασμούς των χαρακτηριστικών των δειγμάτων αυτών $\{y_i\}_{i=1}^{N}$.

Κατ' αυτόν τον τρόπο, σκοπός είναι να παράγουμε ένα μοντέλο ως τον γραμμικό συνδιασμό των χαρακτηριστικών του δείγματος x. Η γραμμική εξίσωση που περιγράφει το μοντέλο αυτό, φαίνεται παρακάτω:

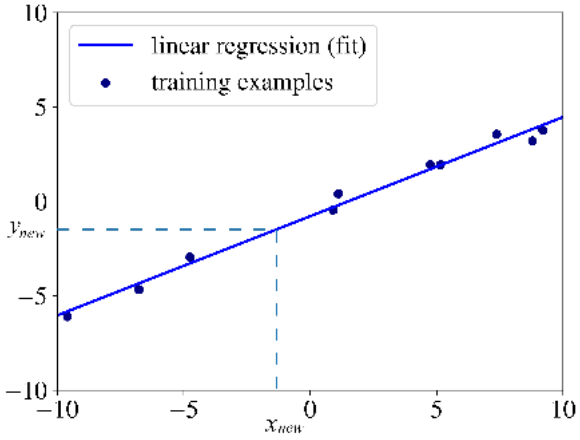
$$f_{w,b}(x) = wx + b \tag{2.1}$$

όπου **w** είναι ένα διάνυσμα παραμέτρων D-διαστάσεων και **b** είναι ένας πραγματικός αριθμός.

Θα χρησιμοποιήσουμε το μοντέλο αυτό για να προβλέψουμε την άγνωστη ετικέτα y για το δεδομένο δείγμα x. Αυτό που μας ενδιαφέρει είναι να κάνουμε μια πρόβλεψη που θα

προσεγγίζει όσο το δυνατόν καλύτερα την πραγματική τιμή της ετικέτας y. Για να γίνει αυτό αρκεί να βρούμε την βέλτιστη τιμή των παραμέτρων (w*,b*) που θα μας δώσουν τις πιο ακριβείς προβλέψεις.

Η παραπάνω εξίσωση-μοντέλο θα μπορούσε γραφικά να αναπαρασταθεί ως μια γραμμή η οποία, δεδομένων των δειγμάτων x, προσπαθεί να προσεγγίζει τις ετικέτες y με τον καλύτερο τρόπο. Παρακάτω φαίνεται μια τέτοια αναπαράσταση για μονοδιάστα δείγματα x [13]:



Σχήμα 2.5: Linear Regression για δείγματα μίας διάστασης.

Αυτή η γραμμή πρακτικά είναι το εργαλείο για να προβλέψουμε την ετικέτα y_{new}, ενός καινούριου δείγματος x_{new}.

Εάν τα παραδείγματα μας είναι διανύσματα χαρακτηριστικών D-διαστάσεων (για D> 1), η μόνη διαφορά από την μονοδιάστατη περίπτωση είναι ότι το μοντέλο παλινδρόμησης δεν είναι γραμμή αλλά επίπεδο (για δύο διαστάσεις) ή υπερεπίπεδο (για D > 2).

Η απαραίτητη προϋπόθεση που πρέπει να τηρείται εδώ, η οποία είναι και το κλειδί για να λύσουμε το πρόβλημά μας και να βρούμε το ζητούμενο μοντέλο, είναι ότι η γραμμή αυτή ή το υπερεπίπεδο πρέπει να βρίσκεται όσο πιο κοντά γίνεται σε όλα τα δεδομένα εκπαίδευσης y_i. Ειδάλλως, οι προβλέψεις y_{new} θα είχαν λιγότερες πιθανότητες να είναι σωστές. Η προϋπόθεση αυτή, εκφράζεται μαθηματικά ως η ελαχιστοποίηση της **συνάρτησης κόστους (Cost Function)** που στην προκειμένη περίπτωση είναι το **μέσο τετραγωνικό σφάλμα (Mean Squared Error(MSE))**:

$$min \quad \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2 \tag{2.2}$$

όπου η έκφραση $(f_{w,b}(x_i) - y_i)^2$, ονομάζεται **συνάρτηση απώλειας (Loss Function)** και είναι η διαφορά μεταξύ της προβλεπόμενης τιμής της ετικέτας και της πραγματικής τιμής της ετικέτας για το δείγμα i. Στην πραγματικότητα, λειτουργεί ως ένα μέτρο ποινής για εσφαλμένη ταξινόμηση του παραδείγματος i.

Από την παραπάνω μαθηματική έκφραση παρατηρούμε ότι για να πάρουμε το μέσο τετραγωνικό σφάλμα, το οποίο τελικά θέλουμε να ελαχιστοποιήσουμε, αθροίζουμε την συνάρτηση απώλειας για όλα τα δείγματα και διαιρούμε αυτήν την τιμή με τον συνολικό αριθμό δειγμάτων.

Η διαδικασία που ακολουθείται ώστε να βρεθούν οι βέλτιστες τιμές των παραμέτρων w* και b* και κατ' επέκταση το βέλτιστο μοντέλο μέσω της παραπάνω μαθηματικής έκφρασης, ονομάζεται διαδικασία βελτιστοποίησης (optimization procedure).

Ο λόγος για τον οποίο χρησιμοποιήθηκε το τετράγωνο της διαφοράς $(f_{w,b}(x_i) - y_i)^2$ και όχι η απόλυτη τιμή είναι επειδή η απόλυτη τιμή δεν έχει συνεχή παράγωγο, κάτι το οποίο θα καθιστούσε την συνάρτηση μη ομαλή. Συναρτήσεις που δεν είναι ομαλές δημιουργούν περιττές δυσκολίες όταν χρησιμοποιούν γραμμική άλγεβρα για την εξεύρεση λύσεων κλειστής μορφής σε προβλήματα βελτιστοποίησης. Διαισθητικά, οι τετράγωνες ποινές είναι καλύτερες επειδή υπερβάλλουν τη διαφορά μεταξύ του πραγματικού στόχου και του προβλεπόμενου σύμφωνα με την αξία αυτής της διαφοράς. Μπορεί επίσης να χρησιμοποιήσουμε τις δυνάμεις 3 ή 4, αλλά τα παράγωγά τους είναι πιο περίπλοκα για να εργαστούμε με αυτά.

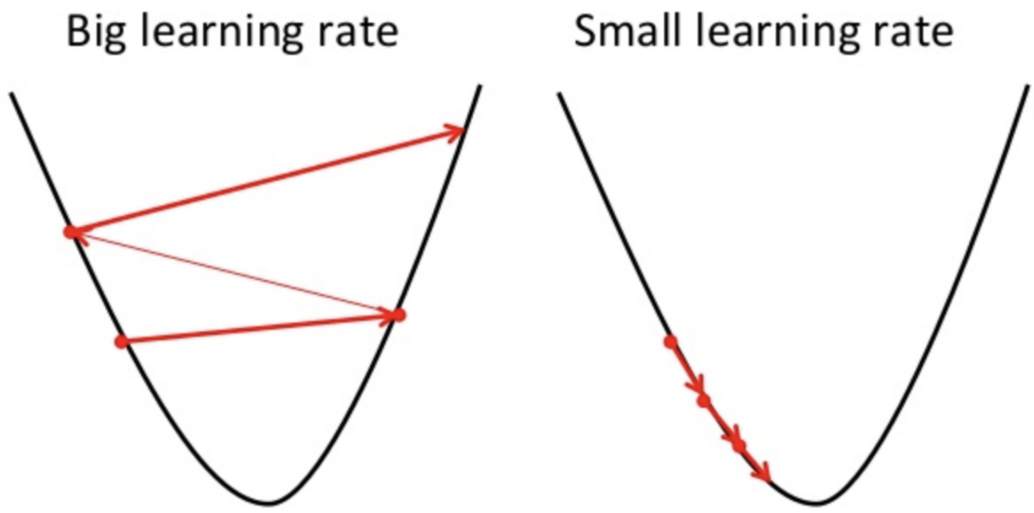
Το ερώτημα εδώ είναι ποιά μέθοδος βελτιστοποίησης θα χρησιμοποιηθεί, ώστε να βρεθούν οι τιμές w* και b* που να ικανοποιούν την (2.2). Η απάντηση στο ερώτημα αυτό δίνεται από τον παρακάτω αλγόριθμο.

Αλγόριθμος Απότομης Καθόδου (Gradient Descent)

Ο λόγος για τον οποίο μας ενδιαφέρουν οι παράγωγοι της συνάρτησης (2.2), όπως αναφέρθηκε παραπάνω, είναι επειδή παραγωγίζοντας την έκφραση που θέλουμε να ελαχιστοποιήσουμε και θέτοντας αυτήν την παράγωγο ίση με 0, μπορούμε να βρούμε την λύση σε ένα σύστημα εξισώσεων που θα μας δώσει την βέλτιστη τιμή των w* και b*. Η παραγώγηση γίνεται υπολογίζοντας τις μερικές παραγώγους ως προς w και ως προς b αντίστοιχα.

Η ιδέα είναι ότι ξεκινάμε με ορισμένες τιμές για τα w και b και μετά αλλάζουμε αυτές τις τιμές επαναληπτικά για να μειώσουμε το κόστος. Ο αλγόριθμος απότομης καθόδου ή αλλιώς αλγόριθμος σύγκλισης με ελάττωση της παραγώγου, μας βοηθά να αλλάξουμε τις τιμές αυτές.

Για να κατανοήσουμε καλύτερα την λειτουργία του αλγορίθμου αυτού, μπορούμε να φανταστούμε ότι βρισκόμαστε στην κορυφή μίας πλαγιάς. Για να φτάσουμε στο κατώτερο σημείο (ελάχιστο), θα ακολουθήσουμε μια συγκεκριμένη διαδρομή. Στην διαδρομή αυτή μπορούμε να πηγαίνουμε με μικρότερα βήματα-ρυθμό, με αποτέλεσμα να φτάσουμε αργότερα ή να πηγαίνουμε με μεγαλύτερα βήματα, με κίνδυνο να ξεπεράσουμε το κατώτερο σημείο. Στον αλγόριθμο απότομης καθόδου, το μέγεθος των βημάτων που ακολουθείται είναι το ποσοστό εκμάθησης (learning rate). Αυτό αποφασίζει για το πόσο γρήγορα ο αλγόριθμος συγκλίνει στα ελάχιστα. Παρακάτω φαίνεται γραφικά το σενάριο αυτό[15]:



Σχήμα 2.6: Ποσοστό εκμάθησης αλγορίθμου απότομης καθόδου.

Παρακάτω φαίνονται οι μαθηματικές πράξεις που πραγματοποιούνται:

$$J = \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2$$

$$J = \frac{1}{N}\sum_{i=1}^{N}(wx_i + b - y_i)^2$$

$$\frac{\partial J}{\partial w} = \frac{2}{N}\sum_{i=1}^{N}(wx_i + b - y_i)x_i \implies \frac{\partial J}{\partial w} = \frac{2}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)x_i$$

$$\frac{\partial J}{\partial b} = \frac{2}{N}\sum_{i=1}^{N}(wx_i + b - y_i) \implies \frac{\partial J}{\partial b} = \frac{2}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)$$

$$w^* = w - \alpha\frac{2}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)x_i \tag{2.3}$$

$$b^* = b - \alpha\frac{2}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i) \tag{2.4}$$

όπου το **α** είναι το ποσοστό εκμάθησης, μια παράμετρος η οποία καθορίζεται από εμάς και τα **w*** και **b*** είναι οι βέλτιστες παράμετροι. Ένα μικρότερο ποσοστό εκμάθησης θα μπορούσε να μας φέρει πιο κοντά στα ελάχιστα, αλλά χρειάζεται περισσότερος χρόνος για να φτάσουμε. Ένα μεγαλύτερο ποσοστό εκμάθησης συγκλίνει νωρίτερα, αλλά υπάρχει η πιθανότητα να υπερβούμε τα ελάχιστα.

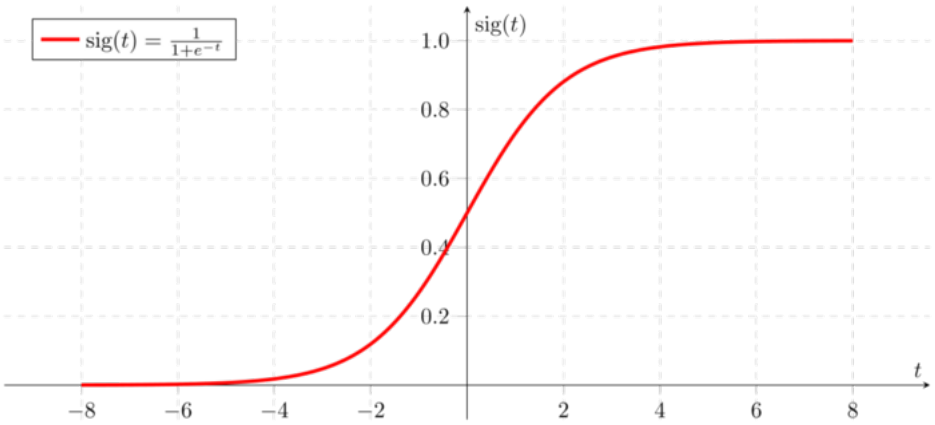
2.2.3.2 Logistic Regression

Ο αλγόριθμος λογιστικής παλινδρόμησης (logistic regression), δεν αναφέρεται σε πρόβλημα παλινδρόμησης αλλά ουσιαστικά σε πρόβλημα ταξινόμησης. Το όνομά της οφείλεται στο γεγονός ότι η μαθηματική διαμόρφωση της λογιστικής παλινδρόμησης είναι παρόμοια με αυτήν της γραμμικής παλινδρόμησης που είδαμε παραπάνω. Θα μελετήσουμε τον αλγόριθμο για το διαδικό πρόβλημα, ωστόσο μπορεί να επεκταθεί και σε προβλήματα ταξινόμησης περισσότερων κλάσεων.

Θα ορίσουμε μια αρνητική ετικέτα ως 0 και μια θετική ετικέτα ως 1. Αυτό που μένει να κάνουμε, είναι να βρούμε μια απλή συνεχή συνάρτηση με πεδίο τιμών το διάστημα [0,1]. Έτσι, για κάθε δείγμα εισόδου x_i, η συνάρτηση-μοντέλο θα επιστρέφει μια τιμή, η οποία εάν βρίσκεται πλησιέστερα στο 0, τότε αντιστοιχίζουμε αρνητική ετικέτα στο x_i. Διαφορετικά, ταξινομούμε το δείγμα ως θετικό. Μια συνάρτηση που έχει μια τέτοια ιδιότητα είναι η **συνάρτηση σιγμοειδούς (sigmoid function)**, η οποία παρουσιάζεται παρακάτω:

$$f(x) = \frac{1}{1 + e^{-x}}$$

όπου το **e** είναι η βάση του φυσικού λογάριθμου (ονομάζεται επίσης αριθμός Euler)[16].



Σχήμα 2.7: Συνάρτηση σιγμοειδούς (sigmoid function)

Οπότε τελικά το μοντέλο λογιστικής παλινδρόμησης θα είναι το παρακάτω:

$$f_{w,b}(x_i) = \frac{1}{1 + e^{-(wx_i+b)}} \tag{2.5}$$

όπου **w** και **b** οι παράμετροι προς βελτιστοποίηση. Εδώ βλέπουμε την γνωστή έκφραση $wx_i + b$ από την γραμμική παλινδρόμηση. Αυτό που μένει είναι να βρούμε τον τρόπο που θα υπολογίσουμε τα βέλτιστα w* και b*.

Στην περίπτωση της γραμμικής παλινδρόμησης θέλαμε να ελαχιστοποιήσουμε το μέσο τετραγωνικό σφάλμα (MSE). Σε αυτή την περίπτωση το κριτήριο βελτιστοποίησης είναι η **μεγιστη πιθανότητα (maximum likelihood)**:

$$max \quad \prod_{i=1}^{N} f_{w,b}(x_i)^{y_i}(1 - f_{w,b}(x_i))^{(1-y_i)} \tag{2.6}$$

Στην ουσία αυτό που θέλουμε είναι να μεγιστοποιήσουμε την πιθανότητα τα δεδομένα εκπαίδευσης να ταξινομηθούν σωστά. Αν δούμε πιο προσεκτικά το $f_{w,b}(x_i)$ είναι η έξοδος του μοντέλου μας που δίνει την πιθανότητα το δείγμα x_i να είναι θετικό. Το $1 - f_{w,b}(x_i)$ δίνει την πιθανότητα το δείγμα x_i να είναι αρνητικό. Εάν η πραγματική τιμή της ετικέτας του δείγματος είναι 1, δηλαδή $y_i = 1$, τότε η παραπάνω έκφραση θα γίνει ως εξής:

$$f_{w,b}(x_i)^{y_i}(1 - f_{w,b}(x_i))^{(1-y_i)} = f_{w,b}(x_i)$$

όπου τελικά καλούμαστε να μεγιστοποιήσουμε την πιθανότητα το δείγμα x_i να ταξινομηθεί με ετικέτα 1 (που είναι και η πραγματική). Αντίστοιχα εάν η πραγματική τιμή της ετικέτας του δείγματος είναι 0, δηλαδή $y_i = 0$, τότε η παραπάνω έκφραση θα γίνει ως εξής:

$$f_{w,b}(x_i)^{y_i}(1 - f_{w,b}(x_i))^{(1-y_i)} = 1 - f_{w,b}(x_i)$$

όπου τελικά καλούμαστε να μεγιστοποιήσουμε την πιθανότητα $1 - f_{w,b}(x_i)$, δηλαδή την πιθανότητα το δείγμα x_i να ταξινομηθεί με ετικέτα 0 (που είναι και η πραγματική).

Ο λόγος που εδώ χρησιμοποιήσαμε γινόμενο $\prod$ και όχι άθροισμα $\sum$ είναι γιατί η πιθανότητα παρατήρησης N ετικετών για N δείγματα είναι το γινόμενο των πιθανοτήτων κάθε παρατήρησης για κάθε δείγμα, όπως ακριβώς συμβαίνει και στην θεωρία πιθανοτήτων με τον πολλαπλασιασμό των πιθανοτικών αποτελεσμάτων σε μια σειρά ανεξάρτητων πειραμάτων.

Λόγω της λειτουργίας της εκθετικής συνάρτησης (exp) που χρησιμοποιείται στο μοντέλο, στην πράξη, είναι πιο βολικό να μεγιστοποιήσουμε την λογαριθμική πιθανότητα (log-likelihood) αντί της πιθανότητας. Το ln είναι μια αυστηρά αυξανόμενη συνάρτηση

και η μεγιστοποίηση αυτής είναι το ίδιο με το να μεγιστοποιούμε το όρισμά της. Η νέα έκφραση προς μεγιστοποίηση ορίζεται ως εξής:

$$max \quad \sum_{i=1}^{N} y_i \ln f_{w,b}(x_i) + (1 - y_i) \ln (1 - f_{w,b}(x_i)) \tag{2.7}$$

Επειδή εξ' ορισμού η συνάρτηση κόστους (cost function) είναι κάτι που έχει αρνητική επίδραση και θα θέλαμε να το ελαχιστοποιήσουμε, θα μετατρέψουμε το παραπάνω πρόβλημα μεγιστοποίησης σε ελαχιστοποίησης:

$$min \quad -\sum_{i=1}^{N} y_i \ln f_{w,b}(x_i) + (1 - y_i) \ln (1 - f_{w,b}(x_i)) \tag{2.8}$$

Ουσιαστικά, κάναμε την έκφραση από θετική, αρνητική και έτσι αντί να την μεγιστοποιήσουμε, θέλουμε να την ελαχιστοποιήσουμε. Η νέα έκφραση προς ελαχιστοποίηση ονομάζεται **εγκάρσια εντροπία (cross entropy)**.

Εν κατακλείδι, για να βρούμε τις βέλτιστες παραμέτρους w* και b* που να ικανοποιούν την (2.8), εφαρμόζουμε και πάλι τον αλγόριθμο απότομης καθόδου (gradient descent):

$$J = -\sum_{i=1}^{N} y_i \ln f_{w,b}(x_i) + (1 - y_i) \ln (1 - f_{w,b}(x_i))$$

$$J = -\sum_{i=1}^{N} y_i \ln \frac{1}{1 + e^{-(wx_i+b)}} + (1 - y_i) \ln (1 - \frac{1}{1 + e^{-(wx_i+b)}})$$

$$w^* = w - \alpha \frac{\partial J}{\partial w} \tag{2.9}$$

$$b^* = b - \alpha \frac{\partial J}{\partial b} \tag{2.10}$$

όπου $\frac{\partial J}{\partial w} = \sum_{i=1}^{N} (f_{w,b}(x_i) - y_i) x_i$, $\frac{\partial J}{\partial b} = \sum_{i=1}^{N} (f_{w,b}(x_i) - y_i)$ και **α** το ποσοστό εκμάθησης.

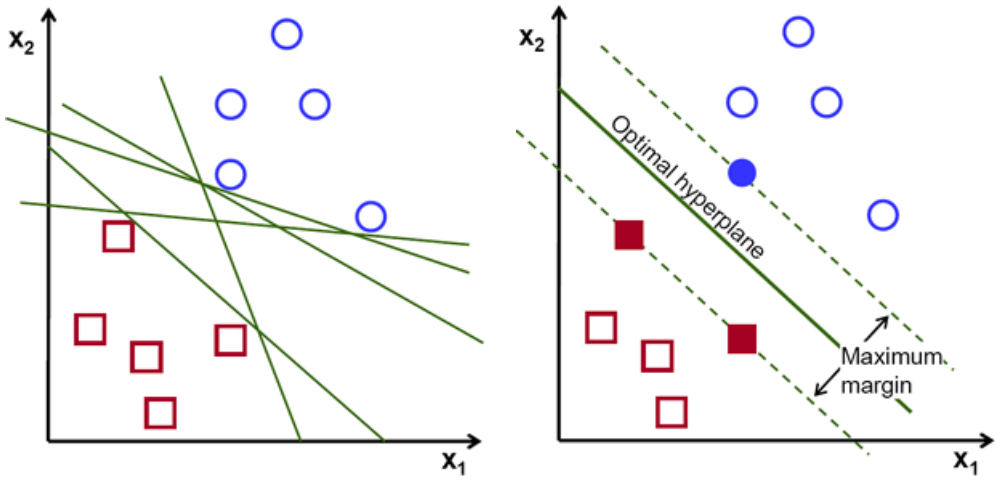
Οι παραπάνω δύο αλγόριθμοι (linear regression, logistic regression) δεν θα χρησιμοποιηθούν άμεσα στην διπλωματική αυτή, αλλά θα βοηθήσουν στην καλύτερη και ευκολότερη κατανόηση των αλγορίθμων που παρουσιάζονται στα κεφάλαια που ακολουθούν.

2.2.3.3 Support Vector Machine (SVM)

Ο αλγόριθμος "Μηχανές Διανυσμάτων Υποστήριξης" (SVM) βλέπει τα διανύσματα χαρακτηριστικών των δειγμάτων, ως σημεία σε έναν χώρο πολλών διαστάσεων (N-διαστάσεων). Έτσι, τοποθετώντας τα διανύσματα χαρακτηριστικών σε ένα φανταστικό γράφημα, ο αλγόριθμος σχεδιάζει μια γραμμή (ένα υπερεπίπεδο)(N-διαστάσεων), το οποίο διαχωρίζει τα δείγματα με θετικές ετικές από τα δείγματα με αρνητικές ετικέτες (ταξινόμηση). Το όριο που διαχωρίζει τα δείγματα διαφορετικών κλάσεων ονομάζεται **όριο απόφασης (decision boundary)**.

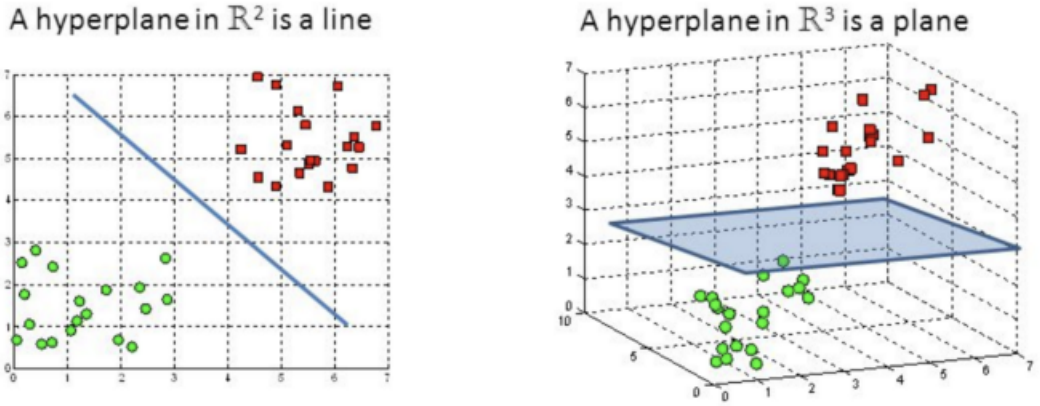
Για να διαχωρίσουμε τα δείγματα σε κλάσεις, υπάρχουν πολλά υπερεπίπεδα που θα μπορούσαμε να σχεδιάσουμε. Στόχος μας είναι να βρούμε ένα επίπεδο που έχει το μέγιστο περιθώριο, δηλαδή τη μέγιστη απόσταση μεταξύ των σημείων δεδομένων και των δύο κατηγοριών. Η μεγιστοποίηση της απόστασης περιθωρίου παρέχει κάποια ενίσχυση

έτσι ώστε τα μελλοντικά σημεία δεδομένων να μπορούν να ταξινομηθούν με μεγαλύτερη εμπιστοσύνη[17].



Σχήμα 2.8: Πιθανά όρια-υπερεπίπεδα

Παρακάτω φαίνονται γραφήματα για δισδιάστατο και τρισδιάστατο χώρο αντίστοιχα. Δηλαδή για διάνυσμα χαρακτηριστικών 2 διαστάσεων και διάνυσμα χαρακτηριστικών 3ων διαστάσεων[17].



Σχήμα 2.9: Υπερεπίπεδα σε χώρο 2 και 3ων διαστάσων

Το μοντέλο σε αυτήν την περίπτωση ορίζεται ως εξής:

$$f_{w,b}(x_i) = sign(wx_i + b) \tag{2.11}$$

όπου $\boldsymbol{wx_i + b}$ είναι η γραμμή-υπερεπίπεδο που εξηγήσαμε παραπάνω και $\boldsymbol{sign}$ είναι η συνάρτηση signum η οποία παίρνει ως είσοδο οποιοδήποτε πραγματικό αριθμό και επιστρέφει +1 αν ο αριθμός αυτός είναι θετικός, -1 αν ο αριθμός είναι αρνητικός και 0 αν ο αριθμός είναι ίσος με το 0. Ο μαθηματικός ορισμός της συνάρτησης αυτής παρουσιάζεται παρακάτω:

$$sign(x) = \begin{cases} -1, x < 0 \\ 0, x = 0 \\ 1, x > 0 \end{cases}$$

Σκοπός είναι να βρεθούν οι βέλτιστες τιμές w* και b* των παραμέτρων w και b , όπως είδαμε και στους προηγούμενους αλγορίθμους.

Με λίγα λόγια, αν το δείγμα x_i βρίσκεται κάτω από το υπερεπίπεδο $w^* x_i + b^*$ τότε θα ταξινομείται ως αρνητικό (ετικέτα -1), ενώ αν βρίσκεται πάνω απ᾽ το υπερεπίπεδο τότε θα ταξινομείται ως θετικό (ετικέτα +1).

Hard-Margin (σκληρό περιθώριο)

Στην περίπτωση που τα δεδομένα είναι γραμμικώς διαχωρίσιμα, για να βρεθούν οι βέλτιστες παράμετροι w* και b*, θα λύσουμε και πάλι ένα πρόβλημα βελτιστοποίησης (optimization problem) με τους παρακάτω περιορισμούς:

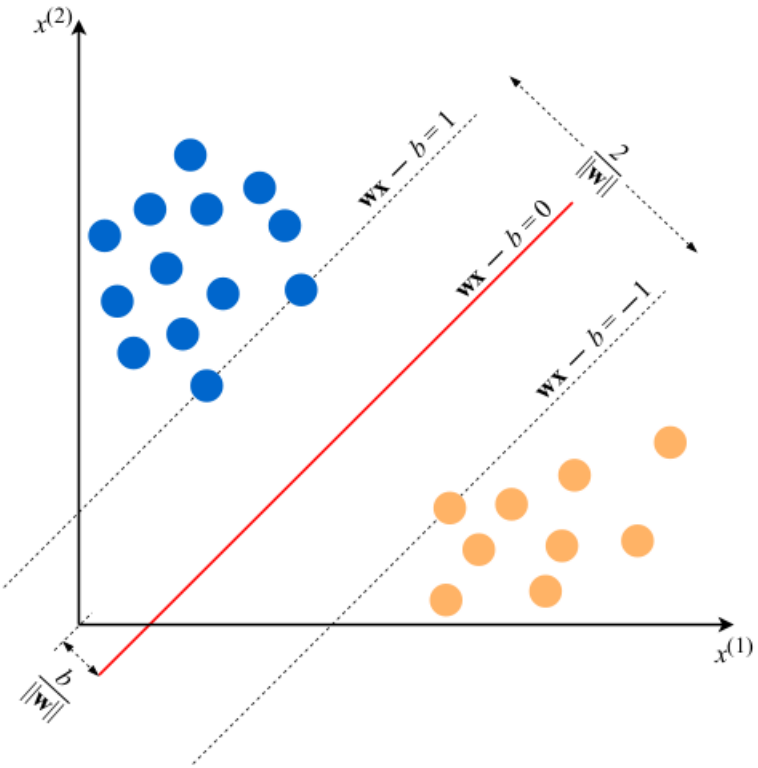
- $wx_i - b \geq 1$, if $y_i = +1$
- $wx_i - b \leq -1$, if $y_i = -1$
- Μέγιστο περιθώριο (margin), του υπερεπιπέδου με τα δείγματα των 2 κλάσεων.

Το παραπάνω πρόβλημα βελτιστοποίησης ορίζεται μαθηματικά ως εξής:

$$min \quad ||w|| \quad \text{subject to} \quad y_i(wx_i - b) \geq 1 \tag{2.12}$$

όπου $\mathbf{||w|| = \sqrt{\sum_{j=1}^{D}(w^{(j)})^2}}$ είναι το περιθώριο του ορίου με τα δείγματα και $\boldsymbol{y_i(wx_i - b) \geq 1}$ εκφράζει τους δύο πρώτου περιορισμούς, αφού αν η πραγματική ετικέτα y_i του δείγματος x_i είναι 1 τότε $y_i(wx_i - b) \geq 1 \implies (wx_i - b) \geq 1$ και αν η πραγματική ετικέτα y_i είναι -1 τότε $y_i(wx_i - b) \geq 1 \implies (wx_i - b) \leq -1$

Οι ισότητες $wx_i - b = 1$ και $wx_i - b = -1$ ορίζουν δυο παράλληλα υπερεπίπεδα στο όριο απόφασής μας $wx_i - b = 0$. Η απόσταση απ᾽ αυτά τα δύο υπερεπίπεδα ορίζεται ως $\frac{2}{||w||}$. Επομένως, όσο πιο μικρή είναι η νόρμα $||w||$, τόσα πιο μεγάλη είναι η απόσταση από τα δύο υπερεπίπεδα [13].



Σχήμα 2.10: Περιθώριο υπερεπιπέδου

Το μοντέλο που περιγράφηκε παραπάνω είναι γραμμικό, αφού χρησιμοποιεί ως όριο απόφασης μια γραμμή (ή ένα επίπεδο ή ενα υπερεπίπεδο) και τα δεδομένα είναι γραμμικώς διαχωρίσιμα. Υπάρχουν περιπτώσεις όμως όπου τα δεδομένα δεν είναι γραμμικώς διαχωρίσιμα σε κλάσεις. Αυτό γιατί μπορεί να υπάρχουν **outliers**, δηλαδή δείγματα που βρίσκονται αρκετά μακρυά στον χώρο, σε σχέση με τα υπόλοιπα, είτε μπορεί να υπάρχει θόρυβος στα δεδομένα.

Soft-Margin (μαλακό περιθώριο)

Θα επεκτείνουμε τον αλγόριθμο σε περιπτώσεις όπου τα δεδόμενα δεν είναι γραμμικώς διαχωρίσιμα.

Για λόγους ευκολίας θα συμβολίζουμε το όριο απόφασης ως:

$$wx_i - b = \beta_0 + \beta_1 x_{i1} + ... + \beta_k x_{ik} = \beta x_i$$

εφόσον το δείγμα x_i είναι διάνυσμα χαρακτηριστικών k-διαστάσεων.

Εδώ θα χρησιμοποιήσουμε την συνάρτηση κόστους **hinge loss**:

$$max(0, 1 - y_i(\beta x_i))$$

η οποία είναι 0 αν οι δύο πρώτοι περιορισμοί που είδαμε παραπάνω, ικανοποιούνται. Δηλαδή, αν το δείγμα x_i είναι θετικό και το $\beta x_i \geq 1$ ή αν το δείγμα x_i είναι αρνητικό και το $\beta x_i \leq -1$. Αυτό σημαίνει πως η συνάρτηση κόστους αυτή, μηδενίζεται όταν το βx_i ταξινομείται στην σωστή πλευρά του ορίου απόφασης (decision boundary).

Η έκφραση που τελικά θέλουμε να ελαχιστοποιήσουμε εδώ είναι η εξής:

$$min \quad C\|\beta\|^2 + \frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_i(\beta x_i)) \tag{2.13}$$

όπου **||β||** είναι και πάλι το περιθώριο του ορίου απόφασης με τα δείγματα.

Η παράμετρος ***C*** καθορίζει την εξισορρόπηση μεταξύ του μεγέθους επιρροής του ορίου απόφασης και της εξασφάλισης ότι κάθε δείγμα x_i βρίσκεται στην σωστή πλευρά του ορίου απόφασης. Η παράμετρος C επιλέγεται συνήθως πειραματικά. Για αρκετά υψηλές τιμές C, ο δεύτερος όρος στη συνάρτηση κόστους θα γίνει αμελητέος, οπότε ο αλγόριθμος SVM θα προσπαθήσει να βρει το υψηλότερο περιθώριο αγνοώντας εντελώς την εσφαλμένη ταξινόμηση. Καθώς μειώνουμε την τιμή του C, η πραγματοποίηση σφαλμάτων ταξινόμησης γίνεται πιο δαπανηρή, οπότε ο αλγόριθμος SVM θα προσπαθήσει να κάνει λιγότερα λάθη θυσιάζοντας το μέγεθος του περιθωρίου. Ένα μεγαλύτερο περιθώριο είναι καλύτερο για τη γενίκευση. Επομένως, το C ρυθμίζει την ανταλλαγή μεταξύ της καλής ταξινόμησης των δεδομένων εκπαίδευσης (ελαχιστοποίηση του εμπειρικού κινδύνου) και της καλής ταξινόμησης μελλοντικών παραδειγμάτων (γενίκευση). Για την κανονικοποίηση (regularization) θα μιλήσουμε αργότερα στο κεφάλαιο 2.2.7.

Εδώ θα πρέπει να σημειωθεί ότι το να ελαχιστοποιήσουμε το $\|\beta\|^2$ είναι το ίδιο με το να ελαχιστοποιούμε το $\|\beta\|$.

Αυτό που τελικά μας απασχολεί, είναι να βρούμε τις βέλτιστες τιμές των παραμέτρων β. Για να βρεθούν αυτές θα χρησιμοποιηθεί και πάλι ο αλγόριθμος απότομης καθόδου (gradient descent), όπως είδαμε και στα προηγούμενα κεφάλαια. Παρακάτω, υπολογίζεται η μερική παράγωγος της έκφρασης (2.13) ως προς β:

$$\frac{\partial}{\partial\beta}(C\|\beta\|^2) = 2C\beta$$

$$\frac{\partial}{\partial\beta}(max(0, 1 - y_i(\beta x_i))) = \begin{cases} 0, & \text{if} \quad y_i(\beta x_i) \geq 1 \\ -y_i x_i, & \text{else} \end{cases}$$

$$\frac{\partial}{\partial\beta}(C\|\beta\|^2 + \frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_i(\beta x_i))) = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} 2C\beta, & \text{if} \quad y_i(\beta x_i) \geq 1 \\ 2C\beta - y_i x_i, & \text{else} \end{cases} \tag{2.14}$$

Βέλτιστη τιμή παραμέτρων β:

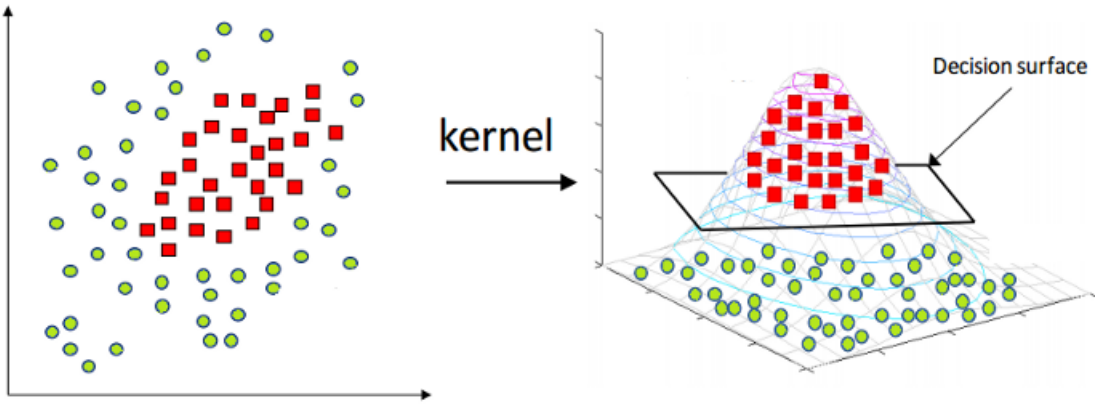
$$\begin{cases} \beta^* = \beta - \alpha(2C\beta), & \text{No misclassification} \\ \beta^* = \beta + \alpha(y_i x_i - 2C\beta), & \text{Misclassification} \end{cases} \tag{2.15}$$

όπου ***α*** είναι το ποσοστό εκμάθησης όπως έχουμε δει και στα προηγούμενα κεφάλαια.

Kernel Trick (κόλπο πυρήνα)

Ο παραπάνω αλγόριθμος του μέγιστου περιθωρίου του υπερεπιπέδου (ορίου απόφασης), δημιουργεί έναν γραμμικό ταξινομητή. Ο SVM όμως μπορεί να προσαρμοστεί ώστε να δουλεύει και σε περιπτώσεις όπου τα δεδομένα δεν μπορούν να διαχωριστούν από ένα υπερεπίπεδο στον αυθεντικό τους χώρο. Αν καταφέρουμε να μετασχηματίσουμε τον αυθεντικό τους χώρο, σε έναν χώρο περισσότερων διαστάσεων, τότε μπορεί τα δεδομένα να είναι πλέον γραμμικώς διαχωρίσιμα στον νέο χώρο. Αυτή η μέθοδος ονομάζεται **kernel trick (κόπλο πυρήνα)**.

Παρακάτω φαίνεται μια τέτοια περίπτωση όπου στον αυθεντικό χώρο 2-διαστάσεων (αριστερά), τα δεδομένα δεν είναι γραμμικώς διαχωρίσιμα, αλλά όταν ο χώρος μετατρέπεται σε 3-διαστάσεις (δεξιά), τότε τα δεδομένα μπορούν να διαχωριστούν με ένα υπερεπίπεδο [14].



Σχήμα 2.11: Kernel Trick (κόλπο πυρήνα)

Ο αλγόριθμος που προκύπτει είναι τυπικά παρόμοιος, εκτός του ότι κάθε εσωτερικό γινόμενο αντικαθίσταται από μια μη γραμμική συνάρτηση πυρήνα (kernel function). Αυτό επιτρέπει στον αλγόριθμο να βρίσκει το μέγιστο περιθώριο υπερεπιπέδου σε ένα μετασχηματισμένο χώρο χαρακτηριστικών.

Για να δούμε πιο αναλυτικά πως λειτουργεί ο αλγόριθμος με την προσθήκη των kernels, θα δούμε πρώτα πως λύνεται η παρακάτω μαθηματική έκφραση:

$$min \quad \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(wx_i - b) - 1 \geq 0, \text{ i=1,...,N} \tag{2.16}$$

η οποία είναι αντίστοιχη της (2.12) και ορίζει τον αλγόριθμο SVM για γραμμικώς διαχωρίσιμα δεδομένα.

Για να λυθεί η (2.16) χρησιμοποιείται η μέθοδος πολλαπλασιαστών Lagrange (Lagrange multipliers method). Έτσι η (2.16) ορίζεται ισοδύναμα ως:

$$\max_{a_i \ldots a_N} \sum_{i=1}^{N} a_i - \sum_{i=1}^{N}\sum_{k=1}^{N} y_i a_i (x_i x_k) y_k a_k \quad \text{subject to} \quad \sum_{i=1}^{N} a_i y_i = 0 \quad \text{and} \quad a_i \geq 0, \text{ i=1,...,N} \tag{2.17}$$

όπου $\boldsymbol{a_i}$ είναι οι πολλαπλασιαστές Lagrange.

Τετραγωνικός Kernel (quadratic kernel)

Ο τετραγωνικός kernel ορίζεται ως εξής:

$$k(x_i, x_k) \stackrel{def}{=} (x_i x_k)^2 \tag{2.18}$$

Στην σχέση (2.17), παρατηρούμε ότι το μόνο σημείο στο οποίο υπάρχουν τα διανύσματα χαρακτηριστικών είναι το εσωτερικό γινόμενο $(x_i x_k)$. Εάν ο αυθεντικός χώρος ήταν 2-διαστάσεων και θέλαμε να αναγάγουμε το πρόβλημα σε 3-διαστάσεις τότε θα έπρεπε να μετατρέψουμε τα διανύσματα χαρακτηριστικών $x_i = (p_i, q_i)$ και $x_k = (p_k, q_k)$, σε $(p_i, q_i) \implies (p_i{}^2, \sqrt{2}p_i q_i, q_i{}^2)$ και $(p_k, q_k) \implies (p_k{}^2, \sqrt{2}p_k q_k, q_k{}^2)$ αντίστοιχα και μετά να υπολογίσουμε το εσωτερικό γινόμενο μεταξύ των νέων διανυσμάτων. Αντ' αυτού, παίρνοντας το εσωτερικό γινόμενο των αρχικών διανυσμάτων (2-διαστάσεων) και υψώνοντας στο τετράγωνο, θα έχουμε ακριβώς το ίδιο αποτέλεσμα με λιγότερο κόπο $x_i x_k \implies (p_i, q_i)(p_k, q_k) \implies (p_i p_k + q_i q_k) \underset{quadrate}{\implies} (p_i{}^2 p_k{}^2 + 2p_i p_k q_i q_k + q_i{}^2 q_k{}^2)$. Έτσι χρησιμοποιήσαμε τον τετραγωνικό kernel.

Στην ουσία, αυτό που κάνει το kernel trick για εμάς είναι να προσφέρει έναν πιο αποτελεσματικό και λιγότερο ακριβό τρόπο μετατροπής των δεδομένων σε υψηλότερες διαστάσεις. Έτσι, η εφαρμογή του kernel trick δεν περιορίζεται στον αλγόριθμο SVM. Τυχόν υπολογισμοί που περιλαμβάνουν τα εσωτερικά γινόμενα (x_i, y_i) μπορούν να χρησιμοποιήσουν το κόλπο του πυρήνα.

Πολυωνυμικός Kernel (polynomial kernel)

Ο πολυωνυμικός kernel είναι η γενική κατηγορία στην οποία ανήκει και ο τετραγωνικός και επεκτείνεται σε διαστάσεις μεγαλύτερες του 2 (d > 2):

$$k(x_i, x_k) \overset{def}{=} (x_i x_k + c)^d \tag{2.19}$$

όπου το $\boldsymbol{c \geq 0}$ είναι μια ελεύθερη παράμετρος που ανταλλάσσει την επίδραση των όρων υψηλότερης τάξης έναντι χαμηλότερης τάξης στο πολυώνυμο. Όταν c = 0, ο πυρήνας ονομάζεται ομοιογενής **(homogeneous kernel)**.

RBF Kernel (radial basis function kernel)

Ο kernel rbf (συνάρτησης ακτινικής βάσης) ή αλλιώς Gaussian kernel ορίζεται ως εξής:

$$k(x_i, x_k) \overset{def}{=} exp(-\frac{||x_i - x_k||^2}{2\sigma^2}) \tag{2.20}$$

όπου το **σ** καθορίζει την μορφή του ορίου απόφασης (decision boundary) στον αρχικό χώρο διαστάσεων (δηλαδή το αν θα είναι ομαλό ή θα έχει πολλές γωνίες) και το $||x_i - x_k||^2$ είναι το τετράγωνο της **Ευκλείδιας απόστασης (Euclidean distance)** μεταξύ των δύο διανυσμάτων χαρακτηριστικών η οποία ορίζεται ως εξής:

$$||x_i - x_k|| = d(x_i, x_k) \overset{def}{=} \sqrt{(x_i{}^{(1)} - x_k{}^{(1)})^2 + (x_i{}^{(2)} - x_k{}^{(2)})^2 + ... + (x_i{}^{(N)} - x_k{}^{(N)})^2} = \sqrt{\sum_{j=1}^{N} (x_i{}^{(j)} - x_k{}^{(j)})^2} \tag{2.21}$$

Σε αυτόν τον kernel βλέπουμε ότι ο διανυσματικός χώρος είναι άπειρων διαστάσεων (N), επειδή μπορεί να επεκταθεί από την σειρά Taylor.

Οι δύο παραπάνω kernels είναι οι πιο συχνά χρησιμοποιούμενοι στο πεδίο της μηχανικής μάθησης. Αν και ο kernel RBF είναι πιο δημοφιλής στην ταξινόμηση SVM από τον πολυωνυμικό kernel, ο τελευταίος είναι αρκετά δημοφιλής στην επεξεργασία φυσικής γλώσσας (Natural Language Processing-NLP). Ο πιο συνηθισμένος βαθμός είναι d = 2 (τετραγωνικός), καθώς οι μεγαλύτεροι βαθμοί τείνουν να κάνουν overfitting [2.2.7] στα προβλήματα NLP.

2.2.3.4 Naive Bayes

Οι ταξινομητές Naive Bayes είναι μια οικογένεια πιθανοτικών μοντέλων που βασίζονται στο θεώρημα του Bayes, θεωρώντας ότι τα χαρακτηριστικά είναι ανεξάρτητα μεταξύ τους.

Το γνωστό θεώρημα bayes δίνεται από την παρακάτω ισότητα:

$$p(A|B) = \frac{p(A)p(B|A)}{p(B)} \tag{2.22}$$

όπου τα A και B είναι γεγονότα και το p(B) είναι διάφορο του μηδενός. Επίσης, το p(A|B) είναι η πιθανότητα υπό συνθήκη (conditional probability), δηλαδή η πιθανότητα του να συμβεί το A δεδομένου ότι συμβαίνει το B, το p(B|A) είναι αντίστοιχα η πιθανότητα

να συμβαίνει το B δεδομένου ότι συμβαίνει το A και τα p(A), p(B) είναι οι πιθανότητες παρατήρησης των A και B, ανεξάρτητων μεταξύ τους, που είναι γνωστές και ως οριακές πιθανότητες (marginal probabilities).

Όταν πρόκειται για μια διεργασία ταξινόμησης, όπου σκοπός είναι να βρεθεί η κλάση παρατήρησης, δεδομένων κάποιων τιμών χαρακτηριστικών, το παραπάνω θεώρημα μπορεί να γραφεί και ως εξής:

$$p(y_i|x_1, x_2, .., x_n) = \frac{p(x_1, x_2, ..., x_n|y_i)p(y_i)}{p(x_1, x_2, ..., x_n)} \tag{2.23}$$

όπου $p(y_i|x_1, x_2, .., x_n)$ είναι η πιθανότητα το δείγμα να ανήκει στην κλάση y_i, δεδομένων των τιμών των χαρακτηριστικών $x_1, x_2, ..., x_n$. Επίσης, το $p(x_1, x_2, ..., x_n|y_i)$ δηλώνει την πιθανότητα εμφάνισης συγκεκριμένου συνδυασμού χαρακτηριστικών, δεδομένης της ετικέτας μιας κλάσης.

Στην παραπάνω σχέση (2.23), ο παρονομαστής μπορεί να αφαιρεθεί καθώς το μόνο που κάνει είναι να κανονικοποιεί την τιμή της πιθανότητας του αριστερού μέλους της εξίσωσης. Επιπλέον, η πιθανότητα $p(y_i)$ υπολογίζεται πολύ απλά ως εξής:

$$p(y_i) = \frac{number \quad of \quad observations \quad with \quad class \quad y_i}{number \quad of \quad all \quad observations} \tag{2.24}$$

Τέλος, όπως ήδη προαναφέρθηκε, όλοι οι ταξινομητές Bayes υποθέτουν ότι η τιμή ενός συγκεκριμένου χαρακτηριστικού είναι ανεξάρτητη από την τιμή οποιουδήποτε άλλου χαρακτηριστικού, δεδομένης της μεταβλητής κλάσης, για αυτό και ονομάζονται αφελείς (naive). Για παράδειγμα, ένα φρούτο μπορεί να θεωρηθεί μήλο εάν είναι κόκκινο, στρογγυλό και περίπου 10 cm σε διάμετρο. Ένας αφελής ταξινομητής Bayes θεωρεί ότι κάθε ένα από αυτά τα χαρακτηριστικά συμβάλλει ανεξάρτητα στην πιθανότητα ότι αυτό το φρούτο είναι ένα μήλο, ανεξάρτητα από τυχόν πιθανές συσχετίσεις μεταξύ των χαρακτηριστικών χρώματος, στρογγυλοποίησης και διαμέτρου. Κάτω από αυτήν την θεώρηση η πιθανότητα $p(x_1, x_2, ..., x_n|y_i)$ της σχέσης (2.23) μπορεί να γραφεί ως εξής:

$$p(x_1, x_2, ..., x_n|y_i) = p(x_1|y_i)p(x_2|y_i)...p(x_n|y_i) \tag{2.25}$$

εφόσον είναι γνωστό πως για ανεξάρτητα γεγονότα A και B, η πιθανότητα σύζευξης τους ισούται με το γινόμενο των πιθανοτήτων τους: $p(A \quad and \quad B) = p(A)p(B)$.

Σύμφωνα με τα παραπάνω, η σχέση (2.23) μετατρέπεται ως εξής:

$$p(y_i|x_1, x_2, .., x_n) = p(x_1|y_i)p(x_2|y_i)...p(x_n|yi)p(y_i) \implies$$

$$p(y_i|x_1, x_2, .., x_n) = p(y_i)\prod_{i=1}^{n} p(x_i|y) \tag{2.26}$$

Από την σχέση (2.26), μπορεί τελικά να δημιουργηθεί μια συνάρτηση απόφασης, η οποία προσδιορίζει την πρόβλεψη του μοντέλου δηλαδή την κλάση στην οποία ταξινομείται το εκάστοτε δείγμα. Ένας κοινός κανόνας απόφασης, είναι να επιλέγεται η υπόθεση που είναι πιο πιθανή. Αυτός ο κανόνας ονομάζεται maximum a posteriori ή αλλιώς MAP decision rule. Με βάση αυτόν, το δείγμα θα ταξινομηθεί στην ετικέτα y_k ως εξής:

$$\hat{y} = \underset{k\in\{1,...,K\}}{\operatorname{argmax}}\, p(y_k)\prod_{i=1}^{n} p(x_i|y_k) \tag{2.27}$$

Με λίγα λόγια, επιλέγεται η ετικέτα-κλάση ανάμεσα στις Κ συνολικά, που έχει την μέγιστη πιθανότητα παρατήρησης, δεδομένων των συγκεκριμένων τιμών των χαρακτηριστικών που παρουσιάζει το εκάστοτε δείγμα προς ταξινόμηση.

Τύποι Naive Bayes Ταξινομητών

Η πιθανότητα υπό συνθήκη για ένα συγκεκριμένο χαρακτηριστικό, δεδομένης της ετικέτας της κλάσης (πχ το $p(x_1|y_i)$ της σχέσης (2.26)), υπολογίζεται πιο εύκολα από τα δεδομένα. Ο αλγόριθμος χρειάζεται να αποθηκεύει τις κατανομές πιθανότητας των χαρακτηριστικών για κάθε κλάση-ετικέτα ξεχωριστά. Έτσι, αν έχουμε 5 κλάσεις και 10 χαρακτηριστικά, θα πρέπει να αποθηκευτούν 50 διαφορετικές κατανομές πιθανότητας. Για να υπολογιστούν οι παράμετροι της κατανομής ενός χαρακτηριστικού, θα πρέπει να υποτεθεί μια κατανομή. Οι υποθέσεις των κατανομών των χαρακτηριστικών ονομάζονται "γεγονός μοντέλου" (event model) του ταξινομητή Naive Bayes.

Ο τύπος της κατανομής εξαρτάται από τα χαρακτηριστικά:

- Για δυαδικά χαρακτηριστικά, όπως είναι η εμφάνιση ή όχι μιας λέξης μέσα σε ένα κείμενο, χρησιμοποιείται η κατανομή Bernoulli. Έτσι η πιθανότητα ενός χαρακτηριστικού δεδομένης μιας κλάσης γίνεται ως εξής:

$$p(x|y_k) = \prod_{i=1}^{n} p(x_i|y_k)^{x_i}(1 - p(x_i|y_k))^{(1-x_i)} \tag{2.28}$$

- Για διακριτά χαρακτηριστικά, όπως είναι η συχνότητα εμφάνισης των λέξεων μέσα σε ένα κείμενο, χρησιμοποιείται η κατανομή Multinomial. Έτσι η πιθανότητα ενός χαρακτηριστικού δεδομένης μιας κλάσης γίνεται ως εξής:

$$p(x|y_k) = \frac{(\sum_{i=1}^{n} x_i)!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} p(x_i|y_k)^{x_i} \tag{2.29}$$

- Για συνεχή χαρακτηριστικά, χρησιμοποιείται η Γκαουσιανή/κανονική κατανομή (Gaussian/Normal Distribution). Έτσι, αν το σύνολο εκπαίδευσης περιλαμβάνει το χαρακτηριστικό x, πρώτα χωρίζονται τα χαρακτηριστικά με βάση την κλάση και στην συνέχεια υπολογίζεται η μέση τιμή και η απόκλιση του x σε κάθε κλάση. Αν το μ_k είναι η μέση τιμή του χαρακτηριστικού x στην κλάση y_k και ${\sigma_k}^2$ είναι η απόκλιση των τιμών του χαρακτηριστικού x και υποθέτοντας πως υπάρχει μια τιμή "υ" παρατήρησης του χαρακτηριστικού αυτού, τότε η πιθανότητα παρατήρησης της τιμής αυτής δεδομένης της κλάσης y_k δίνεται από την παρακάτω σχέση:

$$p(x = \upsilon|y_k) = \frac{1}{\sqrt{2\pi{\sigma_k}^2}} e^{-\frac{(\upsilon-\mu_k)^2}{2{\sigma_k}^2}} \tag{2.30}$$

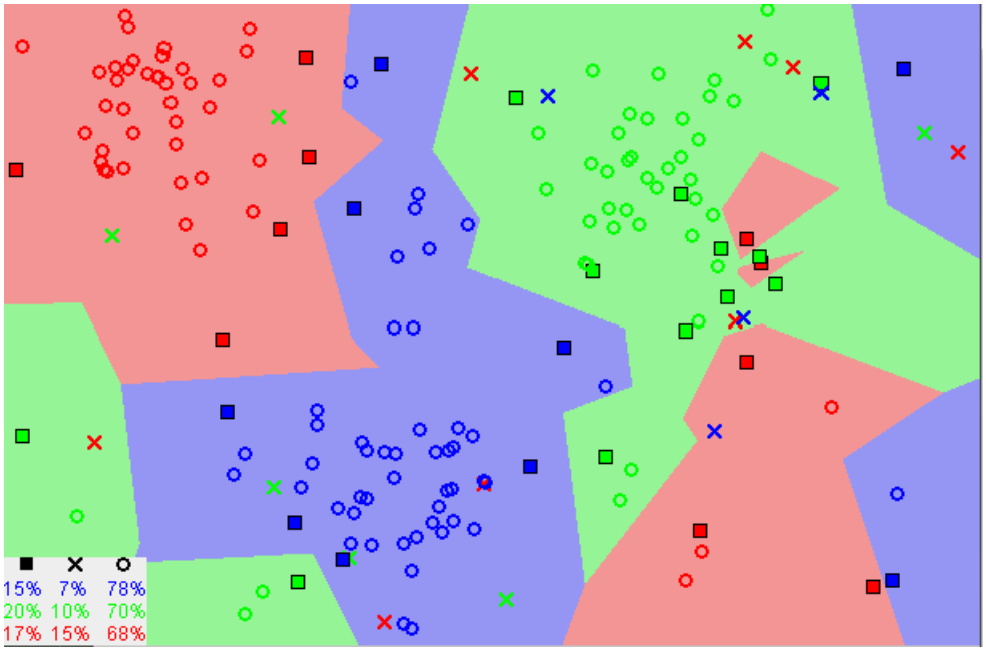
Εν κατακλείδι, η θεώρηση πως όλα τα χαρακτηριστικά είναι ανεξάρτητα μεταξύ τους, κάνει τους ταξινομητές naive bayes πολύ γρήγορους στον υπολογισμό και την εκπαίδευση, αλλά συνήθως δεν ανταποκρίνεται στην πραγματικότητα, καθιστώντας τον αλγόριθμο αυτό λιγότερο αποδοτικό σε σχέση με πιο πολύπλοκους αλγορίθμους. Φυσικά, το πόσο αποδοτικός μπορεί να είναι ένας αλγόριθμος, εξαρτάται από το υπό εξέταση πρόβλημα και μπορεί να διαφέρει από διεργασία σε διεργασία.

2.2.3.5 k-Nearest Neighbors (kNN)

Ο αλγόριθμος kNN βασίζεται στην θεώρηση ότι τα πράγματα που είναι παρόμοια, βρίσκονται σε κοντινή απόσταση. Έτσι για να καθορίσουμε την κλάση-ετικέτα ενός δείγματος,

κοιτάμε τις κλάσεις των k κοντινότερων γειτόνων του και παίρνουμε την κλάση που επικρατεί, δηλαδή αυτήν που έχει η πλειοψηφια των γειτόνων (classification problem). Αντίστοιχα, για να καθορίσουμε την πραγματική τιμή της ετικέτας ενός δείγματος, κοιτάμε τις τιμές των k κοντινότερων γειτόνων του και παίρνουμε τον μέσο όρο αυτών (regression problem).

Στην παρακάτω εικόνα παρατηρούμε ότι τις περισσότερες φορές, παρόμοια σημεία δεδομένων είναι κοντά το ένα με το άλλο [18]:



Σχήμα 2.12: Παρόμοιοι γείτονες

Η εγγύτητα δύο σημείων-δειγμάτων δίνεται από μια συνάρτηση απόστασης. Υπάρχουν πολλές διαφορετικές συναρτήσεις απόστασης που μπορούν να χρησιμοποιηθούν εδώ.
Η **ευκλείδια απόσταση** που αναφέρθηκε και παραπάνω, είναι η πιο συχνά χρησιμοποιούμενη:

$$d(p,q) \overset{def}{=} \sqrt{(p-q)^2} \quad \text{1-dimension} \tag{2.31}$$

$$d(p,q) \overset{def}{=} \sqrt{(q_1-p_1)^2+(q_2-p_2)^2} \quad \text{2-dimensions} \tag{2.32}$$

$$d(q,p) \overset{def}{=} \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\ldots+(q_i-p_i)^2+\ldots+(q_n-p_n)^2} \quad \text{N-dimensions} \tag{2.33}$$

Μια άλλη δημοφιλής επιλογή συνάρτησης απόστασης είναι η **ομοιότητα συνημιτόνου (cosine similarity)**:

$$CoS(p,q) \overset{def}{=} \cos(\angle(p,q)) = \frac{\sqrt{\sum_{i=1}^{n} p_i q_i}}{\sqrt{\sum_{i=1}^{n} {p_i}^2}\sqrt{\sum_{i=1}^{n} {q_i}^2}} \tag{2.34}$$

όπου υπολογίζεται η γωνία μεταξύ δύο διανυσμάτων p και q και στην συνέχεια το συνημίτονο αυτής. Αυτή η μέτρηση κυμαίνεται μεταξύ -1 και 1. Εάν το CoS είναι κοντά στο 1, αυτό δείχνει ότι η γωνία μεταξύ των δύο διανυσμάτων είναι κοντά στο μηδέν και επομένως είναι παρόμοια (γείτονες).

Άλλες διάσημες μετρικές απόστασης είναι η απόσταση Chebychev και η απόσταση Hamming.

Η ομοιότητα συνημιτόνων δύο δειγμάτων ισοδυναμεί με το εσωτερικό γινόμενό τους μετά από ομαλοποίηση (normalization). Υπό αυτές τις παραδοχές, ο kNN κάνει μια τοπικά γραμμική ταξινόμηση με διάνυσμα συντελεστών που ορίζεται ως εξής:

$$w_x = \sum_{(x',y')\in R_k(x)} y'x' \tag{2.35}$$

όπου $\boldsymbol{w_x}$ είναι οι συντελεστές για το δείγμα εισόδου x, $\boldsymbol{R_k(x)}$ είναι το σύνολο των k κοντινότερων γειτόνων, $\boldsymbol{y'}$ και $\boldsymbol{x'}$ είναι η κλάση και το διάνυσμα χαρακτηριστικών του γείτονα x' αντίστοιχα.

Η έξοδος του ταξινομητή, δηλαδή η πρόβλεψη της κλάσης του δείγματος x, αποκτάται από το εσωτερικό γινόμενο $w_x x$ (το οποίο στην περίπτωση των ομαλοποιημένω διανυσμάτων χαρακτηριστικών είναι ίσο με την συνημιτονοειδή ομοιότητα μεταξύ του w_x και του x) όταν αυτό ξεπερνά κάποιο κατώφλι (threshold).

Στην περίπτωση για παράδειγμα που έχουμε ένα δυαδικό πρόβλημα ταξινόμησης με κλάσεις (0,1), το w_x αναπαριστά την κεντροειδή των γειτόνων που ανήκουν στην κλάση 1 (αφού για $y' = 0 \implies y'x' = 0$) και το εσωτερικό γινόμενο $w_x x$ αναπαριστά το πόσο κοντά βρίσκεται η κεντροειδής αυτή, στο δείγμα x, δηλαδή το πόσο κοντά βρίσκεται το δείγμα x στους γείτονες της κλάσης 1. Αν αυτή η εγγύτητα είναι μεγαλύτερη από ένα ορισμένο κατώφλι, τότε το δείγμα x ταξινομείται και αυτό στην κλάση 1.

Τέλος, είναι σημαντικό να οριστεί η **συνάρτηση κόστους** πάνω στην οποία διεξάγεται η **μέθοδος βελτιστοποίησης** ώστε να παραχθούν οι βέλτιστοι συντελεστές w_x. Συνάρτηση κόστους:

$$L = -\sum_{(x',y')\in R_k(x)} y'x'w_x + \frac{1}{2}\|w_x\|^2 \tag{2.36}$$

$$min \quad L \tag{2.37}$$

Σύμφωνα με την (2.37), το **κριτήριο βελτιστοποίησης** είναι να ελαχιστοποιήσουμε την συνάρτηση κόστους, ως συνήθως, η οποία ορίζεται από την (2.36).

Στην συνάρτηση αυτή, ο πρώτος όρος $-\sum_{(x',y')\in R_k(x)} y'x'w_x$ αντικατοπτρίζει το σφάλμα του συνόλου εκμάθησης (training set), ενώ ο δεύτερος όρος $\frac{1}{2}\|w_x\|^2$ είναι μια ποινή πολυπλοκότητας, ακριβώς όπως στην λογική που συναντήσαμε στον SVM (2.13). Συνεπώς, η απώλεια του training-set σε ένα μόνο παράδειγμα εκμάθησης είναι:

$$l = \begin{cases} -y'x'w_x, & \text{when} \quad y' = 1 \quad \text{and} \quad x' \in kNN(x) \\ 0, & \text{Otherwise} \end{cases} \tag{2.38}$$

όπου το $\boldsymbol{-y'x'w_x}$ εκφράζει την απόσταση του γείτονα x' που έχει ετικέτα 1, από την κεντροειδή όλων των γειτόνων του x που έχουν ετικέτα 1. Αν δούμε πιο προσεκτικά αυτήν την σχέση, αυτό που προσπαθεί ο αλγόριθμος να κάνει, είναι να φέρει τους γείτονες που έχουν ίδια ετικέτα, δηλαδή ανήκουν στην ίδια κατηγορία-κλάση, πιο κοντά μεταξύ τους. Δηλαδή να μεγιστοποιήσει την ομοιότητά τους ή αλλιώς να ελαχιστοποιήσει την απόστασή τους ($\min -y'x'w_x$).

Για να βρούμε τελικά τις βέλτιστες τιμές των συντελεστών w_x, θα θέσουμε ως 0 την πρώτη παράγωγο του δεξιού όρου της ισότητας (2.27).

Η παραπάνω ανάλυση της συνάρτησης κόστους του αλγορίθμου kNN βασίζεται στο [19].

2.2.3.6 Decision Tree

Ο αλγόριθμος δέντρου απόφασης λειτουργεί έτσι ακριβώς όπως φανερώνει το όνομά του. Ένα δείγμα έρχεται σαν είσοδος, για το οποίο θέλουμε να προβλέψουμε την κλάση του. Στην συνέχεια ο αλγόριθμος θέτει μια σειρά από ερωτήματα. Μετά από κάθε απάντηση, ακολουθεί η επόμενη ερώτηση, ώσπου στο τέλος καταλήγουμε σε μία κλάση. Ο αλγόριθμος έχει την μορφή ενός δέντρου αποφάσεων, αποτελούμενο από κόμβους και ακμές. Οι κόμβοι διακρίνονται σε τρία είδη:

- Ρίζα (root node)
  Η ρίζα είναι ο πρωταρχικός κόμβος, ο οποίος δεν έχει καμία εισερχόμενη ακμή, ενώ μπορεί να έχει 0 ή περισσότερες εξερχόμενες ακμές.

- Εσωτερικοί κόμβοι (internal nodes)
  Οι εσωτερικοί κόμβοι έχουν ακριβώς μία εισερχόμενη ακμή, ενώ μπορεί να έχουν 2 ή περισσότερες εξερχόμενες ακμές.

- Φύλλα (leaf nodes)
  Τα φύλλα έχουν ακριβώς μία εισερχόμενη ακμή και καμία εξερχόμενη ακμή. Το κάθε φύλλο είναι επισημειωμένο με την κλάση-ετικέτα πρόβλεψης.

Υπάρχουν διαφορετικοί αλγόριθμοι που μπορούν να χρησιμοποιηθούν στα δέντρα απόφασης με βάση την μορφή των χαρακτηριστικών και κατ' επέκταση τα κριτήρια βελτιστοποίησης που χρησιμοποιούνται: **ID3** → επέκταση ID3 (extension of D3), **C4.5** → διάδοχος του ID3 (successor of ID3), **CART** → δέντρο ταξινόμησης και παλινδρόμησης (Classification And Regression Tree), **CHAID** → αυτόματη ανίχνευση αλληλεπίδρασης του Chi-square. Εκτελεί διαχωρισμούς πολλαπλών επιπέδων κατά τον υπολογισμό δέντρων ταξινόμησης(Chi-square automatic interaction detection). Performs multi-level splits when computing classification trees), **MARS** → προσαρμοσμένη παλινδρόμηση πολλαπλών παραλλαγών splines (multivariate adaptive regression splines).

Θα μελετήσουμε την λειτουργία του ID3 για προβλήματα ταξινόμησης. Ο ID3 είναι ένας άπληστος αλγόριθμος, ο οποίος κάνει μια αναζήτηση από πάνω προς τα κάτω, κάνοντας πάντα την επιλογή που φαίνεται να είναι καλύτερη εκείνη την στιγμή, χωρίς να λαμβάνει υπόψιν του την προηγούμενη πορεία.
Ο αλγόριθμος αυτός λειτουργεί ως εξής:

1. Ο αρχικός κόμβος- ρίζα, περιέχει το σύνολο S αποτελούμενο από όλα τα επισημασμένα δείγματα $S \stackrel{def}{=} \{(xi, yi)\}_{i=1}^{N}$.

2. Σε κάθε επανάληψη του αλγορίθμου, υπολογίζονται όλοι οι συνδυασμοί των παραμέτρων (j,t), όπου j είναι η θέση συγκεκριμένου χαρακτηριστικού από το διάνυσμα χαρακτηριστικών (j = 1, . . . , D) και t είναι το κατώφλι.

3. Για κάθε ένα συνδυασμό από τους παραπάνω, σπάμε το σύνολο S σε δύο υποσύνολα $S_- \stackrel{def}{=} \{(x,y)|(x,y) \in S, x^{(j)} < t\}$ και $S_+ \stackrel{def}{=} \{(x,y)|(x,y) \in S, x^{(j)} \geq t\}$

4. Από τους παραπάνω συνδυασμούς, επιλέγουμε τον συνδυασμό των παραμέτρων (j,t) που δίνει την καλύτερη διάσπαση του συνόλου. Ως καλύτερη διάσπαση ορίζουμε αυτήν που δίνει την ελάχιστη εγκάρσια εντροπία (cross entropy).

5. Ο αλγόριθμος συνεχίζει να επαναλαμβάνεται από το βήμα 2 για κάθε υποσύνολο, λαμβάνοντας υπόψη μόνο τα χαρακτηριστικά που δεν έχουν επιλεγεί πριν.

6. Ο αλγόριθμος σταματά όταν όλα τα δείγματα στα φύλλα, έχουν ταξινομηθεί σωστά ή όταν δεν υπάρχει άλλο χαρακτηριστικό με βάση το οποίο να γίνει μια νέα διάσπαση

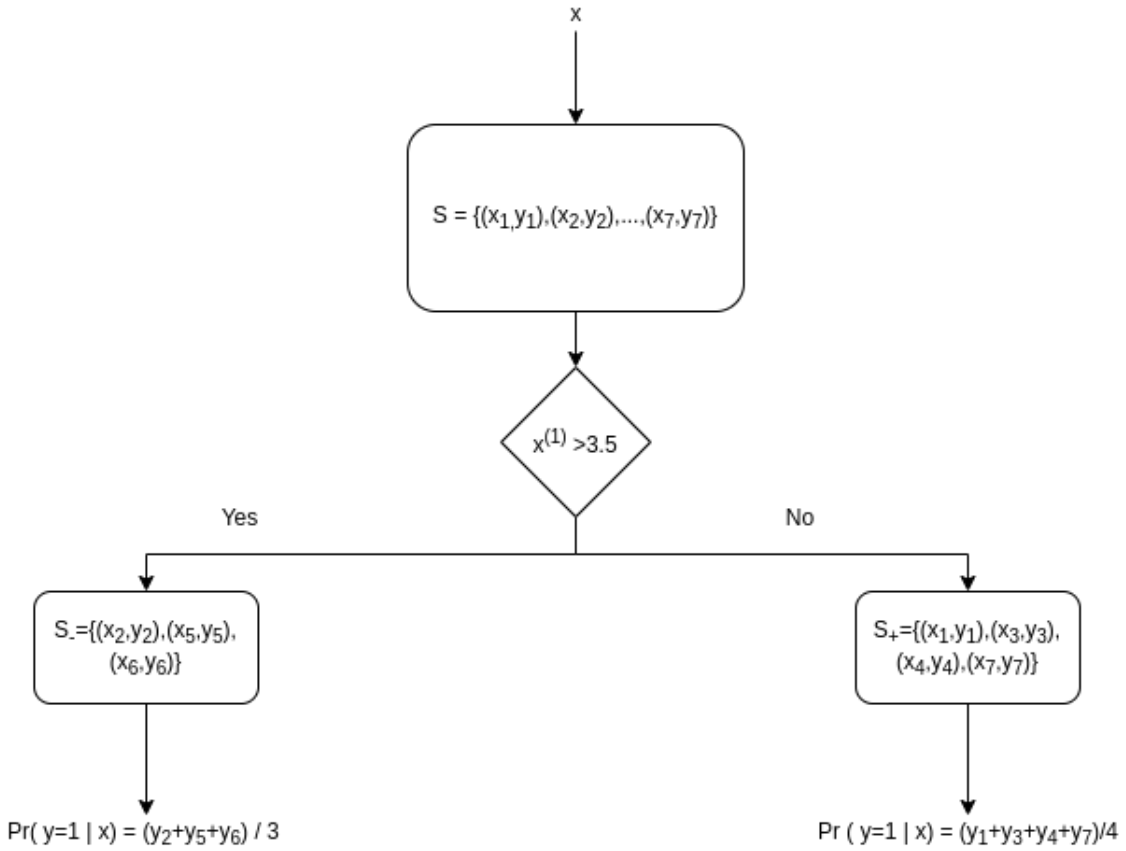
ή όταν το βάθος του δέντρου φτάσει μέχρι κάποια μέγιστη τιμή d ή όταν η εντροπία γίνει μικρότερη από κάποια τιμή ϵ.

Το μοντέλο το οποίο εφαρμόζεται για κάθε σύνολο S, δηλαδή σε κάθε φύλλο του δέντρου το οποίο κάνει μια πρόβλεψη, μπορεί να οριστεί ως εξής:

$$f_{ID3}^{S} = \frac{1}{|S|} \sum_{(x,y)\in S} y \tag{2.39}$$

Η πρόβλεψη που δίνεται απ' το παραπάνω μοντέλο θα είναι η ίδια για κάθε είσοδο x. Η πρόβλεψη αυτή δίνει μία πιθανότητα του κατά πόσο το δείγμα εισόδου x ανήκει στην κλάση 1 Pr(y=1|x). Η πιθανότητα υπολογίζεται από τον μέσο όρο των δειγμάτων του κόμβου που ανήκουν στην κλάση 1.

Παρακάτω φαίνεται ένα παράδειγμα δέντρου απόφασης ID3 με 7 δείγματα εκπαίδευσης:



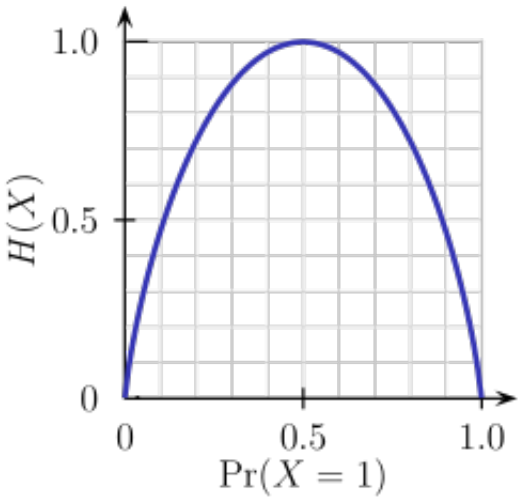
Σχήμα 2.13: Παράδειγμα δέντρου απόφασης

Το κριτήριο βελτιστοποίησης είναι η ελαχιστοποίηση της εντροπίας όπως αναφέρθηκε στο 4. Η εγκάρσια εντροπία έχει οριστεί ξανά στο κεφάλαιο 2.2.3.2, στην σχέση (2.8) για τον αλγόριθμο logistic regression.
Η εντροπία ενός συνόλου δειγμάτων S δίνεται από την παρακάτω σχέση:

$$min \quad H(S) = -f_{ID3}^{S} \ln f_{ID3}^{S} - (1 - f_{ID3}^{S}) \ln (1 - f_{ID3}^{S})$$

όπου θέλουμε να μεγιστοποιήσουμε την λογαριθμική πιθανότητα (log-likelihood) τα δείγματα του συνόλου S του εκάστοτε κόμβου να έχουν ταξινομηθεί σωστά. Στην ουσία θέλουμε να μεγιστοποιήσουμε την πιθανότητα τα δείγματα που βρίσκονται σε αυτό το σύνολο-κόμβο να ανήκουν στην ίδια κλάση ή αλλιώς να ελαχιστοποιήσουμε την εγκάρσια εντροπία (αρνητική λογαριθμική πιθανότητα).

Για να κατανοήσουμε λίγο καλύτερα το ρόλο της εντροπίας εδώ, η εντροπία είναι ένα μέτρο της τυχαιότητας των πληροφοριών που υποβάλλονται σε επεξεργασία. Όσο υψηλότερη είναι η εντροπία, τόσο πιο δύσκολο είναι να εξαχθούν συμπεράσματα από αυτές τις πληροφορίες [20]:



Σχήμα 2.14: Εντροπία

Από το παραπάνω γράφημα, είναι αρκετά προφανές ότι η εντροπία H(X) είναι μηδέν όταν η πιθανότητα είναι είτε 0 είτε 1, δηλαδή στην περίπτωσή μας, όταν όλα τα δείγματα του συνόλου S ανήκουν στην κλάση 0 ή στην κλάση 1 ($f^S_{ID3} = 0$ ή $f^S_{ID3} = 1$) . Αντίθετα, η εντροπία είναι μέγιστη όταν η πιθανότητα είναι 0,5 επειδή προβάλλει τέλεια τυχαιότητα στα δεδομένα και δεν υπάρχει πιθανότητα να προσδιοριστεί τέλεια το αποτέλεσμα, δηλαδή στην περίπτωσή μας, όταν τα μισά δείγματα του συνόλου S ανήκουν στην κλάση 0 και τα άλλα μισά στην κλάση 1 ($f^S_{ID3} = 0.5$).

Χωρίζοντας το σύνολο S σε δύο υποσύνολα S_- και S_+ με βάση ένα χαρακτηριστικό j και ένα κατώφλι t, η εντροπία που προκύπτει από την διάσπαση αυτή ορίζεται ως ένα σταθμισμένο άθροισμα από τις δύο εντροπίες:

$$min \quad H(S_-, S_+) = \frac{|S_-|}{|S|} H(S_-) + \frac{|S_+|}{|S|} H(S_+) \tag{2.40}$$

Ουσιαστικά, σε κάθε βήμα του αλγορίθμου ID3, χρησιμοποιείται η σχέση (2.40), με σκοπό να βρεθεί η διάσπαση συνόλου που δίνει την ελάχιστη εντροπία, δηλαδή να βρεθεί ο βέλτιστος συνδυασμός (j,t) με βάση τον οποίο γίνεται η διάσπαση.

Πέρα από την εντροπία, μπορούν να χρησιμοποιηθούν και άλλα κριτήρια, με βάση τα οποία επιλέγεται το καλύτερο χαρακτηριστικό με το καλύτερο κατώφλι (j,t) για να γίνει η διάσπαση. Άλλα τέτοια κριτήρια είναι: κέρδος πληροφοριών **(Information gain)**, δείκτης Gini **(Gini index)**, αναλογία κέρδους **(Gain Ratio)**, μείωση της διακύμανσης **(Reduction in Variance)** και τετράγωνο-Chi **(Chi-Square)**.

Ensemble Learning

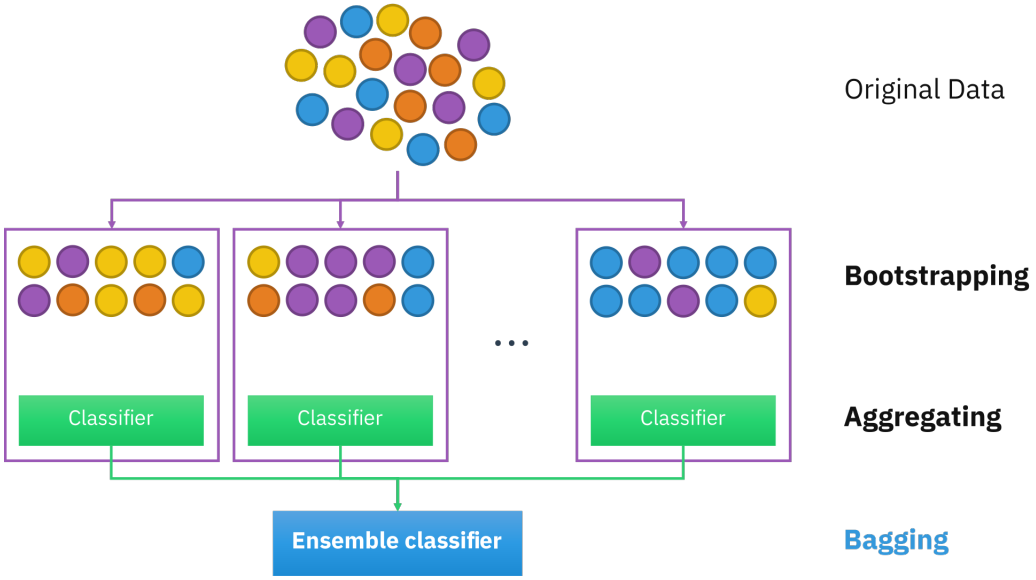
Οι αλγόριθμοι που ακολουθούν στα επόμενα κεφάλαια ανήκουν στην κατηγορία **ensemble learning (συλλογική μάθηση)**. Ο στόχος των αλγορίθμων αυτής της κατηγορίας είναι να αναπτύξουν πολλά μοντέλα, που ίσως να μην είναι τόσο αποτελεσματικά και συνδυάζοντάς τα να δημιουργήσουν ένα ισχυρό μετα-μοντέλο. Τα μοντέλα χαμηλής ακρίβειας (low accuracy), ονομάζονται **αδύναμοι μαθητές (weak learners)** και συνήθως είναι πιο γρήγορα στην εκμάθηση και την πρόβλεψη, σε σχέση με πιο ισχυρά και πιο πολύπλοκα

μοντέλα. Ο τρόπος με τον οποίο αποκτάται μια πρόβλεψη από τέτοιου είδους μοντέλα, είναι συνδυάζοντας τις προβλέψεις από κάθε αδύναμο μοντέλο, χρησιμοποιώντας ένα είδος σταθμισμένης ψηφοφορίας.

Στην συλλογική μάθηση υπάρχουν δύο είδη αλγορίθμων, οι **bagging (bootstrap aggregating)** και οι **boosting**.

Στο **bagging** δημιουργούνται αντίγραφα του αυθεντικού συνόλου εκπαίδευσης (training set) καθένα απ' τα οποία χρησιμοποιείται από έναν αδύναμο μαθητή με σκοπό να δημιουργηθεί ένα αδύναμο μοντέλο. Τα αντίγραφα αυτά διαφέρουν μεταξύ τους, μπορεί να είναι μεγέθους διάφορου του αρχικού συνόλου, καθώς επίσης μπορεί να περιέχουν και κάποια δείγματα του αρχικού συνόλου, πολλαπλές φορές. Συγκεκριμένα υπολογίζεται πως αν τα νέα σύνολα είναι ίσων διαστάσεων με το αυθεντικό (n=n') τότε το 63,2% πρόκειται να είναι μοναδικά δείγματα και τα υπόλοιπα διπλότυπα. Τέλος συνδυάζονται τα αδύναμα μοντέλα για να προκύψει το συλλογικό μετα-μοντέλο.

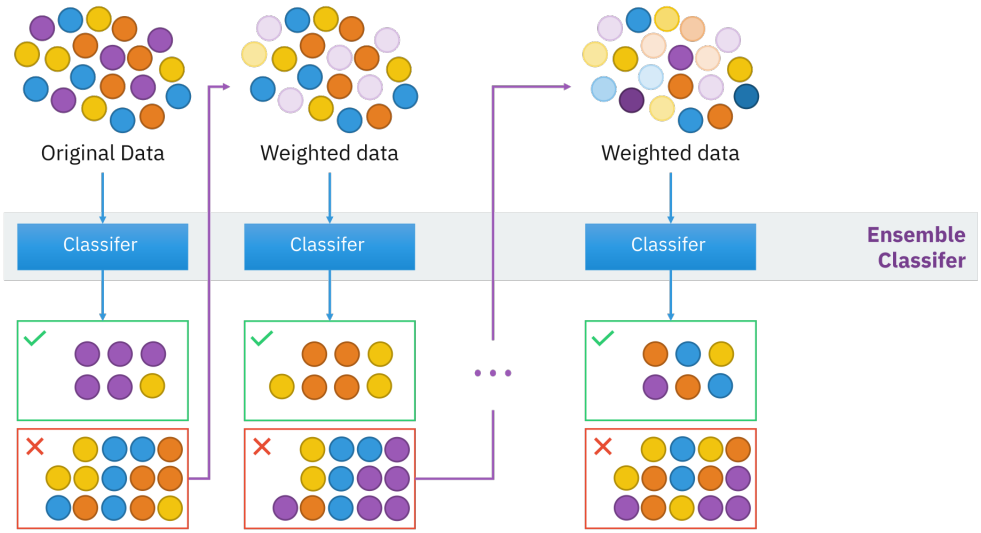
Παρακάτω φαίνεται η μέθοδος bagging [21] :



Σχήμα 2.15: Συλλογική μάθηση με bagging

Στο **boosting**, υπάρχει μια επαναληπτική διαδικασία εκμάθησης αδύναμων ταξινομητών, οι οποίοι προστίθενται δημιουργώντας έναν τελικό ισχυρό ταξινομητή. Μετά την προσθήκη ενός νέου αδύναμου ταξινομητή, τα δεδομένα αναπροσαρμόζουν τα βάρη τους (re-weighting) με τέτοιο τρόπο ώστε τα λάθος ταξινομομημένα δεδομένα να έχουν μεγαλύτερα βάρη από τα σωστά. Έτσι, οι μελλοντικοί αδύναμοι μαθητές εστιάζουν περισσότερο στα παραδείγματα που οι προηγούμενοι αδύναμοι εκπαιδευόμενοι ταξινόμησαν εσφαλμένα.

Παρακάτω φαίνεται η μέθοδος boosting [22]:



Σχήμα 2.16: Συλλογική μάθηση με boosting

2.2.3.7 Random Forest

Ο αλγόριθμος αυτός χρησιμοποιεί bagging, έχοντας σαν αδύναμους μαθητές, δέντρα απόφασης τα οποία είδαμε στο κεφάλαιο 2.2.3.5. Πιο συγκεκριμένα, από ένα αρχικό σύνολο δεδομένων εκπαίδευσης (training set) S, δημιουργούνται B σύνολα με αντικατάσταση (replacement) (S_b,b=1,...,B). Αντικατάσταση σημαίνει ότι διαλέγουμε τυχαία δείγματα από το σύνολο δεδομένων S και τα αντιγράφουμε στο σύνολο δεδομένων S_b, διατηρώντας τα αυθεντικά δείγματα στο αρχικό σύνολο. Ύστερα δημιουργούνται B δέντρα απόφασης, καθένα απ' τα οποία χρησιμοποιεί ως σύνολο εκπαίδευσης (training set) ένα εκ των S_b. Η τελική πρόβλεψη για ένα νέο δείγμα εισόδου x λαμβάνεται από τον μέσο όρο των προβλέψεων των δέντρων, στην περίπτωση παλινδρόμησης (regression), ή από την πλειοψηφία στην περίπτωση ταξινόμησης (classification). Στην περίπτωση παλινδρόμησης (regression), έχουμε:

$$y = \frac{1}{B}\sum_{b=1}^{N} f_b(x) \tag{2.41}$$

όπου $\boldsymbol{y}$ είναι η πρόβλεψη για το δείγμα εισόδου $\mathbf{x}$ και $\boldsymbol{f_b(x)}$ είναι η πρόβλεψη κάθε δέντρου b.

Μια εκτίμηση της αβεβαιότητας της πρόβλεψης μπορεί να οριστεί ως η τυπική απόκλιση των προβλέψεων από όλα τα μεμονωμένα δέντρα παλινδρόμησης στο δείγμα εισόδου x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - y')^2}{B-1}} \tag{2.42}$$

Αξίζει να σημειώσουμε εδώ πως τα δέντρα απόφασης δεν εξετάζουν όλα τα χαρακτηριστικά σε κάθε διάσπαση, όπως έχουμε δει παραπάνω, αλλά ένα τυχαίο υποσύνολο αυτών. Αυτό συμβαίνει γιατί αν κάποιο χαρακτηριστικό είναι πιο ισχυρό-βοηθητικό στο να γίνεται μια πρόβλεψη, τότε αυτό το χαρακτηριστικό θα επιλεχθεί από πολλά διαφορετικά δέντρα ως κριτήριο διάσπασης του συνόλου δεδομένων. Αυτό θα έχει ως αποτέλεσμα,

πολλά κακά μοντέλα (δέντρα), να τείνουν να συμφωνήσουν σε κάποιες λανθασμένες προβλέψεις και έτσι να επηρεάσουν την πλειοψηφία δίνοντας λάθος τελικό αποτέλεσμα. Το φαινόμενο αυτό ονομάζεται συσχέτιση των δέντρων (correlation of the trees) και μπορεί να αποφευχθεί αν η διάσπαση σε κόμβους γίνεται με βάση ένα τυχαίο υποσύνολο των χαρακτηριστικών κάθε φορά.

Από τα παραπάνω καταλαβαίνουμε ότι το όνομα του αλγορίθμου (τυχαίο δάσος), προκύπτει τόσο από την τυχαία δειγματοληψία συνόλου δεδομένων εκπαίδευσης κατά την κατασκευή δέντρων, όσο και από τα τυχαία υποσύνολα χαρακτηριστικών που λαμβάνονται υπόψη κατά τον διαχωρισμό κόμβων.

Οι παράμετροι που μπορούν να τροποποιηθούν στον αλγόριθμο, αλλάζοντας τα αποτελέσματα είναι ο αριθμός των δέντρων B και το μέγεθος των τυχαίων υποσυνόλων των χαρακτηριστικών. Αυτοί οι παράμετροι τίθενται προς διερεύνηση και πειραματισμό, με σκοπό να βρεθούν οι καλύτεροι ανάλογα με την περίπτωση (hyperparameter tuning 2.2.9).

Ο αλγόριθμος τυχαίου δάσους (random forest) είναι ο πιο διάσημος από τους αλγορίθμους συλλογικής μάθησης (ensemble learning). Ο λόγος για τον οποίο είναι τόσο αποτελεσματικός είναι η χρήση πολλών διαφορετικών εκδοχών του συνόλου δεδομένων (training set), οι οποίες έχουν προκύψει με αντικατάσταση. Η λειτουργία αυτή συνεισφέρει στο να εξαλειφθούν φαινόμενα όπως ο θόρυβος στα δεδομένα, οι ακραίες τιμές (μη συνηθισμένες) και όμοια παραδείγματα που εμφανίζονται σε υπερβολικά μεγάλο ποσοστό ή σε υπερβολικά μικρό. Αυτή η διαδικασία εκκίνησης οδηγεί σε καλύτερη απόδοση του μοντέλου επειδή μειώνει τη διακύμανση του μοντέλου (variance) , χωρίς να αυξάνει την προκατάληψη (bias) ή αλλιώς χωρίς να προκαλεί underfitting (2.2.6).

2.2.3.8 Extra Trees

Ο αλγόριθμος **extra trees** ή αλλιώς **extremely randomized trees classifier**, είναι παρόμοιος με τον random forest, αλλάζοντας μόνο στον τρόπο που κατασκευάζονται τα δέντρα απόφασης του δάσους. Πιο συγκεκριμένα οι διαφορές είναι δύο:

1. Στον αλγόριθμο extra trees, κάθε δέντρο εκπαιδεύεται χρησιμοποιώντας όλο το σύνολο εκπαίδευσης (training set) και όχι ένα σύνολο που έχει προκύψει με αντικατάσταση όπως είδαμε παραπάνω.

2. Σε κάθε βήμα διάσπασης, χρησιμοποιείται ναι μεν ένα τυχαίο υποσύνολο των χαρακτηριστικών όπως είδαμε και στο random forest, αλλά το κατώφλι με βάση το οποίο θα γίνει η διάσπαση, επιλέγεται τυχαία. Συγκεκριμένα αυτή η τιμή επιλέγεται από μια ομοιόμορφη κατανομή εντός του εμπειρικού εύρους του χαρακτηριστικού. Στον αλγόριθμο random forest αντίθετα υπολιζόταν το τοπικά βέλτιστο σημείο διακοπής (κατώφλι) για κάθε υπό εξέταση χαρακτηριστικό. Στη συνέχεια, από όλες τις τυχαία παραγόμενες διασπάσεις, επιλέγεται η διάσπαση που αποδίδει την υψηλότερη βαθμολογία για να χωρίσει τον κόμβο (όπως ακριβώς και στο decision tree και το random forest).

Αυτές οι διαφορές ενθαρρύνουν τη μείωση τόσο της προκατάληψης (bias) όσο και της διακύμανσης (variance). Από τη μία πλευρά, η χρήση ολόκληρου του αρχικού συνόλου δειγμάτων αντί του αντιγράφου με αντικατάσταση θα μειώσει την προκατάληψη. Από την άλλη πλευρά, επιλέγοντας τυχαία το σημείο διαχωρισμού κάθε κόμβου θα μειωθεί η διακύμανση. Τα Extra Trees μπορούν κάποιες φορές να γενικεύουν καλύτερα από τα Random Forests, αλλά είναι δύσκολο να μαντέψουμε πότε συμβαίνει αυτό χωρίς να δοκιμάσουμε και τα δύο πρώτα, καθώς επίσης και να πειραματιστούμε με τις παραμέτρους: αριθμός δέντρων, μέγεθος υποσυνόλων χαρακτηριστικών και ελάχιστα δείγματα ανά σύνολο.

Όσον αφορά το υπολογιστικό κόστος, και συνεπώς τον χρόνο εκτέλεσης, ο αλγόριθμος Extra Trees είναι πιο γρήγορος. Αυτός ο αλγόριθμος εξοικονομεί χρόνο επειδή η όλη

διαδικασία είναι η ίδια, αλλά επιλέγει τυχαία το σημείο διαχωρισμού και δεν υπολογίζει το βέλτιστο. Παρ' όλα αυτά , ο αλγόριθμος αυτός παράγει συνήθως ένα μοντέλο πιο μεγάλο σε σχέση με τον Random Forest ο οποίος παράγει ένα πιο συμπαγές.

2.2.3.9 Gradient Boosting

Ο αλγόριθμος αυτός χρησιμοποιεί boosting και συνήθως δέντρα απόφασης όπως οι παραπάνω αλγόριθμοι. Τα δέντρα απόφασης λειτουργούν και εδώ ως αδύναμοι μαθητές, οι οποίοι συνδυάζονται για να δημιουργήσουν έναν αλγόριθμο που ονομάζεται **δέντρα ενισχυμένης κλίσης (gradient boosted trees)**. Μπορεί να εφαρμοστεί τόσο σε προβλήματα παλινδρόμησης όσο και ταξινόμησης.

Στην περίπτωση της παλινδρόμησης το αρχικό μοντέλο-δέντρο ορίζεται ως εξής:

$$f = f_0(x) \stackrel{def}{=} \frac{1}{N}\sum_{i=1}^{N} y_i \tag{2.43}$$

όπου υπολογίζεται ο μέσος όρος των ετικετών των δειγμάτων.

Στην συνέχεια παραποιούνται οι ετικέτες για κάθε δείγμα $x_i, i = 1, ...N$ του συνόλου εκπαίδευσης ως εξής:

$$\hat{y_i} \leftarrow y_i - f(x_i) \tag{2.44}$$

όπου το $\hat{y_i}$ ονομάζεται **residual (υπόλοιπο)** και αντικατοπτρίζει το πόσο καλά ή άσχημα έχει ταξινομηθεί το κάθε δείγμα του συνόλου εκπαίδευσης, αφού είναι η διαφορά μεταξύ της πραγματικής ετικέτας και της προβλεπόμενης ετικέτας του κάθε δείγματος. Με λίγα λόγια, είναι το σφάλμα ταξινόμησης/πρόβλεψης κάθε δείγματος.

Από τις νέες αυτές ετικέτες δημιουργείται ένα καινούριο σύνολο εκπαίδευσης (training set), το οποίο και θα χρησιμοποιηθεί από το επόμενο μοντέλο-δέντρο για να προσπαθήσει να διορθώσει τα σφάλματα του προηγούμενου. Το νέο μοντέλο f_1 θα συνδυαστεί με το αρχικό f_0 δίνοντας το τελικό **μοντέλο ενίσχυσης (boosting model)** : $f = f_0 + \alpha f_1$, όπου α είναι το ποσοστό εκμάθησης (learning rate) που έχουμε δει και σε προηγούμενα κεφάλαια. Στην συνέχεια η διαδικασία επαναλαμβάνεται, φτιάχνοντας ένα νέο σύνολο δεδομένων από τις νέες ετικέτες (2.44) και δημιουργώντας ένα νέο μοντέλο f_2 το οποίο καλείται να διορθώσει τα λάθη του f_1. Το μοντέλο ενίσχυσης αναδιαμορφώνεται αντιστοίχως: $f = f_0 + \alpha f_1 + \alpha f_2$. Η επανάληψη θα σταματήσει όταν φτάσουμε σε ένα μέγιστο αριθμό συνδυαζόμενων δέντρων.

Το όνομα του αλγορίθμου (gradient boosting), δηλώνει κάποιου είδους σχέση με τον υπολογισμό μιας παραγώγου (gradient) καθώς επίσης και με τον αλγόριθμο απότομης καθόδου (gradient descent), ο οποίος χρησιμοποιήθηκε στο κεφάλαιο 2.2.3.1 (linear regression). Πιο συγκεκριμένα, είδαμε ότι ο αλγόριθμος απότομης καθόδου μας βοηθούσε να καθορίσουμε την κατεύθυνση προς την οποία πρέπει να μετακινήσουμε τις τιμές των παραμέτρων (βαρών) του μοντέλου, με σκοπό να μειώσουμε το μέσο τετραγωνικό σφάλμα (mean squared error - MSE) στο ελάχιστο. Η κατεύθυνση αυτή υπολογιζόταν από την αρνητική παράγωγο του σφάλματος. Το ποσοστό εκμάθησης (learning rate) μας έδινε αντίστοιχα το μέγεθος της μεταβολής των παραμέτρων αυτών (μικρό ή μεγάλο βήμα). Ο υπολογισμός της κατεύθυνσης (gradient descent) εφαρμοζόταν σε κάθε βήμα-επανάληψη του αλγορίθμου, επαναπροσδιορίζοντάς τον. Στην περίπτωση του gradient boosting, δεν έχουμε άμεση χρήση παραγώγου, αλλά η σχέση (2.44), προσομοιάζει αυτήν την λειτουργία μέσω των υπολοίπων (residuals), οι οποίοι μας δείχνουν πώς πρέπει να προσαρμοστεί το μοντέλο έτσι ώστε να μειωθεί το σφάλμα (το υπόλοιπο).

Παρακάτω θα μελετήσουμε την εφαρμογή του αλγορίθμου αυτού σε προβλήματα ταξινόμησης και συγκεκριμένα δυαδικής ταξινόμησης (πρόβλεψη ανάμεσα σε 2 κλάσεις). Ενώ στην περίπτωση της παλινδρόμησης, ο αλγόριθμος μοιάζει με την γραμμική παλινδρόμηση (linear regression), στην περίπτωση της ταξινόμησης μοιάζει με την λογιστική παλινδρόμηση (logistic regression).

Έστω ότι έχουμε M δέντρα απόφασης-παλινδρόμησης. Όπως και στην λογιστική παλινδρόμηση, το τελικό μοντέλο ενίσχυσης χρησιμοποιεί την σιγμοειδή συνάρτηση (sigmoid function):

$$Pr(y = 1|x, f) \stackrel{def}{=} \frac{1}{1 + e^{-f(x)}} \tag{2.45}$$

όπου το **Pr** μας δίνει την πιθανότητα ένα δείγμα x να ταξινομηθεί ως θετικό (με ετικέτα 1), $\boldsymbol{f(x) = \sum_{m=1}^{M} f_m(x)}$ είναι το άθροισμα των αποτελεσμάτων που δίνει κάθε δέντρο f_m και η σιγμοειδής συνάρτηση $\frac{1}{1+e^{-f(x)}}$, μετατρέπει το άθροισμα αυτό, των πραγματικών τιμών, σε τιμή εύρους [0,1], δίνοντας έτσι μια πιθανότητα.

Το κριτήριο βελτιστοποίησης εδώ είναι η μεγιστοποίηση της πιθανότητας (likelihood) όλα τα δεδομένα να έχουν ταξινομηθεί σωστά ή αλλιώς η μεγιστοποίηση της λογαριθμικής πιθανότητας (log-likelihood) που μας επιτρέπει να πάρουμε άθροισμα έναντι γινομένου, για πιο εύκολους υπολογισμούς:

$$max \quad L_f = \sum_{i=1}^{N} (y_i \log (Pr(y_i = 1|x, f)) + (1 - y_i) \log (1 - Pr(y_i = 1|x, f))) \tag{2.46}$$

Ο αλγόριθμος ξεκινά και πάλι αρχικοποιώντας ένα μοντέλο-δέντρο $f_0 = \frac{p}{1-p}$, όπου $p = \frac{1}{N} \sum_{i=1}^{N} y_i$ είναι η πρόβλεψη του δέντρου απόφασης όπως έχουμε δει σε προηγούμενα κεφάλαια. Στην συνέχεια για κάθε νέο δέντρο f_m το οποίο προστίθεται στο ενισχυτικό μοντέλο, υπολογίζεται η παράγωγος της συνάρτησης απώλειας (loss function) L_f με σκοπό να οριστούν οι νέες τιμές y_i για τα νέα σύνολα εκπαίδευσης. Η παράγωγος υπολογίζεται ως εξής:

$$g_i = \frac{\partial L_f}{\partial f} \tag{2.47}$$

Η παράγωγος αυτή υπολογίζεται για κάθε i (i=1,...N), δηλαδή για κάθε δείγμα.

Στην συνέχεια αντικαθιστούμε τις τιμές των ετικετών y_i με τα g_i και το νέο σύνολο εκπαίδευσης που δημιουργείται, το χρησιμοποιούμε για το επόμενο δέντρο απόφασης.

Για να συνδυάσουμε όλα τα δέντρα μαζί, θα πρέπει για καθένα από αυτά να υπολογίζουμε την παράμετρο ρ_m ως εξής:

$$\rho_m = \underset{\rho}{\operatorname{argmax}} L_{f+\rho f_m} \tag{2.48}$$

όπου το $\boldsymbol{\rho_m}$ παίρνει την τιμή για την οποία μεγιστοποιείται το log-likelihood $\boldsymbol{L_{f+\rho f_m}}$ του συνολικού ενισχυτικού μοντέλου $\boldsymbol{f + \rho f_m}$. Στην ουσία το ρ_m λειτουργεί ως βάρος που καθορίζει το πόσο μεγάλη επίδραση θα έχει το μοντέλο-δέντρο m στο συνολικό μοντέλο-δάσος.

Μετά την προσθήκη του μοντέλου m, το συνολικό μοντέλο (ensemble model) ενημερώνεται ως εξής:

$$f \leftarrow f + \alpha \rho_m f_m \tag{2.49}$$

όπου έχει προστεθεί το νέο μοντέλο και το **α** που είναι το ποσοστό εκμάθησης.

Οι παράμετροι που επιδέχονται πειραματισμό και αναζήτηση σε αυτόν τον αλγόριθμο είναι ο αριθμός των δέντρων M, το ποσοστό εκμάθησης α και το βάθος των δέντρων. Οι παράμετροι αυτοί μπορούν να επηρεάσουν την απόδοση του μοντέλου και για αυτό πρέπει να διερευνώνται οι τιμές τους (hyperparameter tuning 2.2.9). Το βάθος των δέντρων, πέρα από την απόδοση, επηρεάζει και την ταχύτητα εκπαίδευσης και πρόβλεψη. Όσο πιο μεγάλο είναι το βάθος, τόσο πιο αργός είναι ο αλγόριθμος.

Μέσω αυτού του αλγορίθμου μπορεί να εντοπιστεί η διαφορά μεταξύ του bagging και του boosting. Το boosting μειώνει την προκατάληψη (bias or underfitting), αφού προσπαθεί να διορθώσει σφάλματα που αφορούν την λάθος ταξινόμηση ολόκληρου του συνόλου εκπαίδευσης, με κίνδυνο όμως να αυξήσει την διακύμανση (variance) και να οδηγήσει σε overfitting (2.2.6). Αντίθετα, το bagging μειώνει την διακύμανση του μοντέλου με το να χρησιμοποιεί την τυχαιότητα σε ό,τι αφορά τα δεδομένα εκπαίδευσης. Παρ᾽ όλα αυτά το overfitting μπορεί να αποφευχθεί και στην περίπτωση του boosting βρίσκοντας τις κατάλληλες παραμέτρους που αναφέρθηκαν στην προηγούμενη παράγραφο.

Ο αλγόριθμος gradient boosting, συνήθως τα πηγαίνει καλύτερα από τον random forest, όμως χρειάζεται περισσότερο χρόνο στην εκπαίδευση καθώς τα δέντρα είναι ακολουθιακά (πρέπει να έχει τελειώσει πρώτα η εκπαίδευση του ενός για να ξεκινήσει η εκπαίδευση του επόμενου).

2.2.3.10 XGBoost

Ο αλγόριθμος XGBoost (eXtreme Gradient Boosting), είναι ένα κλιμακούμενο σύστημα ενίσχυσης δέντρων (scalable tree boosting system) που χρησιμοποιεί gradient boosting και δημιουργήθηκε από τους Tianqi Chen και Carlos Guestrin, ως ερευνητικό έργο στο Πανεπιστήμιο της Ουάσιγκτον.

Οι XGBoost και Gradient Boosting Machines (GBMs) είναι και οι δύο συλλογικές μέθοδοι δέντρων που εφαρμόζουν την αρχή της ενίσχυσης των ασθενών μαθητών (CARTs-classification and regression trees) χρησιμοποιώντας την αρχιτεκτονική της απότομης καθόδου (gradient descent). Ωστόσο, το XGBoost βελτιώνει το βασικό πλαίσιο GBM μέσω βελτιστοποίησης συστήματος και αλγοριθμικών βελτιώσεων.

Βελτιστοποιήσεις Συστήματος:

- **Παραλληλισμός**
  Η εκμάθηση δέντρων χρειάζεται δεδομένα σε ταξινομημένη μορφή σύμφωνα με τις τιμές χαρακτηριστικών, ώστε ο αλγόριθμος να επισκεφτεί τα δεδομένα για να συγκεντρώσει τα στατιστικά gradient και να απαριθμήσει όλους τους δυνατούς διαχωρισμούς για τα συνεχή χαρακτηριστικά. Για τη μείωση του κόστους ταξινόμησης, τα δεδομένα χωρίζονται σε συμπιεσμένα μπλοκ (κάθε στήλη με αντίστοιχη τιμή χαρακτηριστικού). Το XGBoost ταξινομεί κάθε μπλοκ παράλληλα χρησιμοποιώντας όλους τους διαθέσιμους πυρήνες / νήματα (cores/threads) της CPU. Αυτή η βελτιστοποίηση είναι πολύτιμη καθώς ένας μεγάλος αριθμός κόμβων δημιουργείται συχνά σε ένα δέντρο. Συνοπτικά, το XGBoost παραλληλίζει τη διαδοχική διαδικασία δημιουργίας δέντρων.

- **Υπολογισμός εκτός πυρήνα**
  Επιτρέπει τον υπολογισμό εκτός πυρήνα (out-of-core computing) για πολύ μεγάλα σύνολα δεδομένων που δεν χωρούν στη μνήμη και για βελτιστοποίηση του διαθέσιμου χώρου στον δίσκο.

- **Cache Aware**
  Με βελτιστοποίηση που έχει επίγνωση της cache, αποθηκεύονται στατιστικά του

gradient (κατεύθυνση και τιμή) για κάθε διαχωριστικό κόμβο σε ένα εσωτερικό buffer κάθε νήματος και εκτελείται συσσώρευση με τρόπο μίνι-παρτίδας. Αυτό βοηθά στη μείωση του χρόνου επιβάρυνσης των άμεσων εργασιών ανάγνωσης / εγγραφής και επίσης στην αποφυγή απώλειας προσωρινής μνήμης. Η επίγνωση της προσωρινής μνήμης επιτυγχάνεται επιλέγοντας το βέλτιστο μέγεθος του μπλοκ (γενικά 2^{16}).

Αλγοριθμικές βελτιώσεις:

- **Column (feature) subsampling**
  Το XGBoost χρησιμοποιεί ένα μέρος των features κάθε φορά, όπως ακριβώς συμβαίνει και στο Random Forest. Η χρήση υπο-δειγματοληψίας στηλών αποτρέπει το overfitting ακόμη περισσότερο από την παραδοσιακή υπο-δειγματοληψία σειράς (η οποία υποστηρίζεται επίσης). Η χρήση υπο-δειγμάτων στηλών επιταχύνει επίσης τους υπολογισμούς του παράλληλου αλγορίθμου.

- **Loss Function**
  Ενώ το Gradient Boosting ακολουθεί αρνητικά gradients για τη βελτιστοποίηση της συνάρτησης απώλειας, το XGBoost χρησιμοποιεί την επέκταση Taylor για να υπολογίσει την τιμή της συνάρτησης απώλειας για διαφορετικούς μαθητές βάσης.

- **Κλάδεμα δέντρων**
  Το κλάδεμα (pruning) είναι μια τεχνική μηχανικής μάθησης για τη μείωση του μεγέθους των δέντρων παλινδρόμησης αντικαθιστώντας κόμβους που δεν συμβάλλουν στη βελτίωση της ταξινόμησης στα φύλλα. Η ιδέα της περικοπής ενός δέντρου παλινδρόμησης είναι να αποφευχθεί το overfitting των δεδομένων εκπαίδευσης. Η πιο αποτελεσματική μέθοδος για το κλάδεμα είναι η πολυπλοκότητα κόστους (Cost Complexity) ή το κλάδεμα της πιο αδύναμης σύνδεσης (Weakest Link Pruning) που εσωτερικά χρησιμοποιεί μέσο τετραγωνικό σφάλμα, k-fold cross-validation και ρυθμό εκμάθησης. Το XGBoost δημιουργεί κόμβους/διασπάσεις έως το καθορισμένο μέγιστο βάθος (max_depth) και ξεκινά το κλάδεμα προς τα πίσω έως ότου η απώλεια είναι κάτω από ένα όριο. Για παράδειγμα, σε μια διάσπαση που έχει απώλεια -3 και ο επόμενος κόμβος έχει απώλεια +7, το XGBoost δεν θα καταργήσει τη διάσπαση μόνο κοιτάζοντας μία από τις αρνητικές απώλειες. Θα υπολογίσει τη συνολική απώλεια (-3 + 7 = +4) και αν αποδειχθεί θετική θα διατηρήσει και τα δύο.

- **Sparsity awareness**
  Είναι πολύ συνηθισμένο τα δεδομένα που συλλέγουμε να έχουν αραιότητα (πολλές ελλείπουσες ή κενές τιμές) ή να γίνουν αραιά μετά την εκτέλεση μηχανικής δεδομένων (κωδικοποίηση χαρακτηριστικών). Σε κάθε δέντρο, για να γνωρίζει τα μοτίβα αραιοτήτων στα δεδομένα, εκχωρείται μια προεπιλεγμένη κατεύθυνση. Το XGBoost χειρίζεται τα δεδομένα που λείπουν αναθέτοντάς τα στην προεπιλεγμένη κατεύθυνση και βρίσκοντας την καλύτερη τιμή καταλογισμού έτσι ώστε να ελαχιστοποιεί την απώλεια εκπαίδευσης. Με άλλα λόγια, μαθαίνει αυτόματα την καλύτερη τιμή που λείπει ανάλογα με την απώλεια εκπαίδευσης και χειρίζεται διαφορετικούς τύπους μοτίβων αραιότητας στα δεδομένα πιο αποτελεσματικά.

- **Regularization**
  Επιβάλλει κυρώσεις σε πιο πολύπλοκα μοντέλα μέσω της κανονικοποίησης τόσο του LASSO (L1) όσο και του Ridge (L2) για την αποφυγή του overfitting. Οι μέθοδοι αυτοί παρουσιάζονται στο κεφάλαιο 2.2.7.

- **Weighted Quantile Sketch**
  Το XGBoost χρησιμοποιεί τον κατανεμημένο σταθμισμένο αλγόριθμο Quantile Sketch για να βρει αποτελεσματικά τα βέλτιστα σημεία διαχωρισμού μεταξύ των σταθμισμένων συνόλων δεδομένων.

- **Approximate Split Algorithm**
  Το XGBoost δίνει την δυνατότητα, πέραν του ακριβούς άπληστου αλγορίθμου (exact greedy algorithm) που χρησιμοποιεί ο Gradient Boosting για την εξεύρεση των διαχωρισμών, να χρησιμοποιηθεί ένας κατά προσέγγιση αλγόριθμος ο οποίος προτείνει πρώτα τα υποδιαιρούμενα σημεία διαχωρισμού σύμφωνα με τα εκατοστημόρια της κατανομής χαρακτηριστικών (weighted quantile sketch). Στη συνέχεια, χαρτογραφεί τα συνεχή χαρακτηριστικά σε κουβάδες (buckets) που χωρίζονται από αυτά τα υποψήφια σημεία, συγκεντρώνει τα στατιστικά στοιχεία και βρίσκει την καλύτερη λύση μεταξύ των προτάσεων με βάση τα συγκεντρωτικά στατιστικά στοιχεία.

- **Cross-Validation**
  Ο αλγόριθμος συνοδεύεται από ενσωματωμένη μέθοδο διασταυρούμενης επικύρωσης σε κάθε επανάληψη, αφαιρώντας την ανάγκη ρητού προγραμματισμού αυτής της αναζήτησης και καθορισμού του ακριβούς αριθμού των επαναλήψεων που απαιτούνται σε μία μόνο εκτέλεση. Περισσότερες πληροφορίες για αυτήν την μέθοδο παρουσιάζονται στο κεφάλαιο 2.2.9.

Χάρη στις παραπάνω βελτιώσεις, ο αλγόριθμος XGBoost είναι πιο γρήγορος στην εκτέλεση από ότι άλλες εφαρμογές του gradient boosting. Επίσης, είναι πιο αποτελεσματικός σε ότι αφορά την μνήμη.

Περισσότερες πληροφορίες για τον συγκεκριμένο αλγόριθμο μπορούν να βρεθούν εδώ [23].

2.2.4 Κλιμάκωση Χαρακτηριστικών

Όπως έχει προαναφερθεί, το κάθε δείγμα εισόδου x_i, είναι ένα διάνυσμα χαρακτηριστικών D διαστάσεων. Το κάθε χαρακτηριστικό συμβολίζεται ως $x^{(j)}$ με j=1,...,D. Τα χαρακτηριστικά αυτά προσδιορίζουν το εκάστοτε δείγμα με τέτοιο τρόπο ώστε να δίνουν πληροφορία στο μοντέλο, για να μπορεί να κάνει μια πρόβλεψη. Παραδείγματος χάριν, για έναν ζώο τα χαρακτηριστικά θα μπορούσαν να είναι το ύψος του, το βάρος του, το χρώμα του, το τρίχωμά του κ.α. Αυτά τα χαρακτηριστικά θα μπορούσαν να αποδοθούν σε κάθε δείγμα εισόδου, δηλαδή σε κάθε ζώο και να εισέλθουν σε ένα μοντέλο που προβλέπει την ταυτότητα του κάθε δείγματος εισόδου, δηλαδή το αν είναι σκύλος, γάτα κλπ. Τα χαρακτηριστικά που ενέχουν μεγάλη πληροφορια **(informative features)**, ονομάζονται επίσης **χαρακτηριστικά με υψηλή προγνωστική ισχύ (features with high predictive power)**.

2.2.4.1 Normalization

Η κανονικοποίηση (normalization) είναι μια τεχνική κλιμάκωσης (scaling technique) κατά την οποία μετατρέπεται το πραγματικό εύρος τιμών ενός χαρακτηριστικού, σε ένα καθορισμένο εύρος τιμών π.χ. [0,1] ή [-1,1]. Είναι επίσης γνωστή και ως **κλιμάκωση Min-Max (Min-Max scaling)**. Αυτό επιτυγχάνεται αφαιρώντας από κάθε χαρακτηριστικό την ελάχιστη τιμή που παίρνει ανάμεσα στα δεδομένα εκπαίδευσης και διαιρώντας με το εύρος τιμών που παίρνει στα δεδομένα εκπαίδευσης (δηλαδή την μέγιστη πλην την ελάχιστη τιμή):

$$\bar{x}^{(j)} = \frac{x^{(j)} - min^{(j)}}{max^{(j)} - min^{(j)}} \tag{2.50}$$

Από τον μαθηματικό ορισμό της κανονικοποίησης προκύπτουν οι εξής τρεις περιπτώσεις:

- Όταν η τιμή του $x^{(j)}$ είναι η ελάχιστη τιμή στη στήλη (η ελάχιστη τιμή του χαρακτηριστικού ανάμεσα σε όλα τα δείγματα) , ο αριθμητής θα είναι 0 και επομένως το $\bar{x}^{(j)}$ είναι 0.

- Από την άλλη πλευρά, όταν η τιμή του $x^{(j)}$ είναι η μέγιστη τιμή στη στήλη, ο αριθμητής είναι ίσος με τον παρονομαστή και επομένως η τιμή του $\bar{x}^{(j)}$ είναι 1.
- Εάν η τιμή του $x^{(j)}$ είναι μεταξύ της ελάχιστης και της μέγιστης τιμής, τότε η τιμή του $\bar{x}^{(j)}$ είναι μεταξύ 0 και 1.

Αν σκεφτούμε τον αλγόριθμο απότομης καθόδου (gradient descent), στον οποίο υπολογίζεται η μερική παράγωγος του σφάλματος ως προς τις παραμέτρους-βάρη $w^{(j)}$, θα δούμε ότι αν κάποιο χαρακτηριστικό έχει πολύ μεγαλύτερο εύρος τιμών από κάποιο άλλο, τότε η μερική παράγωγος του πρώτου θα κυριαρχήσει στην ενημέρωση των παραμέτρων. Αυτό σημαίνει πως τα χαρακτηριστικά με μεγαλύτερο εύρος τιμών θα έχουν μεγαλύτερη δύναμη και ισχύ στην έκβαση του μοντέλου και της κατεύθυνσής του. Έτσι, ο καθορισμός ενός συγκεκριμένου εύρους για όλα τα χαρακτηριστικά θα βοηθήσει στην αποφυγή του φαινομένου αυτού.

Η κανονικοποίηση, βοηθάει επίσης στο να μην γίνονται πράξεις μεταξύ πολύ μεγάλων ή πολύ μικρών αριθμών, πράγμα που μπορεί να οδηγήσει σε υπολογιστικά προβλήματα γνωστά και ως **αριθμητικές υπερχειλίσεις (numerical overlflow)**.

2.2.4.2 Standarization

Αντίστοιχη μέθοδος με την κανονικοποίηση (noramalization) είναι και η τυποποίηση (standarization) ή αλλιώς **κανονικοποίηση βαθμολογίας z (z-score normalization)**. Η τυποποίηση είναι μια άλλη τεχνική κλιμάκωσης όπου οι τιμές επικεντρώνονται γύρω από το μέσο όρο με τυπική απόκλιση μονάδας. Αυτό σημαίνει ότι ο μέσος όρος του χαρακτηριστικού γίνεται μηδέν και η προκύπτουσα κατανομή έχει τυπική απόκλιση μονάδας ($\mu = 0$, $\sigma = 1$). Η τυποποίηση των χαρακτηριστικών ορίζεται ως εξής:

$$\hat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sigma^{(j)}} \tag{2.51}$$

όπου **μ** είναι ο μέσος όρος των τιμών του χαρακτηριστικού και **σ** είναι η τυπική απόκλιση των τιμών αυτών. Σε αυτήν την περίπτωση οι τιμές δεν περιορίζονται σε ένα συγκεκριμένο εύρος.

Το εύλογο ερώτημα εδώ είναι πότε χρησιμοποιούμε κανονικοποίηση και πότε τυποποίηση στα δεδομένα μας. Σε αυτό το ερώτημα δεν υπάρχει κάποιος κανόνας που θα μπορούσε να δοθεί ως απάντηση. Υπάρχουν μόνο κάποιες ενδείξεις: Η κανονικοποίηση (normalization), είναι καλό να χρησιμοποιείται όταν γνωρίζουμε ότι η κατανομή των δεδομένων δεν ακολουθεί μια Γκαουσιανή κατανομή. Αυτό μπορεί να είναι χρήσιμο σε αλγόριθμους που δεν προϋποθέτουν κατανομή δεδομένων όπως το K-Nearest Neighbours και τα Νευρωνικά δίκτυα.

Αντίθετα, η τυποποίηση (standarization) μπορεί να είναι χρήσιμη σε περιπτώσεις όπου τα δεδομένα ακολουθούν μια κατανομή Gauss. Ωστόσο, αυτό δεν πρέπει απαραίτητα να ισχύει. Επίσης, σε αντίθεση με την κανονικοποίηση, η τυποποίηση δεν έχει οριακό εύρος. Έτσι, ακόμη και αν κάποια δεδομένα έχουν ακραίες τιμές (outliers), δεν θα επηρεαστούν από την τυποποίηση, πράγμα που είναι καλύτερο. Επίσης, η τυποποίηση φαίνεται να είναι πιο ευεργετική σε αλγορίθμους μη-επιβλεπόμενης μάθησης.

Παρ' όλες τις παραπάνω παρατηρήσεις, η επιλογή ανάμεσα στην τυποποίηση και την κανονικοποίηση εξαρτάται από το πρόβλημα και τον αλγόριθμο που χρησιμοποιείται. Μια καλή λύση είναι να δοκιμάζονται και οι δυο μέθοδοι στο μοντέλο και να επιλέγεται αυτή που δίνει καλύτερη απόδοση.

Συνήθως ο κλιμακωτής (scaler), εκπαιδεύεται στο σύνολο εκπαίδευσης (training set) και ο ίδιος εφαρμόζεται και στο σύνολο δοκιμής (test set). Αυτό θα αποτρέψει οποιαδήποτε **διαρροή δεδομένων (data leakage)** κατά τη διαδικασία δοκιμής του μοντέλου. Η διαρροή

δεδομένων, είναι το φαινόμενο κατά το οποίο το μοντέλο, λαμβάνει πληροφορίες εκτός του συνόλου εκπαίδευσης οι οποίες του επιτρέπουν να μάθει κάτι, που άλλοτε δεν θα γνώριζε. Αυτό έχει ως αποτέλεσμα το μοντέλο να καθίσταται μη έγκυρο ή πολλές φορές υπερβολικά αποδοτικό, χωρίς όμως η απόδοση αυτή να ανταποκρίνεται στην πραγματικότητα. Για τον λόγο αυτό, στο σύνολο δεδομένων θα πρέπει να εφαρμοστεί ο ίδιος scaler που χρησιμοποιήθηκε και στο σύνολο εκπαίδευσης ώστε τα δεδομένα δοκιμής να μην παράσχουν κάποια επιπλέον πληροφορία. Τέλος, η κλιμάκωση δεν εφαρμόζεται στις τιμές-στόχους, δηλαδή στις ετικέτες των δειγμάτων.

2.2.5 Σύνολα Δεδομένων

Τα σύνολα δεδομένων που χρησιμοποιούνται για την κατασκευή και την αξιολόγηση ενός μοντέλο, έχουν αναφερθεί αρκετές φορές σε προηγούμενα κεφάλαια. Εδώ θα μας δοθεί η ευκαιρία να ξεκαθαρίσουμε τον ρόλο του καθενός και την σημαντικότητά του.

Όταν έχουμε λάβει τα δεδομένα, η πρώτη διεργασία που πραγματοποιείται είναι να τα χωρίσουμε σε σύνολα. Τα ανακατεύουμε και τα χωρίζουμε σε: σύνολο εκπαίδευσης (training set), σύνολο δοκιμής (test set) και σύνολο επικύρωσης (validation set). Τα δύο τελευταία σύνολα ονομάζονται και **σύνολα αναμονής (hold-out sets)**, επειδή δεν χρησιμοποιούνται στο χτίσιμο-εκπαίδευση του μοντέλου. Η αναλογία με την οποία θα χωριστούν τα σύνολα, δεν είναι συγκεκριμένη, υπάρχει όμως ο άτυπος κανόνας πως το σύνολο εκπαίδευσης θα είναι το μεγαλύτερο, αφού για να εκπαιδευτεί ένα μοντέλο χρειάζεται πολλά δείγματα (για παράδειγμα 70% training, 15% test, 15% validation). Η αναλογία διαχωρισμού εξαρτάται και από τον όγκο δεδομένων που έχουμε στα χέρια μας. Αν, για παράδειγμα, το σύνολο των δεδομένων αποτελείται από εκατομμύρια δείγματα, τότε μπορεί να δοθεί ένα ακόμα μεγαλύτερο ποσοστό στο σύνολο εκπαίδευσης (της τάξης του 90%) και τα άλλα δύο σύνολα να πάρουν ο,τι απομένει (5% και 5%).

Πιο αναλυτικά η λειτουργία κάθε συνόλου:

- **Training Set**
  Το σύνολο εκπαίδευσης είναι αυτό που χρησιμοποιείται για την εκπαίδευση του μοντέλου. Είναι ουσιαστικά ένα σύνολο παραδειγμάτων που χρησιμοποιούνται για την προσαρμογή των παραμέτρων του μοντέλου. Στην πράξη, το σύνολο δεδομένων εκπαίδευσης συχνά αποτελείται από ζεύγη ενός διανύσματος εισόδου και του αντίστοιχου διανύσματος εξόδου. Η πρόβλεψη του μοντέλου ονομάζεται συνήθως στόχος (ή ετικέτα). Το τρέχον μοντέλο εκτελείται με το σύνολο δεδομένων εκπαίδευσης και παράγει ένα αποτέλεσμα-πρόβλεψη, το οποίο στη συνέχεια συγκρίνεται με το στόχο, για κάθε δείγμα εισόδου στο σύνολο δεδομένων εκπαίδευσης. Οι παράμετροι του μοντέλου προσαρμόζονται, με βάση το αποτέλεσμα της σύγκρισης και τον ειδικό αλγόριθμο μάθησης που χρησιμοποιείται.

- **Validation Set**
  Το σύνολο επικύρωσης χρησιμοποιείται για την αξιολόγηση του μοντέλου, με σκοπό να βοηθήσει στην εύρεση των βέλτιστων τιμών των υπερπαραμέτρων (hyperparameter tuning), καθώς επίσης και του βέλτιστου αλγορίθμου εκπαίδευσης για το συγκεκριμένο πρόβλημα.

  Αρχικά, εκπαιδεύονται μοντέλα τα οποία χρησιμοποιούν διαφορετικές τιμές υπερπαραμέτρων ή διαφορετικούς αλγορίθμους εκμάθησης. Τα διανύσματα εισόδου (x_i) του συνόλου επικύρωσης, εισάγονται στα εκπαιδευμένα μοντέλα, στην συνέχεια τα μοντέλα κάνουν τις αντίστοιχες προβλέψεις και οι προβλέψεις αυτές συγκρίνονται με το διάνυσμα εξόδου του συνόλου (y_i). Ως αποτέλεσμα, υπολογίζεται μια απόδοση του κάθε μοντέλου, πάνω στα δεδομένα επικύρωσης. Αυτό δίνει μια εικόνα του ποιες

τιμές υπερπαραμέτρων ή ποιοι αλγόριθμοι εκπαίδευσης τα πηγαίνουν καλύτερα στο εκάστοτε πρόβλημα.

Τα σύνολα δεδομένων επικύρωσης μπορούν να χρησιμοποιηθούν και κατά την διάρκεια εκπαίδευσης ενός μοντέλου (διακοπή της εκπαίδευσης όταν αυξάνεται το σφάλμα στο σύνολο δεδομένων επικύρωσης, καθώς αυτό είναι ένδειξη της υπερβολικής προσαρμογής του μοντέλου στο σύνολο δεδομένων εκπαίδευσης (overfitting)).

- **Test Set**
  Το σύνολο δοκιμής χρησιμοποιείται για την τελική αξιολόγηση του μοντέλου που έχει επιλεχθεί ως το βέλτιστο ανάμεσα στα δοκιμαζόμενα μοντέλα. Στην ουσία δίνει μια αναπαράσταση του πόσο καλά τα πηγαίνει το μοντέλο σε δεδομένα που δεν έχει προηγουμένως ξαναδεί, δηλαδή πόσο καλά μπορεί να γενικεύει. Για να γίνει αυτό, το τελικό μοντέλο χρησιμοποιείται για την πρόβλεψη ετικετών των παραδειγμάτων του συνόλου δοκιμών. Αυτές οι προβλέψεις συγκρίνονται με τις αληθινές ετικέτες των παραδειγμάτων για την αξιολόγηση της ακρίβειας του μοντέλου.

Συνοψίζοντας, εάν αναζητείται ο καταλληλότερος ταξινομητής για ένα πρόβλημα, το σύνολο δεδομένων εκπαίδευσης χρησιμοποιείται για την εκπαίδευση των διαφόρων υποψηφίων ταξινομητών, το σύνολο δεδομένων επικύρωσης χρησιμοποιείται για να συγκρίνει τις επιδόσεις τους και να αποφασίσει ποιος θα επιλεγεί και τέλος, χρησιμοποιείται το σύνολο δεδομένων δοκιμής για να ληφθούν τα χαρακτηριστικά απόδοσης όπως ακρίβεια, ευαισθησία, score F και ούτω καθεξής.

Αν κάποιος αναρωτηθεί γιατί να μην χρησιμοποιούμε ένα μόνο σύνολο που να κάνει όλες τις λειτουργίες, η απάντηση είναι απλή. Όταν χτίζουμε ένα μοντέλο, αυτό που δεν θέλουμε είναι το μοντέλο να αποδίδει καλά μόνο στην πρόβλεψη ετικετών παραδειγμάτων που έχουν ήδη δει οι αλγόριθμοι μάθησης. Ένας αλγόριθμος που απλώς απομνημονεύει όλα τα παραδείγματα εκπαίδευσης και στη συνέχεια χρησιμοποιεί τη μνήμη για να «προβλέπει» τις ετικέτες τους δεν θα κάνει λάθη όταν του ζητηθεί να προβλέψει τις ετικέτες των παραδειγμάτων εκπαίδευσης, αλλά ένας τέτοιος αλγόριθμος θα ήταν άχρηστος στην πράξη. Αυτό που πραγματικά θέλουμε είναι το μοντέλο μας να προβλέπει καλά, παραδείγματα που ο αλγόριθμος μάθησης δεν έχει ξαναδεί. Θέλουμε λοιπόν καλή απόδοση στα hold-out σύνολα. Τέλος, υπάρχει ένας διαχωρισμός και μεταξύ του validation και του test set. Ο λόγος για τον οποίο χρησιμοποιούμε δύο ξεχωριστά τέτοια σύνολα είναι γιατί εκτός από τα training δεδομένα, μπορεί επίσης να γίνει υπερβολική προσαρμογή του μοντέλου (overfitting) και στην εύρεση βέλτιστων υπερπαραμέτρων, δηλαδή στην χρήση του validation set. Για αυτό τον λόγω χρειάζεται να τεστάρουμε την απόδοση του μοντέλου με τις βέλτιστες υπερπαραμέτρους, σε ένα ανεξάρτητο σύνολο (test set). Το φαινόμενο της υπερβολικής προσαρμογής (overfitting) θα αναλυθεί στο επόμενο κεφάλαιο.

2.2.6 Underfitting και Overfitting

Underfitting

Όταν το μοντέλο κάνει πολλά λάθη στα δεδομένα εκπαίδευσης (training set) τότε λέμε πως παρουσιάζει **underfitting** ή αλλιώς πως έχει **μεγάλη προκατάληψη (high bias)**. Σε αυτήν την περίπτωση, δεν είναι ικανό να προβλέψει σωστά τις ετικέτες των δεδομένων, πάνω στα οποία εκπαιδεύτηκε.

Οι λόγοι που μπορεί να οδηγήσουν σε αυτό το φαινόμενο είναι είτε ότι έχει επιλεχθεί ένα πολύ απλό μοντέλο για τα δεδομένα που έχουμε, το οποίο δεν είναι ικανό να τα ταξινομήσει σωστά (για παράδειγμα ένα γραμμικό μοντέλο την στιγμή που τα δεδομένα δεν είναι γραμμικώς διαχωρίσιμα), είτε ότι τα χαρακτηριστικά των δεδομένων δεν παρέχουν αρκετή πληροφορία, ώστε το μοντέλο να καταφέρει να εξάγει συμπεράσματα από αυτήν.

Το underfitting συνήθως εντοπίζεται εύκολα από μια καλή μετρική απόδοσης. Αν εντοπιστεί αυτό το φαινόμενο, η λύση είναι να δοκιμαστούν εναλλακτικοί αλγόριθμοι μάθησης, ή να ελεγχθούν τα χαρακτηριστικά και να εμπλουτιστούν κατά περίπτωση.

Overfitting

Στον αντίποδα, όταν το μοντέλο παρουσιάζει **overfitting** δηλαδή υπερβολική προσαρμογή ή αλλιώς **υψηλή διακύμανση (high variance)**, τότε τα πηγαίνει πολύ καλά στα δεδομένα εκπαίδευσης αλλά όχι στα δεδομένα επικύρωσης ή δοκιμής. Σε αυτήν την περίπτωση, το μοντέλο δεν έχει την ικανότητα να γενικεύει, δηλαδή να παράγει σωστές προβλέψεις για δεδομένα που προηγουμένως δεν έχει ξαναδεί. Η ουσία της υπερβολικής προσαρμογής είναι να έχει εξαγάγει εν αγνοία του, μέρος της παραμένουσας παραλλαγής (δηλαδή του θορύβου) σαν αυτή η παραλλαγή να αντιπροσωπεύει την υποκείμενη δομή του μοντέλου. Με άλλα λόγια, το μοντέλο θυμάται έναν τεράστιο αριθμό παραδειγμάτων αντί να μαθαίνει να παρατηρεί χαρακτηριστικά. Αυτό σημαίνει ότι εάν τα δεδομένα εκπαίδευσης είχαν δειγματοληπτηθεί διαφορετικά, η μάθηση θα οδηγούσε σε ένα πολύ διαφορετικό μοντέλο. Αυτός είναι ο λόγος για τον οποίο το μοντέλο που παρουσιάζει overfitting, έχει χαμηλή απόδοση στα δεδομένα δοκιμής: τα δεδομένα δοκιμής και εκπαίδευσης είναι δειγματοληπτημένα από το σύνολο δεδομένων ανεξάρτητα το ένα από το άλλο.

Οι λόγοι που μπορεί να οδηγήσουν στο φαινόμενο είναι είτε το μοντέλο να είναι πολύ πολύπλοκο για τα δεδομένα (για παράδειγμα ένα δέντρο απόφασης μεγάλου ύψους ή ένα πολύ βαθύ νευρωνικό δίκτυο), είτε τα χαρακτηριστικά να είναι πολλά σε σχέση με τον μικρό όγκο δεδομένων εκπαίδευσης.

Παρακάτω παρουσιάζονται κάποιες λύσεις που μπορούν να δοθούν για τον περιορισμό του overfitting:

1. Η χρήση μιας τεχνικής με την οποία ξαναδειγματοληπτούμε τα δεδομένα για να αξιολογήσουμε την απόδοση του μοντέλου. Μια τέτοια τεχνική είναι το cross-validation που θα παρουσιαστεί στο κεφάλαιο 2.2.9.

2. Η χρήση ενός συνόλου επικύρωσης, το οποίο περιγράψαμε στο 2.2.5, για να αξιολογήσουμε την απόδοση των μοντέλων σε καινούρια δεδομένα που το μοντέλο δεν έχει ξαναδεί.

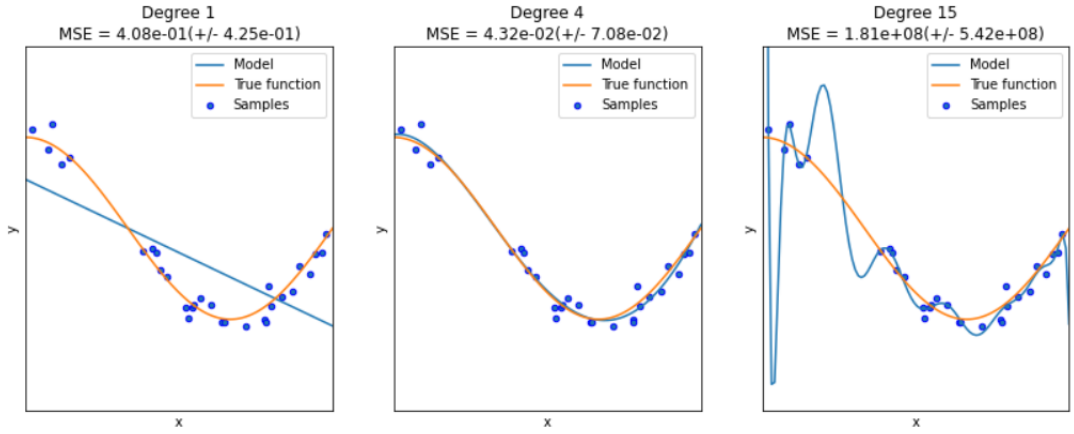
3. Η δοκιμή πιο απλών μοντέλων (για παράδειγμα γραμμικής παλινδρόμησης έναντι πολυωνυμικής ή SVM με γραμμικό kernel έναντι kernel RBF).

4. Η μείωση των διαστάσεων των δειγμάτων στο σύνολο δεδομένων. Δηλαδή, η μείωση των χαρακτηριστικών.

5. Η προσθήκη περισσότερων δεδομένων εκπαίδευσης.

6. Η κανονικοποίηση του μοντέλου (regularization), την οποία θα περιγράψουμε στο επόμενο κεφάλαιο 2.2.7.

Παρακάτω παρουσιάζεται ένα παράδειγμα προβλήματος γραμμικής παλινδρόμησης με τρεις περιπτώσεις [24]:



Σχήμα 2.17: Παράδειγμα overfitting-underfitting σε γραμμική παλινδρόμηση

Στην πρώτη γραφική παράσταση αριστερά, έχει χρησιμοποιηθεί ένα πολυώνυμο 1ου βαθμού. Όπως βλέπουμε, το μοντέλο που έχει σχηματιστεί από αυτό το πολυώνυμο (μπλε γραμμή), δεν πλησιάζει τα δείγματα. Φαίνεται να μην μπορεί να κάνει σωστές προβλέψεις στα δεδομένα εκπαίδευσης και έτσι παρουσιάζει **underfitting**.

Στην δεύτερη γραφική παράσταση βλέπουμε ένα μοντέλο πολυωνυμικού βαθμού 4. Εδώ το μοντέλο έχει πλησιάσει αρκετά τα δείγματα και είναι παρόμοιο με την πραγματική συνάρτηση (πορτοκαλί γραμμή). Σε αυτήν την περίπτωση θεωρούμε ότι έχουμε **"Best Fit"** ή αλλιώς **"Good Model"**.

Στην τρίτη γραφική παράσταση στα δεξιά, έχει χρησιμοποιηθεί πολυώνυμο 15ου βαθμού και όπως βλέπουμε το μοντέλο έχει γίνει αρκετά πολύπλοκο. Περνάει σχεδόν από κάθε δείγμα πράγμα που δείχνει υπερβολική προσαρμογή στα δεδομένα εκπαίδευσης. Σε αυτήν την περίπτωση, αν το μοντέλο δοκιμαστεί σε νέα δεδομένα που δεν μοιάζουν με αυτά της εκπαίδευσης, τότε το πιο πιθανό είναι ότι θα δώσει μια κακή απόδοση. Εδώ το μοντέλο παρουσιάζει **overfitting**.

Αξίζει να σημειωθεί πως το σφάλμα (MSE) βλέπουμε να μειώνεται, όσο αυξάνονται οι βαθμοί του πολυωνύμου, πράγμα αναμενόμενο αφού το μοντέλο προσαρμόζεται όλο και περισσότερο στα δεδομένα εκπαίδευσης. Έτσι είναι φανερό πως ένα μικρότερο σφάλμα στα δεδομένα εκπαίδευσης, δεν σημαίνει απαραίτητα πως το μοντέλο είναι καλύτερο (αφού το δεξιότερο μοντέλο με το μικρότερο σφάλμα δεν είναι το καλύτερο). Για αυτό τον λόγο είναι σημαντικά τα υπόλοιπα σύνολα δεδομένων (validation και test), στην αξιολόγηση της απόδοσης.

2.2.7 Regularization

Με τον όρο regularization εννοούμε τις μεθόδους που προσπαθούν να επιβάλλουν στον αλγόριθμο μάθησης να δημιουργήσει ένα μοντέλο λιγότερο πολύπλοκο, ώστε να αποφευχθεί ο κίνδυνος υπερβολικής προσαρμογής (overfitting). Με αυτόν τον τρόπο ίσως αυξηθεί λίγο το bias αλλά σίγουρα θα μειωθεί το variance. Με άλλα λόγια, η κανονικοποίηση (regularization), είναι οποιαδήποτε τροποποίηση κάνει κάποιος σε έναν αλγόριθμο εκμάθησης που αποσκοπεί στη μείωση του σφάλματος γενίκευσης αλλά όχι του σφάλματος εκπαίδευσης. Αυτό το πρόβλημα είναι γνωστό και ως αντιστάθμιση προκατάληψης-διακύμανσης (bias-variance tradeoff).

Για να δημιουργήσουμε ένα κανονικοποιημένο μοντέλο, τροποποιούμε την αντικειμενική συνάρτηση/συνάρτηση κόστους προσθέτοντας έναν τιμωρητικό όρο του οποίου η αξία είναι υψηλότερη όταν το μοντέλο είναι πιο περίπλοκο.

Οι πιο συχνά χρησιμοποιούμενες τεχνικές κανονικοποίησης είναι η κανονικοποίηση L1

(L1 regularization) και η κανονικοποίηση L2 **(L2 regularization)**. Θα δούμε αυτές τις τεχνικές στο πρόβλημα της γραμμικής παλινδρόμησης:

Η συνάρτηση κόστους που θέλουμε να ελαχιστοποιήσουμε για την γραμμική παλινδρόμηση είναι η παρακάτω, όπως έχουμε ξαναδεί:

$$RSS(w,b) = \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2$$

Ένα πρόβλημα γραμμικής παλινδρόμησης που χρησιμοποιεί **L1 regularization** ονομάζεται **παλινδρόμηση LASSO(Least Absolute Shrinkage and Selection Operator)**. Η αντικειμενική συνάρτηση σε αυτήν την περίπτωση γίνεται ως εξής:

$$RSS_{LASSO}(w,b) = \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2 + \alpha\sum_{\{j=1\}}^{p}|w^{(j)}| \implies \tag{2.52}$$

$$RSS_{LASSO}(w,b) = RSS(w,b) + \alpha\sum_{\{j=1\}}^{p}|w^{(j)}| \tag{2.53}$$

όπου ο πρώτος όρος του αθροίσματος είναι ίδιος με την συνάρτηση κόστους της κανονικής γραμμικής παλινδρόμησης και ο δεύτερος όρος ισούται με το άθροισμα των βαρών όλων των χαρακτηριστικών επί μια μεταβλητή **α**, η οποία καθορίζει το πόσο μεγάλη επίδραση έχει η κανονικοποίηση στο μοντέλο.

Αν τεθεί το α=0 τότε δεν υπάρχει κανονικοποίηση και το πρόβλημα είναι απλή γραμμική παλινδρόμηση. Αντίθετα, αν η παράμετρος α πάρει μια μεγάλη τιμή τότε το μοντέλο θα προσπαθήσει να ελαχιστοποιήσει τον όρο $\sum_{\{j=1\}}^{p}|w^{(j)}|$, με σκοπό να ελαχιστοποίησει όλη την συνάρτηση κόστους, με αποτέλεσμα πολλά βάρη να πάρουν μικρές τιμές ή και να τεθούν ίσα με μηδέν. Στην ουσία το L1 regularization λειτουργεί ως ένας επιλογέας χαρακτηριστικών, που επιλέγει τα χαρακτηριστικά που είναι πιο σημαντικά για την πρόβλεψη, αφαιρώντας τα υπόλοιπα. Αυτό αυτομάτως σημαίνει πως το μοντέλο θα γίνει πολύ απλό πράγμα που μπορεί να οδηγήσει σε underfitting. Επομένως, η παράμετρος α τίθεται προς διερεύνηση και θα πρέπει να βρεθεί μια κατάλληλη τιμή της, η οποία θα μειώνει το overfitting, χωρίς να αυξάνει πολύ το bias (underfitting).

Ένα πρόβλημα γραμμικής παλινδρόμησης που χρησιμοποιεί **L2 regularization** ονομάζεται **παλινδρόμηση κορυφογραμμών (RIDGE regression)**. Η αντικειμενική συνάρτηση σε αυτήν την περίπτωση γίνεται ως εξής:

$$RSS_{RIDGE}(w,b) = \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2 + \alpha\sum_{\{j=1\}}^{p}(w^{(j)})^2 \implies \tag{2.54}$$

$$RSS_{RIDGE}(w,b) = RSS + \alpha\sum_{\{j=1\}}^{p}(w^{(j)})^2 \tag{2.55}$$

όπου και πάλι ο πρώτος όρος του αθροίσματος είναι ίδιος με την συνάρτηση κόστους της κανονικής γραμμικής παλινδρόμησης ενώ ο δεύτερος όρος εδώ ισούται με το άθροισμα των τετραγώνων των βαρών όλων των χαρακτηριστικών επί μια μεταβλητή **α**, η οποία καθορίζει το πόσο μεγάλη επίδραση έχει η κανονικοποίηση στο μοντέλο.

Η τεχνική L2 regularization, συνήθως δίνει καλύτερα αποτελέσματα από την L1 σε ότι αφορά την μεγιστοποίηση της απόδοσης του μοντέλου στα δεδομένα δοκιμής. Επίσης, ο παράγοντας L2 στην συνάρτηση κόστους είναι παραγωγίσιμος, πράγμα που σημαίνει ότι μπορεί να χρησιμοποιηθεί ο αλγόριθμος απότομης καθόδου (gradient descent), ως μέθοδος βελτιστοποίησης.

Άλλες τεχνικές κανονικοποίησης που χρησιμοποιούνται σε νευρωνικά δίκτυα είναι η εγκατάλειψη (dropout), η ομαλοποίηση κατά παρτίδες (batch-normalization), η πρόωρη διακοπή (early stopping) και η αύξηση δεδομένων (data augmentation).

2.2.8 Μετρικές Αξιολόγησης της Απόδοσης ενός Ταξινομητή

Στο κεφάλαιο αυτό θα παρουσιαστούν οι μετρικές με τις οποίες μπορεί να γίνει η αξιολόγηση ενός μοντέλου ταξινόμησης.

2.2.8.1 Confusion Matrix

Το confusion matrix είναι ένας πίνακας NxN που δείχνει τις επιτυχίες και αποτυχίες του μοντέλου στις προβλέψεις που έκανε, ανά κλάση. Πιο συγκεκριμένα, ο πίνακας έχει ως σειρές τις πραγματικές ετικέτες των δειγμάτων (N) και ως στήλες τις ετικέτες που προβλέφθηκαν από το μοντέλο (N).

Θεωρώντας ένα δυαδικό πρόβλημα με δύο κλάσεις, ο πίνακας σύγχυσης είναι διαστάσεων 2x2 και περιέχει τους παρακάτω 4 όρους:

- **True Positives(TP)**
  Τα true positives βρίσκονται στην θέση (0,0) του πίνακα και είναι ο αριθμός των δειγμάτων που ήταν θετικά (ετικέτα 1) και προβλέφθηκαν ως θετικά από το μοντέλο.

- **False Positives(FP)**
  Τα false positives βρίσκονται στην θέση (0,1) και είναι ο αριθμός των δειγμάτων που ήταν αρνητικά (ετικέτα 0) και προβλέφθηκαν ως θετικά από το μοντέλο.

- **False Negatives(FN)**
  Τα false negatives βρίσκονται στην θέση (1,0) και είναι ο αριθμός των δειγμάτων που ήταν θετικά (ετικέτα 1) και προβλέφθηκαν ως αρνητικά από το μοντέλο.

- **True Negatives(TN)**
  Τα true negatives βρίσκονται στην θέση (1,1) και είναι ο αριθμός των δειγμάτων που ήταν αρνητικά (ετικέτα 0) και προβλέφθηκαν ως αρνητικά από το μοντέλο.

Παρακάτω παρουσιάζεται ένα παράδειγμα πίνακα σύγχυσης για την καλύτερη κατανόησή του:

Predicted classes / Actual classes	Cat	Non-Cat
Cat	**9(TP)**	4(FN)
Non-Cat	3(FP)	**4(TN)**

Table 2.1: Παράδειγμα πίνακα σύγχυσης σε δυαδικό πρόβλημα

Το παραπάνω πρόβλημα αφορά την ταξινόμηση σε γάτες και όχι γάτες. Όπως μπορούμε να δούμε από τον πίνακα 9 δείγματα ήταν γάτες και προβλέφθηκαν ως γάτες, 3 δείγματα δεν ήταν γάτες και προβλέφθηκαν ως γάτες, 4 δείγματα ήταν γάτες και προβλέφθηκαν ως όχι-γάτες και 4 δείγματα δεν ήταν γάτες και προβλέφθηκαν ως όχι-γάτες. Ο συνολικός

αριθμός δειγμάτων είναι 9+3+4+4=20, ενώ τα δείγματα που ταξινομήθηκαν-προβλέφθηκαν σωστά απ᾽ το μοντέλο είναι αυτά που παρουσιάζονται στην διαγώνιο, τα true positives και true negatives => 9+4 = 13.

Η μετρική αυτή μπορεί να μας βοηθήσει να εντοπίσουμε συγκεκριμένα λάθη που τείνει να κάνει το μοντέλο. Για παράδειγμα, αν το πρόβλημα αφορούσε την αναγνώριση αντικειμένων και στον πίνακα βλέπαμε πολλά από τα δείγματα που στην πραγματικότητα είναι ποτήρια να έχουν λανθασμένα ταξινομηθεί ως μπουκάλια, τότε αυτό δείχνει πως το μοντέλο δεν έχει μάθει να ξεχωρίζει καλά τα μπουκάλια από τα ποτήρια. Σε αυτήν την περίπτωση θα μπορούσαμε να βοηθήσουμε τον αλγόριθμο με το να προσθέσουμε περισσότερα επισημασμένα δείγματα μπουκαλιών ή να προσθέσουμε περισσότερα χαρακτηριστικά δειγμάτων, ώστε να τον βοηθήσουμε να ξεχωρίσει καλύτερα αυτά τα αντικείμενα.

Από τον πίνακα σύγχυσης προκύπτουν και οι παρακάτω σημαντικές μετρικές.

2.2.8.2 Precision

Το **precision** ονομάζεται και αλλιώς **positive predictive value (PPV)** (ακρίβεια ή θετική προγνωστική αξία) και είναι το ποσοστό των θετικών δειγμάτων που προβλέφθηκαν σωστά (δηλαδή ήταν όντως θετικά), από όλα τα δείγματα που προβλέφθηκαν ως θετικά. Δηλαδή, σύμφωνα με τις τιμές του confusion matrix, ορίζεται ως εξής:

$$Precision = PPV = \frac{TP}{TP + FP} \tag{2.56}$$

Από την (2.47), φαίνεται ότι το precision υπολογίζεται διαιρώντας τα true positives, δηλαδή τον αριθμό των θετικών δειγμάτων που προβλέφθηκαν σωστά, με το άθροισμα των true positives και false positives, δηλαδή όλων των δειγμάτων που προβλέφθηκαν ως θετικά.

2.2.8.3 Negative Predictive Value

Το **negative predictive value (NPV)** ή αλλιώς η αρνητική προγνωστική αξία, είναι το αντίστοιχο με το PPV αλλά για τα αρνητικά δείγματα. Είναι δηλαδή, το ποσοστό των αρνητικών δειγμάτων που προβλέφθηκαν σωστά, από όλα τα δείγματα που προβλέφθηκαν ως αρνητικά:

$$NPV = \frac{TN}{TN + FN} \tag{2.57}$$

Από την (2.48), φαίνεται ότι το NPV υπολογίζεται διαιρώντας τα true negatives, δηλαδή τον αριθμό των αρνητικών δειγμάτων που προβλέφθηκαν σωστά, με το άθροισμα των true negatives και false negatives, δηλαδή όλων των δειγμάτων που προβλέφθηκαν ως αρνητικά.

2.2.8.4 Recall

Το **recall** ή αλλιώς **sensitivity** ή αλλιώς **true positive rate (TPR)** (ανάκληση ή ευαισθησία ή πραγματικό θετικό ποσοστό), είναι το ποσοστό των πραγματικών θετικών δειγμάτων που προβλέφθηκαν σωστά:

$$Recall = Sensitivity = TPR = \frac{TP}{TP + FN} \tag{2.58}$$

Από την (2.49), φαίνεται ότι το Recall υπολογίζεται διαιρώντας τα true positives, δηλαδή τον αριθμό των θετικών δειγμάτων που προβλέφθηκαν σωστά, με το άθροισμα των true positives και false negatives, δηλαδή όλων των πραγματικών θετικών δειγμάτων.

2.2.8.5 Specificity

Το **specificity** ή αλλιώς **true negative rate (TNR)** (ειδικότητα ή πραγματικό αρνητικό ποσοστό), είναι το αντίστοιχο με το recall αλλά για αρνητικά δείγματα, δηλαδή είναι το ποσοστό των πραγματικών αρνητικών δειγμάτων που προβλέφθηκαν σωστά:

$$Specificity = TNR = \frac{TN}{TN + FP} \tag{2.59}$$

Από την (2.50), φαίνεται ότι το Specificity υπολογίζεται διαιρώντας τα true negatives, δηλαδή τον αριθμό των αρνητικών δειγμάτων που προβλέφθηκαν σωστά, με το άθροισμα των true negatives και false positives, δηλαδή όλων των πραγματικών αρνητικών δειγμάτων.

Συνήθως, στην πράξη, καλούμαστε να διαλέξουμε ανάμεσα σε ένα υψηλό precision ή ένα υψηλό recall, καθώς είναι δύσκολο να έχουμε και τα δύο. Ο τρόπος που επιλέγουμε είναι ανάλογος με το πρόβλημα που έχουμε να αντιμετωπίσουμε. Για παράδειγμα, εάν θέλουμε να αναγνωρίσουμε τα spam μηνύματα από τα όχι-spam, τότε αυτό που μας ενδιαφέρει περισσότερο είναι να έχουμε υψηλότερο precision, με κόστος να έχουμε χαμηλότερο recall. Αυτό γιατί είναι πιο σημαντικό για μας, να μην διαγράφουμε κανονικά-σημαντικά μηνύματα επειδή έχουν λανθασμένα προβλεφθεί ως spam (precision), απ᾽ το να δεχόμαστε spam μηνύματα τα οποία έχουν λανθασμένα θεωρηθεί κανονικά (recall).

Υπάρχουν διάφοροι τρόποι για να καταστίσουμε σαφές στον αλγόριθμο, ποιά από τις μετρικές αυτές θέλουμε να αυξήσουμε. Για παράδειγμα, μπορούμε να βάλουμε μεγαλύτερα βάρη στα δείγματα των κλάσεων που μας ενδιαφέρει να προβλεφθούν σωστά (π.χ. στα spam μηνύματα). Μπορούμε, επίσης, να κάνουμε αναζήτηση υπερπαραμέτρων (2.2.9) ώστε να βρεθούν αυτές που δίνουν υψηλότερο recall ή precision στο σύνολο επικύρωσης. Τέλος, στους αλγορίθμους εκμάθησης οι οποίοι επιστρέφουν πιθανότητες (όπως η λογιστική παλινδρόμηση ή το δέντρο απόφασης), μπορούμε να ελέγξουμε την τιμή του κατωφλίου απόφασης για να αυξήσουμε το precision. Παραδείγματος χάριν, μπορούμε να ορίσουμε το κατώφλι απόφασης ως 0.9 και έτσι η πρόβλεψη θα είναι θετική μόνο αν η πιθανότητα που επιστρέφεται απ᾽ το μοντέλο είναι μεγαλύτερη του 0.9 .

2.2.8.6 F1-score

Πολλές φορές προσπαθούμε να πετύχουμε τόσο καλό precision, όσο και καλό recall. Αυτή η επιδίωξη συνοψίζεται σε μια μετρική που ονομάζεται **F1-score** και είναι ο αρμονικός μέσος των τιμών recall και precision:

$$F1 = 2\frac{Precision \quad Recall}{Precision + Recall} \tag{2.60}$$

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{2.61}$$

Ο λόγος για τον οποίο χρησιμοποιείται ένας αρμονικός μέσος, έναντι ενός αριθμητικού ή γεωμετρικού, είναι επειδή ο αρμονικός μέσος τιμωρεί τις πιο ακραίες τιμές.

Υπάρχουν, ωστόσο, καταστάσεις για τις οποίες ένας επιστήμονας δεδομένων θα ήθελε να δώσει μεγαλύτερη σημασία / βάρος είτε στην ακρίβεια είτε στην ανάκληση (precision or recall). Για αυτόν τον λόγο υπάρχει η πιο γενική μορφή του **F-score** ή $\boldsymbol{F_\beta}$ που παρουσιάζεται παρακάτω:

$$F_\beta = (1 + \beta^2)\frac{Precision \quad Recall}{(\beta^2 Precision) + Recall} \tag{2.62}$$

$$F_\beta = \frac{(1+\beta^2)TP}{(1+\beta^2)TP + \beta^2 FN + FP} \tag{2.63}$$

όπου το **β** είναι ένας θετικός πραγματικός αριθμός, που επιτρέπει προσαρμογή της μετρικής, ώστε να ελέγχει την σημαντικότητα του recall έναντι του precision. Πιο συγκεκριμένα, το β επιλέγεται έτσι ώστε η ανάκληση (recall) να θεωρείται β φορές πιο σημαντική από την ακρίβεια (precision). Παραδείγματος χάριν, αν θέλαμε να δώσουμε περισσότερη σημασία στο precison μπορούσαμε να θέσουμε β=0.5 έτσι ώστε τα false positives (FP) να έχουν μεγαλύτερη βαρύτητα από τα false negatives (FN), ενώ αν θέλαμε να δώσουμε περισσότερη σημασία στο recall μπορούσαμε να θέσουμε β=2, έτσι ώστε τα false negatives (FN) να έχουνε μεγαλύτερη βαρύτητα από τα false positives (FP).

2.2.8.7 Accuracy

Η ακρίβεια **(accuracy-ACC)**, είναι το ποσοστό του συνολικού αρθμού προβλέψεων που ήταν σωστές. Όσον αφορά τους συμβολισμούς του πίνακα σύγχυσης, η ακρίβεια ορίζεται ως εξής:

$$Accuracy = ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.64}$$

όπου στην (2.55), ο αριθμητής είναι το άθροισμα του αριθμού των δειγμάτων που προβλέφθηκαν σωστά, δηλαδή τα θετικά δείγματα που προβλέφθηκαν σωστά (true positive) συν τα αρνητικά δείγματα που προβλέφθηκαν σωστά (true negatives) και ο παρονομαστής είναι το άθροισμα όλων των δειγμάτων (true positives και true negatives και false positives και false negatives).

Η μετρική αυτή είναι χρήσιμη όταν τα σφάλματα πρόβλεψης όλων των κλάσεων είναι εξίσου σημαντικά. Στο πρόβλημα του εντοπισμού ανεπιθύμητης αλληλογραφίας, για παράδειγμα, που είδαμε παραπάνω, όπου μας ενδιαφέρει περισσότερο το precision απ' ότι το recall, η μετρική της ακρίβειας (accuracy) δεν μπορεί να μας δώσει καθαρή εικόνα. Σε αυτήν την περίπτωση όμως, μπορεί να χρησιμοποιηθεί η μετρική **cost-sensitive accuracy**, η οποία χρησιμοποιεί ανάλογη λειτουργία με την γενική μορφή του f-score. Στην ουσία τοποθετούμε ένα κόστος (θετικό αριθμό) και στους δύο τύπους λαθών (false positives και false negatives), πριν τα τοποθετήσουμε στην μαθηματική έκφραση (2.64).

Στον παρακάτω πίνακα φαίνονται συνοπτικά οι μετρικές που περιγράφηκαν μέχρι στιγμής:

Confusion Matrix		**Model**			
		Positive	Negative		
Target	Positive	TP	FN	**Recall (Sensitivity-TPR)**	TP/(TP+FN)
	Negative	FP	TN	**Specificity (TNR)**	TN/(TN+FP)
		Presicion (PPV) TP/(TP+FP)	**Negative Predictive Value (NPV)** TN/(TN+FN)	**Accuracy (ACC)** = (TP+TN)/(TP+TN+FP+FN)	

Table 2.2: Μετρικές αξιολόγησης μοντέλου-ταξινομητή

2.2.8.8 AUC-ROC

Αυτή η μετρική ονομάζεται area under the curve (AUC) ή area under the ROC curve, όπου το ROC προέρχεται από το "receiver operating characteristic", και σημαίνει χαρακτηριστική καμπύλη λειτουργίας δέκτη. Η μετρική αυτή χρησιμοποιεί δύο όρους:

1. **True Positive Rate (TPR)**
   Το TPR το είδαμε και παραπάνω. Είναι η μετρική recall, η οποία αναπαριστά το πόσα θετικά δείγματα προβλέφθηκαν σωστά.

$$TPR = \frac{TP}{TP + FN}$$

2. **False Positive Rate (FPR)**
   Το FPR αναπαριστά το πόσα απ' τα αρνητικά δείγματα προβλέφθηκαν λάθος:

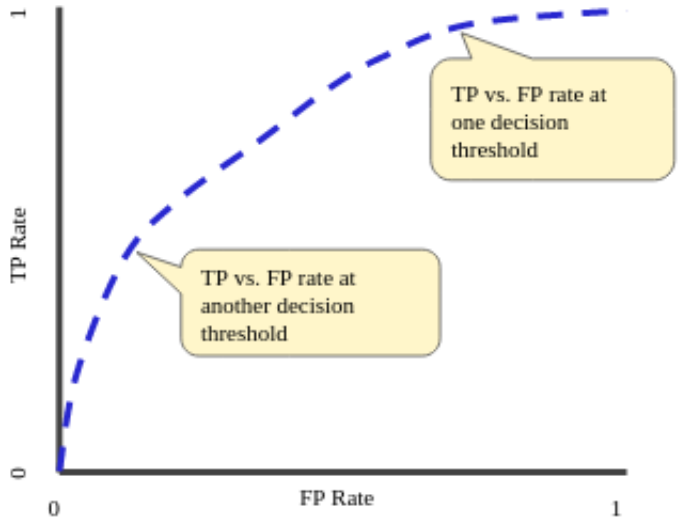
$$FPR = \frac{FP}{FP + TN}$$

   Ισούται επίσης με το 1 - specificity (TNR) που είδαμε στο 2.2.8.5.

Η μετρική αυτή, αφορά αλγορίθμους μάθησης οι οποίοι επιστρέφουν κάποια βαθμολογία εμπιστοσύνης (confidence score) ή πιθανότητα. Τέτοιοι αλγόριθμοι είναι η λογιστική παλινδρόμηση, τα δέντρα απόφασης, καθώς επίσης και όλοι οι αλγόριθμοι συλλογικής μάθησης που έχουμε δει, οι οποίοι χρησιμοποιούν δέντρα απόφασης.

Η καμπύλη σχηματίζεται από μια γραφική παράσταση που έχει στο x άξονα το FPR και στον y άξονα το TPR. Τα ζεύγη αυτά τιμών, υπολογίζονται για διαφορετικές τιμές του κατωφλίου απόφασης (decision threshold). Στην περίπτωση για παράδειγμα που ο αλγόριθμος επιστρέφει μια πιθανότητα μεταξύ [0,1], οι τιμές του κατωφλίου μπορούν να χωριστούν ανά 0.1 στο διάστημα αυτό (δηλαδή [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]). Ο αλγόριθμος προβλέπει την πιθανότητα τα δείγματα να είναι θετικά και ύστερα για καθεμία από τις τιμές κατωφλίου, υπολογίζει τις μετρικές FPR και TPR. Ύστερα τα ζεύγη τιμών των μετρικών αυτών αναπαριστώνται σε μια γραφική παράσταση.

Ένα παράδειγμα φαίνεται στην παρακάτω γραφική παράσταση [25]:



Σχήμα 2.18: Μετρική AUC

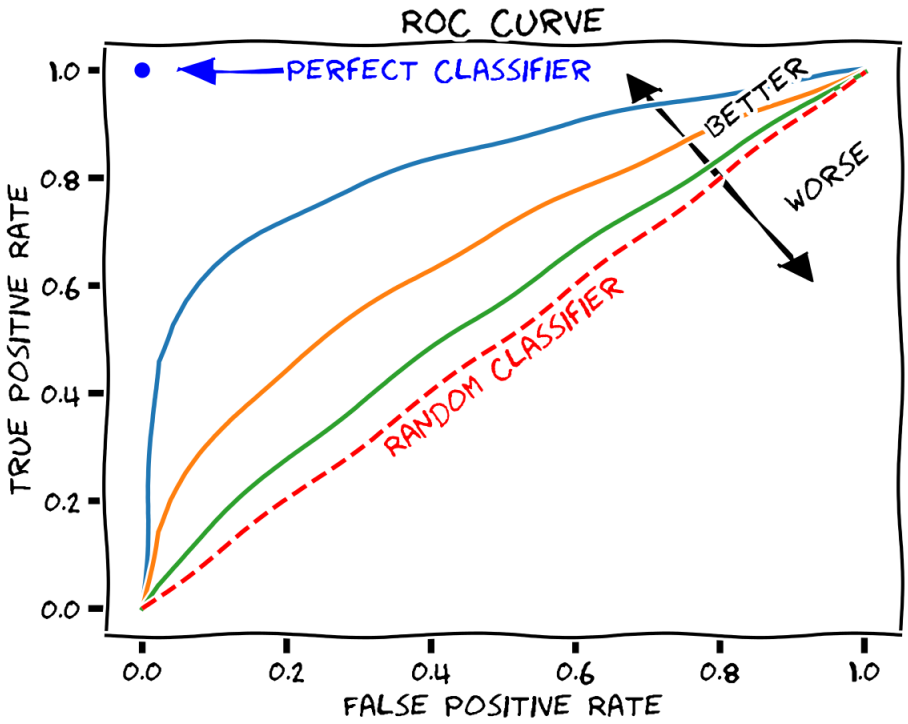
Εάν το κατώφλι είναι ίσο με 1, τότε τα δείγματα θα θεωρούνται θετικά για πιθανότητα μεγαλύτερη του 1, επομένως όλα τα δείγματα θα θεωρούνται αρνητικά και έτσι οι όροι FPR και TPR θα μηδενιστούν. Αυτό γιατί κανένα από τα αρνητικά δείγματα δεν θα χει προβλεφθεί λάθος (άρα FPR = 0) και κανένα από τα θετικά δείγματα δεν θα χει προβλεφθεί σωστά (άρα TPR = 0). Αντίθετα, εάν το κατώφλι απόφασης είναι ίσο με 0,

τότε τα δείγματα θα θεωρούνται θετικά για πιθανότητα μεγαλύτερη του 0, επομένως όλα τα δείγματα θα θεωρούνται θετικά και έτσι οι όροι FPR και TPR θα γίνουν ίσοι με 1. Αυτό γιατί όλα τα αρνητικά δείγματα θα έχουν προβλεφθεί λάθος (άρα FPR = 1) και όλα τα θετικά δείγματα θα έχουν προβλεφθεί σωστά (άρα TPR = 1).

Ο στόχος μας είναι να έχουμε όσο το δυνατόν περισσότερα σωστά ταξινομημένα δείγματα, δηλαδή όσο το δυνατόν μεγαλύτερο TPR και όσο το δυνατόν λιγότερα λάθος ταξινομημένα αρνητικά δείγματα, δηλαδή όσο το δυνατόν μικρότερο FPR. Αυτό συνοψίζεται σε έναν στόχο, που είναι να έχουμε όσο το δυνατόν μεγαλύτερη επιφάνεια κάτω από την καμπύλη που σχηματίζουν οι δυο μετρικές.

Ένας τυχαίος ταξινομητής μπορεί να θεωρηθεί ότι δίνει AUC = 0.5, επειδή στην ουσία όσα θετικά δείγματα ταξινομεί σωστά, τόσα αρνητικά δείγματα ταξινομεί λάθος (TPR = FPR). Για να έχουμε έναν καλύτερο ταξινομητή από τον τυχαίο θα πρέπει να πάρουμε AUC > 0.5, ενώ αν πάρουμε AUC < 0.5, αυτό σημαίνει ότι ο ταξινομητής μας είναι χειρότερος ακόμα και από ένα τυχαίο ταξινομητή, επομένως κάτι δεν πηγαίνει καλά.

Παρακάτω φαίνονται διάφορες περιπτώσεις της καμπύλης ROC, καθώς επίσης και η καμπύλη στην περίπτωσης του τυχαίου ταξινομητή [26]:



Σχήμα 2.19: Ο χώρος ROC για έναν "καλύτερο" και "χειρότερο" ταξινομητή.

Όπως βλέπουμε από το σχήμα 2.19, ο τέλειος ταξινομητής, δηλαδή η βέλτιστη περίπτωση, παρουσιάζεται στην πάνω αριστερά γωνία όπου το TPR ισούται με 1, το FPR ισούται με 0 και η περιοχή κάτω απ' την καμπύλη θα έχει την μέγιστη τιμή. Επιπλέον, οι καμπύλες πάνω από την καμπύλη του τυχαίου ταξινομητή (AUC = 0.5) είναι όλο και καλύτερες, όσο απομακρυνόμαστε από αυτόν, αφού το εμβαδόν κάτω απ' την καμπύλη μεγαλώνει, ενώ οι καμπύλες κάτω από την καμπύλη του τυχαίου ταξινομητή είναι όλο και χειρότερες, όσο απομακρυνόμαστε από αυτόν, αφού το εμβαδόν κάτω από την καμπύλη μειώνεται.

Όλες οι μετρικές αξιολόγησης ενός ταξινομητή που παρουσιάστηκαν στα προηγούμενα υποκεφάλαια, αφορούσαν την περίπτωση ενός δυαδικού ταξινομητή. Οι μετρικές αυτές

μπορούν εύκολα να υπολογιστούν και για προβλήματα με περισσότερες από δύο κλάσεις, αν θεωρηθεί ως θετική η κλάση η οποία μας ενδιαφέρει να αξιολογήσουμε και ως αρνητικές όλες οι υπόλοιπες.

Τέλος, υπάρχουν και άλλες παρόμοιες μετρικές οι οποίες μπορούν να προκύψουν από τους όρους του πίνακα σύγχυσης (confusion matrix). Μια σύνοψη όλων των μετρικών παρουσιάζεται παρακάτω [27]:

		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Σχήμα 2.20: Σύνοψη μετρικών αξιολόγησης ενός ταξινομητή

2.2.9 Hyperparameter Tuning

Ο **συντονισμός υπερπαραμέτρων (hyperparameter tuning)** ή αλλιώς η **βελτιστοποίηση υπερπαραμέτρων (hyperparameter optimization)**, έχει αναφερθεί αρκετές φορές σε προηγούμενα κεφάλαια. Αφορά την επιλογή των βέλτιστων υπερπαραμέτρων για τον αλγόριθμο μάθησης. Με τον όρο υπερπαράμετροι, αναφερόμαστε στις παραμέτρους που χρησιμοποιούνται για να ελέγξουν την διαδικασία της εκμάθησης και όχι στις παραμέτρους οι οποίοι μαθαίνονται κατά την διάρκεια της εκπαίδευσης, όπως είναι τα βάρη. Τέτοιοι υπερπαράμετροι είναι: το ύψος του δέντρου σε ένα δέντρο απόφασης, η παράμετρος C στον soft margin SVM, ο kernel στον SVM με kernel, η παράμετρος α (learning rate) στον αλγόριθμο απότομης καθόδου (gradient descent) κ.α.

Grid Search

Η μέθοδος που ακολουθείται ώστε να βελτιστοποιηθούν οι υπερπαράμετροι ονομάζεται **grid search** και είναι μια διεξοδική αναζήτηση τιμών παραμέτρων, από ένα συγκεκριμένο υποσύνολο του συνόλου τιμών των παραμέτρων αυτών. Με λίγα λόγια, δοκιμάζονται συνδυασμοί υπερπαραμέτρων από μια γκάμα τιμών, εκπαιδεύοντας μοντέλα και στην συνέχεια επιλέγεται ο συνδυασμός που δίνει το μοντέλο με την καλύτερη απόδοση.

Για να γίνει πλήρως κατανοητό, θα δώσουμε ένα παράδειγμα. Έστω ότι έχουμε έναν soft margin SVM με kernel. Οι υπερπαράμετροι που πρέπει να βελτιστοποιηθούν ώστε να δώσουν καλύτερη απόδοση στο μοντέλο, είναι η παράμετρος C και ο kernel που θα χρησιμοποιηθεί. Έτσι επιλέγουμε δυο υποσύνολα τιμών για την καθεμία από αυτές τις υπερπαραμέτρους ως εξής: $C \in \{10, 100, 1000\}$ και $kernel \in \{linear, rbf\}$. Από αυτές τις τιμές προκύπτουν 6 συνδυασμοί οι οποίοι θα πρέπει να δοκιμαστούν. Με τον καθένα από τους συνδυασμούς αυτούς εκπαιδεύουμε ένα μοντέλο (SVM). Εδώ είναι σημαντικό να σημειώσουμε ότι στην περίπτωση που δεν γνωρίζουμε το πιθανό εύρος τιμών των υπερπαραμέτρων (παραδείγματος χάριν της παραμέτρου C), το πιο συχνό φαινόμενο είναι να χρησιμοποιήσουμε μια λογαριθμική κλίμακα.

Στην συνέχεια το καθένα από τα μοντέλα που προκύπτει, θα πρέπει να αξιολογηθεί με κάποια από τις μετρικές που αναφέραμε στο κεφάλαιο 2.2.8. Η αξιολόγηση αυτή μπορεί να γίνει είτε στα δεδομένα επικύρωσης (validation set), εάν υπάρχουν, είτε στα δεδομένα εκπαίδευσης (training set) με την μέθοδο **cross validation**, η οποία θα αναλυθεί παρακάτω.

Μετά την επιλογή του μοντέλου με τον συνδυασμό υπερπαραμέτρων που έδωσαν την καλύτερη απόδοση, αξιολογείται η συνολική απόδοση του μοντέλου (αναξάρτητα από τις υπερπαραμέτρους), στο σύνολο δοκιμής (test set).

Πέρα από τον grid search, υπάρχουν και άλλες τεχνικές βελτιστοποίησης υπερπαραμέτρων όπως: random search, bayesian hyperparameter optimization, gradient-based optimization, evolutionary optimization, population-based, early stopping-based κ.α. [28]. Ο grid search είναι μια αρκετά χρονοβόρα τεχνική αφού δοκιμάζει διεξοδικά όλους τους πιθανούς συνδυασμούς. Αντίθετα, άλλες τεχνικές είναι πιο αποτελεσματικές σε θέμα χρόνου όπως είναι για παράδειγμα ο random search, ο οποίος επιλέγει τυχαία τιμές υπερπαραμέτρων από μια στατιστική κατανομή και δημιουργεί τόσους συνδυασμούς όσους καθορίσουμε.

Cross Validation

Στην περίπτωση, λοιπόν, που δεν έχουμε σύνολο επικύρωσης, δεδομένου ότι αν τα δείγματα είναι πολύ λίγα, δεν μπορούμε να "χαραμίσουμε" δείγματα από το σύνολο εκπαίδευσης για να φτιάξουμε το σύνολο επικύρωσης, η τεχνική που μπορεί να χρησιμοποιηθεί για την αξιολόγηση της απόδοσης του μοντέλου είναι το **cross validation**. Ουσιαστικά, χρησιμοποιούμε διασταυρούμενη επικύρωση (cross validation) στο σύνολο εκπαίδευσης για να προσομοιάσουμε ένα σύνολο επικύρωσης.

Ένας κύκλος διασταυρούμενης επικύρωσης περιλαμβάνει τον διαχωρισμό ενός συνόλου δεδομένων σε συμπληρωματικά υποσύνολα, την εκτέλεση της ανάλυσης σε ένα υποσύνολο (που ονομάζεται σύνολο εκπαίδευσης) και την επικύρωση της ανάλυσης στο άλλο υποσύνολο (που ονομάζεται σύνολο επικύρωσης). Για να μειωθεί η μεταβλητότητα, στις περισσότερες μεθόδους εκτελούνται πολλαπλοί γύροι διασταυρούμενης επικύρωσης με χρήση διαφορετικών κατατμήσεων και τα αποτελέσματα επικύρωσης συνδυάζονται (π.χ. κατά μέσο όρο) στους γύρους για να δώσουν μια εκτίμηση της προγνωστικής απόδοσης του μοντέλου.

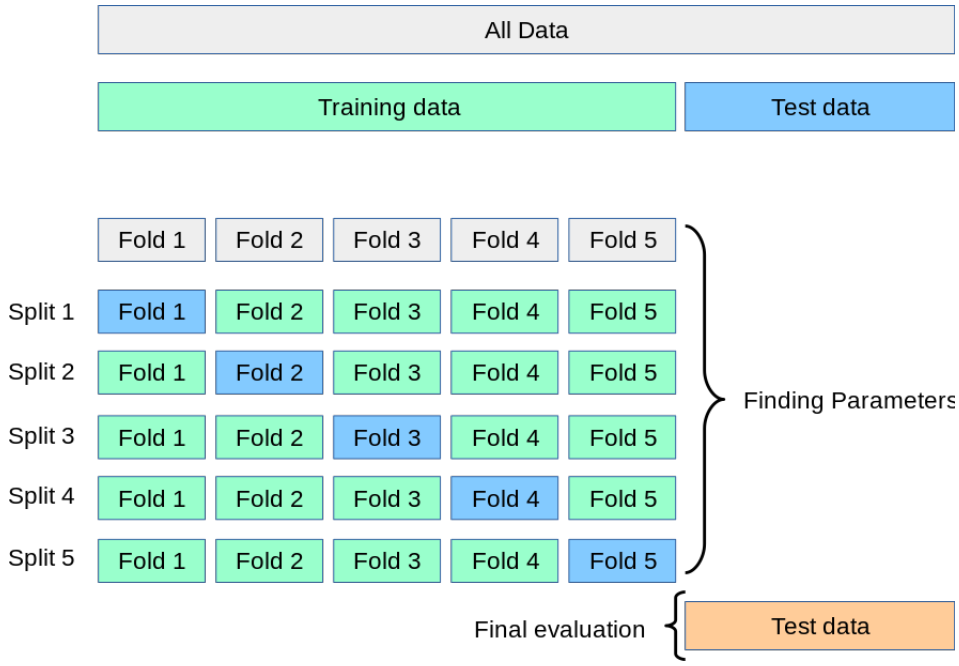
k-Fold Cross Validation

Ο τύπος διασταυρούμενη επικύρωση k-fold λειτουργεί ως εξής: αρχικά, καθορίζουμε τις τιμές των υπερπαραμέτρων που θέλουμε να αξιολογήσουμε. Στη συνέχεια, χωρίζουμε το σύνολο εκπαίδευσης σε πολλά υποσύνολα του ίδιου μεγέθους. Κάθε υποσύνολο ονομάζεται fold. Συνήθως, στην πράξη χρησιμοποιείται 5-fold cross-validation. Με 5-fold διασταυρούμενη επικύρωση, χωρίζουμε τυχαία τα δεδομένα εκπαίδευσης σε πέντε μέρη: $\{F_1, F_2, ..., F_5\}$. Κάθε F_k, k = 1,. . . , 5 περιέχει το 20% των δεδομένων εκπαίδευσης. Στη συνέχεια εκπαιδεύουμε πέντε μοντέλα ως εξής: για να εκπαιδεύσουμε το πρώτο μοντέλο, f_1, χρησιμοποιούμε όλα τα παραδείγματα από τα υποσύνολα F_2, F_3, F_4 και F_5 ως σύνολο εκπαίδευσης και τα παραδείγματα από το F_1 ως σύνολο επικύρωσης. Για να εκπαιδεύσουμε το δεύτερο μοντέλο, f_2, χρησιμοποιούμε τα παραδείγματα από τα υποσύνολα F_1, F_3, F_4 και F_5 ως σύνολο εκπαίδευσης και τα παραδείγματα από το F_2 ως σύνολο επικύρωσης. Συνεχίζουμε να δημιουργούμε μοντέλα με αυτόν τον τρόπο και υπολογίζουμε την τιμή της μετρικής που έχουμε επιλέξει για αξιολόγηση, σε κάθε σύνολο επικύρωσης, από F_1 έως F_5. Στη συνέχεια, παίρνουμε τον μέσο όρο των πέντε τιμών της μετρικής για να λάβουμε την τελική απόδοση του μοντέλου.

Η παραπάνω διαδικασία αφορά μόνο ένα μοντέλο, δηλαδή την δοκιμή μόνο ενός συν-

δυασμού υπερπαραμέτρων. Αν θέλουμε να δοκιμάσουμε πολλούς διαφορετικούς συνδυασμούς, επαναλαμβάνουμε το cross-validation για κάθε συνδυασμό. Δηλαδή, ουσιαστικά χρησιμοποιούμε grid search με cross-validation. Όταν τελικά, έχουμε επιλέξει τις υπερπαραμέτρους που μας έδωσαν την καλύτερη απόδοση μέσω διασταυρούμενης επικύρωσης, χρησιμοποιούμε όλο το σύνολο δεδομένων για να εκπαιδεύσουμε ξανά το μοντέλο και στην συνέχεια το αξιολογούμε στο σύνολο δοκιμής (test set).

Η παραπάνω διαδικασία παρουσιάζεται συνοπτικά στο σχήμα που ακολουθεί [29]:



Σχήμα 2.21: k-Fold Cross-Validation

Υπάρχουν διαφορετικοί τύποι διασταυρούμενης επικύρωσης. Αυτή που περιγράφηκε παραπάνω είναι η κανονική k-fold διασταυρούμενη επικύρωση **(regular k-fold cross validation)**, όπου το αρχικό σύνολο δεδομένων, διασπάται τυχαία σε k ίσα υποσύνολα. Ένας άλλος τύπος είναι η στρωματοποιημένη k-fold διασταυρούμενη επικύρωση **(stratified k-fold cross-validation)**, όπου τα μέρη στα οποία σπάμε το σύνολο, επιλέγονται έτσι ώστε η μέση τιμή απόκρισης να είναι περίπου ίση σε όλα τα μέρη. Δηλαδή, να υπάρχει περίπου ίσος αριθμός δειγμάτων από κάθε κλάση, σε όλα τα μέρη. Αν για παράδειγμα το πρόβλημα ήταν δυαδικό, τότε κάθε μέρος (fold), θα περιείχε περίπου τις ίδιες αναλογίες των δύο κλάσεων.

Και οι δύο παραπάνω τύποι ανήκουν στην κατηγορία **non-exhaustive cross-validation**. Σε αυτήν την κατηγορία δεν υπολογίζονται όλοι οι πιθανοί τρόποι διάσπασης του συνόλου δεδομένων, αλλά επιλέγεται μια τυχαία διάσπαση. Αντίθετα στην κατηγορία **exhaustive cross-validation** ανήκουν μέθοδοι διασταυρούμενης επικύρωσης που μαθαίνουν και δοκιμάζουν όλους τους πιθανούς τρόπους για να χωρίσουν το αρχικό σύνολο σε ένα σύνολο εκπαίδευσης και ένα σύνολο επικύρωσης [30].

2.3 Τεχνητά Νευρωνικά Δίκτυα

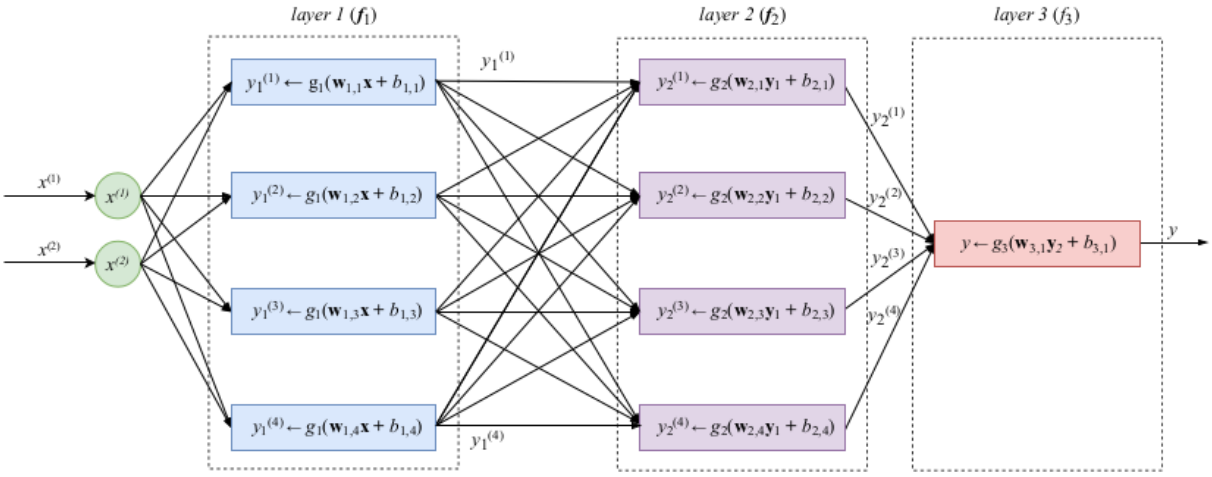
Σε αυτό το κεφάλαιο θα γίνει μια εισαγωγή στα τεχνητά νευρωνικά δίκτυα, ώστε να μπορούν να κατανοηθούν οι αλγόριθμοι που θα χρησιμοποιηθούν στην συνέχεια για την εξαγωγή χαρακτηριστικών από κείμενο, αλλά και τα νευρωνικά δίκτυα βαθιάς μηχανικής μάθησης (deep learning), όπως αυτό που περιγράφεται στο επόμενο κεφάλαιο.

Τα **τεχνητά νευρωνικά δίκτυα (artificial neural networks - ANN)** ή αλλιώς **νευρωνικά δίκτυα (neural networks - NN)**, προσομοιάζουν τα νευρικά δίκτυα ενός βιολογικού εγκεφάλου, εξού και το όνομα τους. Αποτελούνται από κάποια επίπεδα (layers), όπου κάθε επίπεδο περιέχει έναν αριθμό κόμβων ή αλλιώς τεχνητών νευρώνων που συμπεριφέρονται όπως οι βιολογικοί νευρώνες. Οι νευρώνες, ανταλλάσσουν μεταξύ τους σήματα, από το ένα επίπεδο στο επόμενο, μέσω κάποιων συνάψεων-ακμών. Ένας τεχνητός νευρώνας που λαμβάνει ένα σήμα στη συνέχεια το επεξεργάζεται και μπορεί να σηματοδοτήσει νευρώνες που συνδέονται με αυτόν. Το "σήμα" σε μια σύνδεση είναι ένας πραγματικός αριθμός και η έξοδος κάθε νευρώνα υπολογίζεται από κάποια μη γραμμική συνάρτηση του αθροίσματος των εισόδων του. Η κάθε ακμή σύνδεσης, συνήθως συνοδεύεται από κάποιο βάρος (weight), το οποίο προσαρμόζεται με την εκπαίδευση. Τα βάρη αυτά, μειώνουν ή αυξάνουν την ισχύ του σήματος που μεταδίδεται μέσω αυτής της ακμής. Επίσης, διαφορετικά επίπεδα (layers) μπορεί να εκτελούν διαφορετικούς μετασχηματισμούς στις εισόδους τους. Τα σήματα ταξιδεύουν από το πρώτο στρώμα-επίπεδο (το στρώμα εισόδου), στο τελευταίο στρώμα (το στρώμα εξόδου), πιθανώς αφού περάσουν τα στρώματα πολλές φορές.

Ο τρόπος με τον οποίο λειτουργεί το κάθε στρώμα, δεν είναι άλλος από αυτόν που έχουμε δει στην λογιστική παλινδρόμηση. Στην ουσία έχουμε την μαθηματική έκφραση $W_l x + b_l$, όπου W_l και b_l είναι οι παράμετροι για το εκάστοτε στρώμα l, οι οποίοι τίθενται προς βελτιστοποίηση κατά την διάρκεια εκπαίδευσης, μέσω του αλγορίθμου απότομης καθόδου και του κριτηρίου βελτιστοποίησης, όπως έχουμε ξαναδεί. Το W_l αποτελείται από τα βάρη $w_{l,u}$, όπου l είναι το εκάστοτε στρώμα και u είναι ο εκάστοτε νευρώνας-μονάδα του στρώματος αυτού. Το κάθε $w_{l,u}$ είναι τόσων διαστάσεων όσες και οι διαστάσεις των δειγμάτων εισόδου x. Δηλαδή, είναι ένας πίνακας (matrix) (αριθμός δειγμάτων x αριθμός χαρακτηριστικών). Έτσι κάθε στρώμα έχει τόσα βάρη όσοι και οι νευρώνες του. Το b_l αντίστοιχα αποτελείται από τα $b_{l,u}$ για κάθε νευρώνα u του στρώματος l και είναι ένα διάνυσμα (vector) , τόσων διαστάσεων όσα τα δείγματα εισόδου x_i. Στην συνέχεια αυτή η έκφραση περνάει από μία συνάρτηση ενεργοποίησης (activation function) g_l, η οποία είναι μια κλιμακωτή συνάρτηση (scalar function), δηλαδή επιστρέφει έναν απλό αριθμό και όχι ένα διάνυσμα, όπως στην περίπτωση της λογιστικής παλινδρόμησης, στην σχέση (2.5), που είχαμε την σιγμοειδή συνάρτηση. Τελικά, η έξοδος του στρώματος l είναι η $f_l(x) = g_l(W_l x + b_l)$, που είναι ένα διάνυσμα τόσων διαστάσεων όσοι είναι και οι νευρώνες-μονάδες στο στρώμα l.

Multilayer Perceptron

Για να γίνει πιο κατανοητή η αρχιτεκτονική ενός νευρωνικού δικτύου που περιγράφηκε παραπάνω, θα παρασταθεί γραφικά ένα συγκεκριμένο είδος αρχιτεκτονικής που ονομάζεται **πολυεπίπεδο perceptron (multilayer perceptron (MLP))** ή αλλιώς **νευρωνικό δίκτυο vanilla (vanilla neural network)** [13]:



Σχήμα 2.22: Παράδειγμα πολυεπίπεδου perceptron με δύο στρώματα

Στο σχήμα 2.22, βλέπουμε ένα MLP, στο οποίο τα δεδομένα εισόδου x είναι 2 διαστάσεων $x^{(1)}$ και $x^{(2)}$. Και τα δύο δείγματα τροφοδοτούνται στο πρώτο επίπεδο (layer 1) σε καθένα από τους νευρώνες (από 1 εώς 4). Σε κάθε νευρώνα υπολογίζεται αρχικά ο γραμμικός συνδυασμός των δειγμάτων εισόδου $w_{l,u}x + b_{l,u}$ και στην συνέχεια εφαρμόζεται η συνάρτηση ενεργοποίησης σε αυτόν $g_l(w_{l,u}x + b_{l,u})$. Από κάθε νευρώνα προκύπτει η έξοδος $y_1^{(u)}$, όπου u είναι ο δείκτης του νευρώνα. Όλοι οι έξοδοι αυτοί εισάγονται στους νευρώνες του επόμενου επιπέδου, όπου με αντίστοιχο τρόπο υπολογίζονται οι νέοι έξοδοι $y_2^{(u)}$, υπολογισμένοι στα νέα βάρη $w_{2,u}$ και $\beta_{2,u}$ και στις νέες συναρτήσεις ενεργοποίησης g_2. Τελικά, οι έξοδοι του 2ου επιπέδου, εισάγονται στο επίπεδου εξόδου το οποίο περιέχει έναν μόνο νευρώνα και με αντίστοιχες διαδικασίες εξάγει την έξοδο-πρόβλεψη y.

Είναι σημαντικό να αναφερθούν τα εξής:

- Ο αριθμός των νευρώνων μπορεί να διαφέρει από επίπεδο σε επίπεδο. Εδώ έτυχε να είχαν και τα δυο επίπεδα 4 νευρώνες.
- Τα ενδιάμεσα επίπεδα που δεν είναι ούτε εισόδου ούτε εξόδου, ονομάζονται κρυφά επίπεδα (hidden layers).
- Η συνάρτηση ενεργοποίησης, συνήθως είναι η ίδια για όλους τους νευρώνες ενός επιπέδου και μπορεί να διαφέρει ανά επίπεδο. Αυτό όμως δεν είναι κανόνας. Υπάρχει πιθανότητα να έχουμε διαφορετικές συναρτήσεις ενεργοποίησης και ανάμεσα σε νευρώνες του ίδιου επιπέδου.
- Υπάρχουν πολλές διαφορετικές συναρτήσεις ενεργοποίησης που μπορούν να χρησιμοποιηθούν, οι οποίες είναι συνήθως παραγωγίσιμες έτσι ώστε να μπορεί να εφαρμοστεί ο αλγόριθμος απότομης καθόδου (gradient descent) για να βρεθούν οι βέλτιστες τιμές των παραμέτρων w και b.
- Ο λόγος για τον οποίο χρησιμοποιούνται οι συναρτήσεις ενεργοποίησης είναι για να έχουμε μη γραμμικές συνιστώσες που επιτρέπουν στο νευρωνικό δίκτυο να προσεγγίσει μη γραμμικές συναρτήσεις. Ειδάλλως, οι σχέσεις $w_{l,u}x + b_{l,u}$ θα οδηγούσαν πάντα σε γραμμικούς συνδυασμούς.
- Στο πολυεπίπεδο perceptron, όλοι οι έξοδοι ενός επιπέδου συνδέονται με κάθε είσοδο του επόμενου (με κάθε νευρώνα). Αυτά τα επίπεδα ονομάζονται **πλήρως συνδεδεμένα επίπεδα (fully connected layers)**, ενώ ένα νευρωνικό δίκτυο δεν είναι απαραίτητο να απαρτίζεται μόνο από τέτοια.

- Για να ενημερώσουμε τις τιμές των παραμέτρων ενός νευρωνικού δικτύου, χρησιμοποιούμε τον αλγόριθμο εκπαίδευσης **backpropagation**, ο οποίος αποτελείται από δύο βήματα: 1) προώθηση των τιμών (feed forward) προς τα εμπρός 2) υπολογισμός του σφάλματος και μετάδοση του στα προηγούμενα επίπεδα. Στην ουσία, κατά την εκτέλεση του αλγορίθμου απότομης καθόδου (gradient descent), οι παράμετροι του νευρικού δικτύου λαμβάνουν μια ενημέρωση, ανάλογη με την μερική παράγωγο της συνάρτησης κόστους ως προς την τρέχουσα παράμετρο, σε κάθε επανάληψη της εκπαίδευσης. Στο backpropagation, οι παράγωγοι αυτοί υπολογίζονται από πίσω προς τα μπρος (από το τελευταίο επίπεδο προς το πρώτο), με τον κανόνα της αλυσίδας (chain rule).

Activation Functions

Οι πιο συχνά χρησιμοποιούμενες συναρτήσεις ενεργοποίησης είναι οι παρακάτω:

- **Softmax Function**
  Η συνάρτηση softmax εφαρμόζεται σε προβλήματα ταξινόμησης με πολλαπλές κλάσεις (περισσότερες από δύο). Στην περίπτωση πολλαπλών κλάσεων, στο στρώμα εξόδου που είδαμε παραπάνω, δεν θα υπήρχε μόνο ένας νευρώνας αλλά τόσοι νευρώνες όσες και οι κλάσεις προς πρόβλεψη. Η μαθηματική έκφραση της συνάρτησης αυτής έχει ως εξής:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.65}$$

  όπου το $\sigma(z)_i$ είναι η πιθανότητα το δείγμα να ανήκει στην κλάση i. Στην ουσία τα z_i είναι τα σκορ που προκύπτουν για κάθε κλάση. Εφαρμόζοντας την τυπική εκθετική συνάρτηση (exponential) στο σκορ της εκάστοτε κλάσης i και διαιρώντας με το άθροισμα των εκθετικών των σκορ όλων των κλάσεων, παίρνουμε τελικά μια τιμή στο διάστημα [0,1] η οποία αντικατοπτρίζει την πιθανότητα. Το άθροισμα όλων των $\sigma(z)_i$ για κάθε i, δηλαδή για κάθε κλάση, ισούται με 1. Αυτό σημαίνει ότι η συνολική πιθανότητα ταξινόμησης του δείγματος στις κλάσεις είναι 1.

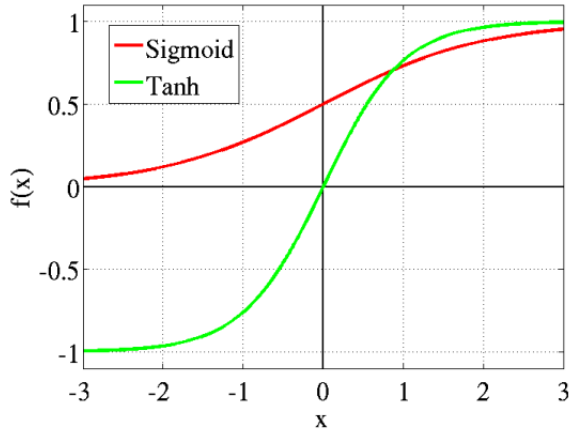
  Μια υποκατηγορία της συνάρτησης softmax είναι και η σιγμοειδής συνάρτηση ή αλλιώς λογιστική συνάρτηση ενεργοποίησης (sigmoid/logistic activation function), την οποία είδαμε στην περίπτωση της λογιστικής παλινδρόμησης. Αυτή η συνάρτηση χρησιμοποιείται για προβλήματα δυαδικής ταξινόμησης αφού έχουμε μονάχα μία πιθανότητα εξόδου που δείχνει την πιθανότητα το δείγμα να ταξινομηθεί ως θετικό:

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{2.66}$$

- **Tanh or Hyperbolic Tangent**
  Η συνάρτηση ενεργοποίησης υπερβολικής εφαπτομένης είναι παρόμοια με την σιγμοειδή με την διαφορά ότι έχει εύρος (-1,1).

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.67}$$

  Παρακάτω παρουσιάζεται η συνάρτηση tanh μαζί με την sigmoid/logistic [31]:

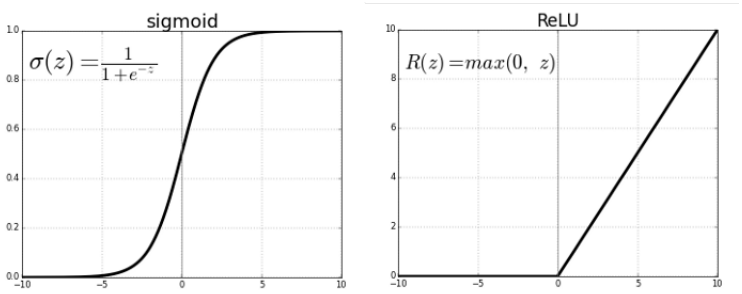


Σχήμα 2.23: Tanh vs Logistic/Sigmoid

- **ReLU (Rectified Linear Unit)**
  Η συνάρτηση διορθωμένης γραμμικής μονάδας είναι ίση με 0 όταν το σκορ z είναι αρνητικό, αλλιώς είναι ίση με το σκορ z:

$$relu(z) = \begin{cases} 0, & \text{if} \quad z < 0 \\ z, & \text{otherwise} \end{cases} \tag{2.68}$$

  Παρακάτω παρουσιάζεται η συνάρτηση relu μαζί με την sigmoid/logistic [31]:



Σχήμα 2.24: Relu vs Logistic/Sigmoid

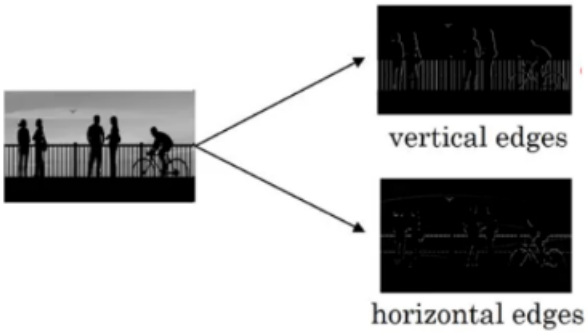
2.4 Συνελικτικά Νευρωνικά Δίκτυα

Η βαθιά μηχανική μάθηση (Deep Learning) αναφέρεται στην εκπαίδευση δικτύων που έχουν περισσότερα από 2 κρυφά στρώματα. Για αυτό και τα νευρωνικά δίκτυα αυτά ονομάζονται βαθιά νευρωνικά δίκτυα (deep neural networks).

Τα **συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Networks - CNN)** είναι ένα είδος τέτοιων νευρωνικών δικτύων τα οποία έχουν βρει εφαρμογή στους τομείς της ανάλυσης εικόνας και κειμένου. Επειδή, λοιπόν, τα CNNs εφευρέθηκαν με την ανάλυση εικόνας, θα εξηγηθούν πάνω σε ένα παράδειγμα τέτοιου είδους ταξινόμησης.

Edge Detection

Οι εικόνες αναπαρίστανται με pixels. Συνήθως, τα γειτονικά pixels αναφέρονται στην ίδια πληροφορία, για παράδειγμα αναπαριστούν ένα πρόσωπο, ένα σύννεφο, το νερό κ.α. Υπάρχουν όμως και τα άκρα (edges), τα σημεία δηλαδή όπου δύο διαφορετικά αντικείμενα αγγίζουν το ένα το άλλο. Αν, λοιπόν, το νευρωνικό δίκτυο μάθει να αναγνωρίζει τόσο περιοχές με την ίδια πληροφορία όσο και περιοχές άκρων, θα είναι μια χρήσιμη οδός που θα του επιτρέψει να προβλέψει το αντικείμενο που αναπαρίσταται στην εικόνα. [32]



Σχήμα 2.25: Τα άκρα μιας εικόνας

Κατ᾽ αυτόν τον τρόπο, η λογική πίσω από ένα CNN είναι ότι τα πρώτα στρώματα ενός νευρικού δικτύου εντοπίζουν άκρα από μια εικόνα. Τα βαθύτερα στρώματα μπορεί να είναι σε θέση να εντοπίσουν την ύπαρξη των αντικειμένων και ακόμη πιο βαθιά στρώματα να ανιχνεύουν την ύπαρξη των πλήρων αντικειμένων (όπως το πρόσωπο ενός ατόμου).

Τα άκρα (edges) μπορεί να είναι οριζόντια ή κατακόρυφα και διακρίνονται με την βοήθεια φίλτρων ή αλλιώς πυρήνων (filter/kernel), τα οποία είναι σε μορφή πινάκων. Τέτοια φίλτρα φαίνονται παρακάτω [32]:

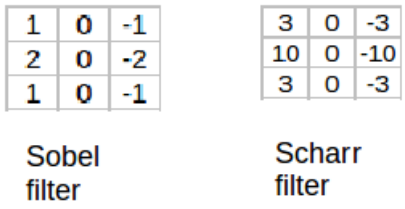
1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

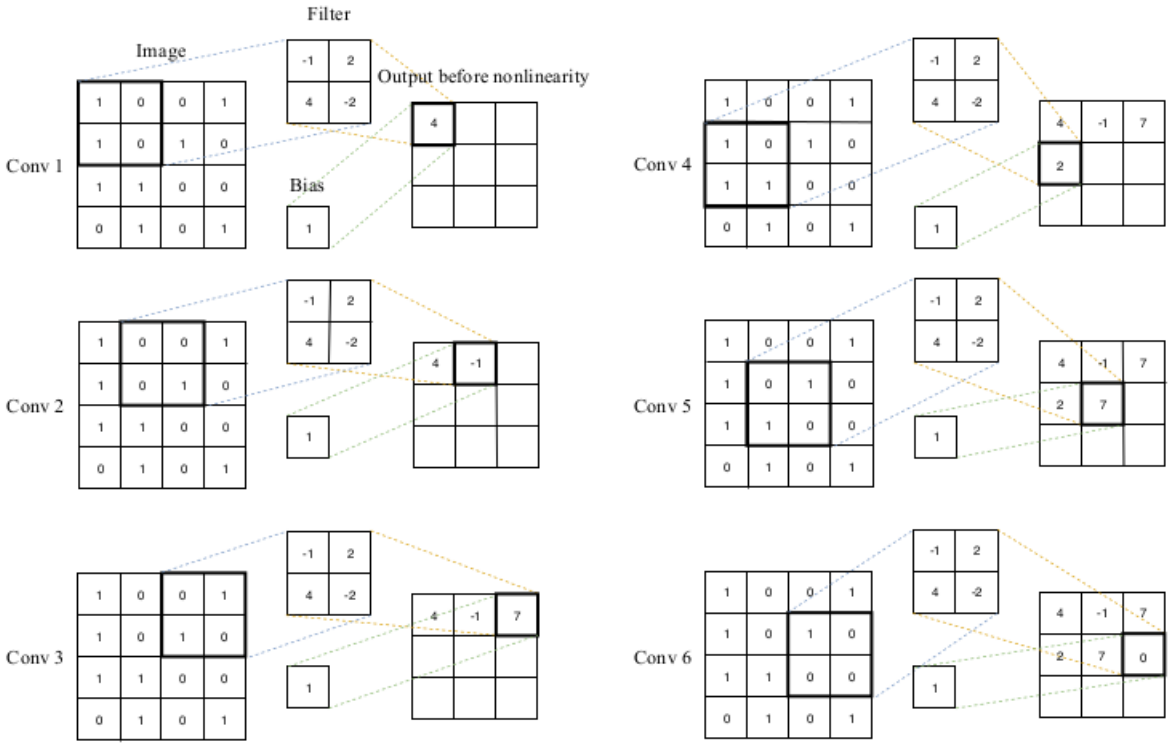
Horizontal

Σχήμα 2.26: Vertical and Horizontal Filter



Σχήμα 2.27: Sobel and Scharr Filter

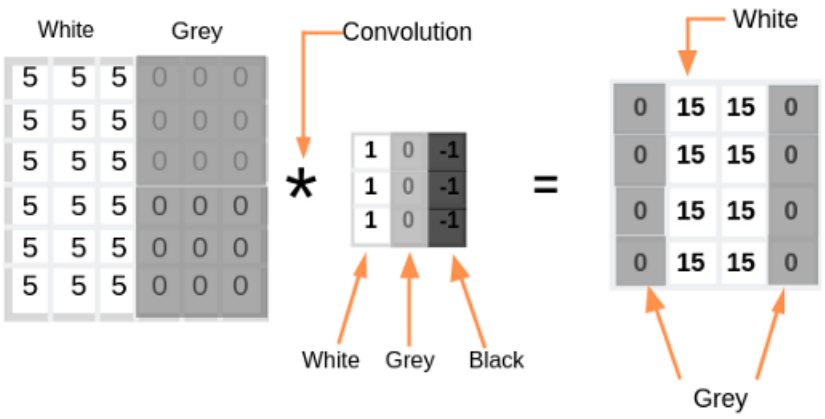
Ο τρόπος με τον οποίο τα φίλτρα αυτά εφαρμόζονται πάνω σε μία εικόνα, είναι κάνοντας μια συνέλιξη της αναπαράστασης της εικόνας σε pixels με το εκάστοτε φίλτρο. Αν για παράδειγμα έχουμε μια εικόνα διάστασης 4×4 και ένα φίλτρο 2×2, το αποτέλεσμα της συνέλιξης (convolution) θα είναι ένας πίνακας διάστασης 3×3. Αυτός ο πίνακας προκύπτει, παίρνοντας στην αρχή το πάνω αριστερά κομμάτι της εικόνας, διάστασης 2×2, πολλαπλασιάζοντάς το με το φίλτρο με γνώμονα το στοιχείο (element-wise) και προσθέτοντας τις τιμές των γινομένων αυτών. Από αυτές τις πράξεις θα προκύψει μόνο ένας αριθμός ο οποίος θα αποτελέσει το πρώτο στοιχείου του 3×3 πίνακα του αποτελέσματος. Στην συνέχεια, επαναλαμβάνουμε την ίδια διαδικασία, μετατοπίζοντας το φίλτρο ένα βήμα προς τα δεξιά, πάνω την αρχική εικόνα, δηλαδή παίρνοντας το αμέσως δεξιότερο κομμάτι 2×2 της εικόνας και εφαρμόζοντας τις ίδιες πράξεις. Οι επαναλήψεις αυτές λαμβάνουν τέλος όταν έχουμε καλύψει την πλήρη εικόνα, από δεξιά προς τα αριστερά και από πάνω προς τα κάτω, και έχει προκύψει ένας πίνακας 3×3. Κάθε φίλτρο συνοδεύεται επίσης και από μια bias παράμετρο, η οποία προστίθεται στο αποτέλεσμα της συνέλιξης κάθε βήματος. Ένα παράδειγμα των παραπάνω υπολογισμών με τις διαστάσεις που δώσαμε, παρουσιάζεται παρακάτω [13]:



Σχήμα 2.28: Συνέλιξη Φίλτρου κατά μήκος Εικόνας

Άλλο ένα παράδειγμα μπορεί να παρουσιαστεί αν στην είσοδο είχαμε μια εικόνα σε

γκρι κλίμακα (grayscale) όπου τα pixels με υψηλότερες τιμές είναι τα πιο φωτεινά μέρη της εικόνας και τα pixels με χαμηλότερες τιμές αναπαριστούν τα πιο σκοτεινά. Στην περίπτωση αυτή, φαίνεται στην παρακάτω εικόνα πως εφαρμόζοντας ένα φίλτρο μπορούν τα διακριθούν τα κατακόρυφα άκρα [33]:



Σχήμα 2.29: Κατακόρυφα Άκρα σε grayscale Εικόνα

Από τα παραπάνω προκύπτει ότι στην γενική περίπτωση, εάν έχουμε είσοδο εικόνας $n \times n$ και φίλτρο διάστασης $f \times f$ τότε η διάσταση της εξόδου θα είναι $(n-f+1) \times (n-f+1)$:

- **Input**: $n \times n$
- **Filter size**: $f \times f$
- **Output**: $(n-f+1) \times (n-f+1)$

Padding

Επειδή οι εικόνες με τον υπολογισμό της συνέλιξης συρρικνώνονται, δηλαδή μειώνεται το μέγεθός τους καθώς επίσης και επειδή τα pixels που παρουσιάζονται στις γωνίες μιας εικόνες, χρησιμοποιούνται λιγότερες φορές κατά την διάρκεια μιας συνέλιξης απ' ότι τα κεντρικά pixels, πράγμα που μπορεί να οδηγήσει σε απώλεια πληροφορίας, χρησιμοποιείται η μέθοδος padding. Με την μέθοδο αυτή, προστίθενται pixels γύρω από τα άκρα της εικόνας. Έτσι, για παράδειγμα, σε μια εικόνα μεγέθους 6×6, εάν εφαρμοστεί padding ενός pixel, η εικόνα θα μετατραπεί σε μέγεθος 8×8 και εφαρμόζοντας ένα φίλτρο 3×3, το αποτέλεσμα θα μας δώσει έναν πίνακα 6×6, όπως και το μέγεθος της αρχικής εικόνας. Οι διαστάσεις αυτές στην γενική περίπτωση υπολογίζονται ως εξής:

- **Input**: $n \times n$
- **Padding**: p
- **Filter size**: $f \times f$
- **Output**: $(n+2p-f+1) \times (n+2p-f+1)$

Το padding είναι προεραιτικό και μπορεί σε περιπτώσεις να μην εφαρμόζεται καθόλου, ενώ στην περίπτωση που θέλουμε ο πίνακας του αποτελέσματος μετά την συνέλιξη να έχει τις ίδιες διαστάσεις με την αρχική εικόνα, όπως είδαμε στο πιο πάνω παράδειγμα, τότε το padding που εφαρμόζεται είναι $p = (f-1)/2$.

Strides

Το stride είναι το μέγεθος που δείχνει τον αριθμό τον βημάτων κατά τα οποία μετακινούμε το φίλτρο πάνω στην εικόνα, σε κάθε επανάληψη της συνέλιξης. Τα βήματα αυτά εφαρμόζονται τόσο στην οριζόντια όσο και στην κατακόρυφη διεύθυνση. Οι τελικές διαστάσεις έπειτα από την εφαρμογή και του stride υπολογίζονται ως εξής:

- **Input**: $n \times n$
- **Padding**: p
- **Stride**: s
- **Filter size**: $f \times f$
- **Output**: $[(n + 2p - f)/s + 1] \times [(n + 2p - f)/s + 1]$

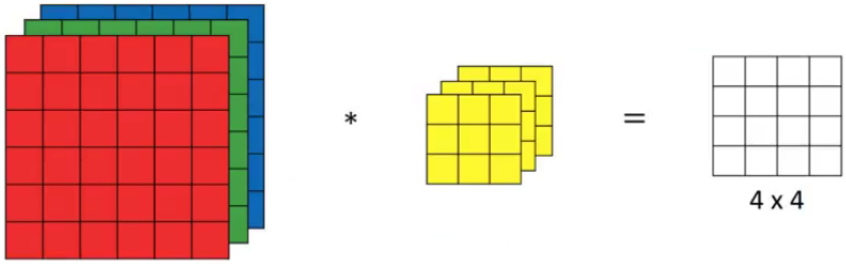
Convolution over Volume

Στα παραδείγματα που είδαμε μέχρι τώρα, οι εικόνες ήταν 2 διαστάσεων (grayscale). Υπάρχουν όμως περιπτώσεις όπου οι εικόνες είναι τριών διαστάσεων, με την τρίτη διάσταση να αναπαριστά το κανάλι (channel). Οι εικόνες κωδικοποιούνται σε κανάλια χρώματος. Η πιο συνηθισμένη απεικόνιση είναι η RGB, που σημαίνει Red-Κόκκινο , Blue-Μπλε και Green-Πράσινο. Οι πληροφορίες που περιέχονται σε μια εικόνα είναι η ένταση κάθε χρώματος καναλιού στο πλάτος και το ύψος της εικόνας.

Η συλλογή μεγέθους l (τρίτη διάσταση) των πινάκων εικόνας (2 διαστάσεις), ονομάζεται **volume**.

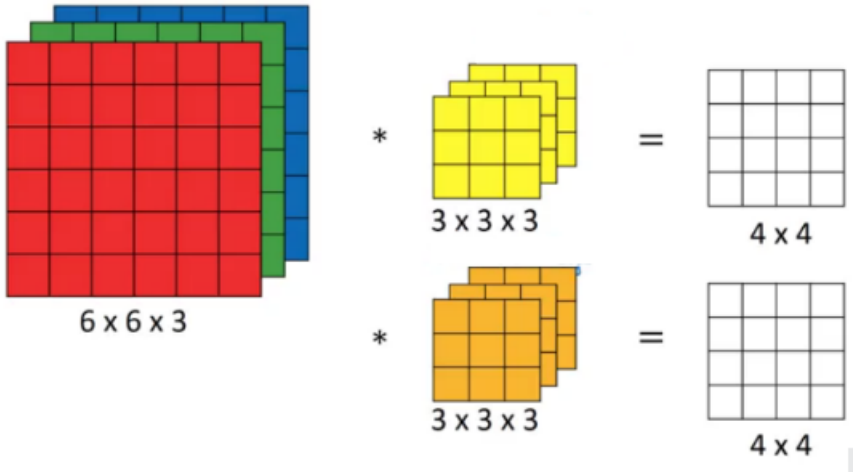
Έτσι αν για παράδειγμα έχουμε μια εικόνα μεγέθους $6 \times 6 \times 3$ (l = 3), τότε θα χρησιμοποιήσουμε ένα φίλτρο το οποίο θα έχει επίσης τρεις διαστάσεις π.χ. $3 \times 3 \times 3$. Ο αριθμός των καναλιών δηλαδή, η τρίτη διάσταση, θα πρέπει να είναι η ίδια για την εικόνα εισόδου και το φίλτρο. Στην ουσία χρησιμοποιείται το ίδιο φίλτρο 3 φορές, μια για κάθε κανάλι ξεχωριστά.

Στο παράδειγμα αυτό, ο πίνακας αποτελέσματος θα είναι και πάλι 4×4, όπως και στην περίπτωση που η εικόνα ήταν δύο διαστάσεων. Για τον υπολογισμό του πρώτου στοιχείου του πίνακα αυτού (σειρά 1 και στήλη 1), υπολογίζεται το element-wise γινόμενο μεταξύ του φίλτρου (3×3) και του 3×3 πάνω αριστερά κομματιού της εικόνας, για κάθε κανάλι. Αυτό θα έχει ως αποτέλεσμα 9 τιμές για κάθε κανάλι, δηλαδή 27 τιμές συνολικά. Αθροίζοντας αυτές τις τιμές, παίρνουμε το πρώτο στοιχείο του πίνακα εξόδου. Η ίδια διαδικασία ακολουθείται μετακινώντας το φίλτρο από αριστερά προς τα δεξιά και από πάνω προς τα κάτω, πάνω στην εικόνα, όπως ήδη ξέρουμε. Ουσιαστικά, γίνονται οι ίδιοι υπολογισμοί με πριν, μόνο που αυτήν την φορά διεξάγονται 3 φορές για κάθε κανάλι ξεχωριστά και τα αποτελέσματα αθροίζονται μεταξύ τους. Η αναπαράσταση αυτή φαίνεται παρακάτω [32]:



Σχήμα 2.30: Συνέλιξη πάνω σε Volume

Μέχρι τώρα στα παραδείγματά μας, χρησιμοποιούσαμε ένα μοναδικό φίλτρο το οποίο στην παραπάνω περίπτωση εφαρμόζεται σε κάθε κανάλι. Αντί, όμως, να χρησιμοποιήσουμε μόνο ένα φίλτρο, υπάρχει η πιθανότητα να χρησιμοποιήσουμε περισσότερα. Για παράδειγμα, μπορούμε να χρησιμοποιήσουμε 2 φίλτρα, ένα για την ανίχνευση οριζόντιων άκρων/ακμών και ένα για την ανίχνευση κατακόρυφων. Σε αυτήν την περίπτωση η έξοδος θα είναι τριών διαστάσεων με την τρίτη διάσταση να είναι ίση με τον αριθμό των φίλτρων. Έτσι, στο παράδειγμα που είδαμε παραπάνω, η έξοδος αντί για 4×4 θα γίνει $4\times4\times2$. Η νέα αυτή αναπαράσταση φαίνεται παρακάτω [32]:



Σχήμα 2.31: Συνέλιξη σε Volume με δύο Φίλτρα

Τελικά, οι γενικευμένες διαστάσεις δίνονται ως εξής:

- **Input**: $n \times n \times n_c$
- **Filter size**: $f \times f \times n_c$
- **Padding**: p
- **Stride**: s
- **Output**: $[(n+2p-f)/s+1] \times [(n+2p-f)/s+1] \times n'_c$

όπου το $\boldsymbol{n_c}$ είναι ο αριθμός των καναλιών της εικόνας εισόδου και $\boldsymbol{n'_c}$ είναι ο αριθμός των φίλτρων.

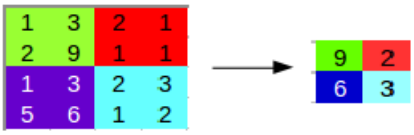
CNN layers

Η διαδικασία που περιγράφηκε μέχρι στιγμής, της συνέλιξης μιας εικόνας με τα φίλτρα, αποτελεί ένα μόνο layer του νευρωνικού δικτύου το οποίο ονομάζεται **convolutional layer**. Τέτοια στρώματα (layers), μπορούν να υπάρχουν διαδοχικά, το ένα μετά το άλλο, μέσα σε ένα CNN και έτσι η έξοδος του ενός να αποτελεί την είσοδο του επόμενου.

Πέραν όμως αυτού του είδους στρώματων, σε ένα CNN μπορούμε να συναντήσουμε και τα στρώματα που ονομάζονται **pooling layers**, τα οποία παρεμβάλλονται μεταξύ των convolutional layers. Τα στρώματα αυτά χρησιμοποιούνται για να μειώσουν το μέγεθος της εισόδου και έτσι να αυξήσουν την υπολογιστική ταχύτητα.

Αν για παράδειγμα έχουμε έναν πίνακα 4×4 και εφαρμόσουμε max pooling μεγέθους 2 και διασκελισμού (stride) 2, τότε το αποτέλεσμα θα είναι ένας πίνακας 2×2. Pooling μεγέθους 2 και διασκελισμού 2, σημαίνει πως εφαρμόζουμε ένα φίλτρο 2×2 πάνω στον πίνακα εισόδου και το μετακινούμε με βήμα 2 όπως ακριβώς συμβαίνει και με φίλτρα/πυρήνες που έχουμε δει μέχρι τώρα. Max pooling σημαίνει πως για κάθε παράθυρο 2×2 του πίνακα

εισόδου, πάνω από το οποίο περνάμε το φίλτρο pooling, λαμβάνουμε την μέγιστη τιμή που εμφανίζεται μέσα στο παράθυρο αυτό. Ένα τέτοιο παράδειγμα παρουσιάζεται παρακάτω [32]:



Σχήμα 2.32: Εφαρμογή Max Pooling Μεγέθους 2 και Βήματος 2

Πέραν του max pooling, μια άλλη μέθοδος που μπορεί να εφαρμοστεί είναι το average pooling το οποίο αντί για την μέγιστη τιμή εντός του παραθύρου, υπολογίζει τον μέσο όρο.

Ως προς τις διαστάσεις για την είσοδο του pooling φίλτρου, το ίδιο το pooling φίλτρο και την έξοδο, έχουμε τα εξής:

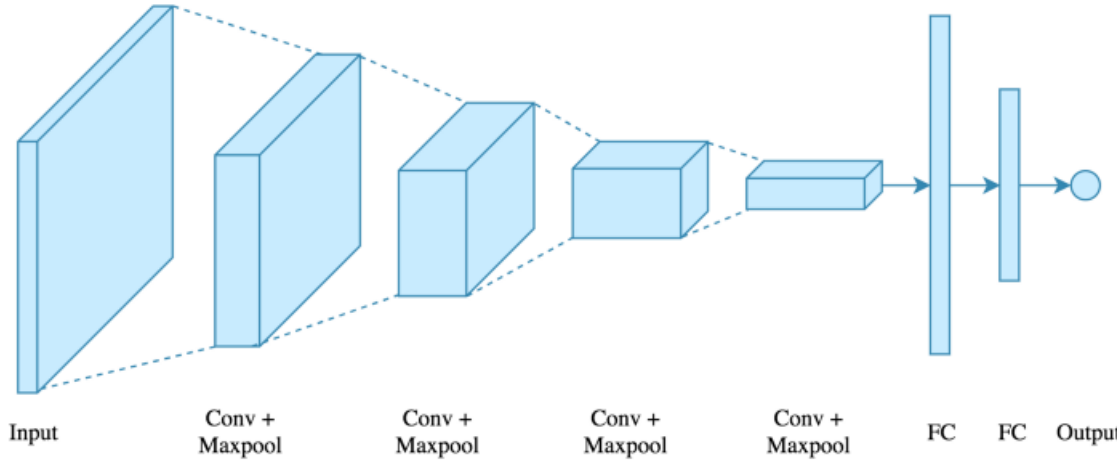
- **Input**: $n_h \times n_w \times n_c$
- **Filter Size of Pooling**: $f \times f \times n_c$
- **Stride of Pooling**: s
- **Output**: $[(n_h - f)/s + 1] \times [(n_w - f)/s + 1] \times n_c$

Μετά από τα convolutional και pooling στρώματα του νευρωνικού, εφαρμόζεται ένας αριθμός από στρώματα που ονομάζονται fully connected layers. Τέτοια στρώματα είναι της μορφής που είδαμε στα τεχνητά νευρωνικά δίκτυα (2.3). Το πλήρως συνδεδεμένο στρώμα περιλαμβάνει βάρη, παραμέτρους bias και νευρώνες. Συνδέει τους νευρώνες ενός στρώματος με τους νευρώνες ενός άλλου στρώματος. Χρησιμοποιείται για την ταξινόμηση εικόνων μεταξύ διαφορετικών κατηγοριών, έπειτα από την εκπαίδευσή του. Έτσι, αν για παράδειγμα το στρώμα k περιέχει 3 νευρώνες και το στρώμα k+1 4, τότε τα δύο στρώματα αυτά συνδέονται μεταξύ τους με ένα fully connected layer που είναι ένας πίνακας βαρών διάστασης 3×4. Σημειώνεται πως η έξοδος τόσο των convolutional όσο και των pooling στρωμάτων είναι τρισδιάστατα volumes, αλλά ένα fully connected στρώμα αναμένει ένα 1D διάνυσμα αριθμών. Έτσι ισοπεδώνουμε (flatten) την έξοδο του τελικού στρώματος pooling σε ένα φορέα και αυτό γίνεται η είσοδος του ακόλουθου πλήρως συνδεδεμένου στρώματος. Η ισοπέδωση απλώς τοποθετεί τον τρισδιάστατο όγκο αριθμών σε ένα 1D διάνυσμα.

Συνηθίζεται να χρησιμοποιούνται πολλαπλά fully connected layers στο τέλος του CNN, τα οποία ολοένα και μειώνουν την διάσταση πριν το τελικό layer το οποίο δίνει στην έξοδό του την τελική πρόβλεψη, διάστασης όσες και οι κλάσεις.

Μετά το fully connected layer, μπορεί φυσικά να εφαρμοστεί μια συνάρτηση ενεργοποίησης όπως η softmax για πολλαπλές κλάσεις ή η sigmoid για δυαδικό πρόβλημα ταξινόμησης, ώστε τα προβλεπόμενα σκορ να μετατραπούν σε πιθανότητες (2.3).

Ένα παράδειγμα ενός cnn με τα layers από τα οποία αποτελείται μπορεί να παρατηρηθεί παρακάτω [34]:

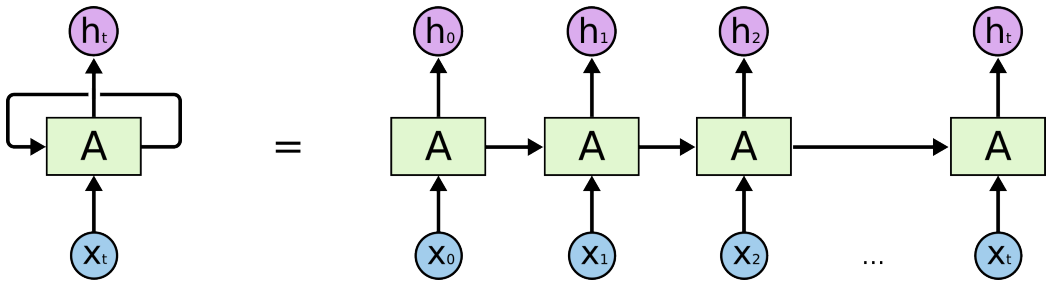


Σχήμα 2.33: Παράδειγμα Στρωμάτων CNN

2.5 Επαναλαμβανόμενα Νευρωνικά Δίκτυα

Τα επαναλαμβανόμενα νευρωνικά δίκτυα (recurrent neural networks - RNN) είναι μια άλλη κατηγορία τεχνητών νευρωνικών δικτύων. Η διαφορά τους από τις υπόλοιπες κατηγορίες και ταυτόχρονα το μεγάλο τους προσόν είναι η ικανότητα να διαχειρίζονται ακολουθιακά δεδομένα, δηλαδή δεδομένα που έχουν μια χρονικά ακολουθιακή σύνδεση, όπως είναι τα καρέ ενός βίντεο και οι λέξεις σε ένα κείμενο. Για τον λόγω αυτόν τα RNNs χρησιμοποιούνται συχνά στην επεξεργασία κειμένου και ομιλίας.

Ο τρόπος με τον οποίο τα δίκτυα αυτά αξιοποιούν την ακολουθιακή πληροφορία, είναι μέσω της έννοιας της μνήμης. Είναι δίκτυα που χρησιμοποιούν βρόχους (loops), επιτρέποντας έτσι να διατηρηθούν οι πληροφορίες στον χρόνο. Ένα επαναλαμβανόμενο νευρικό δίκτυο μπορεί να θεωρηθεί ως πολλαπλά αντίγραφα του ίδιου δικτύου, το καθένα από τα οποία διαβιβάζει ένα μήνυμα σε έναν χρονικό διάδοχο. ΄΄Έτσι, αν ξετυλίξουμε τον βρόχο παίρνουμε την μορφή που φαίνεται στην παρακάτω εικόνα [35]:



Σχήμα 2.34: Βρόχος σε ένα RNN

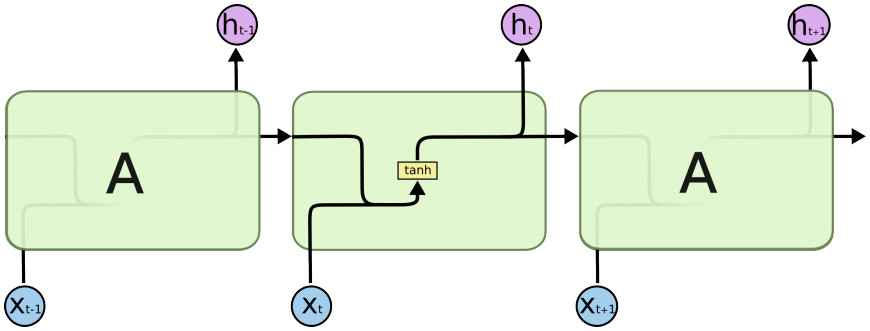
Με αυτόν τον τρόπο, εάν θέλουμε, για παράδειγμα, να μεταφράσουμε κάποιο κείμενο, μπορούμε να ορίσουμε κάθε είσοδο ως μια λέξη σε αυτό το κείμενο. Το RNN μεταβιβάζει τις πληροφορίες των προηγούμενων λέξεων στο επόμενο δίκτυο που μπορεί να χρησιμοποιήσει και να επεξεργαστεί αυτές τις πληροφορίες.

2.5.1 Τρόπος Λειτουργίας

Η έννοια της μνήμης στο επαναλαμβανόμενο νευρωνικό δίκτυο, δίνεται από την "κρυφή κατάσταση" (hidden state). Κάθε έξοδος του δικτύου δεν εξαρτάται μόνο από δεδομένα της εισόδου εκείνη την χρονική στιγμή και τις παραμέτρους του δικτύου αλλά και από ένα διάνυσμα κρυφής κατάστασης που αναπαριστά όλες τις παρελθοντικές εισόδους στο

δίκτυο. Επιπλέον, ο υπολογισμός της επόμενης κρυφής κατάστασης εξαρτάται από την είσοδο και την προηγούμενη κρυφή κατάσταση.

Παρακάτω φαίνεται η δομή ενός μόνο νευρώνα του δικτύου, για διαφορετικές χρονικές στιγμές (x_{t-1},x_t,x_{t+1}) [35]:



Σχήμα 2.35: Δομή Νευρώνα RNN

Ο υπολογισμός της κρυφής κατάσταση h_t αλλά και της εξόδου y_t του συγκεκριμένου νευρώνα για την χρονική στιγμή t δίνεται από τις παρακάτω σχέσεις:

$$h_t = \sigma_h(W_h x_t + U_h h_{t1} + b_h) \tag{2.69}$$

$$y_t = \sigma_y(W_y h_t + b_y) \tag{2.70}$$

όπου $\boldsymbol{x_t}$ είναι η είσοδος, $\boldsymbol{\sigma_h}$ και $\boldsymbol{\sigma_y}$ είναι συναρτήσεις ενεργοποίησης (στην εικόνα 2.35 είναι η σ_h = tanh), $\boldsymbol{W_h}$ $\boldsymbol{W_y}$ $\boldsymbol{U_h}$ $\boldsymbol{b_h}$ $\boldsymbol{b_y}$ είναι διανύσματα παραμέτρων-βαρών και $\boldsymbol{h_{t-1}}$ είναι η κρυφή κατάσταση της προηγούμενης χρονικής στιγμής.

Σημειώνεται πως ο παραπάνω νευρώνας, μπορεί να αποτελεί νευρώνα ενός μόνο στρώματος του επαναλαμβανόμενου νευρωνικού δικτύου και το δίκτυο να αποτελείται από πολλά στρώματα. Σε αυτήν την περίπτωση, το διάνυσμα εισόδου του νευρώνα προέρχεται από την έξοδο του προηγούμενου στρώματος, ενώ το διάνυσμα κρυφής κατάστασης από το ίδιο στρώμα από μια προηγούμενη χρονική στιγμή.

Θεωρητικά, τα RNNs είναι απολύτως ικανά να χειριστούν «μακροπρόθεσμες εξαρτήσεις», δηλαδή εξαρτήσεις από εισόδους που βρίσκονται πολύ μακρυά στον χρόνο ή αλλιώς αποτελούν πολύ μακρινούς προγόνους. Στην πράξη, τα RNNs καθίστανται πολύ αναποτελεσματικά όταν το χάσμα μεταξύ των σχετικών πληροφοριών και του σημείου όπου χρειάζονται είναι πολύ μεγάλο. Αυτό οφείλεται στο γεγονός ότι οι πληροφορίες διαβιβάζονται σε κάθε βήμα και όσο μεγαλύτερη είναι η αλυσίδα, τόσο πιθανότερο είναι οι πληροφορίες να χαθούν κατά μήκος της αλυσίδας. Το πρόβλημα διερευνήθηκε σε βάθος από τους Hochreiter (1991) [German] και Bengio, et al. (1994), οι οποίοι βρήκαν μερικούς πολύ θεμελιώδεις λόγους για τους οποίους μπορεί να συμβαίνει αυτό.

Τα LSTMs, ένας ειδικός τύπος νευρωνικών δικτύων RNN προσπαθούν να λύσουν αυτό το πρόβλημα της μακροπρόθεσμης εξάρτησης.

2.5.2 Δίκτυα Μακράς Βραχυπρόθεσμης Μνήμης

Τα **δίκτυα μακράς βραχυπρόθεσμης μνήμης (Long Short Term Memory - LSTM)** είναι ένα ειδικό είδος RNN, ικανό να μάθει μακροχρόνιες εξαρτήσεις. Δημιουργήθηκαν από τους Hochreiter & Schmidhuber (1997) [36] και βελτιώθηκαν και διαδόθηκαν από

πολλούς ανθρώπους. Λειτουργούν εξαιρετικά καλά σε μια μεγάλη ποικιλία προβλημάτων και χρησιμοποιούνται πλέον ευρέως.

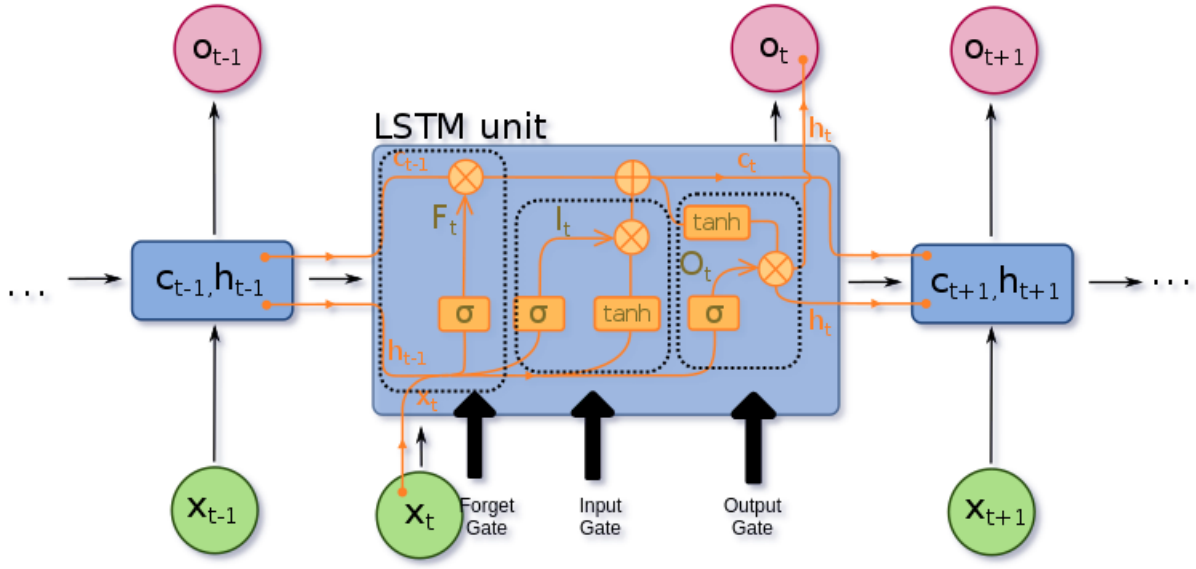
Τα LSTM έχουν σχεδιαστεί ρητά για την αποφυγή του προβλήματος μακροπρόθεσμης εξάρτησης. Η απομνημόνευση πληροφοριών για μεγάλα χρονικά διαστήματα είναι ουσιαστικά η προεπιλεγμένη συμπεριφορά τους, όχι κάτι που δυσκολεύονται να μάθουν.

Εκτός από την κρυφή κατάσταση (h_t), η οποία μπορεί να θεωρηθεί ως η βραχυπρόθεσμη μνήμη, χρησιμοποιούν και μία ακόμα κατάσταση η οποία ονομάζεται κατάσταση κυττάρου (cell state) (c_t), που μπορεί να θεωρηθεί ως η μακροπρόθεσμη μνήμη. Η κατάσταση κυττάρου τρέχει κατά μήκος ολόκληρης της αλυσίδας, με μόνο μερικές μικρές γραμμικές αλληλεπιδράσεις. Έτσι, είναι πολύ εύκολο για τις πληροφορίες να ρέουν αμετάβλητες.

Στην εσωτερική δομή τους τα LSTMs, αποτελούνται από τρεις πύλες:

- Την πύλη άγνοιας (forget gate), η οποία παίρνει ως είσοδο την κατάσταση κυττάρου της προηγούμενης χρονικής στιγμής c_{t-1} και ξεχνάει οτιδήποτε δεν είναι χρήσιμο.
- Την πύλη εισόδου (input gate), η οποία παίρνει ως είσοδο την κρυφή κατάσταση της προηγούμενης χρονικής στιγμής h_{t-1} αλλά και την είσοδο του νευρώνα x_t που είναι η καινούρια πληροφορία και τις τοποθετεί μαζί.
- Την πύλη εξόδου (output gate), η οποία παίρνει ως είσοδο την πληροφορία που μόλις μάθαμε μαζί με την πληροφορία της μακροπρόθεσμης μνήμης (κατάσταση κυττάρου) που δεν έχουν ακόμα ξεχάσει και τις χρησιμοποιεί για να κάνει μία πρόβλεψη, δηλαδή να παράγει την έξοδο και να ενημερώσει την βραχυπρόθεσμη μνήμη (κρυφή κατάσταση).

Στο παρακάτω σχήμα φαίνεται αναλυτικά η εσωτερική δομή του LSTM καθώς και των πυλών από τις οποίες αποτελείται [37]:



Σχήμα 2.36: Δομή Νευρώνα LSTM

Αναλυτικά οι μαθηματικές πράξεις που συνοδεύουν την παραπάνω αρχιτεκτονική είναι οι εξής:

$$F_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{2.71}$$

$$I_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{2.72}$$

$$I'_t = \sigma'_i(W'_i x_t + U'_i h_{t-1} + b'_i) \tag{2.73}$$

$$O_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{2.74}$$

Η νέα κατάσταση κυττάρου c_t, η οποία θα διαδοθεί στον ίδιο νευρώνα, στην επόμενη χρονική στιγμή της ακολουθίας, είναι ουσιαστικά ό,τι δεν ξεχνάμε από την προηγούμενη κατάσταση και ό,τι μάθαμε από την καινούρια πληροφορία. Ορίζεται από την παρακάτω σχέση:

$$c_t = F_t \otimes c_{t-1} + I_t \otimes I'_t \tag{2.75}$$

Η έξοδος του νευρώνα o_t αλλά και η νέα κρυφή κατάσταση h_t που θα διαδοθεί στον ίδιο νευρώνα, στην επόμενη χρονική στιγμή της ακολουθίας ορίζεται ως εξής:

$$o_t = h_t = O_t \otimes \sigma_h(c_t) \tag{2.76}$$

Στις παραπάνω σχέσεις, όλα τα W,U και b αποτελούν παραμέτρους-βάρη που μαθαίνει το μοντέλο κατά την εκπαίδευση. Τα σ είναι συναρτήσεις ενεργοποίησης που στην προκειμένη περίπτωση στο σχήμα 2.36, τα σ_g και σ_h είναι η σιγμοειδής συνάρτηση, ενώ το σ'_i είναι η συνάρτηση tanh. Αυτές οι συναρτήσεις ενεργοποίησης μπορεί βέβαια να έχουν διάφορες παραλλαγές.

Εν κατακλείδι, το ίδιο πρόβλημα που συμβαίνει γενικά με τα RNNs, συμβαίνει και με τα LSTMs, δηλαδή όταν οι προτάσεις είναι πολύ μεγάλες, τα LSTMs εξακολουθούν να μην είναι πολύ αποτελεσματικά. Ο λόγος για αυτό είναι ότι η πιθανότητα διατήρησης του περιβάλλοντος από μια λέξη που απέχει πολύ από την τρέχουσα λέξη υπό επεξεργασία μειώνεται εκθετικά με την απόσταση από αυτήν. Αυτό σημαίνει ότι όταν για παράδειγμα τα δεδομένα είναι κείμενα και οι προτάσεις είναι μεγάλες, το μοντέλο ξεχνά συχνά το περιεχόμενο των απομακρυσμένων θέσεων στην ακολουθία. Ένα άλλο πρόβλημα με τα RNNs και τα LSTMs είναι ότι είναι δύσκολο να παραλληλιστεί η εργασία για την επεξεργασία προτάσεων, καθώς πρέπει να επεξεργαστούν λέξη προς λέξη.

2.6 Transformers

Οι Transformers είναι μία αρχιτεκτονική τεχνητών νευρωνικών δικτύων που δημιουργήθηκε το 2017. Χρησιμοποιούνται κυρίως στην ανάπτυξη συστημάτων στον τομέα της επεξεργασίας φυσικής γλώσσας, αλλά πρόσφατη έρευνα έχει επίσης αναπτύξει την εφαρμογή τους σε άλλες εργασίες όπως η κατανόηση βίντεο. Οι Transformers αναπτύχθηκαν για την αντιμετώπιση ζητημάτων που σχετίζονται με το πρόβλημα της μεταγωγής ακολουθίας (sequence transduction) ή της μετάφρασης νευρικών μηχανών. Αυτό σημαίνει κάθε εργασία που μετατρέπει μια ακολουθία εισόδου σε μια ακολουθία εξόδου. Αυτό περιλαμβάνει αναγνώριση ομιλίας, μετατροπή κειμένου σε ομιλία, μετάφραση κ.λπ.

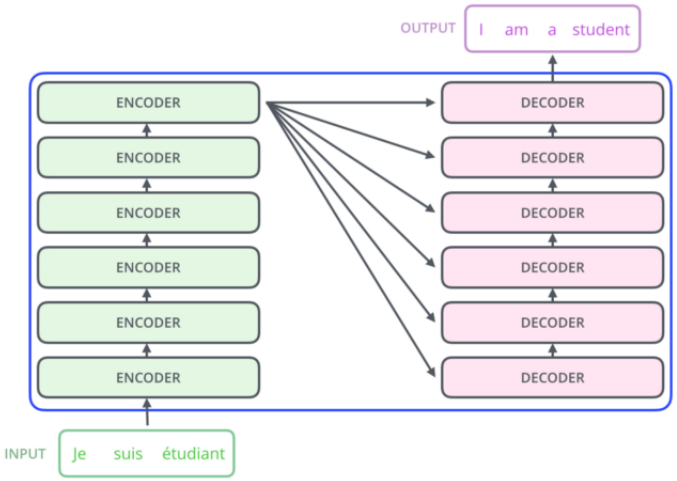
Όπως τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN), οι μετασχηματιστές (transformers) έχουν σχεδιαστεί για να χειρίζονται διαδοχικά δεδομένα εισόδου, όπως η φυσική γλώσσα, για εργασίες όπως μετάφραση και σύνοψη κειμένου. Ωστόσο, σε αντίθεση με τα RNN, οι μετασχηματιστές δεν απαιτούν την επεξεργασία των διαδοχικών δεδομένων

με τη σειρά. Αντίθετα, η λειτουργία προσοχής (attention mechanism) που χρησιμοποιούν, προσδιορίζει το πλαίσιο για οποιαδήποτε θέση στην ακολουθία εισόδου. Για παράδειγμα, εάν τα δεδομένα εισαγωγής είναι μια πρόταση φυσικής γλώσσας, ο μετασχηματιστής δεν χρειάζεται να επεξεργαστεί την αρχή της πριν από το τέλος. Αντίθετα, προσδιορίζει το πλαίσιο που αποδίδει νόημα σε μια λέξη στην πρόταση. Λόγω αυτής της δυνατότητας, ο μετασχηματιστής επιτρέπει πολύ μεγαλύτερο παραλληλισμό από τα RNNs και επομένως μειώνει τους χρόνους εκπαίδευσης.

Οι μετασχηματιστές έχουν γίνει γρήγορα το μοντέλο επιλογής για προβλήματα NLP, αντικαθιστώντας παλαιότερα επαναλαμβανόμενα μοντέλα νευρωνικών δικτύων όπως τα LSTM. Δεδομένου ότι το μοντέλο Transformer διευκολύνει περισσότερο τον παραλληλισμό κατά τη διάρκεια της εκπαίδευσης, επέτρεψε την εκπαίδευση σε μεγαλύτερα σύνολα δεδομένων από ό,τι ήταν δυνατό πριν. Αυτό οδήγησε στην ανάπτυξη προεκπαιδευμένων συστημάτων όπως το BERT (Bidirectional Encoder Representations from Transformers) και το GPT (Generative Pre-trainer Transformer), τα οποία έχουν εκπαιδευτεί με τεράστια σύνολα δεδομένων γλώσσας, όπως η Wikipedia Corpus και το Common Crawl, και μπορούν να προσαρμοστούν μέσω της μεθόδου fine-tuning σε συγκεκριμένες γλωσσικές διεργασίες.

2.6.1 Τρόπος Λειτουργίας

Οι Transformers βασίζονται σε αρχιτεκτονικές **κωδικοποιητή-αποκωδικοποιητή (encoder-decoder)** και σε **μηχανισμούς προσοχής (attention mechanism)**. Πιο συγκεκριμένα αποτελούνται από 6 κωδικοποιητές και 6 αποκωδικοποιητές όπως φαίνεται στο παρακάτω σχήμα [38]:

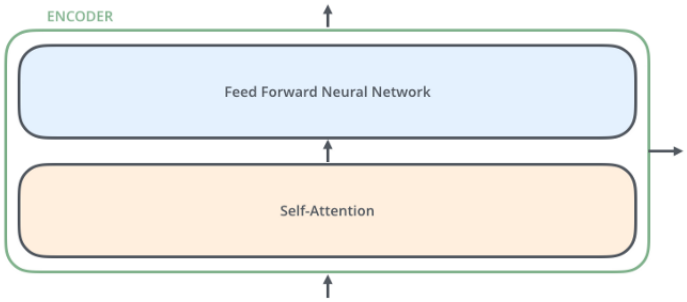


Σχήμα 2.37: Αρχιτεκτονική των Transformers

Γενικά, ο κωδικοποιητής είναι ένα νευρωνικό δίκτυο που δέχεται διαδοχική είσοδο. Μπορεί να είναι RNN, αλλά και CNN ή κάποια άλλη αρχιτεκτονική. Ο ρόλος του κωδικοποιητή είναι να διαβάσει την είσοδο και να δημιουργήσει κάποιο είδος κατάστασης (παρόμοια με την κατάσταση στο RNN) που μπορεί να θεωρηθεί ως μια αριθμητική αναπαράσταση της έννοιας της εισόδου με την οποία μπορεί να δουλέψει το μηχάνημα. Η έννοια ορισμένης οντότητας, είτε πρόκειται για εικόνα, είτε για κείμενο, είτε για βίντεο, είναι συνήθως ένα διάνυσμα ή μια μήτρα που περιέχει πραγματικούς αριθμούς. Αυτός ο φορέας (ή πίνακας) ονομάζεται στη γλώσσα μηχανικής μάθησης **ενσωμάτωση (embedding)** της εισόδου. Ο αποκωδικοποιητής είναι ένα άλλο νευρωνικό δίκτυο που λαμβάνει μια ενσωμάτωση ως είσοδο και είναι ικανό να δημιουργήσει μια ακολουθία εξόδων. Όπως θα μπορούσε να μαντέψει κανείς, η ενσωμάτωση προέρχεται από τον κωδικοποιητή.

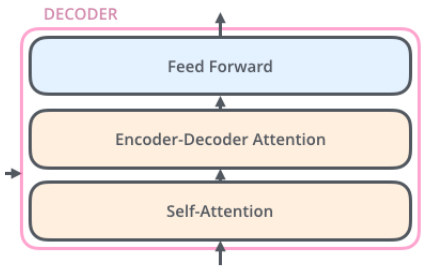
Αυτό που πρακτικά διαφοροποιεί τους transformers από άλλα ακολουθιακά μοντέλα είναι ο μηχανισμός προσοχής και οι ενσωματώσεις θέσης που χρησιμοποιούν, δύο ένοιες που θα αναλύσουμε παρακάτω.

Στην αρχιτεκτονική των Transformers, όλοι οι κωδικοποιητές έχουν την ίδια εσωτερική αρχιτεκτονική. Αποτελούνται από δύο επίπεδα: το πρώτο επίπεδο είναι ένας μηχανισμός προσοχής ο οποίος ονομάζεται **self-attention** και το δεύτερο επίπεδο είναι ένα **πλήρως συνδεδεμένο feed-forward δίκτυο**. Η εσωτερική δομή αυτή παρουσιάζεται στο παρακάτω σχήμα [38]:



Σχήμα 2.38: Αρχιτεκτονική των Encoders

Όλοι οι αποκωδικοποιητές, αντίστοιχα, παρουσιάζουν την ίδια εσωτερική αρχιτεκτονική και αποτελούνται από τα δύο επίπεδα που είδαμε και στους κωδικοποιητές, με ένα επιπλέον επίπεδο που παρεμβάλλεται μεταξύ των άλλων δύο. Το επιπλέον επίπεδο είναι πάλι ένας μηχανισμός προσοχής που παρακολουθεί την έξοδο του σταδίου κωδικοκοποίησης. Σχηματικά η εσωτερική δομή του αποκωδικοποιητή φαίνεται παρακάτω [38]:



Σχήμα 2.39: Αρχιτεκτονική των Decoders

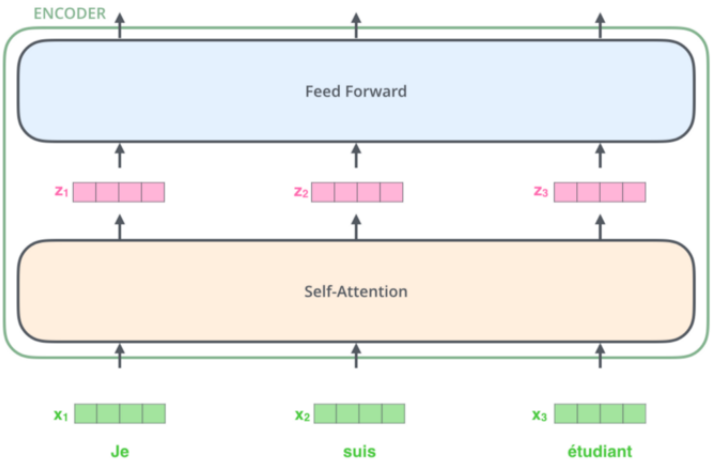
Κάθε κωδικοποιητής όπως και κάθε αποκωδικοποιητής συνδέεται με τον προηγούμενό του και ακόμα η έξοδος του τελευταίου κωδικοποιητή μεταδίδεται και στους έξι αποκωδικοποιητές όπως φαίνεται στο σχήμα 2.37.

Self-Attention

Στο σημείο αυτό, θα δούμε πως δουλεύει ο μηχανισμός προσοχής που παρουσιάστηκε παραπάνω. Ο μηχανισμός προσοχής (Attention mechanism) είναι μια τεχνική που εφαρμόζεται στα τεχνητά νευρωνικά δίκτυα και ουσιαστικά υλοποιεί αυτό που δηλώνει το όνομα του: εστιάζει την προσοχή του σε ένα τμήμα του συνόλου που δέχεται σαν είσοδο κάθε φορά.

Αρχικά, μετατρέπουμε κάθε λέξη εισόδου σε ένα διάνυσμα μέσω αλγορίθμων ενσωμάτωσης.

Αυτές οι λεκτικές ενσωματώσεις της ακολουθίας εισόδου ρέουν μέσα από κάθε ένα από τα δύο στρώματα του κωδικοποιητή όπως φαίνεται παρακάτω [38]:

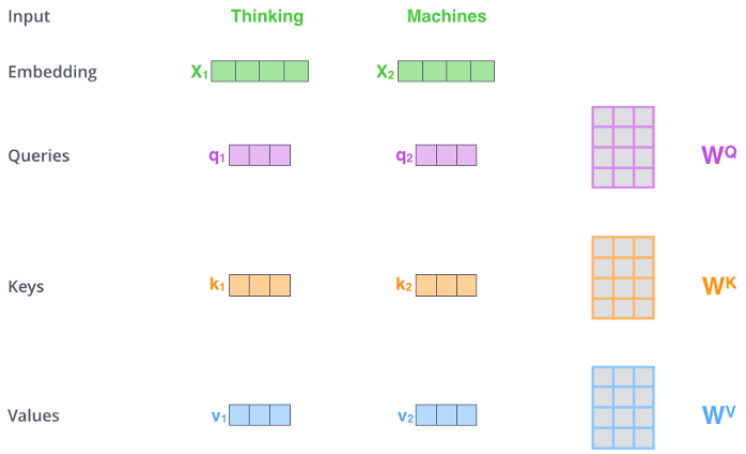


Σχήμα 2.40: Είσοδος στον πρώτο encoder

Εδώ αρχίζουμε να παρατηρούμε μια βασική ιδιότητα του Transformer, δηλαδή ότι η λέξη σε κάθε θέση ρέει μέσω της δικής της διαδρομής στον κωδικοποιητή. Υπάρχουν εξαρτήσεις μεταξύ αυτών των διαδρομών στο επίπεδο self-attention. Το στρώμα feed-forward δεν έχει αυτές τις εξαρτήσεις, ωστόσο, και έτσι οι διάφορες διαδρομές μπορούν να εκτελεστούν παράλληλα ενώ ρέουν μέσω του στρώματος feed-forward.

Το εύλογο ερώτημα που γεννάται εδώ είναι τι λειτουργίες εκτελούνται στο στρώμα self-attention ώστε να παραχθούν τα νέα διανύσματα z που φαίνονται στην εικόνα. Παρακάτω παρουσιάζονται όλα τα βήματα που ακολουθούνται στο στρώμα self-attention :

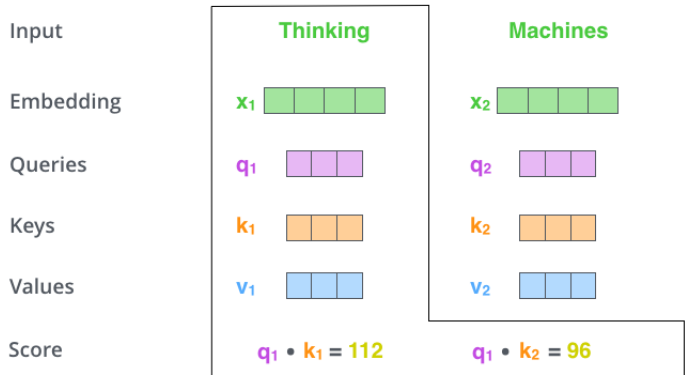
1. Αρχικά δημιουργούνται τρία διανύσματα για καθένα από τα διανύσματα εισόδου του κωδικοποιητή. Στην περίπτωση του πρώτου κωδικοποιητή, τα διανύσματα εισόδου είναι οι λεκτικές ενσωματώσεις, ενώ για τους επόμενους κωδικοποιητές τα διανύσματα εισόδου είναι αυτά που έχουν προκύψει από την έξοδο του αμέσως προηγούμενου κωδικοποιητή. Τα τρία νέα διανύσματα που δημιουργούνται ονομάζονται query,key και value. Αυτά τα διανύσματα προκύπτουν πολλαπλασιάζοντας το διάνυσμα εισόδου με τρεις πίνακες που εκπαιδεύσαμε κατά τη διάρκεια της εκπαιδευτικής διαδικασίας, οι οποίοι είναι οι γνωστοί επονομαζόμενοι πίνακες βαρών που έχουμε ξαναδεί στα νευρωνικά δίκτυα. Ένα παράδειγμα αυτής της διαδικασίας δημιουργίας των τριών διανυσμάτων φαίνεται στο παρακάτω σχήμα [38]:



Σχήμα 2.41: Query, Key and Value Vectors

Στο παραπάνω παράδειγμα, παρουσιάζονται δύο λέξεις της ακολουθίας εισόδου: οι λέξεις thinking και machines. Τα x_1 και x_2 αντίστοιχα, αποτελούν τις διανυσματικές ενσωματώσεις των δύο αυτών λέξεων. Το διάνυσμα query της λέξης thinking που συμβολίζεται ως q_1 προκύπτει πολλαπλασιάζοντας το x_1 με τον πίνακα βαρών W_Q. Τα διανύσματα key (k_1) και value (v_1) προκύπτουν με αντίστοιχο τρόπο, πολλαπλασιάζοντας το διάνυσμα x_1 με τους πίνακες W_K και W_V αντίστοιχα. Η ίδια διαδικασία ακολουθείται και για την λέξη machines που έχει διανυσματική αναπαράσταση x_2 όπως και για κάθε άλλη λέξη της ακολουθίας εισόδου.

2. Το επόμενο βήμα για τον υπολογισμό του self-attention είναι να υπολογίσουμε ένα σκορ. Για παράδειγμα, για την πρώτη λέξη "thinking" πρέπει να βαθμολογήσουμε κάθε λέξη της πρότασης εισαγωγής έναντι αυτής. Το σκορ αυτό καθορίζει το πόσο πρέπει να εστιάσουμε σε άλλα μέρη της πρότασης εισαγωγής όταν κωδικοποιούμε μια λέξη σε μια συγκεκριμένη θέση. Το σκορ υπολογίζεται λαμβάνοντας το εσωτερικό γινόμενο του διανύσματος query της λέξης που κωδικοποιούμε με το διάνυσμα key της αντίστοιχης λέξης που βαθμολογούμε. Επομένως, εάν επεξεργαζόμαστε το self-attention για τη λέξη στη θέση 1, η πρώτη βαθμολογία θα ήταν το εσωτερικό γινόμενο των q1 και k1. Η δεύτερη βαθμολογία θα ήταν το εσωτερικό γινόμενο των q1 και k2 κ.ο.κ. Η διαδικασία αυτή παρουσιάζεται στο παρακάτω σχήμα [38]:

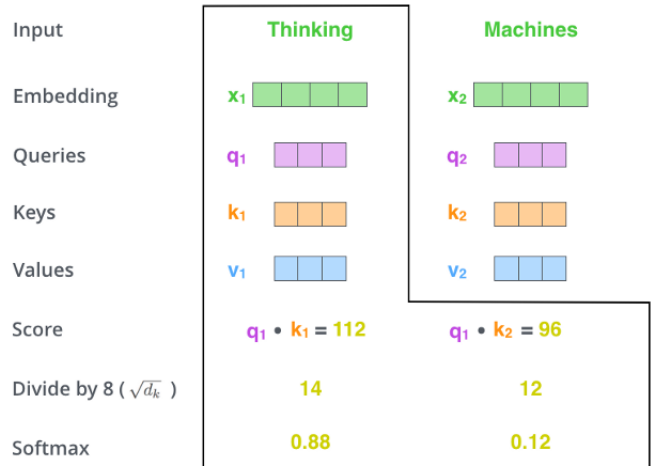


Σχήμα 2.42: Υπολογισμός του σκορ

3. Στην συνέχεια, διαιρούμε το σκορ αυτό με την τετραγωνική ρίζα της διάστασης του διανύσματος key. Στο αυθεντικό paper η διάσταση αυτή είναι 64 και επομένως η

τετραγωνική της ρίζα είναι 8.

4. Το αποτέλεσμα των σκορ που προκύπτει από το προηγούμενο βήμα, το περνάμε από μια συνάρτηση ενεργοποίησης softmax η οποία ομαλοποιεί τα σκορ και δίνει σε όλα θετικές τιμές οι οποίες όλες μαζί αθροίζουν στην τιμή 1. Το αποτελέσμα της συνάρτησης softmax, ουσιαστικά εκφράζει το πόσο θα επηρεάσει η κάθε λέξη την κωδικοποίηση της υπό εξέταση λέξης. Είναι σαφές ότι η υπο εξάταση λέξη σε αυτήν τη θέση θα έχει την υψηλότερη βαθμολογία softmax, αλλά μερικές φορές είναι χρήσιμο να παρακολουθούμε μια άλλη λέξη που σχετίζεται με την τρέχουσα λέξη. Σχηματικά η διαδικασία των δύο τελευταίων βημάτων φαίνεται παρακάτω [38]:

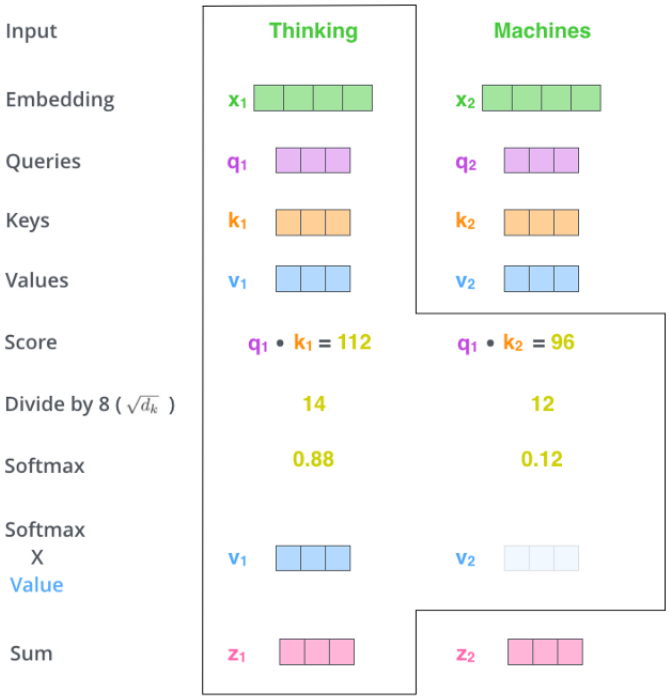


Σχήμα 2.43: Διαίρεση και εφαρμογή συνάρτησης softmax

5. Το τελικό βήμα είναι να πολλαπλασιάσουμε κάθε value διάνυσμα με τη βαθμολογία softmax και στην συνέχεια να αθροίσουμε τα αποτελέσματα αυτά. Αυτό γίνεται ώστε να διατηρήσουμε ανέπαφες τις τιμές των λέξεων (values) στις οποίες θέλουμε να επικεντρωθούμε και να αποδυναμώσουμε τις άσχετες λέξεις (πολλαπλασιάζοντας τις με μικροσκοπικούς αριθμούς όπως, για παράδειγμα, το 0,001).

   Το τελικό διάνυσμα που προκύπτει από το παραπάνω άθροισμα είναι το διάνυσμα z της υπο εξέταση λέξης, το οποίο μπορούμε πλέον να στείλουμε στο στρώμα του feed-forward νευρωνικού δικτύου.

   Η διαδικασία του βήματος αυτού φαίνεται στο παρακάτω σχήμα [38]:



Σχήμα 2.44: Πολλαπλασιασμός των βαθμολογιών softmax με τα διανύσματα values και άθροισμα

Στην πράξη, υπολογίζεται η λειτουργία προσοχής (attention mechanism) σε ένα σύνολο από queries ταυτόχρονα, συσκευασμένα μαζί σε έναν πίνακα Q. Τα keys και τα values συσκευάζονται επίσης σε πίνακες K και V. Έτσι, η μήτρα των εξόδων υπολογίζεται ως εξής:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.77}$$

Multi-Head Attention

Στην πράξη οι Transformers κάνουν παράλληλη χρήση πολλών μηχανισμών προσοχής self-attention με μια τεχνική που ονομάζεται multihead attention.

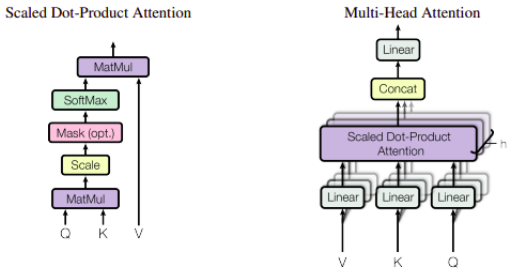
Η ιδέα πίσω από αυτό είναι ότι αν για παράδειγμα μεταφράζουμε μια λέξη, μπορούμε να δώσουμε διαφορετική προσοχή σε κάθε λέξη με βάση τον τύπο της ερώτησης που υποβάλλουμε. Έτσι, κάθε φορά που μεταφράζουμε την λέξη "kicked" στην πρόταση "I kicked the ball", μπορούμε να ρωτήσουμε "Who?". Ανάλογα με την απάντηση, η μετάφραση της λέξης σε άλλη γλώσσα μπορεί να αλλάξει. Αντίστοιχα, μπορούμε να κάνουμε άλλες ερωτήσεις, όπως "Did what;", "To whom;" κλπ.

Το multi-head attention επιτρέπει στο μοντέλο να παρακολουθεί από κοινού πληροφορίες από διαφορετικούς υποχώρους αναπαράστασης σε διαφορετικές θέσεις.

Η μαθηματική έκφραση αυτού του μηχανισμού προσοχής δίνεται ως εξής:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O, \quad where \quad head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.78}$$

Η σύγκριση μεταξύ του ενός μοναδικού attention layer με το multi-head attention φαίνεται στο παρακάτω σχήμα [39]:



Σχήμα 2.45: (αριστερά) Scaled Dot-Product Attention. (δεξιά) Multi-Head Attention αποτελούμενο από πολλά attention στρώματα που τρέχουν παράλληλα

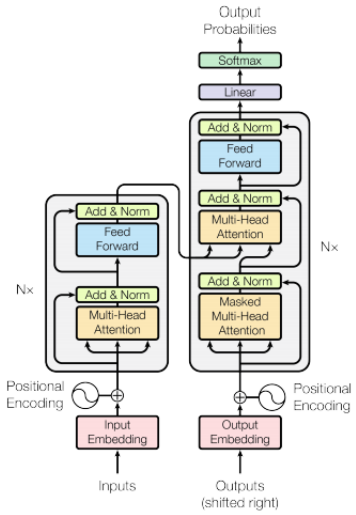
Position Encoding

Ένα άλλο σημαντικό βήμα των transformers είναι η κωδικοποίηση της θέσης, δηλαδή η εισαγωγή πληροφορίας για την θέση της κάθε λέξης στην ακολουθία της εισόδου. Για την επίτευξη αυτού του σκοπού εισάγεται στα διανύσματα των λέξεων μια επιπλέον πληροφορία σχετική με τη θέση κάθε λέξης που το μοντέλο μαθαίνει να αναγνωρίζει. Η διαίσθηση είναι ότι ένα διάνυσμα που εκφράζει θέση προστίθεται σε κάθε λέξη στο πρώτο επίπεδο κωδικοκοποίησης. Η κωδικοποίηση θέσης εισάγει σημαντική πληροφορία στο μοντέλο, την οποία στη συνέχεια θα χρησιμοποιήσει για να αποδώσει σημασία στις λέξεις.

Τέλος, το πλήρως συνδεδομένο feed-forward δίκτυο, το οποίο υπάρχει τόσο στους κωδικοποιητές όσο και στους αποκωδικοποιητές όπως έχουμε ήδη δει, αποτελείται από δύο γραμμικούς μετασχηματισμούς με ενεργοποίηση ReLU ανάμεσά τους, οι οποίοι δίνονται από την παρακάτω σχέση:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.79}$$

Μια πιο αναλυτική και συνολική εικόνα της αρχιτεκτονικής των transformers δίνεται στο παρακάτω σχήμα [39]:



Σχήμα 2.46: Η αρχιτεκτονική του μοντέλου Transformer

Στο σχήμα, αριστερά φαίνεται η εσωτερική αρχιτεκτονική των κωδικοποιητών η οποία επαναλαμβάνεται N φορές, όπου στην προκειμένη περίπτωση N=6. Δεξιά φαίνεται αντίστοιχα η εσωτερική αρχιτεκτονική των N αποκωδικοποιητών.

3 Αρχιτεκτονική Συστήματος

Στο προηγούμενο κεφάλαιο δόθηκε μια εκτενής ανάλυση όλης της θεωρίας που θα χρησιμοποιηθεί για τους σκοπούς της παρούσας διπλωματικής. Η ανάλυση αυτή θα είναι ένα πολύτιμο εργαλείο για την κατανόηση των κεφαλαίων που ακολουθούν.

Για τον σκοπό της αξιολόγησης της ομιλίας, χρησιμοποιούμε πληροφορία που εξάγουμε τόσο από ήχο όσο και από κείμενο. Με τον όρο ήχο εδώ, εννοούμε την ομιλία ενός ανθρώπου σε ηχητική μορφή (ηχητικό σήμα), ενώ με τον όρο κείμενο εννοούμε την ίδια ομιλία σε μορφή κειμένου. Καθένας από τους δύο πυλώνες αυτούς (ήχος και κείμενο) συντάσσουν ένα ξεχωριστό σύστημα επεξεργασίας και ανάλυσης, το οποίο έχει ως απώτερο στόχο να βοηθήσει στην τελική αξιολόγηση της ομιλίας. Παρακάτω παρουσιάζονται τα δύο συστήματα αυτά καθώς και ο συνδυασμός τους.

3.1 Σύστημα Ανάλυσης Ήχου

Σε ο,τι αφορά το κομμάτι του ήχου ακολουθούμε τα παρακάτω βήματα:

1. Το σύστημα παίρνει ως είσοδο την ομιλία σε μορφή ήχου (αρχείο ήχου).

2. Στην συνέχεια ο ήχος αυτός διασπάται σε τμήματα. Η διάσπαση αυτή εκτελείται μέσω προκαθορισμένων χρονικών παραθύρων.

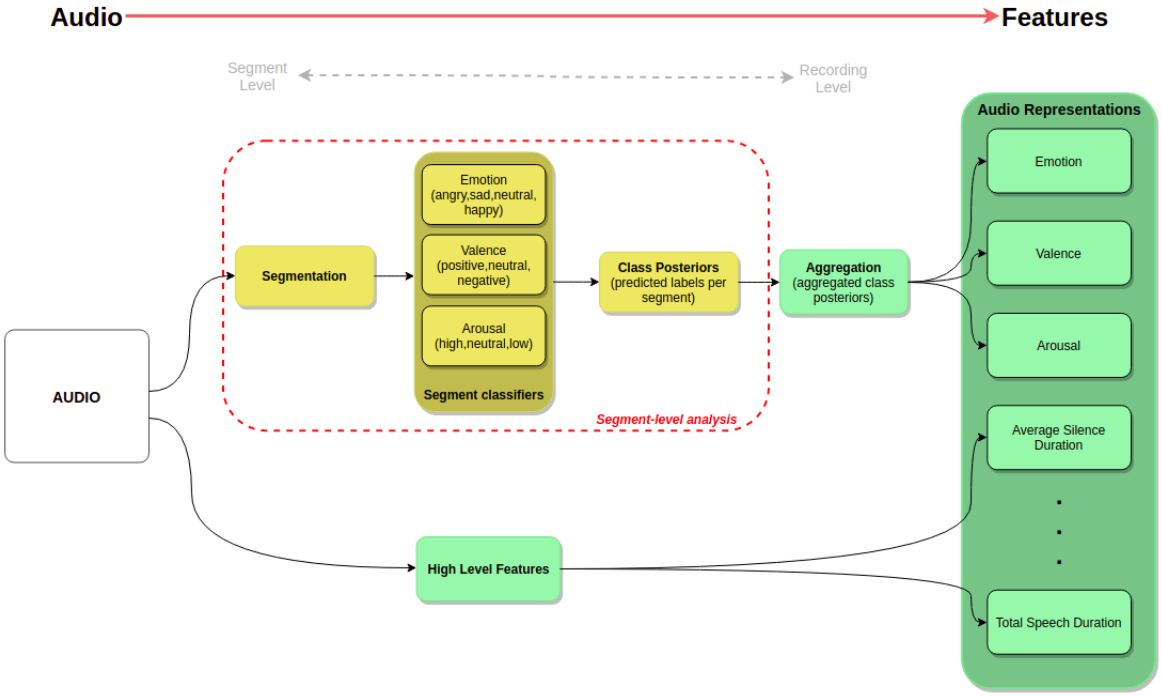
3. Το κάθε τμήμα από αυτά χρησιμοποιείται ως είσοδος σε εκπαιδευμένα μοντέλα τμημάτων, τα οποία είναι ταξινομητές (segment classifiers). Συγκεκριμένα, το σύστημα περιέχει τρία τέτοια μοντέλα: ένα για την πρόβλεψη του συναισθήματος (speech emotion recognition), ένα για την πρόβλεψη της εξέγερσης/έξαψης (arousal) και ένα για την πρόβλεψη του σθένους/δραστικότητας (valence).

4. Οι έξοδοι των ταξινομητών παράγουν μια πρόβλεψη για κάθε τμήμα ήχου ξεχωριστά (class posteriors). Σκοπός όμως του συστήματος είναι να χαρακτηρίσει την συνολική ομιλία. Έτσι συγκεντρώνονται οι ετικέτες που έχουν προβλεφθεί για το κάθε τμήμα, με σκοπό να παράξουν πληροφορία για όλο το αρχείο ήχου. Η συγκέντρωση αυτή γίνεται υπολογίζοντας τον μέσο όρο των δειγμάτων ανά ετικέτα. Για παράδειγμα, εάν ένα αρχείο ήχου περιέχει 2 τμήματα που έχουν χαρακτηριστεί με την ετικέτα "χαρά" και 3 τμήματα με την ετικέτα "άγχος" τότε ο συνολικός ήχος-ομιλία θα χαρακτηριστεί κατά 0.4 ως "χαρά" και 0.6 ως "άγχος".

5. Είναι εμφανές ότι με τα παραπάνω βήματα έχουμε ήδη αποκτήσει πληροφορία ή αλλιώς χαρακτηριστικά για την ομιλία, τα οποία θα μπορούσαν να χρησιμοποιηθούν από ένα μοντέλο που θα αξιολογεί την ποιότητά της. Αυτά τα χαρακτηριστικά όμως, ίσως να μην είναι επαρκή. Για τον λόγο αυτό, στο βήμα αυτό προστίθενται κάποια επιπλέον χαρακτηριστικά που αφορούν τον συνολικό ήχο όπως είναι ο μέσος όρος διάρκειας της σιωπής, ο αριθμός παύσεων, η συνολική διάρκεια ομιλίας, ο ρυθμός λέξεων κ.α. Τα χαρακτηριστικά αυτά τα ονομάζουμε χαρακτηριστικά υψηλού επιπέδου (high level features) ή αλλιώς χαρακτηριστικά επιπέδου εγγραφής (recording-level features), για να τα ξεχωρίζουμε από αυτά του προηγούμενου βήματος που προέκυψαν από τα τμήματα του ήχου (segment-level features).

6. Τέλος, ενώνοντας τα χαρακτηριστικά των δύο προηγούμενων βημάτων, καταλήγουμε σε μια συνολική αναπαράσταση του ήχου.

Το σύστημα που περιγράφηκε στα προηγούμενα βήματα παρουσιάζεται στο παρακάτω σχήμα:



Σχήμα 3.1: Διαδικασία Ανάλυσης της Ηχητικής Πληροφορίας

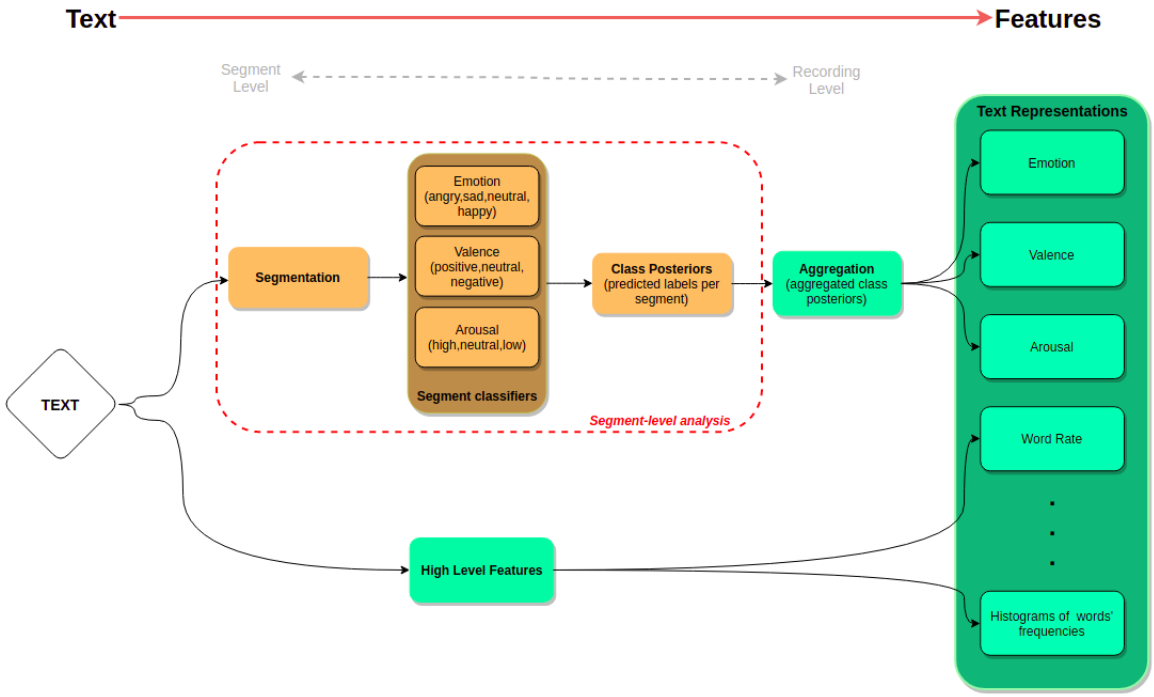
Παρατηρούμε στο σχήμα 3.1, ότι το κομμάτι του συστήματος που αφορά την διάσπαση του ήχου σε τμήματα και την χρήση ταξινομητών τμημάτων έχει σκιαγραφηθεί με διακεκομμένες γραμμές και ονομάζεται ανάλυση επιπέδου τμημάτων (segment-level analysis). Αυτό το κομμάτι θα αναλυθεί εκτενώς στο επόμενο κεφάλαιο (4). Αντίθετα, τα υπόλοιπα κομμάτια που αφορούν τα χαρακτηριστικά του συνολικού ήχου, συνθέτουν την ανάλυση επιπέδου καταγραφής (recording-level analysis), η οποία θα αναπτυχθεί εκτενώς στο κεφάλαιο 5.

3.2 Σύστημα Ανάλυσης Κειμένου

Το κείμενο της ομιλίας προκύπτει από τον ήχο, χρησιμοποιώντας τις τεχνολογίες τεχνητής νοημοσύνης από το Google Cloud. Συγκεκριμένα χρησιμοποιήθηκε το Cloud Speech-to-Text (ομιλία-σε-κείμενο), το οποίο εφαρμόζει τους πιο προηγμένους αλγόριθμους νευρωνικών δικτύων βαθιάς μάθησης της Google για αυτόματη αναγνώριση ομιλίας (Automatic Speech Recognition-ASR) [40].

Σε ο,τι αφορά το κομμάτι του κειμένου ακολουθήθηκαν τα ίδια ακριβώς βήματα με το σύστημα του ήχου, κάνοντας πάλι διάσπαση του κειμένου σε τμήματα και χρησιμοποιώντας ταξινομητές τμημάτων. Αυτό που αλλάζει εδώ είναι το βήμα 5, όπου τα χαρακτηριστικά υψηλού επιπέδου για το κείμενο είναι πλέον ο ρυθμός λέξεων, ο αριθμός μοναδικών λέξεων, τα ιστογράμματα των συχνοτήτων των λέξεων κ.α.

Το σύστημα ανάλυσης κειμένου παρουσιάζεται στο παρακάτω σχήμα:



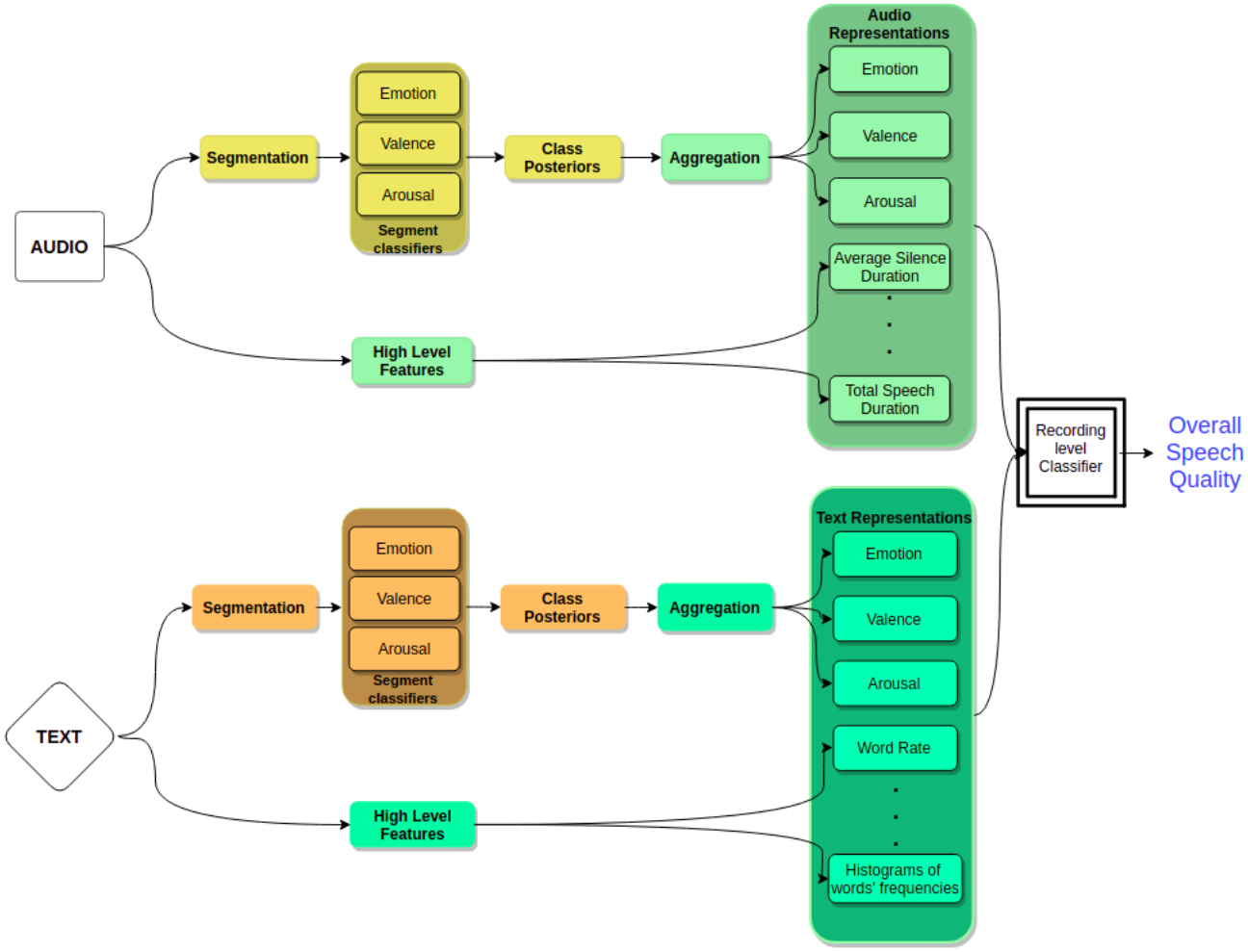
Σχήμα 3.2: Διαδικασία Ανάλυσης της Κειμενικής Πληροφορίας

Στο σχήμα 3.2 διακρίνουμε και πάλι τις περιοχές της ανάλυσης σε επίπεδο τμημάτων και της ανάλυσης σε επίπεδο καταγραφής, οι οποίες θα αναπτυχθούν εκτενώς στα επόμενα κεφάλαια.

3.3 Ολοκλήρωση Αρχιτεκτονικής

Έχοντας πλέον εξάγει χαρακτηριστικά τόσο από το συνολικό ήχο όσο και από το συνολικό κείμενο, ο τελικός στόχος είναι να τα χρησιμοποιήσουμε σε ένα μοντέλο το οποίο θα εκπαιδευτεί για να προβλέπει την ποιότητα της ομιλίας. Με αυτόν τον τρόπο, οι έξοδοι των συστημάτων που παρουσιάστηκαν στα προηγούμενα υποκεφάλαια, θα χρησιμοποιηθούν ως είσοδοι στο τελικό μοντέλο-ταξηνομητή.

Στο παρακάτω σχήμα φαίνεται η συνολική αρχιτεκτονική του συστήματος:



Σχήμα 3.3: Τελικοί Ταξινομητές

Όπως φαίνεται στο σχήμα 3.3, το τελικό μοντέλο ονομάζεται ταξινομητής επιπέδου καταγραφής (recording-level classifier) και αυτό γιατί αφορά την συνολική ομιλία, δηλαδή την συνολική ηχογράφηση/καταγραφή όπως θα μπορούσε κανείς να πει.

Στο σημείο αυτό είναι σημαντικό να επισημανθεί πως δεν είναι υποχρεωτικό να χρησιμοποιηθούν και τα δύο συστήματα ήχου και κειμένου. Απεναντίας, ο τελικός ταξινομητής μπορεί να χρησιμοποιήσει μόνο χαρακτηριστικά ήχου ή μόνο χαρακτηριστικά κειμένου προκειμένου να αξιολογήσει την ποιότητα της ομιλίας. Τίθεται έτσι προς διερεύνηση ποια από τα τρία συστήματα (μόνο ήχος, μόνο κείμενο, συνδυασμός), θα δώσει τα πιο έγκυρα αποτελέσματα, δηλαδή θα βοηθήσει το τελικό μοντέλο να μάθει καλύτερα.

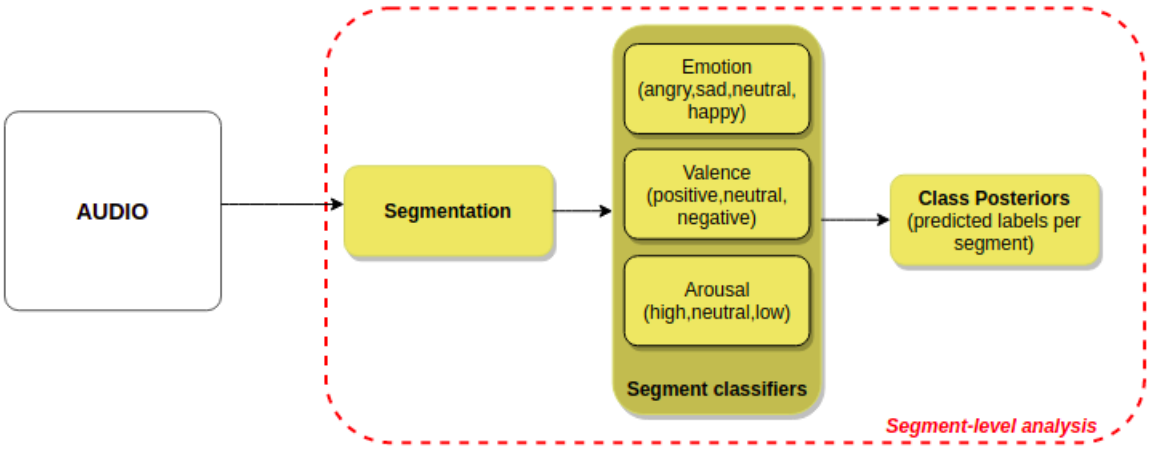
4 Ανάλυση σε Επίπεδο Τμήματος

Στο κεφάλαιο αυτό θα γίνει μια διεξοδική ανάλυση του μέρους της αρχιτεκτονικής που χρησιμοποιεί τμήματα ήχου και τμήματα κειμένου αντίστοιχα (segment-level analysis).

4.1 Ανάλυση Ήχου

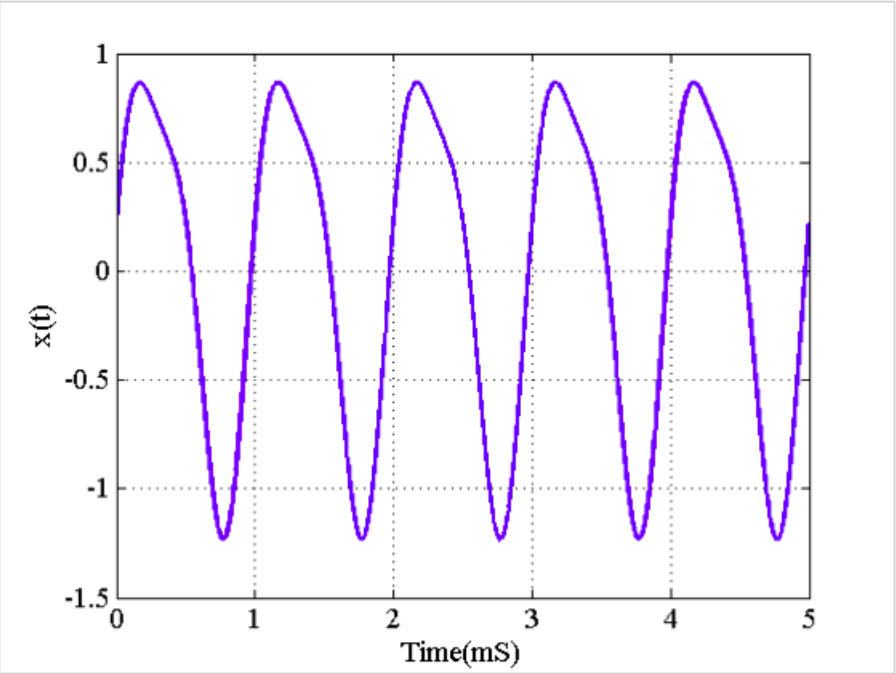
Στο υποκεφάλαιο αυτό θα παρουσιαστούν οι τεχνικές διάσπασης του ηχητικού σήματος σε παράθυρα, το σύνολο των ηχητικών χαρακτηριστικών [41] που χρησιμοποιήθηκαν για τους ταξινομητές τμημάτων, τα σύνολα δεδομένων που χρησιμοποιήθηκαν για την εκπαίδευση των ταξινομητών και τα πειράματα που διεξήχθησαν.

Συγκεκριμένα, θα αναλυθεί το παρακάτω κομμάτι της αρχιτεκτονικής του συστήματος:



Σχήμα 4.1: Ανάλυση Ήχου σε Επίπεδο Τμήματος

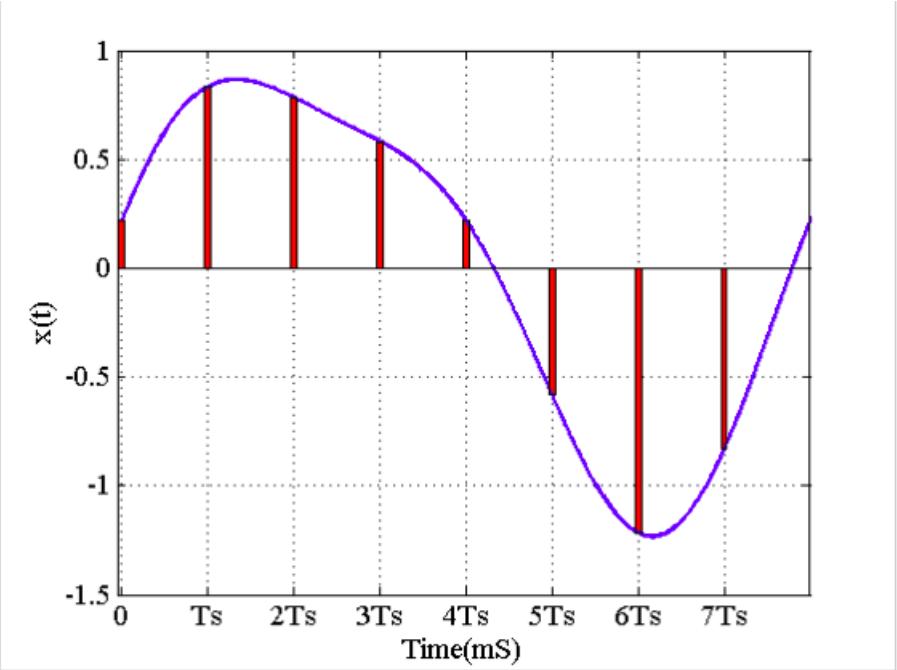
Ως εισαγωγή θα ήταν σημαντικό να αναφέρουμε πως ένα σήμα που συμβολίζουμε ως x(t), είναι ένα σήμα συνεχούς χρόνου όπως φαίνεται στο παρακάτω σχήμα [43]:



Σχήμα 4.2: Σήμα Συνεχούς Χρόνου

Ένας ψηφιακός υπολογιστής, όμως, δεν μπορεί να λειτουργήσει με σήμα συνεχούς χρόνου για αυτόν τον λόγο είναι αναγκαίο να πάρουμε μερικά δείγματα του x(t) και να δουλεύουμε με αυτά τα δείγματα αντί του αρχικού σήματος. Επιπλέον, το αρχικό σήμα επεκτείνεται επ᾽ άπειρον, αλλά ο ψηφιακός υπολογιστής μπορεί να επεξεργαστεί μόνο έναν πεπερασμένο αριθμό δειγμάτων. Επομένως, γενικά, χρησιμοποιείται μια ακολουθία πεπερασμένης διάρκειας για την αναπαράσταση αυτού του αναλογικού σήματος συνεχούς χρόνου που μπορεί να εκτείνεται σε θετικό άπειρο στον άξονα του χρόνου.

Για παράδειγμα, μπορούμε να δειγματοληπτήσουμε το σήμα x(t) του σχήματος 4.2 με συχνότητα δειγματοληψίας f_s = 8000 δείγματα ανά δευτερόλεπτο (samples/second). Έτσι για μία περίοδο του αρχικού σήματος x(t) η οποία ισούται με 1msec θα πάρουμε 8000 samples/sec x 0.001sec = 8 samples (δείγματα). Το αποτέλεσμα εμφανίζεται στο παρακάτω σχήμα με κόκκινο χρώμα [43]:



Σχήμα 4.3: Δειγματοληψία Σήματος Συνεχούς Χρόνου

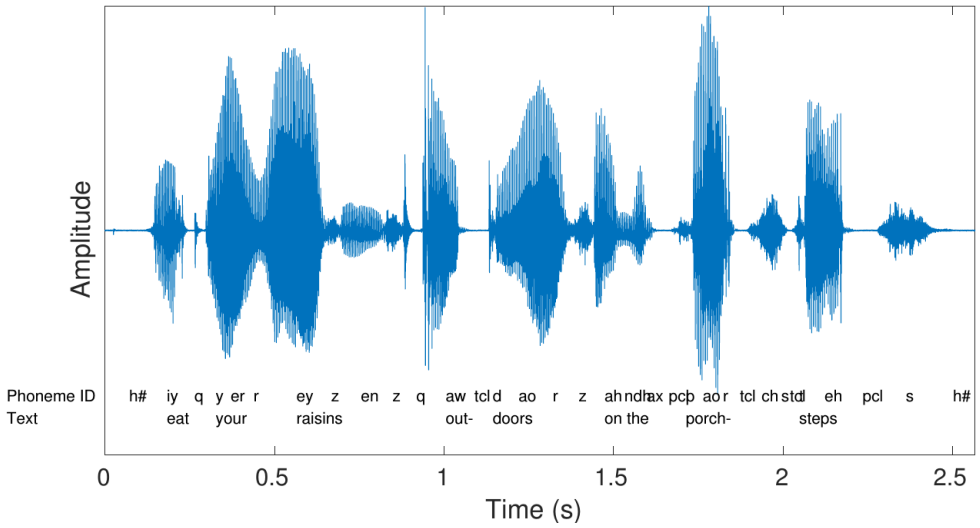
Εάν ομαλοποιήσουμε τον άξονα του χρόνου στην περίοδο δειγματοληψίας $T_s = \frac{1}{f_s}$, θα λάβουμε τη διακριτή ακολουθία x(n) για τα 8 δείγματα (x(n=0) = x(t=0), x(n=1) = x(t=T_S), ..., x(n=7) = x(t=7T_s)).

4.1.1 Βραχυπρόθεσμη Επεξεργασία Ήχου

Η **βραχυπρόθεσμη επεξεργασία ήχου (short-term audio processing)**, είναι μια τεχνική κατά την οποία το ηχητικό σήμα διασπάται σε **βραχυπρόθεσμα παράθυρα (short-term windows)**.

Μια προφορική πρόταση είναι μια ακολουθία φωνημάτων. Τα σήματα ομιλίας, επομένως, αλλάζουν με την πάροδο του χρόνου. Για να εξαγάγουμε πληροφορίες από ένα σήμα, πρέπει να χωρίσουμε το σήμα σε αρκετά σύντομα τμήματα. Με άλλα λόγια, θέλουμε να εξαγάγουμε τμήματα τα οποία είναι αρκετά μικρά ώστε οι ιδιότητες του σήματος ομιλίας να μην αλλάζουν μέσα σε αυτά.

Ένα παράδειγμα σήματος ομιλίας με τα αντίστοιχα φωνήματα παρουσιάζεται παρακάτω [42]:



Σχήμα 4.4: Ηχητικό Σήμα Ομιλίας

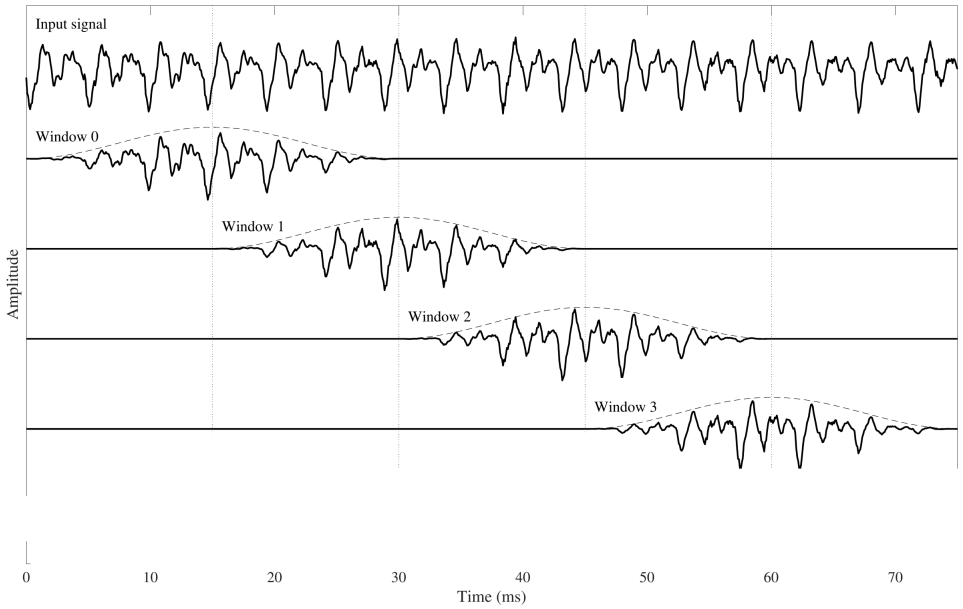
Για τον σκοπό αυτό, στις περισσότερες εφαρμογές το ηχητικό σήμα διασπάται σε πιθανά επικαλυπτόμενα παράθυρα και η ανάλυση του ήχου διεξάγεται σε τμηματική βάση (segment basis). Ο λόγος για τον οποίο γίνεται αυτή η διάσπαση είναι γιατί όπως είπαμε, τα ηχητικά σήματα δεν είναι στάσιμα στην πάροδο του χρόνου, δηλαδή οι ιδιότητές τους αλλάζουν. Για παράδειγμα, το συναίσθημα ενός ομιλητή μπορεί να αλλάζει με την πάροδο του χρόνου, όσο αυτός μιλάει. Το ίδιο και η ένταση, ο τόνος φωνής, ο ρυθμός ομιλίας κ.α. Έτσι θα ήταν ελλιπές και μεροληπτικό να χαρακτηρίζαμε όλη την ομιλία με μία μόνο ετικέτα συναισθήματος, έντασης κ.ο.κ. Πιο συγκεκριμένα, αν είχαμε ένα σήμα, το οποίο περιέχει τμήματα τόσο κανονικής όσο και υψηλής έντασης, και υπολογίζαμε τον μέσο όρο της έντασης του συνολικού σήματος, τότε στο αποτέλεσμα θα κυριαρχούσαν τα δείγματα από το τμήμα υψηλής έντασης, αγνοώντας έτσι το τμήμα του συνολικού σήματος στο οποίο η ένταση ήταν κανονική. Αντίθετα, αν υπολογίζαμε την ένταση ξεχωριστά για κάθε τμήμα, τότε θα είχαμε έναν πιο ακριβές χαρακτηρισμό του συνολικού σήματος.

Από μαθηματικής σκοπιάς, η παραπάνω διεργασία έχει ως εξής: δεδομένου ενός ηχητικού σήματος x(n) με n = 0,...,N-1, το οποίο έχει N δείγματα, σε κάθε βήμα της βραχυπρόθεσμης επεξεργασίας, πολλαπλασιάζουμε το σήμα με ένα παράθυρο προκαθορισμένη διάρκειας w(n). Το παράθυρο-τμήμα του σήματος στο βήμα i προκύπτει ως εξής:

$$x_i(n) = x(n)w(n - m_i), \quad i = 1, ..., K - 1 \tag{4.1}$$

όπου **i** είναι ο αριθμός των βημάτων, δηλαδή ο αριθμός των παραθύρων-τμημάτων στα οποία θα διασπαστεί το σήμα. Το $\boldsymbol{m_i}$ είναι ο δείκτης του δείγματος στο οποίο ξεκινάει το i-οστό παράθυρο. Στην ουσία δείχνει τον αριθμό των δειγμάτων που το παράθυρο w πρέπει να μετατοπιστεί με σκοπό να παράξει το i-οστό τμήμα. Δυο ακόμα μετρικές είναι το **μήκος του παραθύρου (length)**, το οποίο είναι σταθερό και συμβολίζεται ως $\boldsymbol{W_L}$ και το **βήμα (hop size / step)** επίσης σταθερό με το οποίο μετακινείται το παράθυρο, που συμβολίζεται ως $\boldsymbol{W_S}$. Επίσης, ο συνολικός αριθμός των τμημάτων στα οποία χωρίζεται τελικά το ηχητικό σήμα υπολογίζεται ως $\frac{N-W_l}{W_S} + 1$.

Παρακάτω φαίνεται ένα παράδειγμα διάσπασης σήματος σε παράθυρα [42]:



Σχήμα 4.5: Διάσπαση Σήματος σε Παράθυρα

Ακολουθεί ένα αριθμητικό παράδειγμα: αν έχουμε ένα σήμα με συχνότητα δειγματοληψίας $F_s = 8000$ Hz, το οποίο θέλουμε να διασπάσουμε με παράθυρα βήματος $W_S = 10$ ms και μήκους $W_L = 20$ ms ή αλλιώς $W_L = 0.02 \cdot 8000 = 160$ samples, τότε η μεταβλητή m_i θα είναι $m_i = i \cdot W_S \cdot F_s = i \cdot 0.01 \cdot 8000 = i \cdot 80$ samples. Αυτό σημαίνει ότι κάθε βήμα που είναι 0.01 sec μεταφράζεται σε 80 δείγματα και το i-οστό παράθυρο ξεκινάει στο δείγμα $i \cdot 80$ και τελειώνει στο δείγμα $i \cdot 80 + W_L - 1$. Έτσι το 3ο παράθυρο σε αυτήν την περίπτωση θα ξεκινούσε στο δείγμα $3 \cdot 80 = 240$ και θα τελείωνε στο δείγμα $240 + 0.02 \cdot 8000 - 1 = 240 + 160 - 1 = 399$, ενώ το 4ο παράθυρο θα ξεκινούσε στο δείγμα $4 \cdot 80 = 320$ και θα τελείωνε στο δείγμα $320 + 0.02 \cdot 8000 - 1 = 320 + 160 - 1 = 479$.

Όπως μπορεί κανείς να παρατηρήσει, τα δύο αυτά διαδοχικά παράθυρα επικαλύπτονται, αφού το 4ο ξεκινάει πριν τελειώσει το 3ο. Μάλιστα υπάρχει 50% επικάλυψη. Αυτό συμβαίνει γιατί το βήμα του παραθύρου είναι πιο μικρό από το μήκος του ($W_S > W_L$). Για βήματα μεγαλύτερα ή ίσα από το μήκος, δεν παρουσιάζεται επικάλυψη. Αντίστοιχα, όσο πιο μικρό είναι το βήμα σε σχέση με το μήκος του παραθύρου, τόσο μεγαλύτερο ποσοστό επικάλυψης παρουσιάζεται.

Υπάρχουν διαφόρων ειδών παράθυρα που μπορούν να χρησιμοποιηθούν. Το πιο σύνηθες και αυτό που χρησιμοποιείται στην παρούσα διπλωματική εργασία είναι το **ορθογώνιο παράθυρο (rectangular window)**, το οποίο έχει τιμή 1 εντός του μήκους του και 0 οπουδήποτε αλλού:

$$\begin{cases} 1, & 0 \le n \le W_L - 1 \\ 0, & \text{elsewhere} \end{cases}$$

Έχοντας, λοιπόν, διασπάσει το ηχητικό σήμα σε βραχυπρόθεσμα τμήματα (short-term audio segments), μπορούμε πλέον να εξαγάγουμε χαρακτηριστικά για κάθε τμήμα ξεχωριστά, δημιουργώντας έτσι μια ακολουθία από διανύσματα χαρακτηριστικών. Η διάσταση των διανυσμάτων αυτών εξαρτάται από τον αριθμό των χαρακτηριστικών που θα χρησιμοποιήσουμε τα οποία αναλύονται σε επόμενο υποκεφάλαιο.

4.1.2 Μεσοπρόθεσμη Επεξεργασία Ήχου

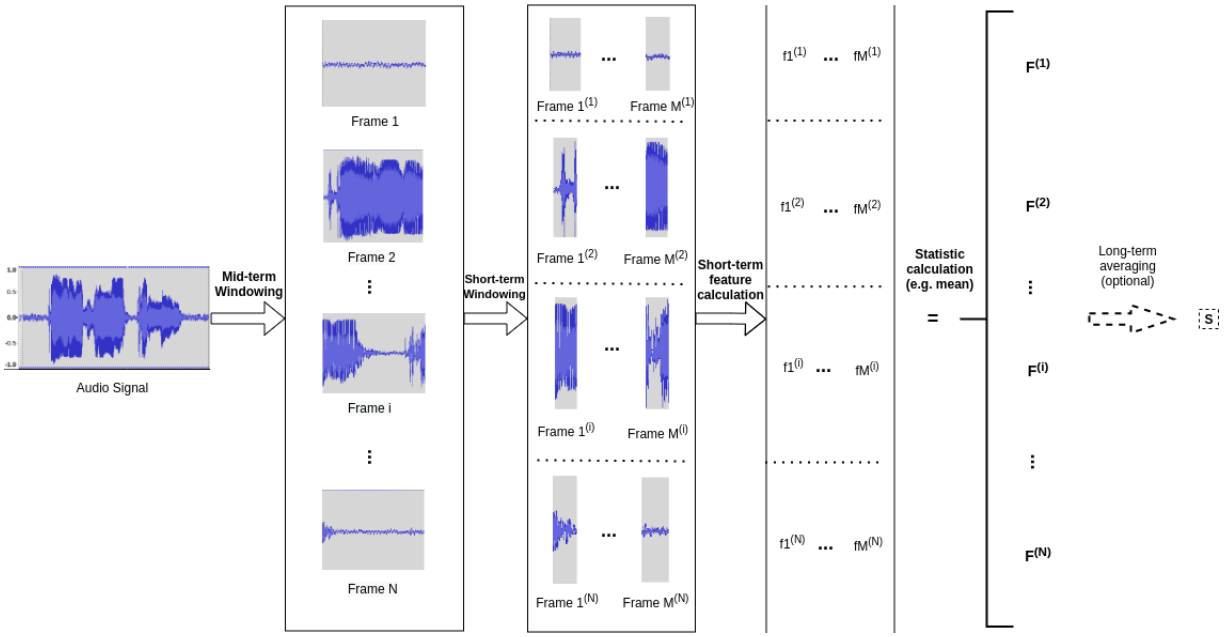
Η **μεσοπρόθεσμη επεξεργασία ήχου (mid-term audio processing)**, είναι η διαδικασία κατά την οποία το ηχητικό σήμα διασπάται σε μεγαλύτερα παράθυρα **(mid-term/texture windows)** απ' ότι στην περίπτωση της βραχυπρόθεσμης επεξεργασίας και στην συνέχεια για κάθε ένα από τα εξαγόμενα τμήματα, ακολουθείται η βραχυπρόθεσμη επεξεργασία.

Ουσιαστικά, το σήμα χωρίζεται σε μεγαλύτερα παράθυρα, καθένα από τα οποία περιέχει έναν αριθμό μικρότερων παραθύρων. Με αυτόν τον τρόπο, εξάγεται ένα διάνυσμα χαρακτηριστικών για το κάθε μεσοπρόθεσμο τμήμα (mid-term segment) του ήχου, το οποίο αναπαριστά τα στατιστικά των χαρακτηριστικών, μπορεί δηλαδή να προκύψει από την μέση τιμή των χαρακτηριστικών στα βραχυπρόθεσμα παράθυρα, από την τυπική απόκλιση τους κ.α.

Αν για παράδειγμα έχουμε ένα σήμα 10 δευτερολέπτων και χρησιμοποιούμε mid-term παράθυρα των 2 δευτερολέπτων και short-term παράθυρα του 1ος δευτερολέπτου, τότε το σήμα θα χωριστεί σε 10/2=5 mid-term παράθυρα, το καθένα από τα οποία περιέχει 2/1=2 short-term παράθυρα. Έτσι αν θέλουμε για παράδειγμα να εξάγουμε το ποσοστό μηδενικής διέλευσης, που είναι ένα ηχητικό χαρακτηριστικό το οποίο θα παρουσιαστεί στην συνέχεια, τότε θα το υπολογίσουμε για καθένα εκ των δύο short-term παραθύρων και θα πάρουμε τον μέσο όρο αυτών των τιμών ή/και την τυπική απόκλισή τους, ώστε να προκύψει μια αναπαράσταση σε mid-term επίπεδο.

Σε πολλές εφαρμογές, η εξαγωγή χαρακτηριστικών σε mid-term επίπεδο, χρησιμοποιείται για μια πιο μακροπρόθεσμη αναπαράσταση. Σε αυτές τις περιπτώσεις, είναι προτιμότερο να εξάγουμε ένα μόνο διάνυσμα χαρακτηριστικών που να αντιπροσωπεύει τον συνολικό ήχο, έναντι πολλών διανυσμάτων από κάθε mid-term τμήμα. Έτσι, υπολογίζεται ο **μακροπρόθεσμος μέσος όρος (long-term averaging)** των μεσοπρόθεσμων στατιστικών (mid-term feature statistics), εξάγοντας με αυτόν τον τρόπο ένα μοναδικό διάνυσμα χαρακτηριστικών. Σε αυτό το τελικό διάνυσμα, θα έχει επιτευχθεί η τόνωση των σημαντικών χαρακτηριστικών και η απόρριψη των χαρακτηριστικών που παρουσιάζονται μόνο σε χρονικά τοπικό επίπεδο (temporal basis).

Η παραπάνω διαδικασία της μεσοπρόθεσμης επεξεργασίας ήχου παρουσιάζεται στο παρακάτω σχήμα:



Σχήμα 4.6: Μεσοπρόθεσμη Ηχητική Ανάλυση

Στο παραπάνω σχήμα (4.6) απεικονίζεται ένα ηχητικό σήμα, το οποίο διασπάται σε N mid-term παράθυρα (mid-term windowing) και στην συνέχεια το κάθε παράθυρο από αυτά διασπάται σε M short-term παράθυρα (short-term windowing). Από κάθε short-term παράθυρο εξάγεται ένα διάνυσμα χαρακτηριστικών f (short-term feature calculation). Στην συνέχεια, γίνεται ένας στατιστικός υπολογισμός (παραδείγματος χάριν υπολογισμός μέσης τιμής), για κάθε mid-term παράθυρο (statistic calculation). Έτσι, καταλήγουμε με N διανύσματα χαρακτηριστικών συνολικά ($F^{(1)}...F^{(N)}$). Στο τελευταίο βήμα μπορεί να υπολογιστεί προαιρετικά ο μέσος όρος των διανυσμάτων αυτών και να καταλήξουμε με ένα μοναδικό διάνυσμα (S) που χαρακτηρίζει το συνολικό σήμα ήχου (long-term averaging).

4.1.3 Εξαγωγή Χαρακτηριστικών Ήχου

Ένα ηχητικό σήμα είναι ένα σήμα που περιέχει πληροφορίες στο ακουστικό εύρος συχνοτήτων. Η αναπαράσταση ήχου αναφέρεται στην εξαγωγή ιδιοτήτων σήματος ήχου, ή χαρακτηριστικών, που είναι αντιπροσωπευτικά της σύνθεσης ακουστικού σήματος (τόσο σε χρονικό όσο και σε φασματικό πεδίο) και στη συμπεριφορά του σήματος ήχου με την πάροδο του χρόνου. Η εξαγωγή χαρακτηριστικών συνδυάζεται συνήθως με την επιλογή χαρακτηριστικών, μέσω της οποίας καθορίζεται το καλύτερο σύνολο χαρακτηριστικών για την επιδιωκόμενη λειτουργία στο ηχητικό σήμα. Στόχος είναι η εξαγωγή χαρακτηριστικών από τα δεδομένα ήχου (ομιλίες), τα οποία θα έχουν πληροφοριακή ισχύ για τα μοντέλα προς εκπαίδευση.

Έχοντας, λοιπόν, διασπάσει το ηχητικό σήμα σε βραχυπρόθεσμα παράθυρα, για το καθένα από αυτά τα παράθυρα πρέπει να υπολογιστούν κάποια χαρακτηριστικά, από τα οποία θα εξαχθούν στην συνέχεια στατιστικές τιμές ανά μεσοπρόθεσμο παράθυρο.

Υπάρχουν πολλές μετρικές που μπορούν να χρησιμοποιηθούν ως χαρακτηριστικά στην ανάλυση ήχου. Αυτή η ενότητα παρέχει μια σύντομη περιγραφή των διαφόρων χαρακτηριστικών που χρησιμοποιήθηκαν στο σχεδιασμό του συστήματος.

4.1.3.1 Χρονικά Χαρακτηριστικά

Τα χρονικά χαρακτηριστικά εξάγονται απευθείας από το ηχητικό σήμα. Συχνά, συνδυάζονται με κάποια φασματικά χαρακτηριστικά που είναι πιο εξελιγμένα, τα οποία θα παρουσιαστούν στο επόμενο κεφάλαιο. Σε αυτό το κεφάλαιο αναλύονται τα χρονικά χαρακτηριστικά του ηχητικού σήματος που χρησιμοποιούνται στο σύστημα για την εκπαίδευση των ταξινομητών τμημάτων (segment classifiers).

Ενέργεια

Δεδομένου ενός ηχητικού σήματος διακριτού χρόνου x(n), η μαθηματική έκφραση της ενέργειάς **(energy)** του ορίζεται ως εξής:

$$E_s = \sum_{n=-\infty}^{\infty} |x(n)|^2 \tag{4.2}$$

Επειδή, όμως, δουλεύουμε σε επίπεδο βραχυπρόθεσμων τμημάτων του σήματος, η βραχυπρόθεσμη ενέργεια (short-term energy) υπολογίζεται ως εξής:

$$E(i) = \sum_{n=1}^{W_L} |x_i(n)|^2$$

όπου $\boldsymbol{x_i(n)}$ για n=1...W_L, είναι η ακολουθία των ηχητικών δειγμάτων του ιοστού παραθύρου, το οποίο έχει μήκος $\boldsymbol{W_L}$.

Για να απεξαρτητοποιηθεί η ενέργεια από το μήκος του εκάστοτε παραθύρου, συνήθως κανονικοποιείται, διαιρώντας την με το μήκος αυτό. Οπότε, τελικά, η ενέργεια δίνεται ως εξής:

$$E(i) = \frac{1}{W_L} \sum_{n=1}^{W_L} |x_i(n)|^2 \tag{4.3}$$

Η ενέργεια μπορεί να παρουσιάσει πολλές εναλλαγές και μεγάλη απόκλιση μεταξύ των τμημάτων ομιλίας, αφού τα σήματα ομιλίας περιέχουν φωνήματα τα οποία μπορεί να είναι πιο ασθενή ή και περιόδους σιωπής/παύσεων ανάμεσα σε λέξεις και προτάσεις. Για αυτόν τον λόγο ένα μεσοπρόθεσμο στατιστικό (mid-term statistic) που μπορεί να εξαχθεί από τις βραχυπρόθεσμες ενέργειες (short-term energy), είναι η τυπική απόκλιση σ^2. Αν, για παράδειγμα, συγκρίνουμε την τυπική απόκλιση ενέργειας ενός τμήματος ομιλίας με ένα τμήμα μουσικής, θα παρατηρήσουμε ότι στην περίπτωση της ομιλίας η τυπική απόκλιση παίρνει μεγαλύτερη τιμή από ότι στην περίπτωση της μουσικής, αφού στην μουσική δεν παρατηρούνται τόσο έντονες εναλλαγές.

Εντροπία Ενέργειας

Η **εντροπία της ενέργειας (entropy of energy)**, είναι μια μετρική που δείχνει τις απότομες αλλαγές στην ενέργεια ενός ηχητικού σήματος. Για τον υπολογισμό της εντροπίας ενέργειας ενός βραχυπρόθεσμου παραθύρου, σπάμε το παράθυρο σε K υπο-παράθυρα συγκεκριμένης διάρκειας. Στην συνέχεια για καθένα από τα υπο-παράθυρα αυτά υπολογίζουμε την ενέργεια σύμφωνα με την εξίσωση (4.2) και διαιρούμε την τιμή αυτή με την συνολική ενέργεια του βραχυπρόθεσμου παραθύρου. Οπότε έχουμε τελικά την ακολουθία τιμών για κάθε υποπαράθυρο j ως εξής:

$$e_j = \frac{E_{subFrame_j}}{E_{shortFrame_i}} \tag{4.4}$$

όπου το j είναι ένα εκ των K υπο-παραθύρων (j=1,...,K) και i είναι ένα εκ των βραχυπρόθεσμων παραθύρων του συνολικού σήματος. Η διαίρεση αυτή μετατρέπει την ενέργεια του υπο-παραθύρου σε πιθανότητα. Ο παρονομαστής της παραπάνω εξίσωσης δίνεται από την έκφραση:

$$E_{shortFrame_i} = \sum_{j=1}^{K} E_{subFrame_j} \tag{4.5}$$

όπου η ενέργεια του συνολικού βραχυπρόθεσμου παραθύρου i, υπολογίζεται από το άθροισμα των ενεργειών όλων των υπο-παραθύρων του.

Ως τελικό βήμα υπολογίζεται η εντροπία της ακολουθίας e_j ως εξής:

$$H(i) = -\sum_{j=1}^{K} e_j \log_2(e_j) \tag{4.6}$$

Αυτό που μπορεί κανείς να παρατηρήσει είναι ότι αν κάποιο υπο-παράθυρο έχει υψηλή ενέργεια, τότε μία από της πιθανότητες της ακολουθίας e_j θα είναι υψηλή, με αποτέλεσμα η εντροπία της ακολουθίας (4.6) να μειωθεί. Επομένως, το χαρακτηριστικό αυτό παρουσιάζει μικρότερη τιμή αν υπάρχουν απότομες αλλαγές της ενέργειας του παραθύρου, για αυτό και χρησιμοποιείται για την ανίχνευση σημαντικών ενεργειακών αλλαγών.

Έτσι, είναι ένα χαρακτηριστικό που μπορεί να βοηθήσει έναν ταξινομητή στην εκμάθηση διαχωρισμού κλάσεων. Αν για παράδειγμα, θέλαμε να διαχωρίσουμε τα είδη μουσικής ανάμεσα σε ηλεκτρονική και κλασσική, η εντροπία της ενέργειας πιθανότατα να έδινε μικρότερη τιμή για τα δείγματα ηλεκτρονικής μουσικής, τα οποία παρουσιάζουν απότομες αλλαγές, απ' ότι για τα δείγματα κλασσικής μουσικής.

Ποσοστό μηδενικής διέλευσης

Το **ποσοστό μηδενικής διέλευσης (zero-crossing rate)** απεικονίζει το ποσοστό κατά το οποίο το πλάτος ενός ηχητικού σήματος αλλάζει πρόσημο μέσα σε ένα τμήμα (frame), δηλαδή γίνεται από θετικό, μηδέν και αρνητικό ή το αντίστροφο. Μαθηματικά ορίζεται από την παρακάτω έκφραση:

$$Z(i) = \frac{1}{2W_L} \sum_{n=1}^{W_L} |sgn[x_i(n)] - sgn[x_i(n-1)]| \tag{4.7}$$

όπου το **sgn** είναι η συνάρτηση sign που δίνει 1 όταν το όρισμά της είναι μη αρνητικό και -1 όταν το όρισμά της είναι αρνητικό:

$$sgn[x_i(n)] = \begin{cases} 1, x_i(n) \geq 0 \\ -1, x_i(n) < 0 \end{cases}$$

επίσης η έκφραση έχει διαιρεθεί και πάλι με το μήκος του παραθύρου W_L.

Το ποσοστό μηδενικής διέλευσης είναι ένα χαρακτηριστικό που μπορεί να δείξει το κατά πόσο θορυβώδες είναι ένα σήμα. Σε θορυβώδη τμήματα ενός σήματος, παρουσιάζει μεγαλύτερη τιμή απ' ότι σε μη θορυβώδη. Επίσης, χρησιμοποιείται συχνά ως χαρακτηριστικό στην αναγνώριση ομιλίας (speech recognition) για την ανίχνευση φωνητικής δραστηριότητας (voice activity detection). Από την άλλη, αν πάρουμε την τυπική απόκλιση αυτού του χαρακτηριστικού, όπως κάναμε και στο προηγούμενο, θα παρατηρήσουμε στην περίπτωση της ομιλίας, μεγαλύτερη τιμή από ότι στην περίπτωση της μουσικής. Αυτό γιατί η ομιλία έχει μεγαλύτερο ποσοστό εναλλαγής προσήμου του σήματος. Επομένως, το χαρακτηριστικό αυτό μπορεί να αποβεί αρκετά πληροφοριακό και βοηθητικό για τους ταξινομητές, όπως στην περίπτωση της δυαδικής ταξινόμησης ανάμεσα σε ομιλία και μουσική.

4.1.3.2 Φασματικά Χαρακτηριστικά

Για την εξαγωγή φασματικών χαρακτηριστικών θα πρέπει πρώτα να υπολογίσουμε τον **διακριτό μετασχηματισμό Fourier (discrete Fourier transform - DFT)** των ηχητικών παραθύρων [43] [44].

Διακριτός Μετασχηματισμός Fourier

Ο μετασχηματισμός αυτός παράγει μια φασματική αναπαράσταση του σήματος. Μετατρέπει, δηλαδή, μια πεπερασμένη ακολουθία δειγμάτων, ισοδύναμης απόστασης, μίας συνάρτησης, σε μια ακολουθία ίδιου μήκους, αποτελούμενη από ισοδύναμης απόστασης δείγματα μιας πολύπλοκης συνάρτησης συχνότητας. Αυτή η συνάρτηση συχνότητας είναι ο **μετασχηματισμός Fourier διακριτού-χρόνου (discrete-time Fourier transform - DTFT)**. Το διάστημα στο οποίο γίνεται δειγματοληψία του DTFT είναι το αντίστροφο της διάρκειας της ακολουθίας εισόδου.

Επομένως, ο DFT είναι ο μετασχηματισμός, δηλαδή ο τρόπος με τον οποίο γίνεται η μετατροπή σε φασματική αναπαράσταση (πεπερασμένη ακολουθία δειγμάτων), ενώ ο

DTFT είναι η συνάρτηση συχνότητας από την οποία εξάγεται η ακολουθία του αποτελέσματος. Εάν η αρχική ακολουθία εκτείνεται σε όλες τις μη μηδενικές τιμές μιας συνάρτησης, ο DTFT της είναι συνεχής (και περιοδικός) και ο DFT παρέχει διακριτά δείγματα ενός κύκλου. Εάν η αρχική ακολουθία είναι ένας κύκλος μιας περιοδικής συνάρτησης, ο DFT παρέχει όλες τις μη μηδενικές τιμές ενός κύκλου της DTFT.

Σημειώνουμε ότι ένας αποτελεσματικός αλγόριθμος για τον υπολογισμό των συντελεστών του μετασχηματισμού αυτού, είναι ο **γρήγορος μετασχηματισμός Fourier (fast Fourier transform - FFT)**.

Ο DTFT μιας ακολουθίας εισόδου x(n),n=0,...,N-1, δίνεται ως εξής:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} x(n)e^{(-jn\omega)} \tag{4.8}$$

Το $\boldsymbol{X(e^{j\omega})}$ που δίνεται από την παραπάνω σχέση είναι μια συνεχής συνάρτηση του ω. Ως εκ τούτου, ένας ψηφιακός υπολογιστής δεν μπορεί να χρησιμοποιήσει απευθείας την σχέση αυτήν για να αναλύσει το $x(n)$. Ωστόσο, μπορούμε να χρησιμοποιήσουμε δείγματα του $X(e^{j\omega})$, για να βρούμε μια προσέγγιση του φάσματος του $x(n)$. Η ιδέα του να δειγματοληπτήσουμε το $X(e^{j\omega})$ σε σημεία με ίσες αποστάσεις είναι στην πραγματικότητα η βάση του DFT.

Όπως συζητήθηκε παραπάνω, το DFT βασίζεται στη δειγματοληψία του DTFT, που δίνεται από την εξίσωση (4.8), σε σημεία ίσων αποστάσεων. Το $X(e^{j\omega})$ είναι μια περιοδική συνάρτηση στο ω με περίοδο 2π. Εάν πάρουμε N δείγματα σε κάθε περίοδο του $X(e^{j\omega})$, η απόσταση μεταξύ των σημείων συχνότητας θα είναι $\frac{2\pi}{N}$. Ως εκ τούτου, η συχνότητα του συνόλου των ημιτονοειδών που αναζητούμε, για να μετατρέψουμε το x(n) σε φασματική μορφή, θα είναι της μορφής $\frac{2\pi}{N} \times$ k, όπου μπορούμε να επιλέξουμε k = 0,1, ..., N-1, με k το εκάστοτε σημείο/δείγμα συχνότητας της ακολουθίας. Χρησιμοποιώντας σύνθετα εκθετικά παρόμοια με της εξίσωσης (4.8), οι συναρτήσεις βάσης θα είναι $e^{(-j\frac{2\pi}{N}kn)}$.

Επομένως, με λίγα λόγια, ο DFT μετατρέπει μια χρονική ακολουθία/σήμα από N δείγματα x(n),n=0,...,N-1, σε μια άλλη φασματική ακολουθία πολύπλοκων αριθμών X(k), k=0,...,N-1 ως εξής:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{(-j\frac{2\pi}{N}kn)}, k = 0, ..., N-1 \tag{4.9}$$

Τελικά τα **X(k)** είναι μια ακολουθία από N συντελεστές. Το k είναι ένα εκ των N δειγμάτων στον τομέα συχνότητας (frequency-domain).

Ο **αντίστροφος DFT (inverse DFT - IDFT)**, κάνει την αντίστροφη διαδικασία, παρέχοντας την ακριβή ανακατασκευή του αυθεντικού σήματος x(n). Ουσιαστικά, είναι μια σειρά Fourier, που χρησιμοποιεί τα δείγματα DTFT ως συντελεστές σύνθετων ημιτονοειδών στις αντίστοιχες συχνότητες DTFT.

Παρακάτω φαίνεται ο αντίστροφος DTFT (inverse of DTFT):

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})e^{jn\omega}d\omega, n = 0, ..., N-1 \tag{4.10}$$

Παρακάτω φαίνεται ο αντίστροφος DFT (inverse DFT - IDFT):

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)e^{(j\frac{2\pi}{N}kn)}, n = 0, ..., N-1 \tag{4.11}$$

όπου το X(k) δηλώνει τον συντελεστή/βάρος που χρησιμοποιείται για το σύνθετο εκθετικό $e^{(j\frac{2\pi}{N}kn)}$. Το n είναι ένα εκ των N δειγμάτων στον τομέα χρόνου (time-domain).

Με μια άλλη προσέγγιση, αν θέλαμε να βρούμε την σχέση (4.11) με την οποία αποκτάμε ένα σήμα διακριτού χρόνου από την αντίστοιχη φασματική αναπαράστασή του, θα μπορούσαμε να σκεφτούμε ως εξής:

1. Το μόνο που χρειαζόμαστε είναι να φτιάξουμε ένα περιοδικό σήμα από τα N δείγματα της -πεπερασμένης διάρκειας- ακολουθίας x(n). Έτσι, για να υπολογίσουμε τον DFT N-σημείων, φτιάχνουμε ένα περιοδικό σήμα p(n) το οποίο είναι ίσο με το x(n) για n=0,1,..,N-1.

2. Στην συνέχεια εφαρμόζοντας την επέκταση της σειράς Fourier διακριτού χρόνου (discrete-time Fourier series), μπορούμε να καταλήξουμε σε μια -φασματικού επιπέδου- αναπαράσταση του περιοδικού σήματος, η οποία ορίζεται ως εξής:

$$\alpha_k = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{(-j\frac{2\pi}{N}kn)} \tag{4.12}$$

   όπου το N είναι η περίοδος του σήματος.

3. Από την παραπάνω σχέση, το σήμα χρονικού τομέα (time-domain) ορίζεται ως εξής:

$$x(n) = \sum_{k=0}^{N-1} \alpha_k e^{(j\frac{2\pi}{N}kn)} \tag{4.13}$$

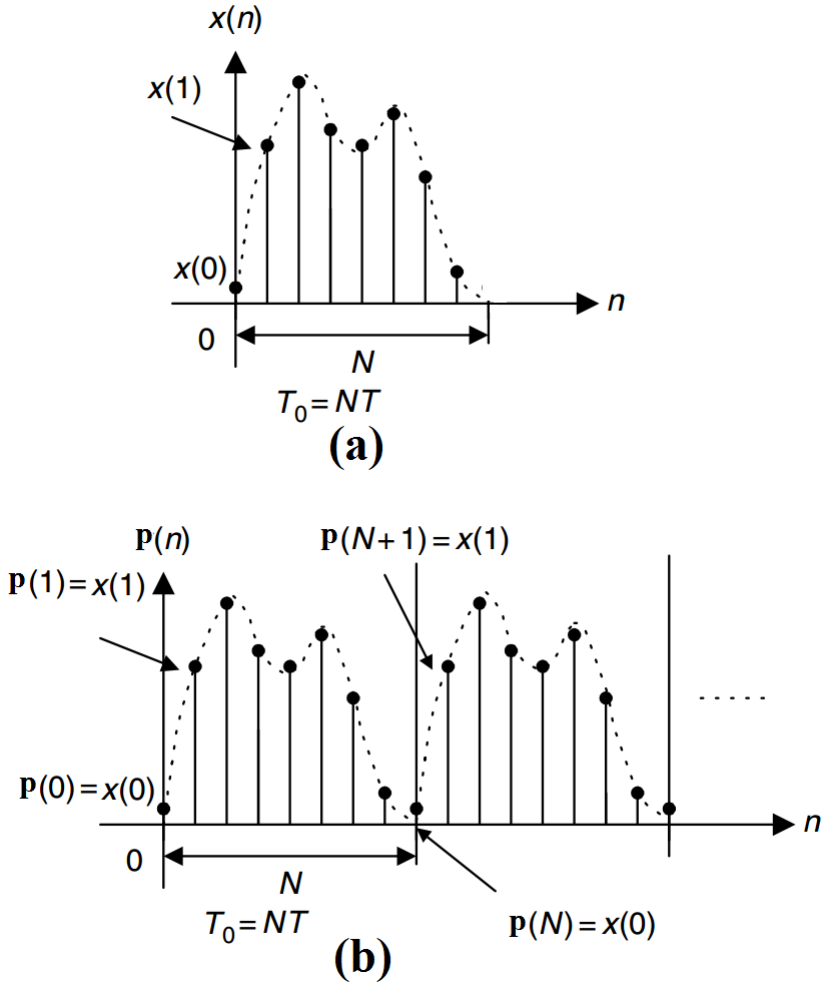
4. Τα α_k διαφέρουν από τους συντελεστές DFT που ορίζονται στην σχέση (4.9), μόνο κατά τον παράγοντα $\frac{1}{N}$. Αυτό γιατί η σειρά Fourier διακριτού χρόνου είναι περιοδική, ενώ οι συντελεστές DFT, X(k), είναι πεπερασμένης διάρκειας ακολουθία. Επομένως, πολλαπλασιάζοντας την σχέση (4.12) με το N, καταλήγουμε στα X(k):

$$X(k) = N\alpha_k = \sum_{n=0}^{N-1} x(n)e^{(-j\frac{2\pi}{N}kn)} \tag{4.14}$$

5. Εν τέλει, ο αντίστροφος DFT υπολογίζεται σύμφωνα με τις δύο παραπάνω σχέσεις (4.13) (4.14) ως εξής:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{(j\frac{2\pi}{N}kn)}$$

Στο παρακάτω σχήμα απεικονίζονται οι διεργασίες που ακολουθήθηκαν στο βήμα 1, δηλαδή η αρχική ακολουθία διακριτού χρόνου x(n) από την οποία δημιουργούμε το περιοδικό σήμα p(n) [43]:



Σχήμα 4.7: (a)Η πεπερασμένη ακολουθία x(n). (b) Το περιοδικό σήμα p(n), που αποκάταται από το x(n).

Τέλος, μια σημαντική παρατήρηση που μπορούμε να κάνουμε είναι η εξής: η δειγματοληψία ενός σήματος στον τομέα χρόνου οδηγεί σε αντίγραφα του σήματος προέλευσης στον τομέα συχνότητας. Αντίστοιχα η δειγματοληψία $X(e^{j\omega})$ (4.8) στον τομέα συχνότητας οδηγεί σε αντίγραφα του x(n) στον τομέα χρόνου.

Φασματικό Κεντροειδές

Έχοντας πλέον αναλύσει τους συντελεστές DFT, είμαστε έτοιμοι να παρουσιάσουμε όλα τα φασματικά χαρακτηριστικά, τα οποία τους χρησιμοποιούν.

Ένα από αυτά είναι και το **φασματικό κεντροειδές (spectral centroid)**. Το φασματικό κεντροειδές είναι το κέντρο "βαρύτητας" του φάσματος. Δείχνει, δηλαδή, πού βρίσκεται το κέντρο μάζας του φάσματος. Αντίστοιχα, έχει μια ισχυρή σύνδεση με την εντύπωση της φωτεινότητας ενός ήχου (brightness of a sound). Η φωτεινότητα αντανακλά την ποσότητα πληροφοριών υψηλής συχνότητας και μετράται συσχετίζοντας την ενέργεια πάνω από μια προκαθορισμένη συχνότητα αποκοπής με τη συνολική ενέργεια. Έτσι, υψηλότερες τιμές φασματικού κεντροειδούς αντιστοιχούν σε φωτεινότερους ήχους. Για παράδειγμα, ένας παρασκηνιακός θόρυβος ή μια σιωπή θα έχουν μικρότερες τιμές αυτού του στατιστικού από κάποιους απότομους ήχους.

Το φασματικό κεντροειδές C_i του ιοστού ηχητικού παραθύρου (short-term window) ορίζεται ως η μέση συχνότητα σταθμισμένη με τα πλάτη, διαιρούμενη με το άθροισμα των πλατών:

$$C_i = \frac{\sum_{k=1}^{Wf_L} kX_i(k)}{\sum_{k=1}^{Wf_L} X_i(k)} \tag{4.15}$$

όπου το $X_i(k)$ με k=1,...,Wf_L είναι το μέγεθος των DFT συντελεστών του ιοστού παραθύρου, ή αλλιώς το πλάτος που αντιστοιχεί στο bin k στο φάσμα DFT, και το k είναι ένα εκ των Wf_L δειγμάτων στον τομέα συχνότητας (frequency-domain).

Σημειώνεται πως κανονικά το μήκος του βραχυπρόθεσμου παραθύρου είναι W_L, όσος και ο αριθμός των DFT συντελεστών του παραθύρου. Όμως, τα φασματικά χαρακτηριστικά αρκούνται στο να χρησιμοποιούν μόνο το πρώτο μισό αυτών των συντελεστών, αφού το δεύτερο μισό εξυπηρετεί στην ανακατασκευή του αρχικού σήματος. Έτσι το Wf_L είναι τελικά, ο αριθμός των συντελεστών που χρησιμοποιούνται.

Φασματική Εξάπλωση

Η **φασματική εξάπλωση (spectral spread)**, είναι ένα χαρακτηριστικό που δείχνει την μέση απόκλιση του φάσματος από το φασματικό κεντροειδές, η οποία συνήθως σχετίζεται με το εύρος ζώνης του σήματος. Ορίζεται ως εξής:

$$S_i = \sqrt{\frac{\sum_{k=1}^{Wf_L}(k - C_i)^2 X_i(k)}{\sum_{k=1}^{Wf_L} X_i(k)}} \tag{4.16}$$

Η φασματική εξάπλωση μετράει πως το φάσμα εξαπλώνεται γύρω από τον κεντροειδή. Έτσι, χαμηλές τιμές της φασματικής εξάπλωσης, αντιστοιχούν σε σήματα που το φάσμα τους είναι σφιχτά συγκεντρωμένο γύρω από το φασματικό κεντροειδές. Τα σήματα που μοιάζουν με θόρυβο έχουν συνήθως μεγάλη φασματική εξάπλωση, ενώ μεμονωμένοι τονικοί ήχοι με απομονωμένες κορυφές θα οδηγήσουν σε χαμηλή φασματική εξάπλωση. Για παράδειγμα, αν συγκρίναμε την τιμή της φασματικής εξάπλωσης για ένα παράθυρο ηλεκτρονικής μουσικής, πιθανότατα να παίρναμε μεγαλύτερη τιμή σε σύγκριση με αντίστοιχα παράθυρα κλασσικής ή jazz μουσικής. Αυτό γιατί η ηλεκτρονική μουσική έχει συνήθως μια πιο ευρεία εξάπλωση γύρω από τον φασματικό κεντροειδή.

Φασματική Εντροπία

Η **φασματική εντροπία (spectral entropy)**, υπολογίζεται με αντίστοιχο τρόπο όπως η εντροπία ενέργειας που είδαμε στα χρονικά χαρακτηριστικά, με την διαφορά ότι λαμβάνει χώρα στον τομέα συχνότητας.

Αρχικά το φάσμα κάθε βραχυπρόθεσμου παραθύρου χωρίζεται σε L υπο-παράθυρα (sub-bands/bins). Στην συνέχεια υπολογίζεται η ενέργεια του f υπο-παραθύρου για f=0,...,L-1 η οποία κανονικοποιείται από την συνολική φασματική ενέργεια. Η σχέση αυτή ορίζεται ως εξής:

$$n_f = \frac{E_f}{\sum_{f=0}^{L-1} E_f} \tag{4.17}$$

όπου n_f είναι η κανονικοποιημένη φασματική ενέργεια του f υπο-παραθύρου.

Στην συνέχεια υπολογίζεται η εντροπία αυτής την ενέργειας για όλα τα υπο-παράθυρα ως εξής:

$$H = -\sum_{f=0}^{L-1} n_f \log_2(n_f) \tag{4.18}$$

Η εντροπία μπορεί να χρησιμοποιηθεί για να αναπαραστήσει το πόσες κορυφές έχει η φασματική αναπαράσταση (peakiness). Το προκύπτον χαρακτηριστικό είναι χαμηλό για ένα φάσμα με πολλές διαφορετικές φασματικές κορυφές και υψηλό για ένα επίπεδο φάσμα. Αυτό μπορεί να δικαιολογηθεί από το γεγονός ότι αν έχουμε υψηλή φασματική ενέργεια σε κάποιο υπο-παράθυρο, η πιθανότητα n_f στο υπο-παράθυρο αυτό αυξάνεται και ως εκ τούτου, η εντροπία μειώνεται για το συνολικό βραχυπρόθεσμο παράθυρο. Επομένως, όσο περισσότερα -υψηλής ενέργειας- υπο-παράθυρα έχουμε (peaks), τόσο μικρότερη θα είναι η τιμή της εντροπίας και αντιστρόφως.

Ως ένα παράδειγμα, αν υπολογίζαμε την τυπική απόκλιση των ακολουθιών της φασματικής εντροπίας κάποιων τμημάτων-παραθύρων ομιλίας ή περιβαλλοντικών ήχων, τότε πολύ πιθανόν η τιμή αυτή να ήταν χαμηλότερη για τους περιβαλλοντικούς ήχους και υψηλότερη για την ομιλία. Αυτό γιατί η ομιλία παρουσιάζει πολλές διαφορετικές φασματικές κορυφές και αλλαγές. Ως εκ τούτου, η φασματική εντροπία αλλάζει από παράθυρο σε παράθυρο (τα παράθυρα με περισσότερες κορυφές έχουν πιο χαμηλή φασματική εντροπία, ενώ τα παράθυρα χωρίς πολλές κορυφές έχουν πιο υψηλή φασματική εντροπία), με αποτέλεσμα η τυπική απόκλιση να μεγαλώνει. Αντίθετα, στην περίπτωση των περιβαλλοντικών ήχων, παρουσιάζεται ένα πιο ομαλοποιημένο φάσμα (όχι απότομες κορυφές), το οποίο δίνει μικρή τυπική απόκλιση φασματικής εντροπίας.

Φασματική Ροή

Η **φασματική ροή (spectral flux)** είναι ένα μέτρο της αλλαγής του φάσματος ισχύος, μεταξύ δύο διαδοχικών πλαισίων/παραθύρων, το οποίο υπολογίζεται ως η τετραγωνική διαφορά μεταξύ δύο κανονικοποιημένων μεγεθών του φάσματος. Η μαθηματική έκφρασή της ορίζεται ως εξής:

$$Fl_{(i,i-1)} = \sum_{k=1}^{Wf_L} (EN_i(k) - EN_{i-1}(k))^2 \tag{4.19}$$

όπου το $EN_i(k)$ είναι ο k κανονικοποιημένος DFT συντελεστής του i-οστού πλαισίου/παραθύρου και ορίζεται ως εξής:

$$EN_i(k) = \frac{X_i(k)}{\sum_{l=1}^{Wf_L} X_i(l)} \tag{4.20}$$

Ως ένα παράδειγμα, αν συγκρίναμε την μέση τιμή των ακολουθιών φασματικών ροών που έχουν υπολογιστεί από τμήματα (segments), δύο κλάσεων ήχου (μουσικής και ομιλίας), τότε το πιο πιθανό είναι ότι οι τιμές της φασματικής ροής θα ήταν υψηλότερες για την κλάση της ομιλίας. Αυτό γιατί η ομιλία περιέχει πολλές τοπικές φασματικές αλλαγές λόγω της γρήγορης εναλλαγής των φωνημάτων.

Φασματική Εξασθένηση

Η **φασματική εξασθένηση (spectral roll-off)**, είναι η συχνότητα που ορίζεται για κάθε πλαίσιο/παράθυρο ως η κεντρική συχνότητα για ένα bin του φασματογραφήματος, ώστε ένα συγκεκριμένο ποσοστό (roll percentage) της ενέργειας του φάσματος σε αυτό το πλαίσιο, περιέχεται σε αυτό το bin και στα μικρότερα bins. Οι συνήθεις τιμές που χρησιμοποιούνται ως ποσοστό είναι μεταξύ του 0.85 και του 0.95.

Δίνοντας έναν μαθηματικό ορισμό της φασματικής εξασθένησης, αν ο m-οστός DFT συντελεστής αντιστοιχεί στην φασματική εξασθένηση του i-οστού πλαισίου, τότε ικανοποιεί την παρακάτω εξίσωση:

$$\sum_{k=1}^{m} X_i(k) = C \sum_{k=1}^{Wf_L} X_i(k) \tag{4.21}$$

όπου το C είναι το προαναφερθέν ποσοστό. Ουσιαστικά, αν είχαμε ορίσει C=90%, τότε η παραπάνω εξίσωση μας δείχνει πως για να είναι το m-οστό bin η συχνότητα που ψάχνουμε (δηλαδή η φασματική εξασθένηση), θα πρέπει το άθροισμα όλων των DFT συντελεστών από το bin 1 μέχρι το bin m να ισούται με το 90% του αθροίσματος όλων των DFT συντελεστών μέσα στο εκάστοτε πλαίσιο (i).

Η συχνότητα φασματικής εξασθένησης, συνήθως κανονικοποιείται διαιρώντας την με το μήκος του παραθύρου Wf_L, παίρνοντας έτσι τιμές από 0 έως 1. Η τιμή 1 αντιστοιχεί στην μέγιστη συχνότητα του σήματος.

Η φασματική εξασθένηση περιγράφει το σχήμα του φάσματος ενός ηχητικού σήματος και μπορεί επίσης να χρησιμοποιηθεί για την διαφοροποίηση μεταξύ φωνητικών (voiced) και μη-φωνητικών (unvoiced) ήχων.

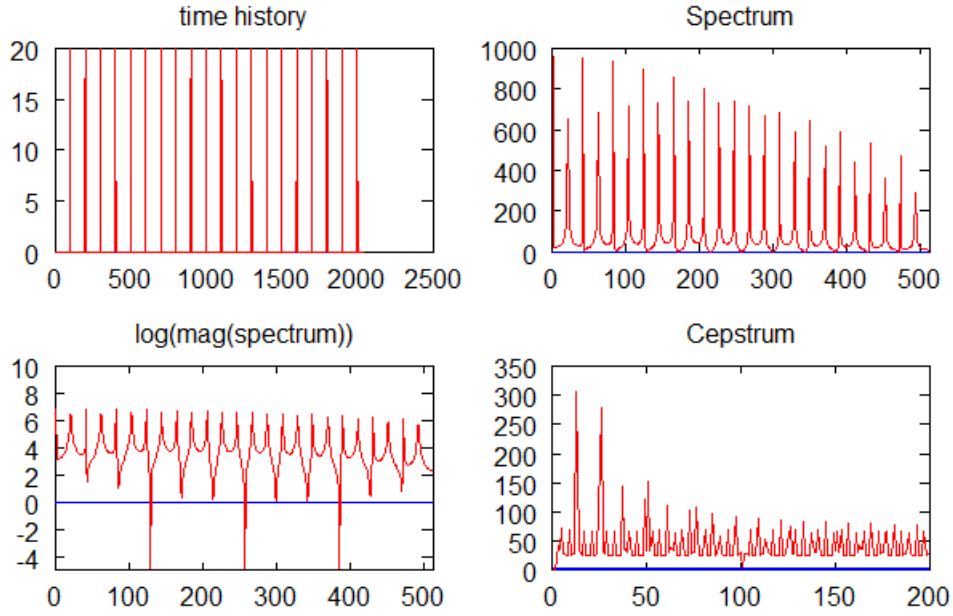
Ως ένα παράδειγμα, αν υπολογίζαμε την φασματική εξασθένηση ακολουθιών τμημάτων ήχου από διαφορετικές κλάσεις όπως η κλασσική μουσική, η ηλεκτρονική μουσική και η jazz μουσική, πιθανότατα να παρατηρούσαμε μεγαλύτερες τιμές στα κομμάτια της ηλεκτρονικής μουσικής απ' ότι στα υπόλοιπα. Αυτό συμβαίνει γιατί το μεγαλύτερο ποσοστό της ενέργειας του φάσματος, της ηλεκτρονικής μουσικής, παρουσιάζεται κάτω από υψηλές συχνότητες και κατά συνέπεια οι τιμές των φασματικών εξασθενήσεών του είναι υψηλότερες. Αντίθετα, η κλασσική και η jazz μουσική, συγκεντρώνουν την φασματική τους ενέργεια σε χαμηλότερες συχνότητες κατά κύριο λόγο. Επίσης, η απόκλιση της φασματικής εξασθένησης, θα ήταν πιο έντονη στην περίπτωση της ηλεκτρονικής μουσικής, αφού έχει ένα ευρύτερο φάσμα.

MFCCs

Οι **συντελεστές Mel-Frequency Cepstrum (Mel-Frequency Cepstrum Coefficients - MFCCs)** είναι πολύ διάσημες στον τομέα της ανάλυσης της ομιλίας και είναι ένα είδος **cepstral** αναπαράστασης του σήματος.

Πριν μπούμε σε λεπτομέρειες των MFCCs, θα ήταν χρήσιμο σε αυτό το σημείο να εξηγήσουμε τι ακριβώς είναι το **cepstrum**. Το cepstrum είναι η πληροφορία του ρυθμού αλλαγής των φασματικών ζωνών. Το cepstrum προέρχεται από την λέξη spectrum (φάσμα), αντιστρέφοντας τα τέσσερα πρώτα γράμματα (spec –> ceps). Τα περιοδικά σήματα στον τομέα του χρόνου, απεικονίζονται ως αιχμηρές κορυφές στον τομέα συχνότητας (φάσμα συχνότητας), έπειτα από εφαρμογή του μετασχηματισμού Fourier όπως έχουμε δει. Παίρνοντας τον λογάριθμο των πλατών του φάσματος και έπειτα υπολογίζοντας εκ νέου το φάσμα του λογαρίθμου αυτού μέσω συνημιτονοειδούς μετασχηματισμού, το νέο φάσμα εμφανίζει μια κορυφή οποτεδήποτε υπάρχει ένα περιοδικό στοιχείο στο αρχικό χρονικό σήμα. Αυτό το νέο φάσμα δεν βρίσκεται ούτε στον τομέα του χρόνου (time domain), ούτε στον τομέα της συχνότητας (frequency domain). Έτσι οι Bogert et al [45] το ονόμασαν "quefrency domain". Cepstrum, λοιπόν, είναι το φάσμα του λογαρίθμου του φάσματος (spectrum) του χρονικού σήματος.

Το παρακάτω σχήμα παρουσιάζει τα βήματα που ακολουθήθηκαν για την παραγωγή του cepstrum [46]:



Σχήμα 4.8: Βήματα για τον Σχηματισμό Cepstrum από το Ιστορικό Χρόνου

Η επόμενη παράμετρος που θα πρέπει να αναλύσουμε είναι το Mel-frequency. Ο όρος αυτός αναφέρεται σε συχνότητες οι οποίες έχουν κλιμακωθεί σύμφωνα με την **κλίμακα Mel (Mel scale)**. Η κλίμακα αυτή συσχετίζει την συχνότητα του τόνου η οποία γίνεται αντιληπτή από το ανθρώπινο αυτί, με την πραγματική συχνότητα. Ουσιαστικά, κλιμακώνει την συχνότητα έτσι ώστε να είναι πιο κοντά σε αυτό που το ανθρώπινο αυτί μπορεί να ακούσει. Για παράδειγμα, είναι γνωστό ότι οι άνθρωποι μπορούν να ξεχωρίσουν συχνότητες πιο εύκολα στην περιοχή χαμηλών συχνοτήτων. Έτσι, αν ο άνθρωπος ακούσει δύο ήχους στα 300 Hz και 400Hz αντίστοιχα και δύο ήχους στα 900Hz και 1kHz αντίστοιχα, τότε η αντιληπτή απόσταση ανάμεσα στους δύο τελευταίους ήχους, πιθανότατα να μοιάζει πολύ μεγαλύτερη από αυτήν ανάμεσα στους δύο πρώτους ήχους, παρ' ότι είναι η ίδια (100Hz).

Με άλλα λόγια, η κλίμακα Mel είναι μια αντιληπτή κλίμακα διαστημάτων συχνότητας, που αν κριθούν από την ανθρώπινη ακοή, γίνονται αντιληπτά ως διαστήματα ίσης απόστασης.

Υπάρχουν διάφοροι **τύποι (formula)** για την μετατροπή σε κλίμακα Mel ή αλλιώς **συναρτήσεις στρέβλωσης συχνότητας (frequency warping functions)** που έχουν προταθεί ανά τα χρόνια. Ένας γνωστός τέτοιος τύπος είναι ο παρακάτω [47]:

$$m = 2595 \log_{10}(1 + \frac{f}{700}) = 1127 \ln(1 + \frac{f}{700}) \tag{4.22}$$

όπου **m** είναι η συχνότητα σε Mels και **f** η συχνότητα σε Hz.

Ο παραπάνω τύπος μπορεί να οριστεί και για την αντίστροφη διαδικασία (από Mels σε Hz) ως εξής:

$$f = 700(10^{\frac{m}{2595}} - 1) = 700(e^{\frac{m}{1127}} - 1) \tag{4.23}$$

Έχοντας εξηγήσει τις βασικές έννοιες, τα βήματα που ακολουθούνται για την παραγωγή των χαρακτηριστικών MFCCs είναι τα εξής:

1. Υπολογίζεται ο DFT του χρονικού σήματος για κάθε παράθυρο (short-term window), αφού όπως έχουμε πει η εξαγωγή των χαρακτηριστικών γίνεται σε επίπεδο βραχυπρόθεσμων παραθύρων. Στα επόμενα βήματα φαίνεται η διαδικασία που ακολουθείται για ένα παράθυρο.

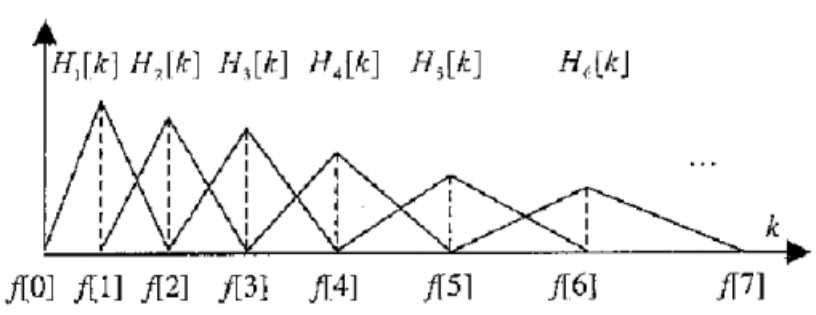
2. Το φάσμα που εξάγεται μετά τον DFT, δίνεται ως είσοδος σε μια **τράπεζα φίλτρων** κλίμακας Mel, η οποία αποτελείται από M φίλτρα, που συνήθως είναι επικαλυπτόμενες τριγωνικές αποκρίσεις συχνότητας.

   Για να δημιουργήσουμε αυτά τα φίλτρα, πρώτα ορίζουμε την χαμηλότερη (f_1) και την υψηλότερη (f_h) συχνότητα της τράπεζας φίλτρων σε Hz. Στην συνέχεια μετατρέπουμε αυτές τις συχνότητες σε Mels σύμφωνα με την σχέση (4.22) και ανάλογα με τον αριθμό των φίλτρων που θα χρησιμοποιήσουμε, χωρίζουμε το εύρος ($f_h - f_1$) σε σημεία γραμμικής απόστασης (ίσης απόστασης). Κάθε φίλτρο στην τράπεζα φίλτρων, όπως έχουμε πει, είναι τριγωνικό με απόκριση 1 στην κεντρική συχνότητα που μειώνεται γραμμικά προς το 0 έως ότου φτάσει στις κεντρικές συχνότητες των δύο γειτονικών φίλτρων όπου η απόκριση γίνεται 0. Έτσι, ο αριθμός των σημείων στα οποία πρέπει να χωρίσουμε το εύρος είναι M + 2 , όπου M ο αριθμός των φίλτρων. Αν για παράδειγμα έχουμε 10 φίλτρα, τότε χωρίζουμε σε 12 σημεία. Συνήθως, στην πραγματικότητα χρησιμοποιούνται 26-40 φίλτρα. Στην συνέχεια μετατρέπουμε τα Mels σε Hz σύμφωνα με την σχέση (4.23) και τέλος δημιουργούμε τα φίλτρα ως εξής:

$$H_m(k) = \begin{cases} 0 & , k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & , f(m-1) \le k \le f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & , f(m) \le k \le f(m+1) \\ 0 & , k > f(m+1) \end{cases} \tag{4.24}$$

   όπου **m** είναι ο αριθμός του φίλτρου, **f(.)** είναι η λίστα των M+2 mel-spaced συχνοτήτων που δημιουργήσαμε παραπάνω και **k** είναι τα σημεία του αρχικού spectrum (συχνότητες).

   Ένα παράδειγμα φίλτρων φαίνεται παρακάτω:



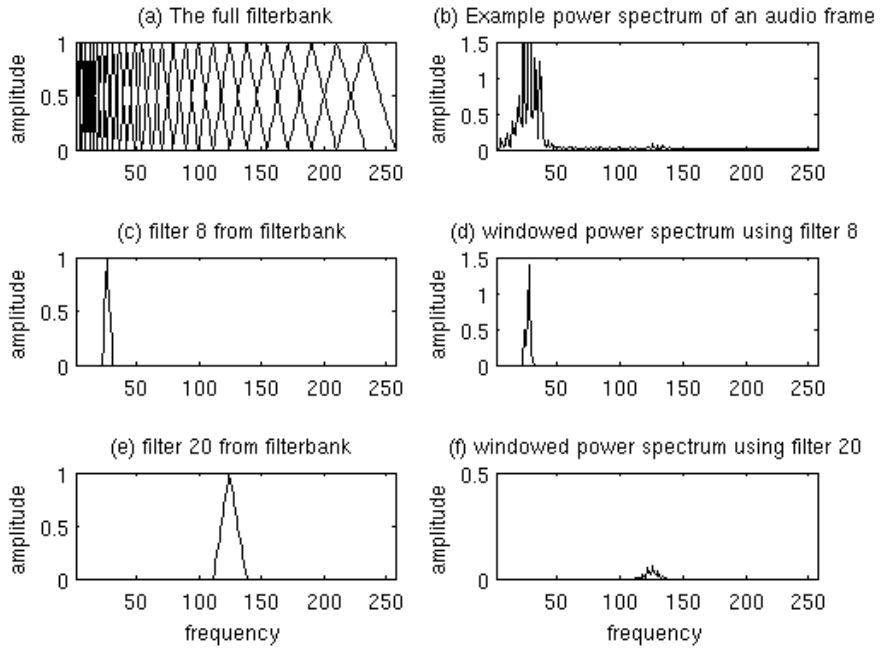
Σχήμα 4.9: Τριγωνικά Φίλτρα για τον Υπολογισμό του Mel-Cepstrum

   Το mel spectrum που δημιουργείται μετά την εφαρμογή των φίλτρων δίνεται από την παρακάτω εξίσωση:

$$s(m) = \sum_{k=0}^{N-1} [|X(k)|^2 H_m(k)] \quad , 0 \le m \le M-1 \tag{4.25}$$

   όπου **X(k)** είναι τα πλάτη του φάσματος (συντελεστές), $\boldsymbol{H_m(k)}$ είναι οι μαθηματικές εκφράσεις των φίλτρων και **m** καθένα εκ των M συνολικά φίλτρων.

   Ένα παράδειγμα εφαρμογής των φίλτρων πάνω στο spectrum (φάσμα), φαίνεται παρακάτω [48]:



Σχήμα 4.10: Mel Φίλτρα πάνω σε παράθυρα του φάσματος ισχύος

3. Στο επόμενο βήμα υπολογίζουμε τον λογάριθμο για καθεμία από τις ενέργειες που εξάγουν τα φίλτρα.

4. Τέλος, υπολογίζουμε τον **διακριτό συνιμιτονοειδή μετασχηματισμό (discrete cosine transform - DCT)** ή αλλιώς τον **γρήγορο μετασχηματισμό Fourier (Fast Fourier Transform - FFT)** των παραπάνω λογαρίθμων για να καταλήξουμε στους cepstral συντελεστές.

   Η μαθηματική έκφραση των συντελεστών ορίζεται ως εξής:

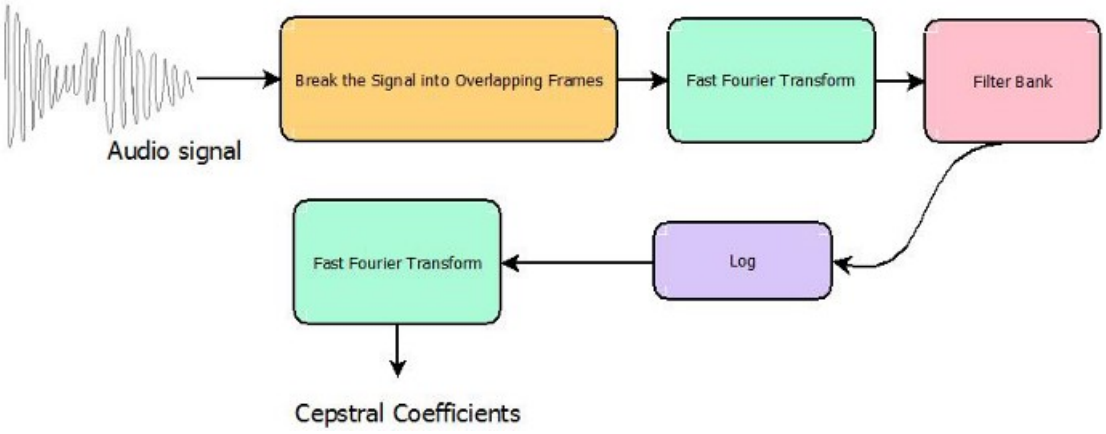
$$c(n) = \sum_{m=0}^{M-1} \log_{10}(s(m)) cos(\frac{\pi n(m-0.5)}{M}) \quad , n = 0, ..., C-1 \tag{4.26}$$

   όπου **C** είναι ο αριθμός των MFCCs συντελεστών που χρησιμοποιούνται.

   Στις περισσότερες εφαρμογές, έχει καθιερωθεί να επιλέγονται οι πρώτοι 12 με 13 συντελεστές, επειδή θεωρείται ότι δίνουν αρκετή πληροφορία στους ταξινομητές για να τους βοηθήσουν με τον διαχωρισμό των κλάσεων.

   Τα τελικά χαρακτηριστικά (13 αριθμοί ανά βραχυπρόθεσμο παράθυρο) ονομάζονται Mel Frequency Coefficients - MFCCs.

Όλα τα παραπάνω βήματα που ακολουθήθηκαν για την εξαγωγή των MFCCs συντελεστών παρουσιάζονται συνοπτικά στο παρακάτω σχήμα [49]:



Σχήμα 4.11: Βήματα για την Εξαγωγή των MFCCs

Διάνυσμα Χρώματος

Το **διάνυσμα χρώματος (chroma vector)** ή αλλιώς **χρωμογράφημα (chromogram)** ή αλλιώς **προφίλ τονικής κλάσης (pich class profile - PCP)** είναι μια αναπαράσταση 12 στοιχείων της φασματικής ενέργειας ενός ηχητικού σήματος.

Για την καλύτερη και ευκολότερη κατανόηση αυτού του χαρακτηριστικού, είναι σημαντικό να αναλυθούν πρωτίστως κάποιες έννοιες.

Ο **τόνος (pitch)** είναι μια αντιληπτική ιδιότητα ήχων που επιτρέπει την ταξινόμησή τους σε κλίμακα που σχετίζεται με τη συχνότητα. Με άλλα λόγια, ο τόνος είναι η ποιότητα που καθιστά δυνατή την εκτίμηση των ήχων ως "υψηλότερων" και "χαμηλότερων" με την έννοια που σχετίζεται με τις μουσικές μελωδίες. Ο τόνος μπορεί να προσδιοριστεί μόνο σε ήχους που έχουν συχνότητα που είναι καθαρή και αρκετά σταθερή ώστε να ξεχωρίζει από το θόρυβο [50].

Στην μουσική, μια **οκτάβα (ovtave)** ή μια τέλεια οκτάβα (perfect octave) είναι το διάστημα ανάμεσα σε έναν μουσικό τόνο και έναν άλλον με διπλάσια συχνότητα [51].

Τονική τάξη (pitch class), είναι ένα σύνολο από όλους τους τόνους που διαφέρουν κατά μία η περισσότερες οκτάβες. Για παράδειγμα, η τονική κλάση C, αποτελείται από τα Cs σε όλες τις οκτάβες και ορίζεται ως εξής: $C_n : nisaninteger = ..., C_{-2}, C_{-1}, C_0, C_1, C_2, C_3, ...$ [52].

Ένα **ίσο μουσικό ταμπεραμέντο (equal temperament)** είναι ένα μουσικό **σύστημα συντονισμού (tunning system)** το οποίο προσεγγίζει τα διαστήματα, χωρίζοντας μια οκτάβα (ή ένα διάστημα), σε ίσα βήματα. Αυτό σημαίνει πως η αναλογία οποιονδήποτε δύο γειτονικών συχνοτήτων νοτών (notes) είναι η ίδια, πράγμα που δίνει την αντίληψη ενός ίσου μεγέθους βήματος, αφού ο τόνος -που είναι η αντιληπτική ιδιότητα- θεωρείται περίπου ίσος με τον λογάριθμο της συχνότητας.

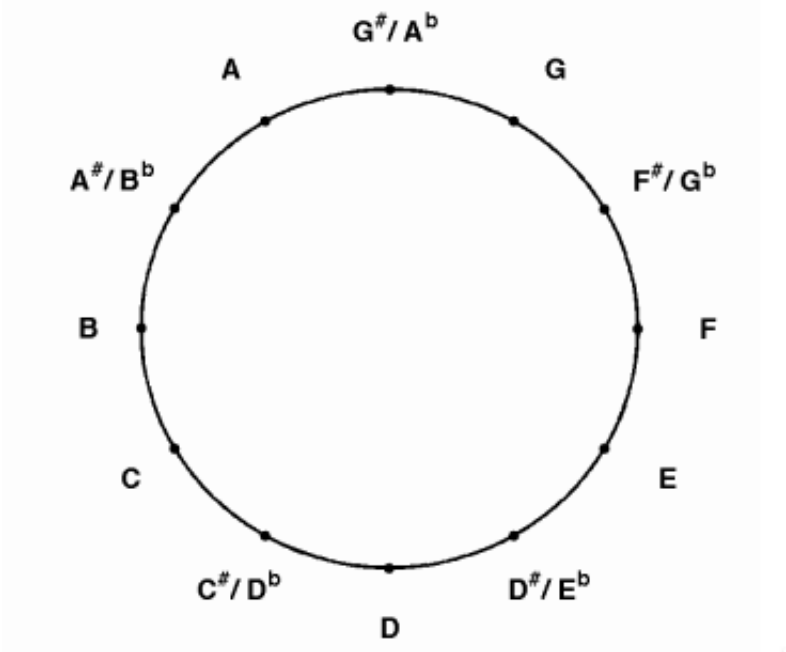
Στην κλασσική μουσική και τη δυτική μουσική (Western-type music) γενικότερα, το πιο κοινό σύστημα συντονισμού είναι το **12-τόνων ίσο ταπεραμέντο (twelve-tone equal temperament / 12-TET)**, το οποίο χωρίζει την οκτάβα σε 12 μέρη που είναι ισοδύναμα σε μια λογαριθμική κλίμακα. Το καθένα από αυτά τα διαστήματα ονομάζεται **ημιτόνιο (semitone)** [53].

Η ανθρώπινη αντίληψη του τόνου είναι περιοδική υπό την έννοια ότι 2 τόνοι γίνονται αντιληπτοί ως όμοιοι σε "χρώμα" (παίζουν έναν όμοιο αρμονικό ρόλο - similar harmonic role), αν διαφέρουν κατά μία ή περισσότερες οκτάβες, όπου στην δική μας περίπτωση, μια οκτάβα ορίζεται ως η απόσταση 12 τόνων.

Ένας τόνος μπορεί να διαχωριστεί σε δύο συστατικά: το **ύψος του τόνου (tone height)** και το **χρώμα (chroma)**. Το ύψος του τόνου αναφέρεται στον αριθμό οκτάβας, ενώ το χρώμα στο αντίστοιχο ορθογραφικό χαρακτηριστικό τόνου. Στην δυτική μουσική

σημειογραφία, τα 12 χαρακτηριστικά τόνου δίνονται από το σύνολο: {C, C#, D, D#, E , F, F#, G, G#, A, A#, B}. Με απαρίθμηση των τιμών χρώματος, ορίζουμε αυτό το σύνολο με δείκτες [0: 11] όπου το c = 0 αναφέρεται στο χρώμα C, c = 1 στο C# και ούτω καθεξής. Σημειώνετε ότι στην **ισότιμα-αντιληπτή κλίμακα (equal-tempered scale)** διαφορετικές ορθογραφίες τόνου, όπως το C# και το Db αναφέρονται στο ίδιο χρώμα.

Στο παρακάτω σχήμα, φαίνεται ο κύκλος του χρώματος (chroma circle), δηλαδή η περιοδικότητα των τόνων που γίνεται αντιληπτή από τον άνθρωπο και η οποία δεν μπορεί να αναπαρασταθεί με μια γραμμική τονική κλίμακα, παρά μόνο με μια ισότιμα-αντιληπτή (equal-tempered scale) κλίμακα που είδαμε παραπάνω [54]:



Σχήμα 4.12: Κύκλος Χρώματος

Τελικά, η κύρια ιδέα των χαρακτηριστικών χρώματος είναι να συγκεντρωθούν όλες οι φασματικές πληροφορίες που σχετίζονται με μια δεδομένη τονική τάξη, σε έναν ενιαίο συντελεστή. Έτσι για τον υπολογισμό του διανύσματος χρώματος, ομαδοποιούνται οι DFT συντελεστές ενός βραχυπρόθεσμου παραθύρου σε 12 σημεία (bins), όπου κάθε σημείο αναπαριστά μία από τις 12 ισότιμης κλίμακας τονική τάξη της δυτικής μουσικής που είδαμε παραπάνω. Στην συνέχεια υπολογίζεται για κάθε σημείο (bin), η μέση τιμή του λογαρίθμου του πλάτους των αντίστοιχων DFT συντελεστών ($\log_{10} X_i(k)$).

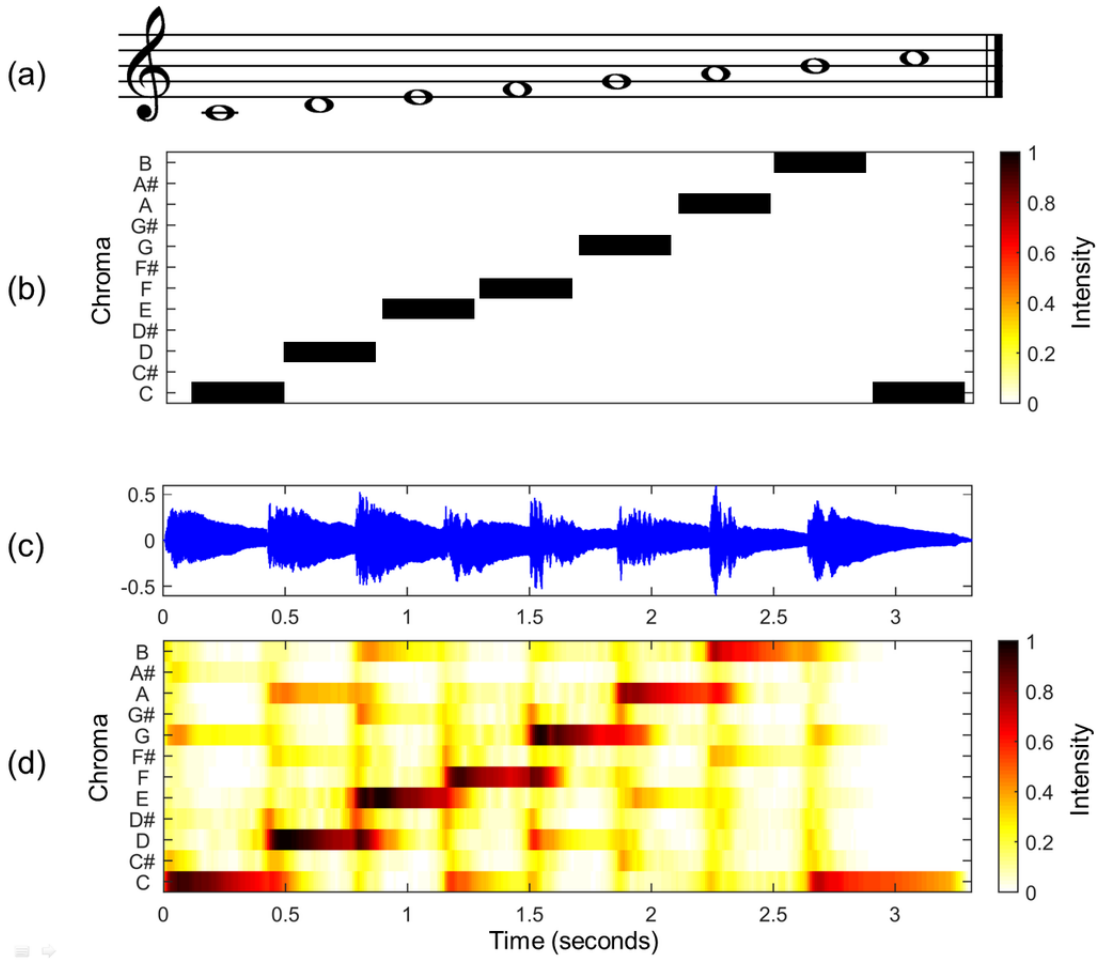
Η μαθηματική έκφραση για κάθε bin/τονική κλάση δίνεται ως εξής:

$$v_z = \sum_{n \in S_z} \frac{\log_{10} X_i(n)}{N_z} \quad , z \in 0, ..., 11 \tag{4.27}$$

όπου $\boldsymbol{v_z}$ είναι το άθροισμα των **τονικών συντελεστών (pitch coefficients)** που ανήκουν στο χρώμα (chroma) z, $\boldsymbol{X_i(n)}$ είναι ο DFT συντελεστής της συχνότητας **n** για το βραχυπρόθεσμο παράθυρο **i**, **z** είναι ο δείκτης της οκτάβας (από 0 εώς 11), $\boldsymbol{S_z}$ είναι ένα υποσύνολο των συχνοτήτων που αντιστοιχούν στους DFT συντελεστές του bin z και $\boldsymbol{N_k}$ είναι ο αριθμός των στοιχείων του υποσυνόλου S_z (δηλαδή ο αριθμός των συχνοτήτων που ανήκουν σε αυτό το υποσύνολο).

Το διάνυσμα χρώματος ή αλλιώς το χρωμογράφημα απαρτίζεται από το παραπάνω v_z για κάθε z, δηλαδή για καθένα εκ των 12 χρωμάτων (chroma).

Μια αναπαράσταση του χρωμογραφήματος φαίνεται στο παρακάτω σχήμα [55]:



Σχήμα 4.13: (α) Μουσικό σκορ μιας C-ματζόρε κλίμακας.(β) Χρωμογράφημα που έχει εξαχθεί απο το σκορ.(γ) Ηχητικη ηχογράφηση της C-ματζόρε κλίμακας από πιάνο.(δ) Χρωμογράφημα που έχει αποκτηθεί από την ηχητική ηχογράφηση.

Η εικόνα 4.13 τα χρωμογραφήματα για μια C-ματζόρε κλίμακα, τα οποία έχουν εξαχθεί είτε από το μουσικό σκορ (α –> β), είτε από την ηχητική ηχογράφηση (γ –> δ).

Ως παράδειγμα εφαρμογής αυτού του χαρακτηριστικού, αν είχαμε ένα χρωμογράφημα ενός μουσικού σήματος έναντι ενός χρωμογραφήματος μιας ομιλίας, το πιθανότερο είναι να βλέπαμε στην πρώτη περίπτωση να κυριαρχούν κάποιοι συγκεκριμένοι συντελεστές χρώματος, οι οποίοι θα ήταν σταθεροί για μικρές χρονικές στιγμές, ενώ στην δεύτερη περίπτωση, θορυβώδεις συντελεστές χρώματος. Αυτό μπορεί να εξηγηθεί από το γεγονός ότι στην μουσική, υπάρχουν διαστήματα όπου κυριαρχεί μια συγκεκριμένη νότα ή νότες από συγκεκριμένες τονικές κλάσεις, ενώ στην ομιλία τα φωνήματα διαφέρουν μεταξύ τους κατά τονικότητα και οι εναλλαγές είναι πολύ γρήγορες.

Αφού περιγράφηκαν όλα τα ηχητικά χαρακτηριστικά, μπορούμε να συνοψίσουμε πόσα και ποια χαρακτηριστικά εξάγονται από τα ηχητικά σήματα και χρησιμοποιούνται για την εκπαίδευση των ταξινομητών του συστήματος:

1. Zero-crossing rate (ποσοστό μηδενικής διέλευσης)
2. Energy (ενέργεια)
3. Energy entropy (εντροπία ενέργειας)

4. Spectral centroid (φασματικό κεντροειδές)
5. Spectral spread (φασματική εξάπλωση)
6. Spectral entropy (φασματική εντροπία)
7. Spectral flux (φασματική ροή)
8. Spectral rolloff (φασματική εξασθένηση)
9. MFCCs (13)
10. Chroma (διάνυσμα χρώματος) (12)
11. Chroma standard deviation-std (τυπική απόκλιση διανύσματος χρώματος)

Από τους συντελεστές Mel-Frequency Cepstrum (MFCCs) χρησιμοποιούνται οι 13 πρώτοι ως χαρακτηριστικά. Το διάνυσμα χρώματος (chroma), επίσης περιέχει 12 τιμές όπως έχουμε δει, ενώ τέλος υπολογίζεται και η τυπική απόκλιση των τιμών του διανύσματος αυτού, η οποία προσμετράται στα χαρακτηριστικά.

Συμπερασματικά, καταλήγουμε σε 34 χαρακτηριστικά συνολικά, τα οποία όπως έχουμε ήδη αναφέρει, εξάγονται για κάθε βραχυπρόθεσμο παράθυρο ξεχωριστά.

Για κάθε βραχυπρόθεσμο παράθυρο, υπολογίζεται επίσης η διαφορά μεταξύ των 34ων αυτών τιμών χαρακτηριστικών του εκάστοτε παραθύρου και των τιμών των χαρακτηριστικών του αμέσως προηγούμενου παραθύρου. Επομένως, σε κάθε short-term feature sequence, προσθέτουμε την παράγωγό του, δηλαδή την διαφορά του από το προηγούμενο frame. Έτσι, στα 34 χαρακτηριστικά προστίθενται άλλα 34 (οι 34 παράγωγοι/deltas) και καταλήγουμε σε 68 χαρακτηριστικά ανά short-term παράθυρο.

Στην συνέχεια, για κάθε μεσοπρόθεσμο παράθυρο, υπολογίζονται 2 στατιστικά: ο μέσος όρος και η τυπική απόκλιση των τιμών που παίρνει το κάθε χαρακτηριστικό στα βραχυπρόθεσμα παράθυρα. Παραδείγματος χάριν, αν κάθε μεσοπρόθεσμο παράθυρο περιέχει 2 βραχυπρόθεσμα παράθυρα, τότε θα υπολογιστεί ο μέσος όρος και η τυπική απόκλιση του ποσοστού μηδενικής διέλευσης στα δύο παράθυρα αυτά, ώστε να εξαχθούν 2 στατιστικά για το μεσοπρόθεσμο παράθυρο. Το ίδιο ισχύει και για τα υπόλοιπα χαρακτηριστικά. Επομένως, κάθε χαρακτηριστικό δίνει 2 στατιστικά σε mid-term επίπεδο (μεσοπρόθεσμο επίπεδο). Έτσι καταλήγουμε σε 68 x 2 = 136 χαρακτηριστικά ανά μεσοπρόθεσμο παράθυρο.

Ο λόγος για τον οποίο χρησιμοποιούνται και τα δύο στατιστικά (μέσος όρος και τυπική απόκλιση), είναι γιατί όπως έχουμε δει, το καθένα από αυτά μπορεί να δώσει διαφορετική πληροφορία για τα ηχητικά σήματα και κατ' επέκταση να συμβάλλει στον διαχωρισμό κλάσεων.

Για την εξαγωγή των χαρακτηριστικών αυτών χρησιμοποιήθηκε η ανοιχτού κώδικα βιβλιοθήκη της python που ονομάζεται pyaudioanalysis [56].

4.1.4 Ταξινομητές Τμημάτων - Αναγνώριση Συναισθήματος

Οι ταξινομητές που χρησιμοποιούνται στο σύστημα αφορούν τα συναισθηματικά χαρακτηριστικά, είτε τα κατηγορηματικά είτε τα διαστατικά. Για την καλύτερη κατανόηση αυτών των εννοιών θα εξηγήσουμε την ψυχολογία του συναισθήματος [63].

Η ψυχολογία του συναισθήματος

Λόγω της υποκειμενικότητας του συναισθήματος, διαφορετικοί μελετητές τείνουν να έχουν διαφορετικές αντιλήψεις και έτσι σπάνια επιτυγχάνουν συναίνεση στη βιβλιογραφία σχετικά με τον ορισμό του συναισθήματος. Σύμφωνα με τη σύγχρονη μελέτη [64], το

συναίσθημα ορίζεται ως «το περίπλοκο μοτίβο σωματικών και διανοητικών αλλαγών που περιλαμβάνει φυσιολογική διέγερση, συναισθήματα, γνωστικές διεργασίες, ορατές εκφράσεις και συγκεκριμένες συμπεριφορικές αντιδράσεις που γίνονται σε απάντηση σε μια κατάσταση που θεωρείται ως προσωπικά σημαντική» .

Υπάρχουν δύο κατηγορίες χαρακτηριστικών συναισθημάτων: τα **κατηγορηματικά χαρακτηριστικά (categorical attributes)** και τα **διαστατικά χαρακτηριστικά (dimensional attributes)**.

Τα **κατηγορηματικά χαρακτηριστικά** αποτελούνται από μερικές βασικές τάξεις συναισθημάτων. Μπορεί να απαιτείται ένα διαφορετικό σύνολο συναισθημάτων για διαφορετικά πεδία και διαφορετικά συστήματα/μοντέλα. Σύμφωνα με τους Ortony και Turner [65], τα βασικά συναισθήματα είναι συχνά τα πρωτόγονα δομικά στοιχεία άλλων μη βασικών συναισθημάτων, τα οποία θεωρούνται παραλλαγές, ή μίγματα των βασικών συναισθημάτων. Ο Ekman [66] στη μελέτη του για την ανάλυση των εκφράσεων του προσώπου, πρότεινε έξι βασικά συναισθήματα: θυμό, αηδία, φόβο, χαρά, θλίψη και έκπληξη. Παρόλο που το βασικό σύνολο συναισθημάτων του Ekman χρησιμοποιείται συχνά για εξόρυξη συναισθημάτων και μοντελοποίηση, υπάρχει επίσης μια μεγάλη ποικιλία εναλλακτικών συστημάτων ταξινόμησης βασικών συναισθημάτων που έχουν συγκεντρωθεί από διάφορους άλλους μελετητές.

Το κύριο μειονέκτημα του κατηγορηματικού μοντέλου είναι ότι έχει χαμηλότερη ανάλυση από το διαστατικό μοντέλο, επειδή χρησιμοποιεί κατηγορίες. Ο αριθμός των συναισθημάτων και οι αποχρώσεις τους που συναντώνται σε διάφορους τύπους επικοινωνίας είναι πολύ πλουσιότερες από τον περιορισμένο αριθμό κατηγοριών συναισθημάτων στο μοντέλο. Όσο μικρότερος είναι ο αριθμός των ομάδων στο κατηγορηματικό μοντέλο, τόσο μεγαλύτερη είναι η απλοποίηση της περιγραφής των συναισθημάτων [67].

Για να κατανοήσουν καλύτερα τα ανθρώπινα συναισθήματα, τις τελευταίες δεκαετίες, οι ερευνητές προσπάθησαν να δημιουργήσουν διαστατικούς χώρους που μπορούν κυρίως να συλλάβουν τις ομοιότητες και τις διαφορές μεταξύ των διαφόρων συναισθηματικών εμπειριών. Ο Wundt [68], πρότεινε το πρώτο **διαστατικό μοντέλο** αποσυνθέτοντας τον χώρο συναισθημάτων κατά μήκος τριών αξόνων, συγκεκριμένα: **σθένος (θετικό-αρνητικό)**, **διέγερση (ηρεμία-ενθουσιασμός)** και **ένταση (ένταση-χαλαρότητα)**. Παρόλο που προτάθηκε πριν από περισσότερο από έναν αιώνα, το μοντέλο του Wundt έθεσε με επιτυχία τα θεμέλια για μεταγενέστερη εργασία και έχει γίνει η βάση πολλών μεταγενέστερων θεωρητικών εξελίξεων. Εμπνευσμένα από το έργο του Wundt, αναπτύχθηκαν αρκετά διαστατικά μοντέλα συναισθημάτων, αν και πολλά από τα οποία ενσωμάτωσαν μόνο δύο από τις τρεις διαστάσεις όπως προτείνονται στο έργο του Wundt (διαστάσεις σθένους και διέγερσης, αλλά όχι την διάσταση της έντασης). Τα κυρίαρχα δισδιάστατα μοντέλα συναισθήματος περιλαμβάνουν: το κυκλικό μοντέλο [69] και τις παραλλαγές του [70], το διανυσματικό μοντέλο [71] και το μοντέλο θετικής ενεργοποίησης-αρνητικής ενεργοποίησης (Positive Activation Negative Activation - PANA) [72]. Μεταξύ όλων των διαστατικών μοντέλων, το **κυκλικό μοντέλο** είναι το πιο ευρέως χρησιμοποιούμενο στην έρευνα συναισθημάτων. Το **μοντέλο circumplex** [69] πρότεινε ότι όλες οι συναισθηματικές εμπειρίες κατανέμονται σε έναν κυκλικό χώρο που αποτελείται από κάθετους άξονες, **σθένους και διέγερσης**. Αν και υπάρχουν και άλλα τρισδιάστατα μοντέλα συναισθημάτων, υποστηρίζεται ότι η πλειοψηφία των συναισθημάτων μπορεί να εξηγηθεί καλύτερα όσον αφορά τις διαστάσεις διέγερσης και σθένους χρησιμοποιώντας το δισδιάστατο μοντέλο.

Στην βιβλιογραφία, έχει απασχολήσει πολύ ο τομέας της αναγνώρισης του συναισθήματος από την ομιλία (Speech Emotion Recognition - SER), με πολλές να είναι οι έρευνες που έχουν διεξαχθεί και δημοσιευθεί πάνω σε αυτό το κομμάτι. Μερικές από τις πιο υπερσύγχρονες υλοποιήσεις χρησιμοποιούν νευρωνικά δίκτυα, είτε συνδυάζοντας συνελικτικά δίκτυα (CNN) με δίκτυα μακράς βραχυπρόθεσμης μνήμης (LSTM) [73], είτε εξετάζοντας

βαθιά νευρωνικά δίκτυα πολλαπλών επιπέδων [74], είτε χρησιμοποιώντας επαναλαμβανόμενους κωδικοποιητές και συνδυάζοντας πληροφορία τόσο από τον ήχο όσο και από το κείμενο της ομιλίας (Multimodal Dual Recurrent Encoder) [75].

Πέραν βέβαια από την πληροφορία του ήχου ή του κειμένου της ομιλίας, αντίστοιχες έρευνες έχουν διεξαχθεί και στο κομμάτι της ανάλυσης της συναισθηματικής συμπεριφοράς (affective behavior analysis) από οπτικοακουστική πληροφορία, δηλαδή της αυτόματης ανάλυσης των τριών κύριων εργασιών συμπεριφοράς: εκτίμηση διέγερσης και σθένους (valence-arousal), αναγνώριση βασικών συναισθηματικών εκφράσεων (κατηγορηματικά χαρακτηριστικά) και ανίχνευση μονάδας δράσης (εκφράσεις και κινήσεις του προσώπου). Μια οπτικοακουστική βάση δεδομένων που περιέχει ετικέτες και των τριών διαφορετικών εργασιών συμπεριφοράς που αναφέρθηκαν παραπάνω είναι η Aff-Wild2 [76]. Πάνω σε αυτήν την βάση έχουν διεξαχθεί πειράματα με διαφορετικών ειδών αρχιτεκτονικές βαθιών νευρωνικών δικτύων, χρησιμοποιώντας κυρίως κομμένες εικόνες από τα αρχικά βίντεο του συνόλου δεδομένων και ερευνώντας είτε το κάθε task ξεχωριστά [77] [78] είτε αξιοποιώντας την χρήση multi-task μάθησης [79]. Σε κάποιες άλλες προτεινόμενες μεθόδους, εξετάζεται η οπτική πληροφορία ξεχωριστά από την ακουστική, με χρήση συνελικτικού δικτύου και επαναλαμβανόμενου δικτύου (CNN-RNN) ή ακόμα και ο συνδυασμός των δύο αυτών πληροφοριών [80].

Στην παρούσα διπλωματική εργασία θα χρησιμοποιηθεί η ηχητική και η κειμενική πληροφορία της ομιλίας, με σκοπό να διερευνηθεί το πρόβλημα της αναγνώρισης του συναισθήματος από την ομιλία τόσο σε ο,τι αφορά τα κατηγορηματικά χαρακτηριστικά όσο και σε ο,τι αφορά τα διαστατικά χαρακτηριστικά του συναισθήματος. Έχοντας κατανοήσει τις έννοιες των κατηγορηματικών και διαστατικών χαρακτηριστικών, αλλά και την έννοια του δισδιάστατου μοντέλου circumplex, μπορούμε πλέον να αναλύσουμε τους ταξινομητές που χρησιμοποιεί το σύστημά μας.

Όπως φαίνεται στο σχήμα 4.1, στο σύστημα χρησιμοποιούνται τρεις ταξινομητές τμημάτων για τον ήχο. Ένας ταξινομητής ο οποίος ταξινομεί τους ήχους σε τέσσερα **συναισθήματα (emotion)** ή αλλιώς κατηγορηματικά χαρακτηριστικά: θυμός, λύπη, ουδετερότητα, χαρά. Ένας ταξινομητής ο οποίος ταξινομεί τους ήχους ανάλογα με το **σθένος (valence)**: θετικό, μέτριο, αρνητικό και ένας ο οποίος τους ταξινομεί με βάση την **διέγερση (arousal)**: υψηλή, μέτρια, χαμηλή. Το σθένος και η διέγερση, όπως είδαμε παραπάνω, αποτελούν ένα δισδιάστατο χώρο χαρακτηριστικών συναισθήματος.

4.1.5 Σύνολα Δεδομένων

Για την εκπαίδευση των τριών αυτών μοντέλων-ταξινομητών χρησιμοποιήθηκαν τα παρακάτω σύνολα δεδομένων:

- **Emovo**
  Το emovo, είναι μια ιταλική βάση δεδομένων συναισθηματικής ομιλίας. Δημιουργήθηκε από φωνές 6 ηθοποιών (3 γυναίκες και 3 άντρες), οι οποίοι είπαν 14 προτάσεις που προσομοιώνουν 6 συναισθηματικές καταστάσεις (αηδία, φόβο, θυμό, χαρά, έκπληξη, θλίψη) συν την ουδέτερη κατάσταση. Αυτά τα συναισθήματα είναι τα γνωστά Big Six που βρίσκονται στο μεγαλύτερο μέρος της βιβλιογραφίας που σχετίζεται με τη συναισθηματική ομιλία [57].

- **Emo-DB**
  Η emo-db είναι μια βάση δεδομένων συναισθηματικής ομιλίας η οποία δημιουργήθηκε από το τεχνικό πανεπιστήμιο του Βερολίνο. Περιέχει ηχογραφήσεις από 10 ομιλητές (5 γυναίκες και 5 άντρες) ηλικίας 21 έως 35 ετών, οι οποίοι είπαν 10 προτάσεις που προσομοιώνουν 6 συναισθηματικές καταστάσεις (θυμός, βαρεμάρα, αηδία,

άγχος/φόβος, χαρά, λύπη) συν την ουδέτερη κατάσταση [58].

- **SAVEE**
  Η βάση δεδομένων SAVEE καταγράφηκε από τέσσερις γηγενείς αγγλικούς άντρες ομιλητές, μεταπτυχιακούς φοιτητές και ερευνητές στο πανεπιστήμιο του Surrey ηλικίας 27 έως 31 ετών. Το συναίσθημα έχει περιγραφεί ψυχολογικά σε 6 διακριτές κατηγορίες: θυμός, αηδία, φόβος, ευτυχία, θλίψη και έκπληξη συν την ουδέτερη κατάσταση. Η βάση αυτή περιλαμβάνει 15 TIMIT προτάσεις ανά συναίσθημα: 3 κοινές, 2 βασισμένες στο εκάστοτε συναίσθημα και 10 γενικές που είναι διαφορετικές για κάθε συναίσθημα και φωνητικά ισορροπημένες. Οι 3 κοινές και οι $2 \times 6 = 12$ συγκεκριμένες προτάσεις συναισθημάτων καταγράφηκαν και ως ουδέτερες για να δώσουν 30 ουδέτερες προτάσεις [59].

- **RAVDESS**
  Η βάση RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song - οπτικοακουστική βάση δεδομένων Ryerson για συναισθηματική ομιλία και τραγούδι) παρέχει ομιλίες και τραγούδια σε μορφή ήχου αλλά και βίντεο. Για το συγκεκριμένο σύστημα αξιολόγησης της ποιότητας της ομιλίας, χρησιμοποιήθηκαν μόνο τα δείγματα ομιλίας σε μορφή ήχου από αυτήν την βάση. Η βάση RAVDESS περιέχει 1440 αρχεία: 60 δοκιμές ανά ηθοποιό x 24 ηθοποιοί = 1440 δείγματα. Οι ομιλητές είναι 24 επαγγελματίες ηθοποιοί (12 γυναίκες, 12 άνδρες). Τα συναισθήματα ομιλίας περιλαμβάνουν 6 εκφράσεις: ηρεμίας, χαράς, λύπης, θυμού, φόβου, έκπληξης και αηδίας. Κάθε έκφραση παράγεται σε δύο επίπεδα συναισθηματικής έντασης (φυσιολογική, ισχυρή), με μια επιπλέον ουδέτερη έκφραση. Επομένως, 7 συναισθηματικές εκφράσεις συνολικά [60].

- **IEMOCAP**
  Η βάση δεδομένων IEMOCAP (Interactive Emotional Dyadic Motion Capture) είναι μια ενεργή, πολυτροπική και πολυφωνική βάση δεδομένων, που συλλέχθηκε στο πανεπιστήμιο Νότιας Καλιφόρνια. Περιέχει περίπου 12 ώρες οπτικοακουστικών δεδομένων, συμπεριλαμβανομένων βίντεο, ομιλίας, λήψης κίνησης προσώπου, μεταγραφών κειμένου. Αποτελείται από δυαδικές συνεδρίες όπου οι ηθοποιοί εκτελούν αυτοσχεδιασμούς ή σενάρια, ειδικά επιλεγμένα για να προκαλέσουν συναισθηματικές εκφράσεις. Περιέχει ηχογραφήσεις από 10 ηθοποιούς (5 γυναίκες και 5 άντρες), οι οποίες επισημαίνονται από πολλούς σχολιαστές σε κατηγορηματικά χαρακτηριστικά-ετικέτες (categorical attributes): θυμός, χαρά, ενθουσιασμός, λύπη, απογοήτευση, φόβος, έκπληξη, αλλά και ουδέτερη κατάσταση, καθώς και διαστατικές ετικέτες (dimensional attributes) όπως σθένος, ενεργοποίηση και κυριαρχία [61].

- **Emotion Speech Movies**
  Η βάση αυτή δεν είναι ανοιχτού κώδικα όπως οι προηγούμενες, δηλαδή δεν διατίθεται δωρεάν για ερευνητική χρήση. Είναι μια βάση που έχει δημιουργηθεί στα πλαίσια ενός διδακτορικού από τον ερευνητή Θεόδωρο Γιαννακόπουλο [62]. Περιέχει αρχεία ήχου, σκηνών από ταινίες τα οποία είναι χωρισμένα στις εξής κλάσεις συναισθημάτων: θυμός, φόβος, χαρά, λύπη και ουδέτερη κατάσταση.

Στο σύστημα, ο ταξινομητής συναισθήματος (emotion) εκπαιδεύεται σε δείγματα τεσσάρων κλάσεων όπως είπαμε παραπάνω: θυμός, λύπη, ουδετερότητα και χαρά. Παρ' όλα αυτά, μπορεί κανείς να παρατηρήσει πως τα σύνολα δεδομένων που χρησιμοποιούνται περιέχουν περισσότερες κλάσεις συναισθημάτων. Έτσι, χρησιμοποιήθηκαν από τα σύνολα δεδομένων, μόνο τα δείγματα των κλάσεων που αντιστοιχούν στις κλάσεις ενδιαφέροντος, ενώ όπου υπήρχε η κλάση "ενθουσιασμός" συνενώθηκε με την χαρά, μια και είναι αρκετά συγγενικές εκφράσεις.

Στον παρακάτω πίνακα φαίνονται οι κλάσεις συναισθήματος στις οποίες τελικά καταλήγουμε, καθώς επίσης και πως προκύπτουν αυτές από τις κλάσεις συναισθημάτων που περιέχουν τα σύνολα δεδομένων:

	Final Classes	Emotions (all datasets)
Emotion	**Sad**	Sad
	Angry	Angry
	Neutral	Neutral
	Happy	Happy, Excitement

Table 4.1: Τελικές Κλάσεις Συναισθήματος

Για τον ταξινομητή σθένους (valence), μπορεί κανείς να παρατηρήσει ότι τα μόνα δεδομένα που υπάρχουν είναι αυτά από την βάση iemocap, η οποία πέρα από συναισθήματα, περιέχει και διαστατικές ετικέτες όπως το σθένος. Η ετικέτα αυτή είναι ένας συνεχής πραγματικός αριθμός με μέγιστη τιμή το 5.5 και ελάχιστη τιμή το 1, ο οποίος υποδηλώνει τον βαθμό σθένους. Όσο μεγαλύτερη είναι η τιμή της ετικέτας αυτής, τόσο πιο έντονο είναι το σθένος και αντιστρόφως.

Το μοντέλο που χρησιμοποιείται στο σύστημα για να προβλέψει το σθένος, είναι ταξινομητής (classification) και όχι μοντέλο παλινδρόμησης (regression). Αυτό σημαίνει πως δεν προβλέπει συνεχείς τιμές αλλά διακριτές κλάσεις. Για τον λόγο αυτό, η παραπάνω ετικέτα η οποία είναι μια συνεχής τιμή, μετατράπηκε σε τρεις διακριτές κλάσεις. Η μετατροπή αυτή έγινε χωρίζοντας το εύρος τιμών (1 έως 5.5) σε τρία ίσα διαστήματα. Έτσι, ως "αρνητικό σθένος" θεωρούνται τα δείγματα με ετικέτα σθένους στο διάστημα [1,2.5) , ως "μέτριο/ουδέτερο σθένος" θεωρούνται τα δείγματα που φέρουν τιμή ετικέτας στο διάστημα [2.5,4) και ως "θετικό σθένος" προσμετρώνται τα δείγματα με ετικέτα στο διάστημα [4,5.5].

Επειδή όμως αυτά τα δεδομένα δεν είναι αρκετά, έγινε μια σύμπτυξη των συναισθηματικών κλάσεων των υπολοίπων βάσεων ώστε να παραχθούν κατηγορίες αρνητικού σθένους, μέτριου σθένους και θετικού σθένους. Πιο συγκεκριμένα, στην κλάση-ετικέτα αρνητικού σθένους συμπεριλήφθηκαν τα δείγματα από τις κλάσεις: λύπη, θυμός, φόβος, απδία, βαρεμάρα, από όλες τις βάσεις δεδομένων (πέραν της iemocap). Στην κλάση-ετικέτα μέτριου/ουδέτερου σθένους συμπεριλήφθηκαν τα δείγματα από τις κλάσεις: ουδετερότητα, έκπληξη. Τέλος στην κλάση-ετικέτα θετικού σθένους συμπεριλήφθηκαν τα δείγματα από τις κλάσεις: χαρά και ηρεμία.

Μια παρόμοια διαδικασία ακολουθήθηκε και για την συλλογή δεδομένων για τον ταξινομητή διέγερσης. Σε αυτήν την περίπτωση, η βάση δεδομένων iemocap περιέχει και πάλι δείγματα με την ετικέτα "ενεργοποίηση" που είναι αντίστοιχη με την διέγερση. Η ετικέτα αυτή είναι συνεχής, όπως και στην περίπτωση του σθένους, με μέγιστη τιμή το 5 και ελάχιστη τιμή το 1. Έτσι έγινε και πάλι η μετατροπή σε διακριτές κλάσεις ως εξης: ως δείγματα "χαμηλής διέγερσης" θεωρήθηκαν τα δείγματα με ετικέτα ενεργοποίησης στο διάστημα [1,2.3). Ως δείγματα "ουδέτερης/μέτριας διέγερσης" θεωρήθηκαν τα δείγματα με ετικέτα ενεργοποίησης στο διάστημα [2.3,3.6). Τέλος, ως δείγματα "υψηλής διέγερσης", θεωρήθηκαν τα δείγματα με ετικέτα ενεργοποίησης στο διάστημα [3.6,5].

Οι υπόλοιπες βάσεις (εκτός από την iemocap), δεν έχουν δείγματα ταξινομημένα με βάση την διέγερση, παρά μόνο με βάση το συναίσθημα. Έτσι, με τον ίδιο τρόπο όπως πριν, συμπτύχθηκαν οι κατηγορίες συναισθημάτων για να παράξουν τις επιθυμητές κλάσεις. Πιο συγκεκριμένα, σχηματίστηκαν οι κατηγορίες χαμηλής διέγερσης, μέτριας/ουδέτερης διέγερσης και υψηλής διέγερσης ως εξής: στην κλάση "χαμηλή διέγερση" συμπεριλήφθηκαν τα δείγματα από τις παρακάτω συναισθηματικές εκφράσεις: λύπη, βαρεμάρα, ηρεμία. Στην κλάση "μέτρια/ουδέτερη διέγερση" συμπεριλήφθηκαν τα δείγματα με συναισθηματικές

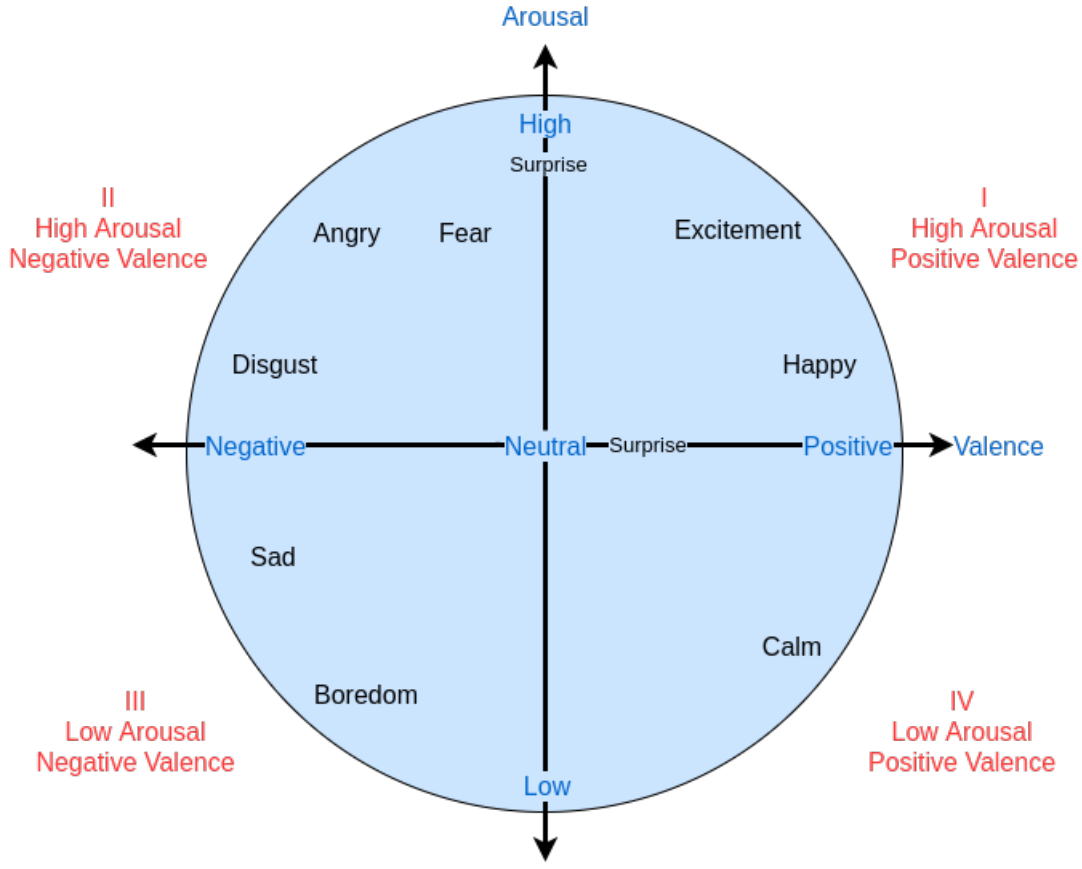
εκφράσεις: ουδετερότητα. Τέλος, στην κλάση "υψηλή διέγερση" συμπεριλήφθηκαν τα δείγματα που είναι επισημειωμένα ως: χαρά, θυμός, έκπληξη, φόβος, αηδία.

Στον παρακάτω πίνακα φαίνονται οι τελικές κλάσεις για το σθένος (valence) και την διέγερση (arousal), καθώς επίσης και πως προκύπτουν από τα δεδομένα που περιέχουν οι βάσεις:

	Final Classes	Values of **Valence (V)** and **Arousal (A)** (iemocap)	Emotions (rest datasets)
Valence	**Negative Valence**	$1 \leq V < 2.5$	Sad, Angry, Fear, Disgust, Boredom
	Neutral Valence	$2.5 \leq V < 4$	Neutral, Surprise
	Positive Valence	$4 \leq V \leq 5.5$	Happy, Calm
Arousal	**Low Arousal**	$1 \leq A < 2.3$	Sad, Boredom, Calm
	Neutral Arousal	$2.3 \leq A < 3.6$	Neutral
	High Arousal	$3.6 \leq A \leq 5$	Happy, Angry, Surprise, Fear, Disgust

Table 4.2: Τελικές Κλάσεις Σθένους και Διέγερσης

Η παραπάνω κατανομή των δεδομένων συναισθήματος στις κλάσεις της ετικέτας σθένους (valence) και της ετικέτας διέγερσης (arousal) ακολουθεί το δισδιάστατο κυκλικό μοντέλο (circumplex) που περιγράφηκε στο προηγούμενο υποκεφάλαιο, το οποίο στην προκειμένη περίπτωση, αφορά στο σύνολο των συναισθημάτων που παρουσιάζονται στις βάσεις δεδομένων που χρησιμοποιούνται. Έτσι, στην προκειμένη περίπτωση σχηματίζεται ως εξής:



Σχήμα 4.14: Συναισθήματα των βάσεων δεδομένων που καθιερώνονται στο κυκλικό μοντέλο (χώρο circumplex)

Τέλος, μετά την νέα διαμόρφωση των δεδομένων, κάθε δείγμα ακολουθεί την διαδικασία που περιγράφηκε στα προηγούμενα υποκεφάλαια: την διάσπαση σε βραχυπρόθεσμα παράθυρα, την εξαγωγή χαρακτηριστικών για καθένα από τα παράθυρα αυτά και τέλος την παραγωγή στατιστικών για κάθε μεσοπρόθεσμο παράθυρο. Για τα μοντέλα ταξινομητές τμημάτων που εκπαιδεύτηκαν στην παρούσα διπλωματική εργασία, χρησιμοποιήθηκαν μεσοπρόθεσμα παράθυρα (mid-term windows) των 3ων δευτερολέπτων και βραχυπρόθεσμα παράθυρα (short-term windows) των 0.05 δευτερολέπτων. Αυτές οι διάρκειες παραθύρων είναι ιδανικές για ομιλίες, καθώς δεν είναι ούτε πολύ μικρές ώστε να είναι δύσκολος ο εντοπισμός πληροφορίας, αλλά ούτε πολύ μεγάλες ώστε να συρρικνώνεται πολύ πληροφορία σε ένα παράθυρο. Για την εκπαίδευση, ακολουθείται ένα ακόμα βήμα, αυτό του μακροπρόθεσμου μέσου όρου (long-term averaging) που είδαμε στο τέλος του κεφαλαίου 4.1.2. Εφόσον, στα σύνολα δεδομένων, το κάθε αρχείο ήχου ταξινομείται σε μια κλάση, τα δεδομένα θεωρούνται ήδη τμηματοποιημένα (presegmented) και για αυτόν τον λόγο χρειαζόμαστε μια μακροπρόθεσμη αναπαράσταση, δηλαδή ένα μόνο διάνυσμα χαρακτηριστικών για κάθε αρχείο. Για να επιτευχθεί αυτό, παίρνουμε τον μέσο όρο των χαρακτηριστικών-στατιστικών των μεσοπρόθεσμων παραθύρων. Έχοντας πλέον 66 χαρακτηριστικά ανά αρχείο ήχου, είμαστε έτοιμοι να εκπαιδεύσουμε τα μοντέλα.

4.1.6 Πειράματα και Αποτελέσματα

Για την διαδικασία την εκπαίδευσης αλλά και της αξιολόγησης των μοντέλων εκτελέστηκαν πειράματα τριών διαφορετικών ειδών: πειράματα για κάθε σύνολο δεδομένων ξεχωριστά (inner dataset evaluation), πειράματα εκπαίδευσης σε όλα τα σύνολα δεδομένων εκτός από ένα και αξιολόγησης (testing) στο εναπομένον σύνολο (cross-dataset evaluation) και τέλος πειράματα σύμπτυξης όλων στων συνόλων μαζί (merged dataset evaluation).

Inner-Dataset Evaluation

Στον παρακάτω πίνακα παρουσιάζεται η απόδοση των μοντέλων για emotion, valence και arousal, σε κάθε dataset ξεχωριστά (inner dataset evaluation). Οι μετρικές που παρουσιάζονται είναι οι f1 macro, δηλαδή υπολογίζεται το f1 για κάθε κλάση ξεχωριστά και στο τέλος ο μέσος όρος των f1 αυτών. Ο αλγόριθμος μηχανικής μάθησης που χρησιμοποιήθηκε είναι ο svm με kernel rbf. Έγινε hyperparameter tuning (συντονισμός υπερπαραμέτρων) μέσω grid search για διαφορετικές τιμές της παραμέτρου C ([0.001, 0.01, 0.5, 1.0, 5.0, 10.0, 20.0]). Χρησιμοποιήθηκε cross validation (διασταυρούμενη επικύρωση) με 2 επαναλήψεις για τις περιπτώσεις που τα δείγματα ξεπερνούν τα 10000 σε αριθμό, 5 επαναλήψεις για τις περιπτώσεις που τα δείγματα ξεπερνούν τα 2000 σε αριθμό, 10 επαναλήψεις για τις περιπτώσεις που τα δείγματα ξεπερνούν τα 1000 σε αριθμό και 100 επαναλήψεις αν τα δείγματα είναι λιγότερα ή ίσα του 1000. Αυτό σημαίνει πως για κάθε τιμή παραμέτρου, τα δεδομένα χωρίστηκαν τυχαία k φορές (όσες και οι επαναλήψεις). Σε κάθε διαχωρισμό, προέκυψαν δύο sets από τον τυχαίο διαχωρισμό των δεδομένα σε 80% που ήταν για εκπαίδευση και 20% που ήταν για αξιολόγηση. Στην συνέχεια, εκπαιδεύτηκε και αξιολογήθηκε το μοντέλο για την εκάστοτε τιμή υπερπαραμέτρου και την δεδομένη διάσπαση. Η τελική απόδοση του μοντέλου για την εκάστοτε τιμή υπερπαραμέτρου, δόθηκε από τον μέσο όρο των αποδόσεων σε όλες τις επαναλήψεις/διασπάσεις. Τελικά, επιλέχθηκε η υπερπαράμετρος με την μέγιστη συνολική απόδοση. Τα αποτελέσματα των καλύτερων μοντέλων παρουσιάζονται παρακάτω:

Classification Task	Dataset						Average
	Iemocap	Savee	Emovo	Emo-db	Ravdess	EmotionSpeechMovies	
Emotion	79.7	71.2	75.5	80.6	67	50.4	**70.7**
Valence	52.9	62.3	59.5	76	63.9	53.2	**61.3**
Arousal	60.3	75.3	79.9	81.9	68.3	64.1	**70**

Table 4.3: Inner dataset Evaluation

Cross-Dataset Evaluation

Στον παρακάτω πίνακα παρουσιάζονται τα αποτελέσματα για την δεύτερη κατηγορία πειραμάτων (cross-dataset evaluation). Σε αυτήν την κατηγορία όπως είπαμε, στην εκπαίδευση λαμβάνονται υπόψιν όλα τα dataset πέρα από ένα το οποίο χρησιμοποιείται κατά το testing του μοντέλου. Οι μετρικές που παρουσιάζονται αφορούν και πάλι το f1 macro. Ο αλγόριθμος που χρησιμοποιήθηκε είναι και εδώ ο svm rbf και ακολουθήθηκε και πάλι η ίδια διαδικασία grid search με cross-validation όπως παραπάνω, κατά την εκπαίδευση του μοντέλου. Τα αποτελέσματα, δεν αφορούν πλέον αυτά που προέκυψαν από το cross validation κατά την εκπαίδευση, αλλά αυτά που προέκυψαν από το testing στο test dataset, μετά την εκπαίδευση του μοντέλου:

Classification Task	Test Dataset						Average
	Iemocap	Savee	Emovo	Emo-db	Ravdess	EmotionSpeechMovies	
Emotion	39	36	45.5	57.6	29.8	36.6	**40.8**
Valence	39.7	37.9	32.7	37	26.3	42	**35.9**
Arousal	40.1	41.8	40.3	51.1	38.4	38.6	**41.7**

Table 4.4: Cross-dataset Evaluation

Συγκρίνοντας τα αποτελέσματα των δύο παραπάνω πινάκων, μπορούμε να δούμε ότι στο cross dataset evaluation (πίνακας 4.4), τα αποτελέσματα είναι πολύ κακά. Φαίνεται να είναι λίγο καλύτερα από έναν random classifier. Αντίθετα, στην περίπτωση του inner dataset evaluation τα ποσοστά είναι αρκετά ικανοποιητικά. Αυτό σημαίνει πως το πρόβλημα το οποίο καλούνται τα συγκεκριμένα μοντέλα να λύσουν είναι άμεσα εξαρτώμενο από το συγκεκριμένο υποτομέα. Ως συγκεκριμένο υποτομέα, εννοούμε τον τύπο της ομιλίας ο οποίος διαφέρει ανά dataset, τις συνθήκες κάτω από τις οποίες έχουν δημιουργηθεί τα δείγματα, το θέμα των ομιλιών, τους ομιλητές και το φύλο τους, τον τρόπο επισημείωσης των δειγμάτων κ.α. Έτσι εξηγείται το γεγονός ότι το μοντέλο δίνει πολύ μεγαλύτερη απόδοση όταν αξιολογείται στο ίδιο dataset στο οποίο εκπαιδεύτηκε, ενώ όταν αξιολογείται σε διαφορετικό, η απόδοσή του πέφτει δραματικά. Αυτό είναι ένα αναμενόμενο φαινόμενο καθώς είναι γνωστό πως το πρόβλημα της αναγνώρισης συναισθήματος από ομιλία (Speech Emotion Recognition - SER), είναι δύσκολο ακριβώς επειδή εξαρτάται και επηρεάζεται από πολλούς παράγοντες.

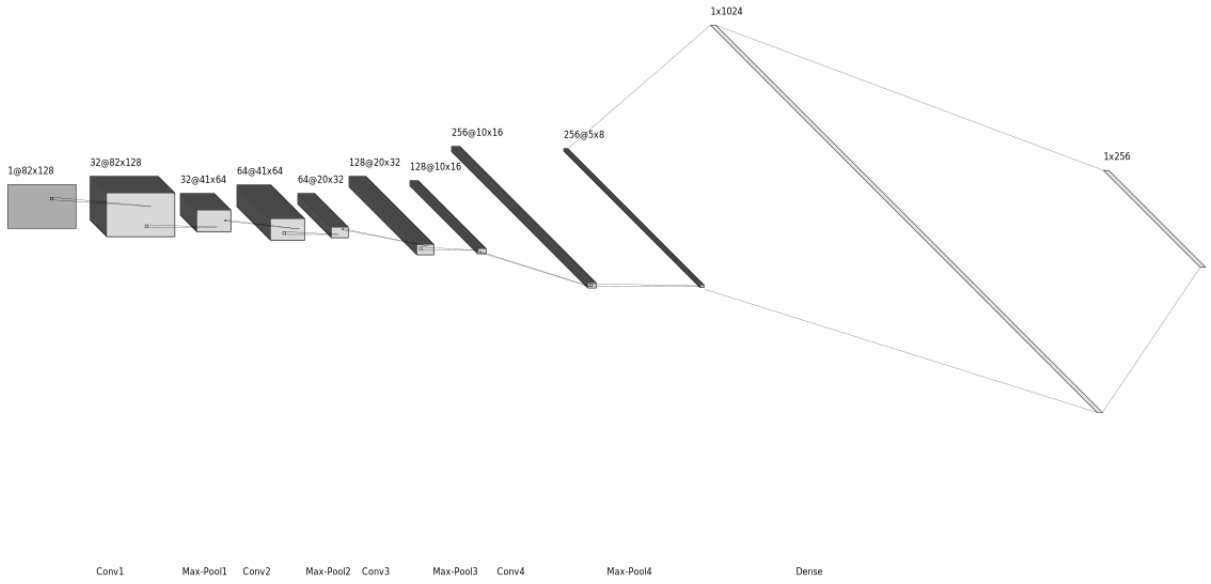
Merged Dataset Evaluation

Για τους παραπάνω λόγους, δημιουργήθηκε το τρίτο είδος πειραμάτων, το οποίο περιέχει όλα τα dataset ενωμένα. Ο λόγος για τον οποίο έγινε η ένωση των datasets είναι για να συμπεριλάβουμε όσο το δυνατόν περισσότερους τομείς γίνεται, από αυτούς που αντιπροσωπεύει το κάθε dataset ξεχωριστά. Τα δεδομένα συνενώθηκαν και στην συνέχεια διαχωρίστηκαν σε train και test set. Εδώ εκτελέστηκε και πάλι η ίδια διαδικασία grid search με cross-validation που είδαμε προηγουμένως, κατά το train, ενώ τα αποτελέσματα που παρουσιάζονται αφορούν την f1 macro μετρική η οποία υπολογίστηκε στο test set. Σε αυτά τα πειράματα που δίνουν και τα τελικά μοντέλα που θα χρησιμοποιηθούν από το σύστημά μας, συμπεριλήφθηκαν και δοκιμές με τον αλγόριθμο xgboost καθώς επίσης και

με το νευρωνικό cnn, πέρα από τον svm rbf.

Για την εκπαίδευση του cnn χρησιμοποιήθηκε η βιβλιοθήκη της python που ονομάζεται deep_audio_features [81], η οποία χρησιμοποιεί σαν χαρακτηριστικά τα mel spectograms, κάνει ένα resizing ώστε το height κάθε δείγματος να είναι ίσο με τον μέσο όρο των lengths όλων των δειγμάτων και χρησιμοποιεί 4 convolutional layers με kernels 5×5, stride =1, padding = 2 και maxpooling = 2. Τα κανάλια εξόδου (δηλαδή η τρίτη διάσταση), για το πρώτο layer είναι 32 και για καθένα από τα επόμενα layers πολλαπλασιάζεται με μια αυξανόμενη δύναμη του 2 (π.χ. για το δεύτερο layer θα είναι 32×2^1, για το τρίτο 32×2^2 κοκ). Στο τέλος, υπάρχουν και 3 γραμμικά layers (linear layers), με το πρώτο να έχει διάσταση εξόδου 1024, το δεύτερο 256 και το τρίτο ίση με τον αριθμό κλάσεων.

Στο παρακάτω σχήμα φαίνεται αναλυτικά η αρχιτεκτονική του cnn αυτού μέχρι το δεύτερο linear layer. Το τρίτο το οποίο δίνει στην έξοδο ένα διάνυσμα διαστάσεων όσες και οι κλάσεις, δεν απεικονίζεται καθώς είναι διαφορετικό για κάθε classification task (συγκεκριμένα έχει διάσταση 4 για τα emotions και 3 για τα arousal και valence):



Σχήμα 4.15: Αρχιτεκτονική του CNN που χρησιμοποιήθηκε

Τα αποτελέσματα από όλα τα πειράματα φαίνονται παρακάτω:

Classification Task	Merged Dataset			
	Train/Val Xgboost	Train/Val CNN	Train/Val SVM	Test SVM
Emotion	60.4	60.5	64.9	**62.8**
Valence	51.7	52.6	56	**49.6**
Arousal	64.2	69.3	68	**56.7**

Table 4.5: Merged dataset Evaluation

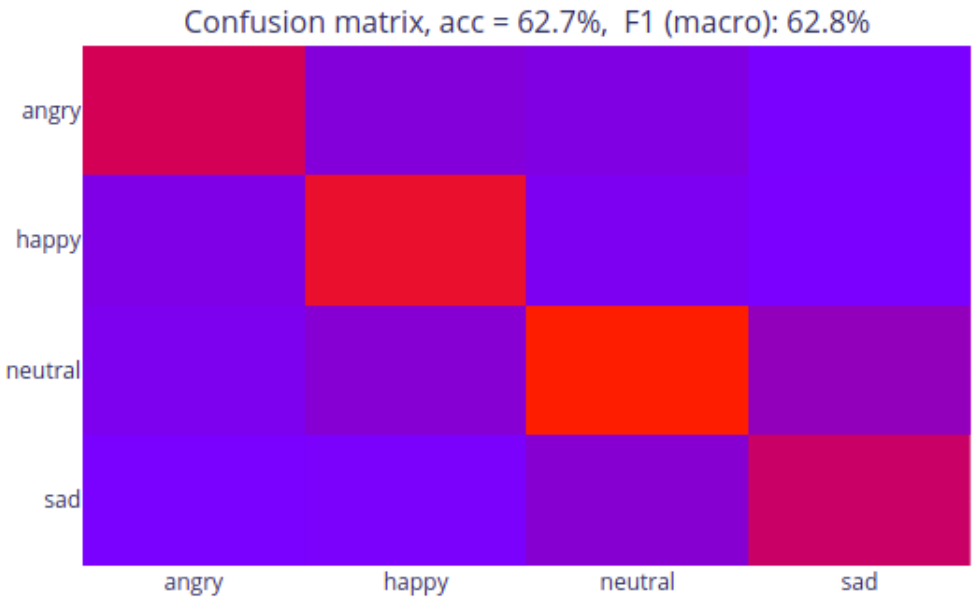
Από τα παραπάνω αποτελέσματα φαίνεται ο svm να τα πηγαίνει καλύτερα από τον xgboost κατά το validation. Ο CNN τα πηγαίνει λίγο καλύτερα μόνο στο arousal. Η διαφορά αυτή όμως είναι αμελητέα καθώς μιλάμε για 1%, για αυτόν τον λόγο θα κρατήσουμε το μοντέλο του svm επειδή είναι λιγότερο κοστοβόρο σε χρόνο αλλά και σε μνήμη και επειδή το συγκεκριμένο dataset μπορεί να περιείχε αρκετά δεδομένα για την εκπαίδευση του cnn αλλά σε μελλοντικές χρήσεις ίσως να μην υπάρχουν τόσα δεδομένα και επομένως ο svm είναι μια πιο γενική και ασφαλής επιλογή. Σε σύγκριση με τα πειράματα του πίνακα

4.3, τα οποία όπως είπαμε αφορούν εκπαιδεύσεις σε μεμονωμένα datasets και είναι πολύ εξειδικευμένα, η απόδοση εδώ (από την στήλη train/val svm) φαίνεται να είναι πολύ κοντά στον μέσο όρο (στήλη average). Αυτό είναι πολύ ενθαρρυντικό, καθώς παρ' όλο που εμπλέκουμε πολλούς διαφορετικούς τομείς (domains) μέσω των διαφόρων datasets, καταφέρνουν να τα πάνε σχεδόν τόσο καλά όσα τα μοντέλα που αφορούν έναν μόνο τομέα. Τέλος, σημαντικό είναι να αναλύσουμε και τα αποτελέσματα της στήλης "test svm". Το test set δημιουργήθηκε από ομιλητές οι οποίοι δεν εμπεριέχονται στο train set. Συγκεκριμένα, επιλέχθηκαν από 1 έως 2 ομιλητές από κάθε dataset και τα δείγματα που αφορούσαν αυτούς τους ομιλητές, συγκεντρώθηκαν όλα στο test set. Μάλιστα, η επιλογή των ομιλητών έγινε και με μια ισορροπία ως προς το φύλο (4 άντρες και 4 γυναίκες). Με αυτόν τον τρόπο, η απόδοση που δίνει το μοντέλο στο test set είναι πιο ρεαλιστική καθώς αφορά ομιλητές που το μοντέλο δεν έχει προηγουμένως ξαναδεί κατά την εκπαίδευση. Αυτός είναι και ο λόγος για τον οποίο η απόδοση εμφανίζεται πιο χαμηλή σε σχέση με την στήλη Train/Val SVM (2-10% απόκλιση), καθώς το πρόβλημα της αναγνώρισης συναισθήματος από την ομιλία είναι εξαρτώμενο και από τον ομιλητή (speaker dependent).

Για όλους τους παραπάνω λόγους, τα μοντέλα που τελικά επιλέγονται για να πλαισιώσουν το σύστημά μας είναι τα svm rbf από τα merged dataset που είδαμε στον τελευταίο πίνακα.

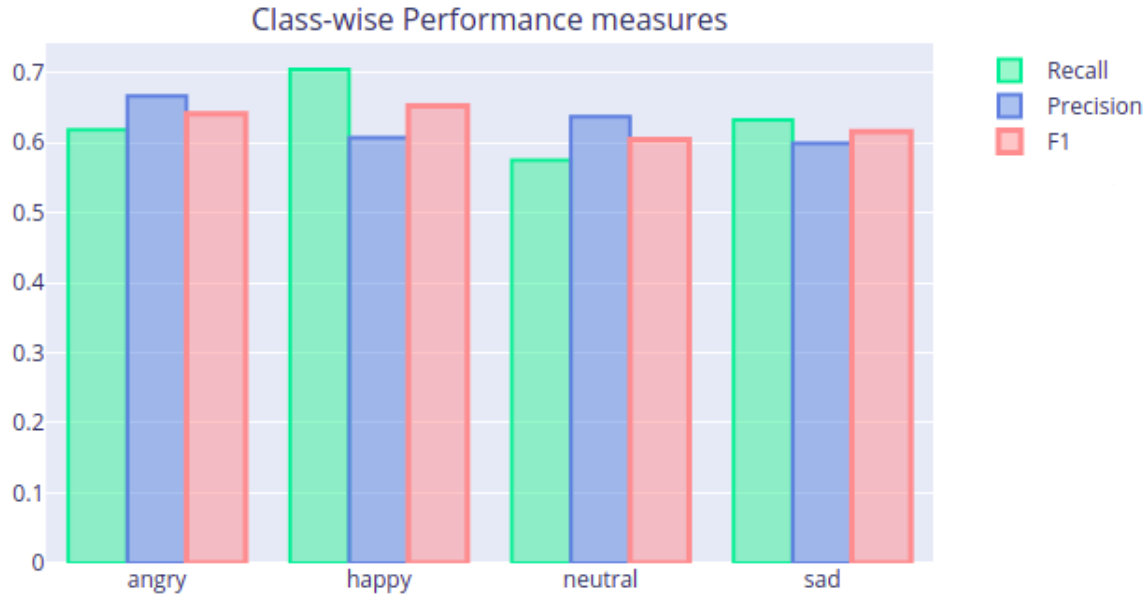
Παρακάτω παρουσιάζονται τα αποτελέσματα του test svm σε γραφήματα.

Emotions



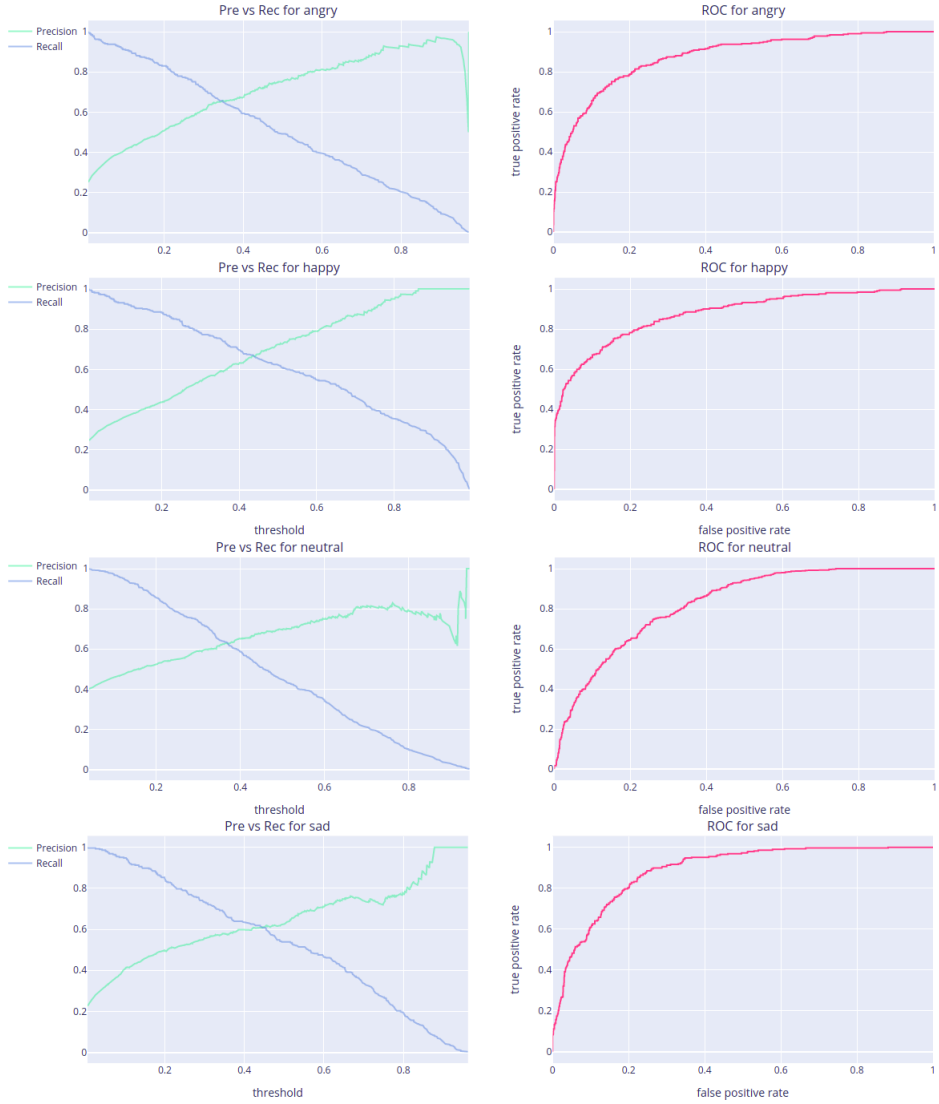
Σχήμα 4.16: Confusion Matrix Συναισθημάτων

όπως βλέπουμε από το confusion matrix, στην διαγώνιο έχουν ταξινομηθεί τα περισσότερα δείγματα αφού αυτή έχει πιο έντονα και φωτεινά χρώματα. Η διαγώνιος αποτελείται από τα δείγματα που έχουν ταξινομηθεί σωστά επομένως αυτό το αποτέλεσμα είναι αρκετά ενθαρρυντικό.



Σχήμα 4.17: Μετρικές Απόδοσης ανά Κλάση Συναισθήματος

Από το παραπάνω σχήμα μπορούμε πλέον να δούμε ξεκάθαρα τα performance measures ανά κλάση και μπορούμε να διακρίνουμε πως η απόδοση είναι σχεδόν ίδια για όλες τις κλάσεις, με μικρές διαφορές. Η πιο έντονες διαφορές που μπορούμε να παρατηρήσουμε είναι πως η κλάση happy ξεπερνά όλες τις άλλες κλάσεις ως προς την μετρική recall, πράγμα που σημαίνει ότι έχει τα πιο πολλά σωστά ταξινομημένα δείγματα, ενώ η κλάση neutral είναι η πιο χαμηλή ως προς το recall και το f1, χωρίς βέβαια αυτές οι διαφορές να είναι μεγάλες. Η τελευταία παρατήρηση ήταν αναμενόμενη για την κλάση neutral, καθώς είναι η πιο ουδέτερη κλάση, για αυτό και κάποια από τα δείγματά των άλλων κλάσεων μπορεί να ταξινομούνται λανθασμένα σε αυτήν.



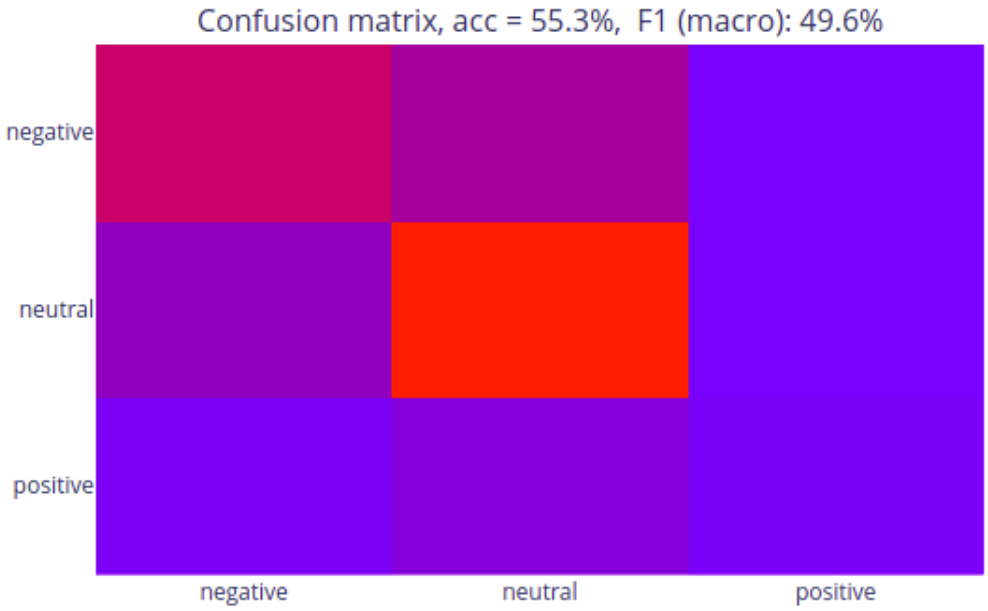
Σχήμα 4.18: Precision vs Recall και ROC curves

Στις παραπάνω γραφικές μπορούμε να δούμε την απόδοση του μοντέλου για κάθε κλάση ξεχωριστά, αν εφαρμόζονταν διαφορετικά κατώφλια πιθανότητας (probability threshold). Το κατώφλι αυτό, πρακτικά καθορίζει την τιμή της πιθανότητας πάνω από την οποία το δείγμα θα ταξινομηθεί ως θετικό, δηλαδή στην εκάστοτε κλάση.

Ως προς τις γραφικές precision vs recall, αν θέλαμε να έχουμε τόσο καλό precision όσο και καλό recall, τότε θα επιλέγαμε το threshold στο οποίο οι δυο καμπύλες τέμνονται.

Ως προς τις γραφικές ROC, μπορούμε να δούμε σίγουρα πως όλες οι κλάσεις τα πηγαίνουν πολύ καλύτερα απ' ότι θα τα πήγαιναν σε έναν random classifier, καθώς είναι πιο πάνω από την ευθεία x = y, που αντιπροσωπεύει τον classifier αυτόν. Επίσης, μπορούμε να συγκρίνουμε τις κλάσεις μεταξύ τους, καθώς όταν η καμπύλη βρίσκεται πιο πάνω από μία άλλη, δηλαδή το εμβαδόν κάτω από την καμπύλη αυτήν είναι μεγαλύτερο, αυτό σημαίνει πως έχει μεγαλύτερα true positive rate και άρα έχει καλύτερααποτελέσματα και θα έχει πάντα καλύτερα αποτε όποιο καλέσματατώφλι και αν εφαρμόσουμε. Στην συγκεκριμένη περίπτωση, φαίνεται όλες οι κλάσεις να είναι σχεδόν ισοδύναμες, με την κλάση neutral να έχει ίσως μια λίγο χειρότερη απόδοση, αφού η καμπύλη της παρουσιάζεται λίγο πιο κάτω από τις άλλες.

Valence



Σχήμα 4.19: Confusion Matrix του Σθένους

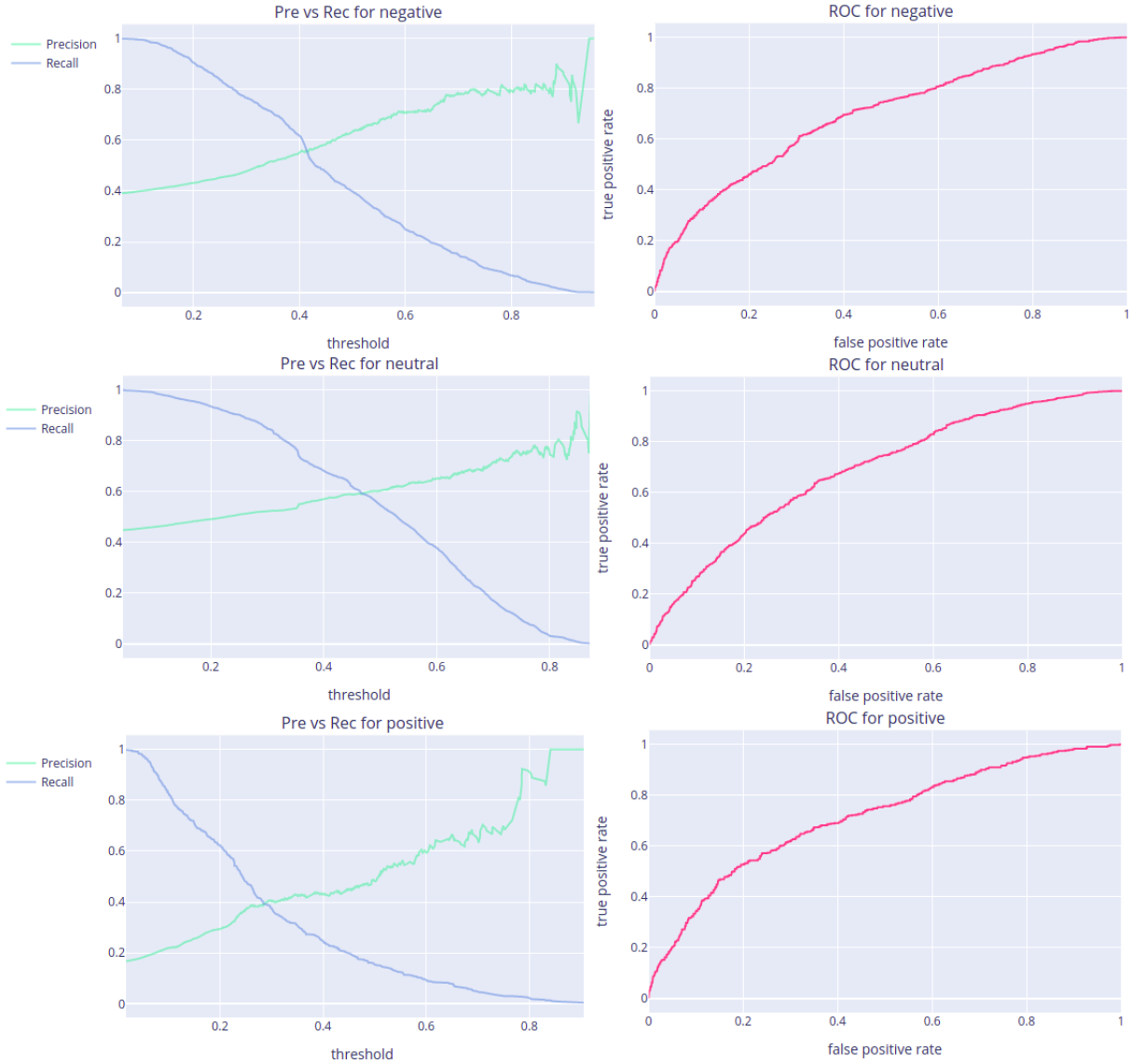
Από το confuson matrix αυτό μπορούμε να παρατηρήσουμε πως η κλάση positive δεν τα πηγαίνει και τόσο καλά, καθώς δεν υπάρχουν πολλά δείγματα που να έχουν ταξινομηθεί ως true positive σε αυτήν την κλάση. Αντίθετα, φαίνεται τα πιο πολλά δείγματα της κλάσης αυτής να ταξινομούνται ως neutral.

Σχήμα 4.20: Μετρικές Απόδοσης ανά Κλάση Σθένους

Αυτό που παρατηρήσαμε παραπάνω με το confusion matrix, επιβεβαιώνεται με τις μετρικές απόδοσης που φαίνονται εδώ ανά κλάση. Βλέπουμε πως η κλάση positive έχει αρκετά χαμηλό recall, ενώ παρουσιάζει μεγαλύτερο precision. Αυτό σημαίνει πως το μοντέλο ταξινόμησε αρκετά δείγματα της κλάσης positive σε άλλες κλάσεις, αλλά είχε καλύτερα αποτελέσματα στα δείγματα που ταξινόμησε στην κλάση αυτή. Συνεπώς, είναι πιο δύσκολο να ταξινομηθούν δείγματα άλλης κλάσης στην κλάση positive, παρά το αν-

τίστροφο. Τέλος, παρατηρούμε ότι η κλάση neutral τα έχει καλύτερα αποτελέσματα από τις άλλες δύο, με ίσως λίγο χαμηλότερο precision από την negative.

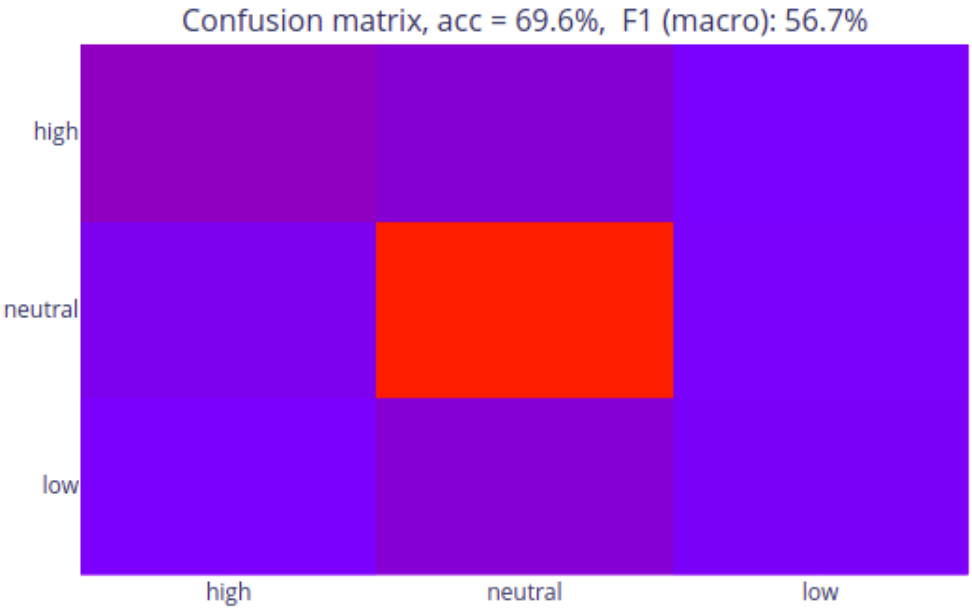
Έναν λόγο για τον οποίο, η κλάση positive φαίνεται να μην τα πηγαίνει καλά αποτελεί και το γεγονός πως το dataset είναι imbalanced, αφού τα δείγματα που άνηκαν στην κλάση negative ήταν 4746 σε αριθμό, τα δείγματα που άνηκαν στην κλάση neutral ήταν 4695 και τα δείγματα που άνηκαν στην κλάση positive ήταν 2189. Αυτό σημαίνει ότι περίπου το 41% του train set ήταν δείγματα της κλάσης negative, 40% της κλάσης neutral και 19% της κλάσης positive. Επομένως, η κλάση positive βρισκόταν σε μειονεκτική θέση, καθώς είχε πολύ λιγότερα δείγματα σε σχέση με τις άλλες δύο και αυτό, αυτόματα, δυσκόλεψε το μοντέλο στο να μάθει καλύτερα τα χαρακτηριστικά της και κατ'επέκταση να την ταξινομεί σωστά.



Σχήμα 4.21: Precision vs Recall και ROC curves

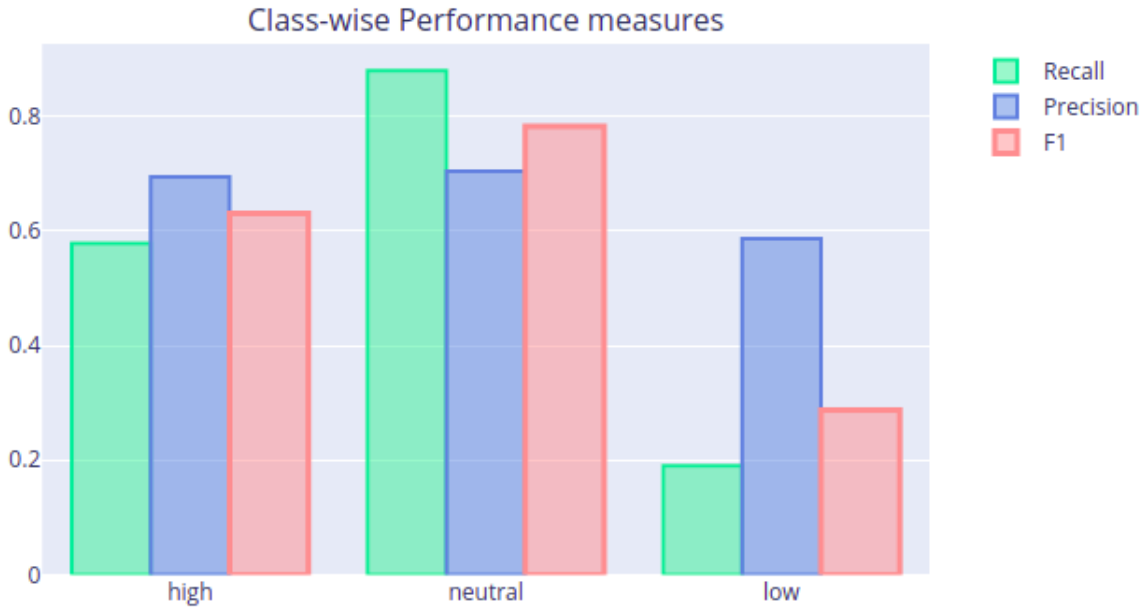
Από τα ROC curves, μπορούμε να δούμε πως οι τρεις κλάσεις έχουν σχεδόν ισοδύναμη απόδοση σε όλα τα κατώφλια.

Arousal



Σχήμα 4.22: Confusion Matrix της Διέγερσης

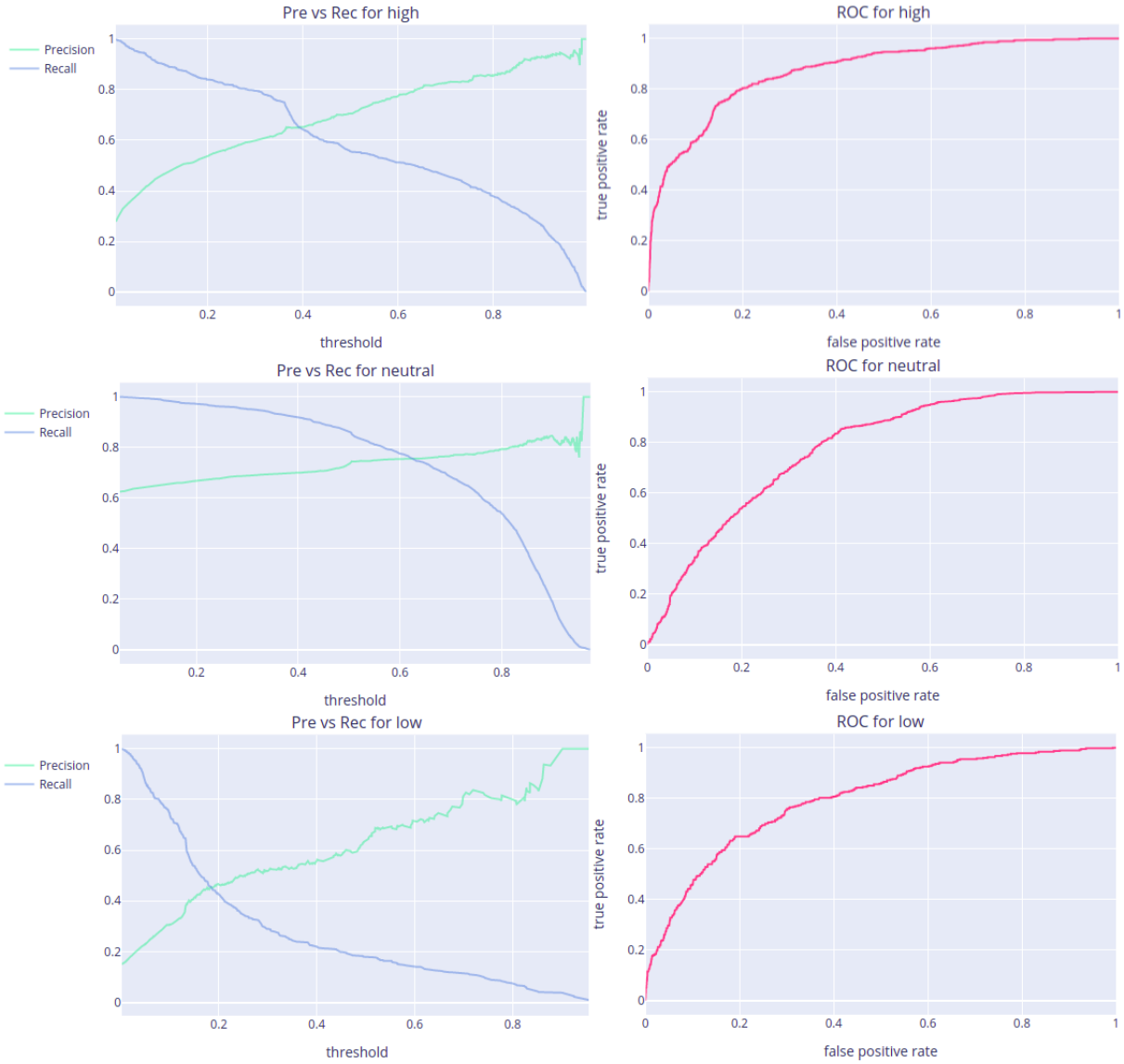
Εδώ παρατηρούμε ότι η κλάση που τα πηγαίνει καλύτερα είναι και πάλι η neutral, η αμέσως επόμενη είναι η high, ενώ η κλάση low τα πηγαίνει χειρότερα από όλες. Για τις κλάσεις high και low φαίνεται πως πολλά δείγματα ταξινομούνται λάθος ως neutral, πράγμα αναμενόμενο καθώς όπως έχουμε ξαναπεί, η κλάση neutral είναι ουδέτερη και μπορεί εύκολα να "παραπλανήσει".



Σχήμα 4.23: Μετρικές Απόδοσης ανά Κλάση Διέγερσης

Απο τις μετρικές απόδοσης ανά κλάση, παρατηρούμε την κλάση low να έχει ένα πολύ κακό recall, πράγμα που σημαίνει πως πολύ λίγα από τα δείγματα της κλάσης αυτής προβλέφθηκαν σωστά, ενώ το precision είναι πολύ πιο υψηλό που δείχνει ότι τα περισσότερα από τα δείγματα που προβλέφθηκαν στην κλάση αυτή, ταξινομήθηκαν σωστά.

Και σε αυτήν την περίπτωση, όπως ήταν και στο valence, το dataset είναι imbalanced και για αυτόν τον λόγο η κλάση low φαίνεται να τα πηγαίνει πολύ χειρότερα σε σχέση με τις άλλες δύο. Πιο συγκεκριμένα, η κλάση high περιείχε 3690 δείγματα, η κλάση neutral 6413 και η κλάση low 1527. Δηλαδή, περίπου το 32% του train set ήταν δείγματα της κλάσης high, το 55% της κλάσης neutral και το 13% της κλάσης low.



Σχήμα 4.24: Precision vs Recall και ROC curves

Από τα παραπάνω ROC curves μπορούμε να δούμε πως και οι τρεις κλάσεις είναι σχεδόν ισοδύναμες, με την κλάση neutral να τα πηγαίνει λίγο χειρότερα σε χαμηλότερα threshold, σε σχέση με τις άλλες κλάσεις, αφού παρουσιάζει χαμηλότερο true positive rate.

Με σκοπό να αυξηθεί η απόδοση των παραπάνω τριών μοντέλων που τελικά θα χρησιμοποιηθούν από το σύστημα, αλλά και να γίνει μια διαχείριση της ανισορροπίας των datasets, εφαρμόστηκαν κάποιες επιπλέον μέθοδοι. Συγκεκριμένα, έγινε εκ νέου εκπαίδευση μοντέλων svm με kernel rbf και πάλι στο merged dataset.

Αυτήν την φορά εφαρμόστηκε ένα pipeline, το οποίο χρησιμοποιεί τα παρακάτω βήματα:

- Τον sampler SMOTETomek [82], ο οποίος εφαρμόζεται μόνο σε περίπτωση που το dataset είναι imbalanced, κάνοντας upsampling ή downsampling κατά περίπτωση, εξαλείφοντας έτσι την ανισορροπία του dataset. Η ανισορροπία του dataset ελέγχεται ως εξής: υπολογίζεται αρχικά μια ισότιμη αναλογία δειγμάτων ανά κλάση. Δηλαδή, αν έχουμε 2 κλάσεις η ιδανική αναλογία θα ήταν το 50% των δειγμάτων να άνηκε στην πρώτη κλάση και το υπόλοιπο 50% στην δεύτερη. Αν έχουμε 3 κλάσεις η ιδανική αναλογία θα ήταν ένα $\tilde{3}3\%$ των δειγμάτων να άνηκε στις τρεις κλάσεις αντίστοιχα. Στην συνέχεια, υπολογίζεται η πραγματική αναλογία, διαιρώντας τον αριθμό δειγμάτων της κάθε κλάσης με τον συνολικό αριθμό δειγμάτων και αν αυτή η αναλογία είναι μικρότερη από το 80% της ιδανικής, για οποιαδήποτε κλάση, τότε το dataset θεωρείται imbalanced.

- Τον StandardScaler [83] για normalization των χαρακτηριστικών, ο οποίος ομαλοποιεί τα χαρακτηριστικά, αφαιρώντας την μέση τιμή και διαιρώντας με την τυπική απόκλιση.

- Το VarianceThreshold [84] με τιμή threshold=0, ο οποίος αφαιρεί ουσιαστικά όλα τα features που έχουν μηδενικό variance, δηλαδή αυτά που παρουσιάζουν ίδια τιμή σε όλα τα δείγματα και επομένως δεν προσφέρουν κάποια πληροφορία.

- Τον PCA [85], ο οποίος είναι υπεύθυνος για την μείωση της γραμμικής διάστασης με χρήση Singular Value Decomposition των δεδομένων για να τα προβάλλει σε έναν χώρο χαμηλότερης διάστασης.

Κατά την εκπαίδευση, εφαρμόστηκε ένα gridsearch το οποίο χρησιμοποεί RepeatedStratifiedKFold cross validation [86] και πιο συγκεκριμένα, χρησιμοποιεί 5 folds, δηλαδή κάθε φορά διασπά το dataset σε 5 μέρη και επαναλαμβάνει το cross validation 3 φορές, με διαφορετική τυχαιοποίηση των folds, σε κάθε επανάληψη. Σε αυτό το gridsearch γίνεται hyperparameter tuning σε τρεις υπερπαραμέτρους: τον αριθμό components του pca που διατηρούνται, την υπερπαράμετρο γ και την υπερπαράμετρο C του SVM. Για τις παραμέτρους αυτές εξετάζονται οι εξής τιμές: n_components = [0.98, 0.99, 'mle', None], 'gamma': ['auto', 'scale'], 'C': [0.001, 0.01, 0.5, 1.0, 5.0, 10.0, 20.0]. Επιλέγονται τελικά οι παράμετροι που δίνουν την καλύτερη μέση απόδοση, από τον μέσο όρο απόδοσης των επαναλήψεων. Η μετρική απόδοσης που έχει επιλεχθεί είναι και πάλι η f1 macro.

Οι τιμές των υπερπαραμέτρων που έδωσαν τα καλύτερα μοντέλα για κάθε classification task είναι οι εξής:

- **Emotion**:
  Parameters of best svm model: 'SVM C': 5.0, 'SVM gamma': 'auto', 'pca n_components': 'mle'

- **Valence**:
  Parameters of best svm model: 'SVM C': 1.0, 'SVM gamma': 'scale', 'pca n_components': 'mle'

- **Arousal**:
  Parameters of best svm model: 'SVM C': 1.0, 'SVM gamma': 'scale', 'pca n_components': 'mle'

Στον παρακάτω πίνακα παρουσιάζονται τα νέα αποτελέσματα, για τα καλύτερα μοντέλα που προέκυψαν από την διαδικασία που περιγράφηκε παραπάνω, στο merged dataset, τόσο στο validation κατά τον training (training set με cross validation), όσο και στο evaluation κατά το testing (test set):

Classification Task	Merged Dataset	
	Train/Val SVM	Test SVM
Emotion	64.4 (+/-0.9)	**62.9**
Valence	55.2 (+/-1.2)	**51.8**
Arousal	66.8 (+/-1.1)	**60**

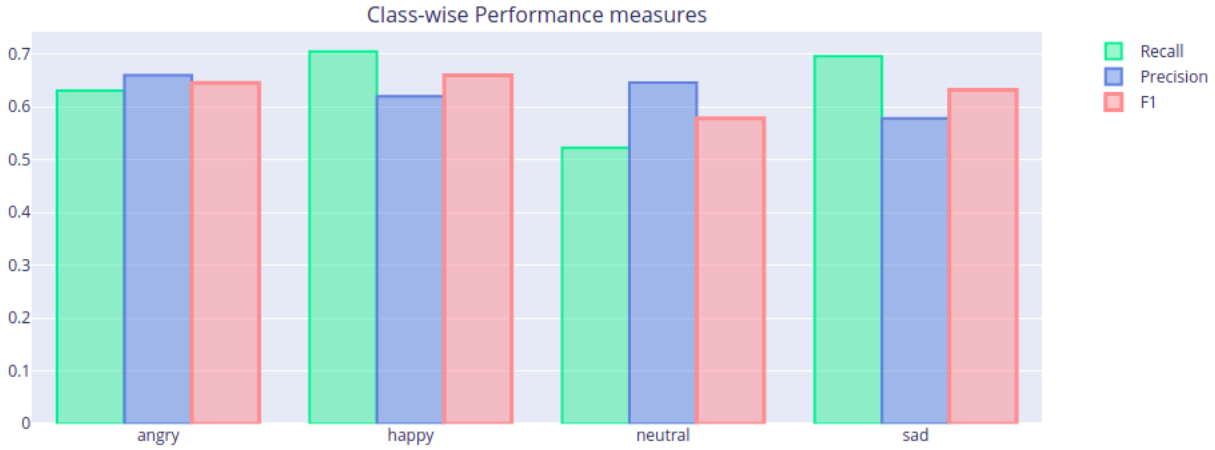
Table 4.6: Merged dataset - Βελτίωση Απόδοσης

Αν συγκρίνουμε τα αποτελέσματα στο test set του παραπάνω πίνακα, με αυτά του πίνακα 4.5, θα δούμε ότι έχουν βελτιωθεί όλα τα classification tasks, με το emotion να παρουσιάζει την μικρότερη βελτίωση πράγμα αναμενόμενο, καθώς το συγκεκριμένο dataset δεν παρουσίαζε προηγουμένως το πρόβλημα της ανισορροπίας, όπως τα άλλα δύο.

Παρακάτω φαίνονται πιο αναλυτικά τα confusion matrices καθώς επίσης και οι μετρικές απόδοσης ανά κλάση, για το κάθε classification task:

Emotions

Σχήμα 4.25: Confusion Matrix Συναισθημάτων

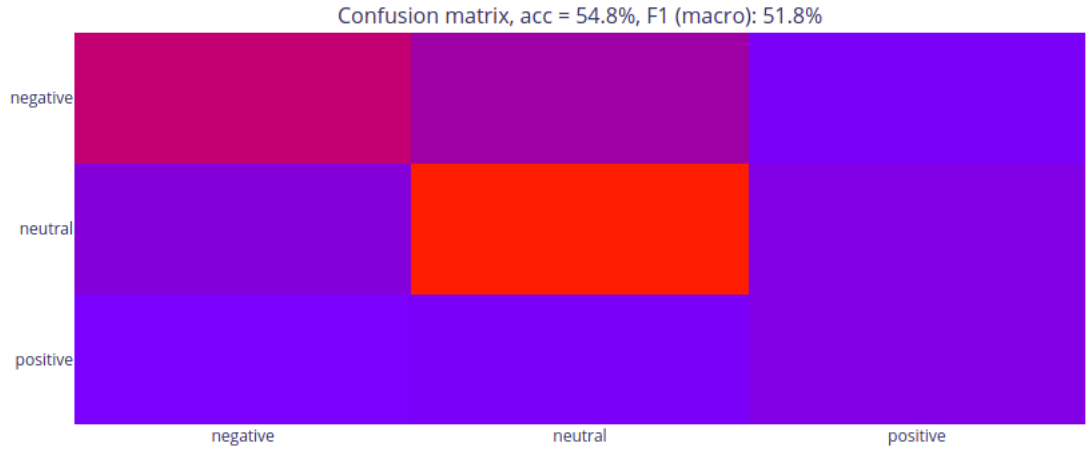


Σχήμα 4.26: Μετρικές Απόδοσης ανά Κλάση Συναισθήματος

Αν συγκρίνουμε τις παραπάνω γραφικές, με τις γραφικές 4.16 και 4.17, βλέπουμε μια

βελτίωση σε όλη την διαγώνιο του confusion matrix, καθώς παρουσιάζεται με πιο ανοιχτό κόκκινο χρώμα σε σχέση με πριν, πράγμα που σημαίνει πως περισσότερα δείγματα έχουν ταξινομηθεί ως true positives σε κάθε κλάση, απ' ότι πριν. Η βελτίωση αυτή φαίνεται πιο έντονη στην κλάση sad. Επίσης, παρατηρώντας τις μετρικές απόδοσης ανά κλάση, βλέπουμε μια μικρή βελτίωση σε όλα τα recall εκτός από της κλάσης neutral όπου το recall υπέστη κάποια μείωση και μια μικρή βελτίωση σε όλα τα precision, εκτός από της κλάσης sad όπου το precision υπέστη μια μικρή μείωση σε σχέση με πριν. Τα f1 έχουν λίγο πολύ διατηρηθεί τα ίδια, πέραν από την κλάση sad όπου το f1 αυξήθηκε.

Valence

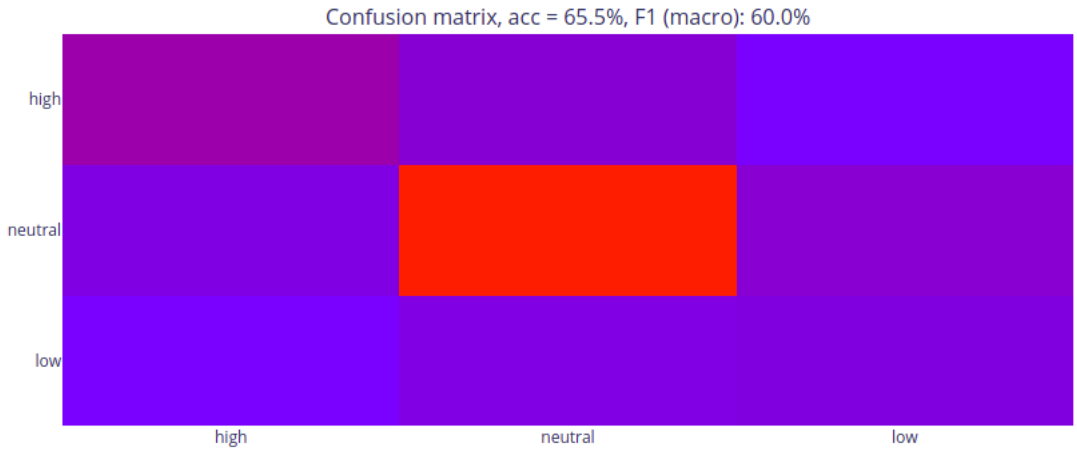


Σχήμα 4.27: Confusion Matrix ΣΘένους

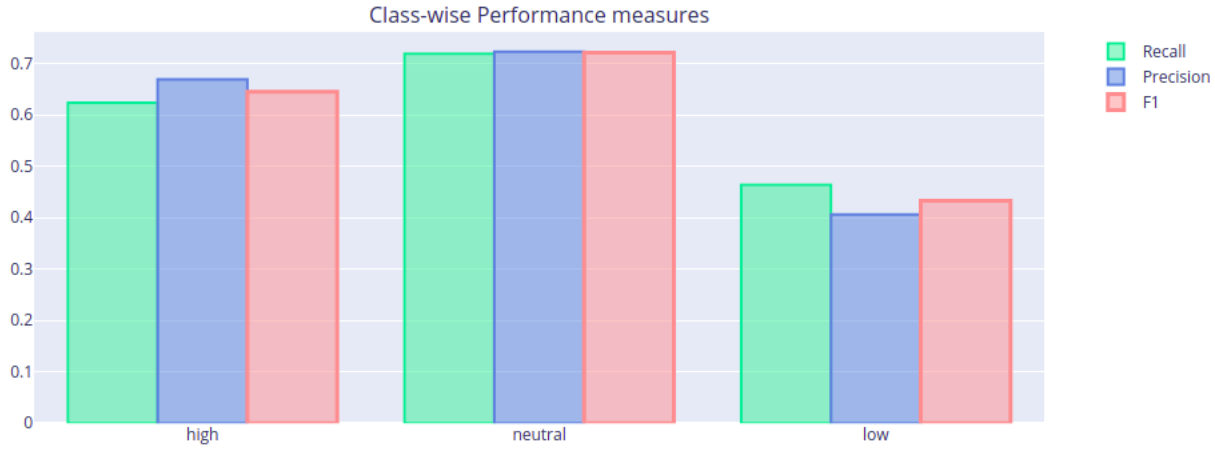
Σχήμα 4.28: Μετρικές Απόδοσης ανά Κλάση Σθένους

Αν συγκρίνουμε τις παραπάνω γραφικές, με τις γραφικές 4.19 και 4.20, βλέπουμε πως τα recall των κλάσεων negative και neutral έχουν μειωθεί σε σχέση με πριν, αλλά έχουν αυξηθεί τα precision τους, ενώ για την κλάση positive που ήταν και αυτή με τα λιγότερα δείγματα βλέπουμε μια αισθητή βελτίωση όπου το recall έχει ανέβει κατά ένα 20% με το precision να έχει πέσει ελάχιστα. Τα f1 έχουν και πάλι διατηρηθεί λίγο πολύ τα ίδια, εκτός από το f1 της κλάσης positive το οποίο βελτιώθηκε.

Arousal



Σχήμα 4.29: Confusion Matrix Διέγερσης



Σχήμα 4.30: Μετρικές Απόδοσης ανά Κλάση Διέγερσης

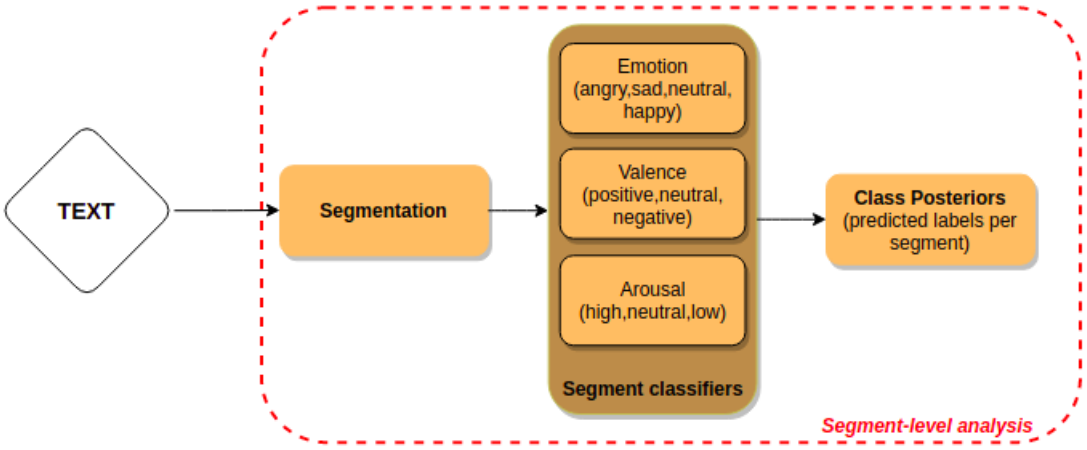
Αν συγκρίνουμε τις παραπάνω γραφικές, με τις γραφικές 4.22 και 4.23, βλέπουμε πως το recall έχει αυξηθεί τόσο για την κλάση high όσο και για την κλάση low, στην οποία μάλιστα παρατηρούμε μια τεράστια διαφορά αφού από 19% ανέβηκε στο 46%. Αυτό δείχνει πως το πρόβλημα της ανισορροπίας του dataset, όπου η κλάση low ήταν αυτή με τα λιγότερα δείγματα, λύθηκε ως έναν βαθμό. Στην κλάση neutral, το recall έπεσε και πάλι όμως διατηρείται σε μια αρκετά ικανοποιητική τιμή. Ως προς το precision, βλέπουμε να έχει ανέβει στην κλάση high και neutral και να έχει πέσει στην κλάση low. Τα f1 έχουν βελτιωθεί για τις κλάσεις high και low, με την τελευταία να παρουσιάζει μεγάλη βελτίωση, ενώ έχει μειωθεί λίγο για την κλάση neutral.
Σαν μια συνολική εικόνα, τα μοντέλα φαίνεται να βελτιώθηκαν σε σχέση με πριν και κυρίως αντιμετώπισαν το πρόβλημα της ανισορροπίας των datasets. Ως εκ τούτου, αυτά είναι τα μοντέλα που θα πλαισιώσουν τελικά το σύστημά μας, από πλευράς ταξινομητών τμημάτων ήχου.

4.2 Ανάλυση Κειμένου

Όπως είδαμε και στο κεφάλαιο 3.2, το κείμενο της ομιλίας προκύπτει από τον ήχο κάνοντας μια μετατροπή speech to text.

Στο υποκεφάλαιο αυτό θα παρουσιαστούν οι τεχνικές διάσπασης του κειμένου σε τμήματα, το σύνολο των κειμενικών χαρακτηριστικών που χρησιμοποιήθηκαν για τους ταξινομητές τμημάτων,το σύνολο δεδομένων που χρησιμοποιήθηκε για την εκπαίδευση των ταξινομητών και τα πειράματα που διεξήχθησαν.

Συγκεκριμένα, θα αναλυθεί το παρακάτω κομμάτι της αρχιτεκτονικής του συστήματος:



Σχήμα 4.31: Ανάλυση Κειμένου σε Επίπεδο Τμήματος

4.2.1 Τμηματοποίση Κειμένου

Όπως στον ήχο, έτσι και στο κείμενο είναι σημαντικό να χωριστεί η πληροφορία σε τμήματα (segments). Ο λόγος για τον οποίο γίνεται αυτή η διάσπαση είναι γιατί η κειμενική πληροφορία, η οποία έχει προέλθει από το ηχητικό σήμα, δεν είναι στάσιμη στην πάροδο του χρόνου, δηλαδή οι ιδιότητές της αλλάζουν. Για παράδειγμα, το συναίσθημα ενός ομιλητή μπορεί να αλλάξει από πρόταση σε πρόταση και από λεπτό σε λεπτό καθώς αυτός μιλάει. Το ίδιο και η ένταση, ο τόνος φωνής, ο ρυθμός ομιλίας κ.α.Έτσι θα ήταν ελλιπές και μεροληπτικό να χαρακτηρίζαμε όλο το κείμενο με μία μόνο ετικέτα συναισθήματος,έντασης κ.ο.κ

Η διάσπαση του κειμένου σε τμήματα, μπορεί να επιτευχθεί μέσω του συστήματος με τρεις τρόπους:

1. **Διάσπαση ανά πρόταση**
   Σε αυτήν την περίπτωση κάθε πρόταση αποτελεί ένα ξεχωριστό δείγμα για τον ταξινομητή και χαρακτηρίζεται ξεχωριστά.

2. **Διάσπαση σε τμήματα σταθερού μεγέθους**
   Σε αυτήν την περίπτωση καθορίζεται ένα κατώφλι αριθμού λέξεων ανά τμήμα. Για παράδειγμα, το κείμενο μπορεί να διασπαστεί σε τμήματα 3ων λέξεων.

3. **Διάσπαση σε σταθερά χρονικά παράθυρα**
   Σε αυτήν την περίπτωση καθορίζεται ένα κατώφλι δευτερολέπτων που είναι η διάρκεια κάθε τμήματος/παραθύρου. Για παράδειγμα, το κείμενο μπορεί να διασπαστεί σε τμήματα 3ων δευτερολέπτων. Έτσι σε κάθε τμήμα θα περιέχονται οι λέξεις που ειπώθηκαν μέσα στα εκάστοτε 3 δευτερόλεπτα.

Υπάρχει η δυνατότητα επιλογής ενός εκ των τριών τρόπων διάσπασης. Για τα μοντέλα ταξινομητές τμημάτων που εκπαιδεύτηκαν στην παρούσα διπλωματική εργασία, χρησιμοποιήθηκε ο πρώτος τρόπος διάσπασης (ανά πρόταση). Αυτός ο τρόπος διάσπασης είναι πιο ασφαλής, καθώς είναι πιθανότερο, οποιαδήποτε αλλαγή στα χαρακτηριστικά της ομιλίας, να γίνεται σε επίπεδο πρότασης. Μια μικρότερη διάσπαση, ίσως είναι ελλιπής για τον εντοπισμό συγκεκριμένων μοτίβων/χαρακτηριστικών, ενώ μια μεγαλύτερη διάσπαση θα μπορούσε να περιέχει περισσότερες από μία ιδιότητες με κίνδυνο κάποιες από αυτές να παραλειφθούν. Αυτό βέβαια δεν σημαίνει πως οι δύο άλλοι τρόποι διάσπασης δεν είναι ωφέλιμοι. Σε κάθε περίπτωση όμως, το κατώφλι διάσπασης στους τρόπους αυτούς, επιδέχεται διερεύνησης, εξαρτάται από την εκάστοτε διεργασία και πρέπει να επιλέγεται συνετά, έπειτα από δοκιμές.

4.2.2 Εξαγωγή Χαρακτηριστικών Κειμένου

Στα μοντέλα που επεξεργάζονται κείμενα, οι λέξεις που συνθέτουν τα κείμενα ή οι χαρακτήρες που συνθέτουν τις λέξεις δεν παρουσιάζουν κάποια άμεση αριθμητική αντιστοιχία. Στην **επεξεργασία φυσικής γλώσσας (Natural Language Processing - NLP)**, η **ενσωμάτωση λέξεων (Word Embedding)** είναι ένας όρος που χρησιμοποιείται για την αναπαράσταση λέξεων στην ανάλυση κειμένου, συνήθως με τη μορφή ενός διανύσματος πραγματικής τιμής που κωδικοποιεί την έννοια της λέξης έτσι ώστε οι λέξεις που είναι πιο κοντά στον διανυσματικό χώρο αναμένεται να έχουν παρόμοιο νόημα. Οι ενσωματώσεις λέξεων μπορούν να ληφθούν χρησιμοποιώντας ένα σύνολο γλωσσικής μοντελοποίησης και εκμάθησης χαρακτηριστικών, όπου λέξεις ή φράσεις από το λεξιλόγιο χαρτογραφούνται σε διανύσματα πραγματικών αριθμών. Εννοιολογικά περιλαμβάνει μια μαθηματική ενσωμάτωση από έναν χώρο με πολλές διαστάσεις ανά λέξη σε έναν συνεχή διανυσματικό χώρο με πολύ χαμηλότερη διάσταση [87].

Οι ενσωματώσεις λέξεων και φράσεων, όταν χρησιμοποιούνται ως αναπαράσταση εισόδου, έχουν αποδειχθεί ότι ενισχύουν την απόδοση σε εργασίες επεξεργασίας φυσικής γλώσσας, όπως η συντακτική ανάλυση και η ανάλυση συναισθημάτων.

Στην παρούσα διπλωματική εργασία, οι ταξινομητές-μοντέλα τμημάτων που χρησιμοποιούνται στο κομμάτι του κειμένου, είναι αναλυτές συναισθημάτων, όπως και στην περίπτωση της ηχητικής πληροφορίας που είδαμε παραπάνω. Έτσι, χρησιμοποιείται η ενσωμάτωση λέξεων ως κειμενική αναπαράσταση εισόδου (ως χαρακτηριστικά κειμένου) για τα μοντέλα.

Για τον υπολογισμό των διανυσμάτων των λέξεων (ενσωμάτωση κειμένου), υπάρχουν διάφορες μεθοδολογίες. Παρακάτω παρουσιάζονται οι πιο γνωστές από αυτές, καθώς και αυτή που τελικά χρησιμοποιείται από το σύστημα.

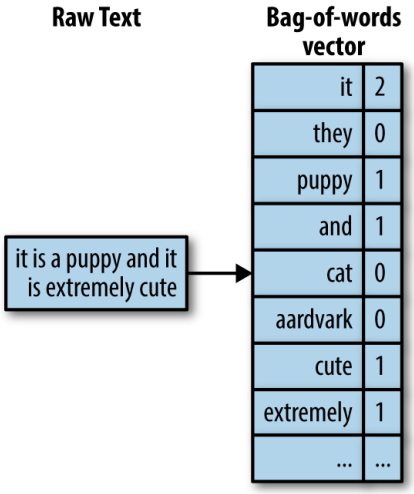
4.2.2.1 Bag of Words

Η μέθοδος **Bag of Words (BoW) (σακίδιο λέξεων)**, είναι η πιο απλή τεχνική μετατροπής κειμένου σε διάνυσμα. Με τη μέθοδο αυτή λέξεις, φράσεις, προτάσεις και κείμενα αναπαρίστανται με αραιά διανύσματα μεγάλου μήκους σε ένα διανυσματικό χώρο υψηλής διάστασης. Όπως δηλώνει και το όνομα, το κομμάτι κειμένου αντιμετωπίζεται σαν ένας σάκος με λέξεις. Αυτό σημαίνει πρακτικά ότι αγνοούμε τη σειρά των λέξεων και ενδιαφερόμαστε μόνο για την παρουσία ή τη συχνότητα λέξεων στο κείμενο.

Δεδομένου ενός λεξικού V = $[v_1, v_2, ..., v_n]$ αποτελούμενου από n λέξεις, το BoW διασπά τις λέξεις ενός κειμένου K που αποτελείται από k<n λέξεις σε μεμονωμένα στατιστικά στοιχεία καταμέτρησης λέξεων. Δηλαδή, κατασκευάζει ένα διάνυσμα w = $[w_1, w_2, ..., w_n]$ μήκους n, όπου κάθε στοιχείο w_i του διανύσματος εκφράζει πόσες φορές βρίσκεται στο κείμενο η λέξη v_i. Για παράδειγμα, αν έχουμε το εξής λεξικό: V = ['you','my','great','very']

και το κείμενο K = "you are very very beatiful", τότε το παραγόμενο διάνυσμα θα είναι: w = [1,0,0,2]. Αυτή η αναπαράσταση ονομάζεται **Term Frequency (συχνότητα όρου)**.

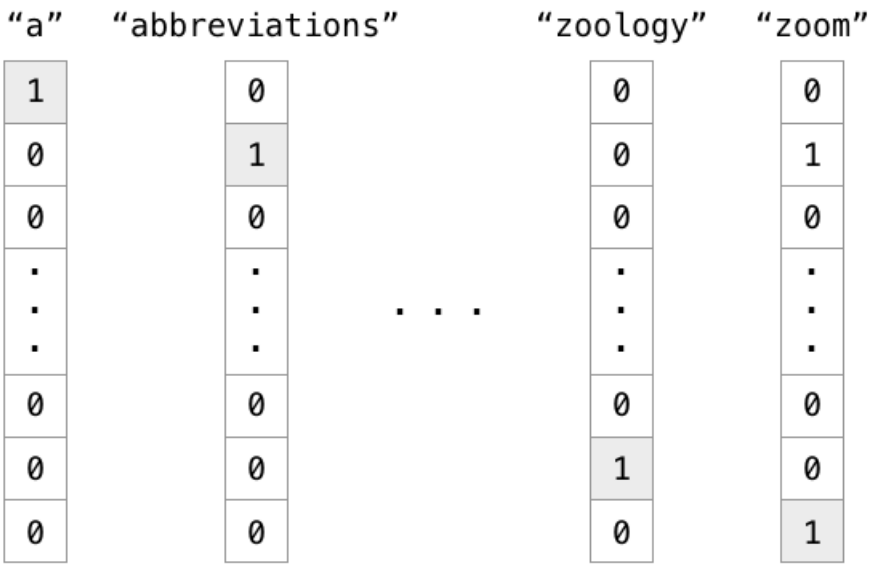
Ένα άλλο παράδειγμα αναπαράστασης BoW με term frequency φαίνεται σχηματικά παρακάτω [88]:



Σχήμα 4.32: Μετατροπή ακατέργαστου κειμένου σε αναπαράσταση σάκου λέξεων

Ένα άλλο είδος αναπαράστασης BoW είναι η αναπαράσταση λέξεων με διανύσματα one-hot **one-hot vectors** που είναι ουσιαστικά ένα διάνυσμα με μόνο ένα στοιχείο να είναι 1 και όλα τα υπόλοιπα να είναι 0. Το μήκος του διανύσματος είναι ίσο με το μέγεθος του συνολικού μοναδικού λεξικού. Συμβατικά, αυτές οι μοναδικές λέξεις κωδικοποιούνται με αλφαβητική σειρά. Δηλαδή, θα πρέπει να περιμένουμε τα one-hot διανύσματα για λέξεις που ξεκινούν με "a" να έχουν τιμή "1" σε χαμηλότερο δείκτη, ενώ για λέξεις που ξεκινούν με "z" να έχουν τιμή "1" σε υψηλότερο δείκτη.

Ένα παράδειγμα one-hot αναπαράστασης λέξεων φαίνεται στο παρακάτω σχήμα [89]:



Σχήμα 4.33: One-hot διανύσματα λέξεων

Με τον ίδιο τρόπο, αν θέλαμε να αναπαραστήσουμε με ένα διάνυσμα, ολόκληρο το

έγγραφο και όχι μόνο μεμονωμένες λέξεις, τότε το διάνυσμα θα είχε τιμή 1 στις θέσεις που αντιστοιχούν στις λέξεις του λεξικού, οι οποίες εμφανίζονται μέσα στο έγγραφο. Αυτή η αναπαράσταση ονομάζεται **Term Occurrence (ύπαρξη όρου)**.

Οι αναπαραστάσεις BoW χρησιμοποιούνται συχνά σε μεθόδους ταξινόμησης εγγράφων όπου η συχνότητα κάθε λέξης είναι ένα χρήσιμο χαρακτηριστικό για την εκπαίδευση ταξινομητών. Μία πρόκληση με την BoW αναπαράσταση λέξεων είναι ότι δεν κωδικοποιούν πληροφορίες σχετικά με την έννοια μιας δεδομένης λέξης, καθώς επίσης και πληροφορίες σχετικά με την σύνταξη/σειρά λέξεων μιας πρότασης ή ενός κειμένου. Είναι γνωστό, παρ' όλα αυτά, ότι δύο προτάσεις που μπορεί να έχουν ακριβώς τις ίδιες λέξεις σε διαφορετική σειρά, μπορούν να δίνουν ένα τελείως διαφορετικό νόημα. Αυτή η αναπαράσταση όμως, θα δώσει δύο αυτούσια διανύσματα για τις δυο αυτές προτάσεις.

Στο BoW, οι εμφανίσεις λέξεων σταθμίζονται ομοιόμορφα. Ωστόσο, στις περισσότερες εργασίες NLP ορισμένες λέξεις είναι πιο σχετικές από άλλες.

4.2.2.2 Bag of n-grams

Η μέθοδος Bag of n-grams αποτελεί μια επέκταση της μεθόδου BoW, η οποία αντιμετωπίζει εν μέρη το πρόβλημα της συντακτικής δομής. Το n-gram είναι απλώς οποιαδήποτε ακολουθία από n λέξεις. Αντί δηλαδή να διασπάμε το κείμενο σε έναν σάκο λέξεων, το διασπάμε σε έναν σάκο από ακολουθίες λέξεων. Για παράδειγμα, η πρόταση "You are very very beatiful" που είδαμε πριν θα μπορούσε να διασπαστεί σε:

- **1-grams**: 'You', 'are', 'very','very','beautiful'
- **2-grams**: 'You are', 'are very', 'very very', 'very beautiful'
- **3-grams**: 'You are very', 'are very very', 'very very beautiful'
- **etcetera**

Ο σάκος των n-grams μπορεί να είναι πιο ενημερωτικός από ό,τι η σάκος των λέξεων που είδαμε πρίν, επειδή αποτυπώνει περισσότερο περιεχόμενο γύρω από κάθε λέξη (δηλ. "Αγαπώ αυτό το φόρεμα" είναι πιο ενημερωτικό από το "φόρεμα"), δηλαδή αξιοποιεί και τα συμφραζόμενα, ή αλλιώς δίνει σημασία και στην σύνταξη. Ωστόσο, αυτό έχει και κάποιο κόστος, καθώς ο σάκος n-grams μπορεί να παράγει ένα πολύ μεγαλύτερο και πιο αραιό σύνολο χαρακτηριστικών από το σάκο λέξεων. Συνήθως, τα 3-grams είναι περίπου τόσο υψηλά όσο θέλουμε, καθώς η χρήση υψηλότερων n-grams αυξάνει σπάνια την απόδοση λόγω της αραιότητας.

4.2.2.3 TF-IDF

Η μεθοδολογία **TF-IDF (Term Frequency - Inverse Document Frequency)** είναι μια στατιστική διαδικασία που υπολογίζει την σημαντικότητα μιας λέξης ή ενός n-gram σε ένα κείμενο από ένα σύνολο κειμένων. Η αναπαράσταση αυτή αποτελείται από δύο μεγέθη: την **συχνότητα του όρου (Term Frequency)** και την **αντίστροφη συχνότητα των εγγράφων (Inverse Document Frequency)**. Το πρώτο μέγεθος είναι ο αριθμός των φορών που η εκάστοτε λέξη εμφανίζεται στο έγγραφο, ενώ το δεύτερο μέγεθος είναι η συχνότητα της λέξης στο σύνολο των εγγράφων και υπολογίζεται ως ο λογάριθμος του λόγου του συνολικού αριθμού των εγγράφων προς τον αριθμό των εγγράφων στα οποία εμφανίζεται η λέξη.

Η μαθηματική αναπαράστασή της φαίνεται παρακάτω:

$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_i} \tag{4.28}$$

όπου το $\boldsymbol{tf_{i,j}}$, είναι ο αριθμός των φορών που η λέξη i εμφανίζεται στο έγγραφο j, $\boldsymbol{df_i}$ είναι ο αριθμός των εγγράφων στα οποία εμφανίζεται η λέξη i και **N** είναι ο συνολικός αριθμός εγγράφων.

Η παραπάνω μαθηματική έκφραση όταν δίνει τιμές κοντά στο 0 σημαίνει ότι η σημαντικότητα της λέξης στο κείμενο είναι μικρή, ενώ όταν δίνει τιμές κοντά στο 1 δηλώνει μεγάλη σημαντικότητα

Κύριο προτέρημα που παρουσιάζει η μέθοδος tf-idf είναι ότι εξετάζει τη σημαντικότητα των λέξεων απέναντι σε ένα μεγάλο σύνολο από έγγραφα. Για παράδειγμα, λέξεις που εμφανίζονται συχνά σε ένα κείμενο όπως οι 'is', 'the' είναι μικρής σημασίας για την σημασιολογία του εγγράφου. Αυτό ο αλγόριθμος καταφέρνει να το δείξει, αφού οι λέξεις αυτές είναι συχνές σε όλα τα έγγραφα και επομένως ο λόγος $\frac{N}{df_i}$ που απεικονίζει την αντίστροφη συχνότητα των εγγράφων (idf), θα είναι μικρός, μικραίνοντας έτσι και την τελική τιμή του w (tf-idf). Αντίθετα, η εμφάνιση μιας λέξης που είναι άμεσα συνδεδεμένη με την θεματολογία ενός συγκεκριμένου εγγράφου, θα είναι συχνή στο έγγραφο αυτό αλλά όχι στα υπόλοιπα, δίνοντας έτσι μεγάλη τιμή στο idf και αντίστοιχα μεγάλη τιμή στην τελική μετρική tf-idf.

Ωστόσο, παρόλο που οι αναπαραστάσεις tf-idf και BoW παρέχουν βάρη σε διαφορετικές λέξεις, δεν είναι σε θέση να αναπαραστήσουν την έννοια της λέξης και να λάβουν υπόψιν τα συμφραζόμενα.Ένα ακόμα πρόβλημα που παρουσιάζει η tf-idf είναι ότι για να είναι αποδοτική αυτή η μεθοδολογία χρειάζεται μεγάλο αριθμό εγγράφων.

Όπως είπε ο διάσημος γλωσσολόγος J. R. Firth το 1935, «Το πλήρες νόημα μιας λέξης είναι πάντοτε συμφραζόμενο και δεν μπορεί να ληφθεί σοβαρά υπόψη η μελέτη του νοήματος εκτός του περιεχομένου/πλαισίου».

4.2.2.4 Lexical Co-occurrence Matrix

Ο **πίνακας συνύπαρξης (co-occurrence matrix)** για ένα δεδομένο σώμα κειμένου (corpus), είναι ένας πίνακας που δείχνει ανά ζευγάρια λέξεων, τον αριθμό φορών που αυτές οι λέξεις συνυπάρχουν/εμφανίζονται μαζί μέσα στο κείμενο. Μαθηματικά εκφράζεται από την σχέση: C = $\{c_{ij}\}$ με $c_{ij} = f(w_i, w_j, S)$, όπου $f(w_i, w_j, S)$ είναι η συχνότητα εμφάνισης της ακολουθίας $w_i w_j$ ή $w_j w_i$ σε όλο το σώμα κειμένου.

Αν για παράδειγμα έχουμε το σώμα/corpus = **He is not very pretty. He is happy. He is intelligent**, τότε ο πίνακας συνύπαρξης, αγνοώντας τις τελείες θα ήταν ο εξής:

	He	is	not	very	pretty	happy	intelligent
He	0	3	0	0	1	1	0
is	3	0	1	0	0	1	1
not	0	1	0	1	0	0	0
very	0	0	1	0	1	0	0
pretty	1	0	0	1	0	0	0
happy	1	1	0	0	0	0	0
intelligent	0	1	0	0	0	0	0

Table 4.7: Παράδειγμα Πίνακα Συνύπαρξης

Όμως, αυτός ο πίνακας συνύπαρξης δεν είναι η διανυσματική αναπαράσταση της λέξης που χρησιμοποιείται γενικά. Αντ 'αυτού, αυτός ο πίνακας συνύπαρξης αποσυντίθεται χρησιμοποιώντας τεχνικές όπως PCA, SVD κ.λπ. σε παράγοντες και ο συνδυασμός αυτών των παραγόντων σχηματίζει τη διανυσματική αναπαράσταση της λέξης (word vector representation).

Με τη μέθοδο SVD (Singular Value Decomposition) ο παραπάνω αραιός πίνακας μπορεί να παραγοντοποιηθεί ως εξής [92]:

$$\underset{|V|}{\begin{bmatrix} \\ X \\ \\ \end{bmatrix}} = \underset{|V|}{\begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix}} \underset{|V|}{\begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}} \underset{|V|}{\begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix}}$$

Σχήμα 4.34: Εφαρμογή SVD πάνω σε τετραγωνικό πίνακα συνύπαρξης X

όπου **Σ** διαγώνιος πίνακας, $\boldsymbol{\sigma_i}$ καλούνται ιδιάζουσες τιμές (singular values), $\boldsymbol{u_i}$ αριστερά ιδιάζοντα διανύσματα (left-singular vectors) και $\boldsymbol{v_i}$ δεξιά ιδιάζοντα διανύσματα (right-singular vectors). Επιλέγοντας τις k μεγαλύτερες ιδιάζουσες τιμές και τα ιδιάζοντα διανύσματα από αριστερά και δεξιά που αντιστοιχούν σε αυτές, παίρνουμε την προσέγγιση βαθμού k του πίνακα X με το μικρότερο σφάλμα κατά τη νόρμα Frobenius . Αυτό πρακτικά σημαίνει ότι αν ο αρχικός πίνακας είναι τετραγωνικός V × V, δηλαδή περιέχει V στήλες V διαστάσεων ή V γραμμές V διαστάσεων, επιλέγοντας τις k μεγαλύτερες ιδιάζουσες τιμές και τα αντίστοιχα k διανύσματα u και k διανύσματα v, τα k διανύσματα u δίνουν την k-διάστατη προσέγγιση των V γραμμών και τα k διανύσματα v την k-διάστατη προσέγγιση των V στηλών. Η παραγοντοποίηση SVD λοιπόν είναι μία μέθοδος μείωσης της διαστατικότητας.

Το αποτέλεσμα της μείωσης της διαστατικότητας με την επιλογή των πρώτων k ιδιαζόντων τιμών φαίνεται παρακάτω [92]:

$$\underset{|V|}{\begin{bmatrix} \\ \hat{X} \\ \\ \end{bmatrix}} = \underset{|V|}{\underbrace{\begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix}}_{\text{Word embedding matrix}}} \underset{k}{\begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}} \underset{|V|}{\begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix}}$$

Σχήμα 4.35: Μείωση Διαστατικότητας με SVD

όπου πλέον η διάσταση του X είναι |V|*|V|, η διάσταση του U είναι |V|*k, η διάσταση του Σ είναι k*k και η διάσταση του V είναι k*|V|.

Επομένως, στο παράδειγμά μας όπου ο πίνακας C είναι διάστασης 7x7, μπορούμε να πάρουμε k-διάστατες (k<7) προσεγγίσεις των στηλών ή των γραμμών, δηλαδή των λέξεων και να καταλήξουμε σε k-διάστατα word vectors (π.χ. για k=2 ή k=3). Έτσι, μια μόνο λέξη, αντί να αντιπροσωπεύεται σε διαστάσεις V, θα εκπροσωπείται σε διαστάσεις k, ενώ θα αποτυπώνει σχεδόν την ίδια σημασιολογική έννοια. Το k είναι γενικά της τάξης των εκατοντάδων. Αυτό που, επίσης, έχει σημασία είναι ότι το εσωτερικό γινόμενο των U και Σ δίνει την διανυσματική αναπαράσταση της λέξης **word vector representation** και το V δίνει την αναπαράσταση περιβάλλοντος της λέξης **word context representation**.

Η μείωση της διαστατικότητας του co-occurrence πίνακα είναι μία απλή μέθοδος για την εξαγωγή διανυσματικών αναπαραστάσεων λέξεων. Ωστόσο μπορεί να επεκταθεί και σε αναπαραστάσεις εγγράφων. Σε αυτή την περίπτωση ο co-occurrence πίνακας είναι μη τετραγωνικός και κάθε γραμμή του αφορά μία λέξη ή γενικότερα έναν όρο (n-gram) ενώ κάθε στήλη ένα έγγραφο. Κάθε στοιχείο του πίνακα αναφέρεται στη συνύπαρξη όρων με έγγραφα δηλαδή κάθε στήλη είναι μία term occurrence, term frequency ή tf-idf αναπαράσ-

ταση. Συνεπώς με την SVD παραγοντοποίηση παίρνουμε k-διάστατες προσεγγίσεις όρων και εγγράφων. Η μέθοδος αυτή των καθολικών διανυσμάτων παραγοντοποίησης (global matrix factorizations), όταν εφαρμόζεται σε πίνακες συχνότητας όρων (term frequency matrices) είναι ευρύτερα γνωστή ως **Λανθάνουσα Σημασιολογική Ανάλυση (Latent Semantic Analysis - LSA)**.

4.2.2.5 Word2Vec

Οι μεθοδολογίες που παρουσιάζονται σε αυτό το υποκεφάλαιο και στα επόμενα δύο υποκεφάλαια, για τον υπολογισμό των διανυσμάτων λέξεων είναι **νευρωνικές ενσωματώσεις (neural embeddings)** ή αλλιώς **νευρωνικά γλωσσικά μοντέλα**, βασίζονται δηλαδή σε νευρωνικά δίκτυα/μοντέλα. Τα νευρωνικά μοντέλα πρόβλεψης μαθαίνουν να προβλέπουν τα διανύματά τους, με το να βελτιώνουν την απώλεια πρόβλεψης, όπως είναι η απώλεια του να προβλέπεις ένα διάνυσμα μιας λέξης στόχου από τα διανύσματα των γύρω λέξεων.

Το Word2Vec είναι ένα τέτοιο προγνωστικό μοντέλο ενσωμάτωσης (predictive embedding model). Αναπτύχθηκε από τους Tomas Mikolov, et al. στην Google το 2013.

Ο αλγόριθμος εργάζεται με **context windows (παράθυρα περιβάλλοντος)**. Ουσιαστικά, επιλέγεται μια κεντρική λέξη-στόχος (target word) και οι γειτονικές λέξεις περιβάλλοντός της (context words), αριστερά και δεξιά της. Ο αριθμός των γειτονικών λέξεων καθορίζεται από το μήκος του παραθύρου. Παρακάτω φαίνεται ένα παράδειγμα για context window = 2:

Quick	rabit	jump	over	the	turtle

Table 4.8: Παράδειγμα Context Window

Quick	rabit	jump	over	the	turtle

Table 4.9: Παράδειγμα Context Window

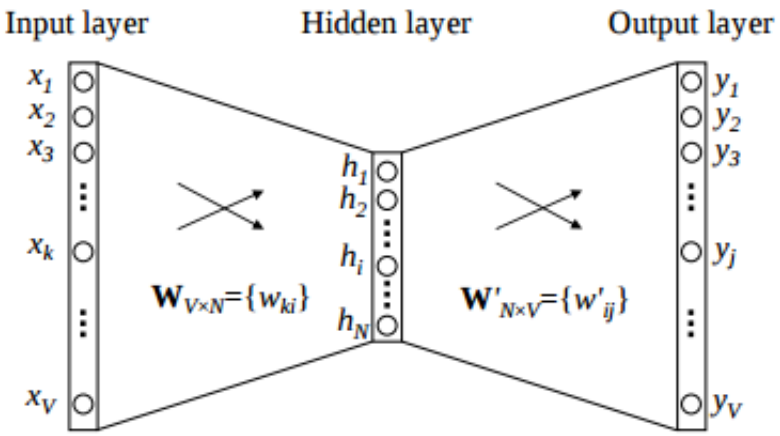
όπου στα παραπάνω παραδείγματα, οι πράσινες λέξεις αποτελούν το context window (2 around) γύρω από τις κεντρικές λέξεις που είναι χρωματισμένες πορτοκαλί.

Υπάρχουν δύο κύριες αρχιτεκτονικές του Word2Vec που χρησιμοποιούνται για την παραγωγή μιας κατανεμημένης αναπαράστασης λέξεων: η CBOW και η Skip-gram.

CBOW

Το **CBOW (Continuous bag-of-words)** βασίζεται στην αρχιτεκτονική ενός απλού νευρωνικού δικτύου. Η ιδέα είναι ότι δεδομένου ενός πλαισίου (context), θέλουμε να μάθουμε ποια λέξη είναι πιο πιθανό να εμφανίζεται σε αυτό. Το νευρωνικό λαμβάνει ως είσοδο τις λέξεις γύρω από την λέξη-στόχο (context words) και προβλέπει στην έξοδο την λέξη-στόχο (target word). Για παράδειγμα, στην πρόταση “I have a cute dog”, η είσοδος θα μπορούσε να είναι τα “I”, “have”, “cute”, “dog” και η έξοδος το “a”. Τα δεδομένα εισόδου και εξόδου είναι της ίδιας διάστασης και κωδικοποιημένα με one-hot όπως είδαμε στο κεφάλαιο 4.2.2.1. Το δίκτυο περιέχει 1 κρυφό επίπεδο (hidden layer) του οποίου η διάσταση είναι ίση με το μέγεθος ενσωμάτωσης (embedding size), το οποίο είναι μικρότερο από το μέγεθος του διανύσματος εισόδου / εξόδου.

Στο παρακάτω σχήμα, φαίνεται η αρχιτεκτονική αυτου του νευρωνικού για μία λέξη εισόδου (context word) και μία λέξη εξόδου (target word) [90]:



Σχήμα 4.36: Αρχιτεκτονική CBOW για μία Context Λέξη

Από το παραπάνω σχήμα της αρχιτεκτονικής μπορούμε να επισημάνουμε τα εξής:

1. Στο στρώμα εισόδου (input layer), είναι η context λέξη ενώ στο στρώμα εξόδου (output layer) είναι η target λέξη την οποία επιδιώκει το νευρωνικό να προβλέψει, δεδομένου της context λέξης. Και οι δυο λέξεις είναι κωδικοποιημένες με one-hot encoding, δηλαδή φέρουν 1 σε μία θέση και 0 σε όλες τις υπόλοιπες. Η διάσταση του διανύσματος εισόδου και εξόδου είναι V [1 X V] όσες και οι λέξεις του λεξικού, το οποίο κατασκευάζεται από το σώμα του κειμένου (context). Για παράδειγμα, για την πρόταση "I have a cute dog", το V είναι ίσο με 5.

2. Υπάρχουν δύο ομάδες βαρών (weights). Η μία είναι μεταξύ του στρώματος εισόδου (input layer) και του κρυφού στρώματος (hidden layer) και η δεύτερη μεταξύ του κρυφού στρώματος (hidden layer) και του στρώματος εξόδου (output layer). Input-Hidden layer matrix size =[V X N] , hidden-Output layer matrix size =[N X V] : όπου N είναι η διάσταση που επιλέγουμε να αναπαραστήσουμε την λέξη. Είναι μια υπερπαράμετρος του νευρωνικού συστήματος, αφού είναι και ο αριθμός των νευρώνων στο κρυφό στρώμα.

3. Η είσοδος πολλαπλασιάζεται με τα βάρη μεταξύ των input-hidden στρωμάτων. Στην ουσία αυτό το γινόμενο δίνει την k σειρά των βαρών $W_{V\times N}$, όπου k είναι η θέση στην οποία το διάνυσμα εισόδου x φέρει 1, δηλαδή $x_k = 1$. Από τον πολλαπλασιασμό αυτό προκύπτει το κρυφό διάνυσμα h διάστασης N.

4. Στην συνέχεια το κρυφό διάνυσμα πολλαπλασιάζεται με την δεύτερη ομάδα βαρών μεταξύ των hidden-output στρωμάτων ($W'_{N\times V}$). Από το γινόμενο αυτό προκύπτει ένα διάνυσμα μήκους V στο οποίο εφαρμόζεται μια συνάρτηση ενεργοποίησης soft-max έτσι ώστε όλες οι τιμές να αθροίζουν στην τιμή 1. Μετά την εφαρμογή της συνάρτησης ενεργοποίησης, προκύπτει το διάνυσμα εξόδου y. Με αυτό τον τρόπο, η κάθε τιμή y_j του διανύσματος εξόδου y, είναι μια πιθανότητα που περιγράφει πόσο πιθανό είναι η λέξη j του λεξιλογίου να είναι η λέξη στόχος που ψάχνουμε.

5. Το κριτήριο βελτιστοποίησης που χρησιμοποιείται στο νευρωνικό αυτό είναι μεγιστοποίηση της πιθανότητας παρατήρησης της πραγματικής λέξης-στόχου που όπως έχουμε δει και στην θεωρία ισοδυναμεί με την μεγιστοποίηση του λογαρίθμου της πιθανότητας. Επειδή, όμως, η συνάρτηση κόστους πρέπει να ελαχιστοποιείται, το ίδιο πρόβλημα

ορίζεται ως η ελαχιστοποίηση του αρνητικού λογαρίθμου της πιθανότητας ή αλλιώς η ελαχιστοποίηση της εγκάρσιας εντροπίας (cross entropy). Η συνάρτηση κόστους, λοιπόν, ή αλλιώς αντικειμενική συνάρτηση (objective function) ορίζεται ως εξής:

$$-\log(p(w_0 = w_{j^*}|w_I)) \tag{4.29}$$

όπου το $p(w_0 = w_{j^*}|w_I)$ ορίζεται ως εξής:

$$p(w_0 = w_{j^*}|w_I) = \frac{\exp(v'^{\mathsf{T}}_{w_{j^*}} \cdot v_{w_I})}{\sum_{j'=1}^{V} \exp(v'^{\mathsf{T}}_{w_{j'}} \cdot v_{w_I})} \tag{4.30}$$

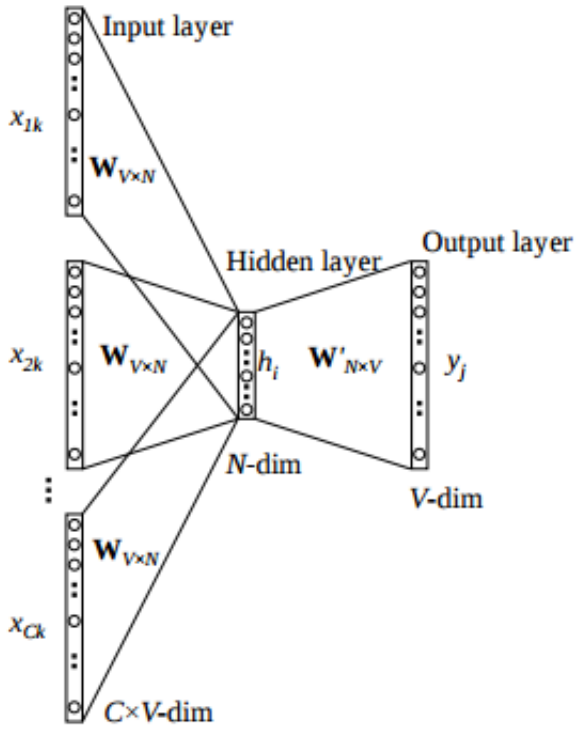
όπου w_0 είναι η λέξη εξόδου, w_I είναι η λέξη εισόδου (context word), j^* δηλώνεται ο δείκτης της λέξης w_{j^*}, που είναι πράγματι η λέξη-στόχος στο λεξικό, το v είναι κομμάτι των βαρών W και το v των βαρών W, ενώ v_w και v'_w, είναι το διάνυσμα εισόδου και εξόδου (βάρη εισόδου και εξόδου) αντίστοιχα, που χαρακτηρίζουν την λέξη w.

Η σχέση 4.30, ουσιαστικά είναι η πιθανότητα η λέξη εξόδου w_0, να είναι ίση με την λέξη-στόχο w_{j^*}, δεδομένης της context λέξης εισόδου w_I. Η πιθανότητα αυτή όπως είπαμε και στο προηγούμενο βήμα, έχει προκύψει έπειτα από την εφαρμογή της softmax συνάρτησης ενεργοποίησης, που είδαμε στο κεφάλαιο 2.3, για αυτό και η έκφραση 4.30, έχει αυτήν την μορφή.

6. Το σφάλμα που είδαμε στο προηγούμενο βήμα (cross entropy loss), διαδίδεται προς τα πίσω χρησιμοποιώντας stochastic gradient descent και backpropagation μεθόδους ώστε να ενημερωθούν/επαναρυθμιστούν όλα τα βάρη.

7. Μετά το τέλος της εκπαίδευσης του νευρωνικού, το διάνυσμα που τελικά μας δίνει την αναπαράσταση της λέξης στόχου, είναι το βάρος μεταξύ του κρυφού στρώματος και του στρώματος εξόδου και έχει διάσταση N. Δηλαδή είναι το $v'_{w_{j^*}}$ όπου j^* είναι η θέση της λέξης στόχου.

Στην πραγματικότητα, το παραπάνω παράδειγμα ήταν μια απλούστευση της αρχιτεκτονικής με μία μόνο λέξη εισόδου. Δεδομένου ενός σώματος κειμένου, ο αλγόριθμος εργάζεται σε παράθυρα όπου κάθε παράθυρο περιλαμβάνει μία κεντρική λέξη-στόχο και C συνολικά λέξεις δεξιά και αριστερά της κεντρικής (context words). Στην γενικότερη περίπτωση όπου ως είσοδο έχουμε περισσότερες από μία context λέξεις, η αρχιτεκτονική φαίνεται παρακάτω [90]:



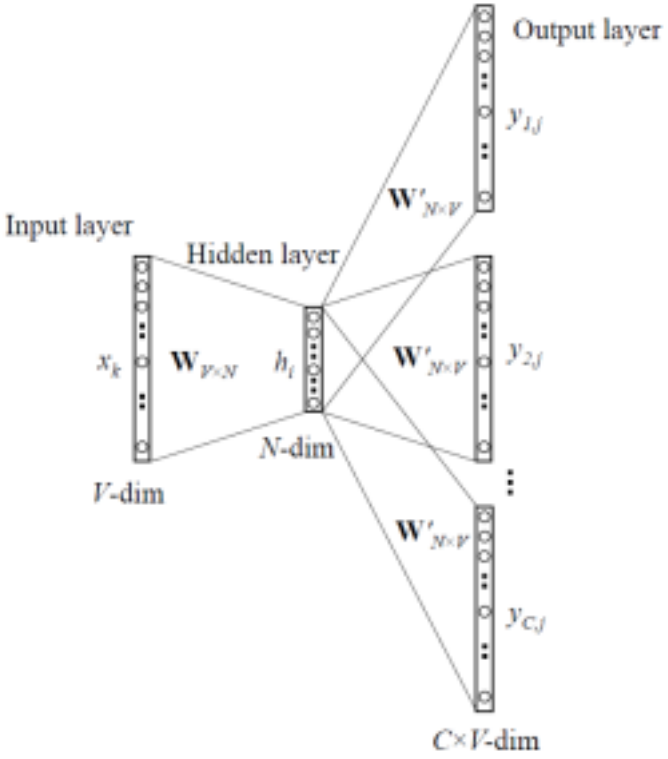
Σχήμα 4.37: Αρχιτεκτονική CBOW για πολλές Context Λέξεις

Στο παραπάνω σχήμα βλέπουμε ότι πλέον έχουμε C context λέξεις εισόδου, καθεμία από τις οποίες είναι διάστασης V. Τα βήματα παραμένουν τα ίδια όπως και πριν, αλλάζει μόνο το βήμα 3 που είναι ο υπολογισμός του κρυφού διανύσματος h. Αντί να αντιγράφουμε απλώς τις αντίστοιχες σειρές του input-hidden weight matrix ($W_{V\times N}$) στο κρυφό στρώμα, λαμβάνεται ένας μέσος όρος όλων των αντίστοιχων σειρών του πίνακα. Στην ουσία, πριν είχαμε μόνο ένα διάνυσμα εισόδου x το οποίο είχε τιμή 1 σε μια θέση k και έτσι ο πολλαπλασιασμός $x \times W_{V\times N}$ έδινε την σειρά k του πίνακα $W_{V\times N}$. Τώρα, πραγματοποιούνται C τέτοιοι πολλαπλασιασμοί (όσα και τα διανύσματα x), καθένας από τους οποίους δίνει μια διαφορετική σειρά του πίνακα. Στο τέλος υπολογίζεται ο μέσος όρος των σειρών αυτών με γνώμονα τα στοιχεία (element-wise) και προκύπτει και πάλι ένα διάνυσμα h, N διαστάσεων.

Skip-gram

Το continuous skip-gram είναι πολύ παρόμοιο με το CBOW, εκτός του ότι ανταλλάσσει την είσοδο και την έξοδο. Η αρχιτεκτονική του skip-gram είναι ένα νευρωνικό που δέχεται ως είσοδο μια λέξη στόχο και προβλέπει στην έξοδο τις λέξεις (context words) γύρω από την λέξη-στόχο. Για παράδειγμα,στην “I have a cute dog”,η είσοδος θα μπορούσε να είναι το “a” και η έξοδος τα “I”, “have”, “cute”, “dog”.

Στο παρακάτω σχήμα φαίνεται η αρχιτεκτονική αυτού του νευρωνικού [90]:



Σχήμα 4.38: Αρχιτεκτονική skip-gram

Εδώ ακολουθούνται και πάλι τα ίδια βήματα με πριν, από το 1 μέχρι το 3. Αφού έχει υπολογιστεί το διάνυσμα h, στην συνέχεια στο βήμα 4, το διάνυσμα αυτό πολλαπλασιάζεται με όλα τα κρυφά βάρη εξόδου $W'_{N\times V}$. Από τον πολλαπλασιασμό αυτό προκύπτουν C διανύσματα διάστασης V. Στο καθένα από τα διανύσματα αυτά εφαρμόζεται και πάλι η softmax συνάρτηση και από εκεί προκύπτουν τα C-σε πλήθος διανύσματα εξόδου y. Έτσι ως $y_{c,j}$ χαρακτηρίζεται η πιθανότητα η context λέξη c να είναι η λέξη j του λεξικού με c = 1,2, ... ,C και j = 1,2, ... , V. Ο στόχος της εκπαίδευσης στο skip-gram μοντέλο είναι, πλέον, η μεγιστοποίηση της πιθανότητας οι context λέξεις να είναι οι λέξεις που πράγματι παρατηρούνται στο δείγμα. Επομένως, η συνάρτηση κόστους εδώ ορίζεται ως εξής:

$$-\log(P(w_{0,1}, w_{0,2}, ..., w_{0,C}|w_I)) = -\log(\prod_{c=1}^{C} P(w_{0,c}|w_I)) = -\sum_{c=1}^{C} \log(P(w_{0,c}|w_I)) \qquad (4.31)$$

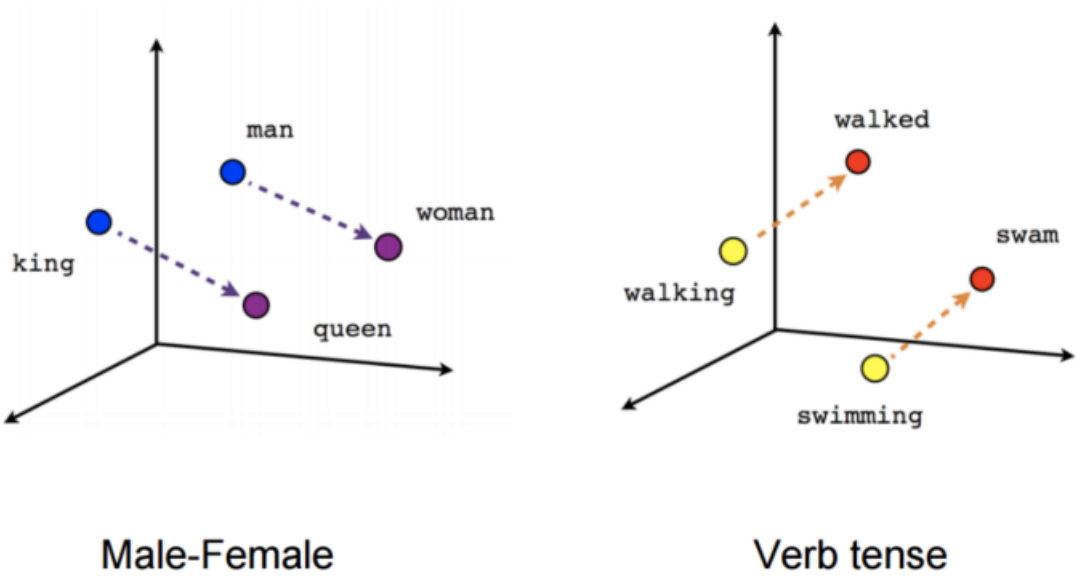
όπου στην ουσία το συνολικό σφάλμα υπολογίζεται αθροίζοντας τα επιμέρους σφάλματα για κάθε context λέξη. Δηλαδή, υπολογίζεται η σχέση 4.30, για κάθε λέξη εξόδου $w_{0,c}$ με c=1,...,C, που είναι η πιθανότητα παρατήρησης της πραγματικής λέξης-context, και έπειτα αθροίζονται οι αρνητικοί λογάριθμοι των πιθανοτήτων αυτών.

Στην συνέχεια επαναπροσδιορίζονται τα βάρη του νευρωνικού με μεθόδους stochastic gradient descent και backpropagation, όπως είδαμε πριν.

Μετά την εκπαίδευση, το διάνυσμα που τελικά μας δίνει την αναπαράσταση της λέξης στόχου, είναι το βάρος μεταξύ του στρώματος εισόδου και του κρυφού στρώματος, που έχει διάσταση N. Δηλαδή είναι το $v_{w_{j^*}}$ όπου j^* είναι η θέση της λέξης στόχου.

Και τα δύο μοντέλα επικεντρώνονται στην εκμάθηση λέξεων δεδομένου του τοπικού τους περιβάλλοντος χρήσης, όπου το περιβάλλον καθορίζεται από ένα παράθυρο γειτονικών λέξεων. Αυτό το παράθυρο είναι μια διαμορφώσιμη παράμετρος του μοντέλου.

Με το word2vec, η διάσταση αναπαράστασης μιας λέξης μειώνεται από το μέγεθος του λεξιλογίου (V) στο μήκος του κρυφού επιπέδου (N). Επιπλέον, τα διανύσματα έχουν περισσότερο «νόημα» όσον αφορά την περιγραφή της σχέσης μεταξύ λέξεων. Τα διανύσματα που λαμβάνονται αφαιρώντας δύο σχετικές λέξεις μερικές φορές εκφράζουν μια ουσιαστική έννοια όπως το φύλο ή το ρήμα, όπως φαίνεται στο παρακάτω σχήμα [91]:



Σχήμα 4.39: Οπτικοποίηση Διανυσμάτων Λέξεων

Έτσι, σε αυτήν την μέθοδο βλέπουμε να λαμβάνονται υπόψιν και τα συμφραζόμενα ή αλλιώς η σύνταξη και η σειρά των λέξεων, κάτι το οποίο δεν συνέβαινε στις προηγούμενες μεθόδους που είδαμε. Επίσης η μέθοδος αυτή, μπορεί να δώσει περισσότερες από μία αναπαραστάσεις/σημασιολογίες σε μία λέξη, αφού για κάθε context εξάγεται ένα διαφορετικό διάνυσμα αναπαράστασης. Επομένως, η ίδια λέξη σε διαφορετικά contexts μπορεί να αναπαρασταθεί διαφορετικά. Τέλος, υποστηρίζεται ότι το CBOW είναι πιο γρήγορο από το skip-gram, αλλά το skip-gram τείνει να τα πηγαίνει καλύτερα για σπάνιες λέξεις. Το Skip-gram με αρνητική υπο-δειγματοληψία ξεπερνά γενικά κάθε άλλη μέθοδο.

4.2.2.6 Glove

Ο αλγόριθμος **Global Vectors ή GloVe** είναι μια επέκταση της μεθόδου word2vec για την αποτελεσματική εκμάθηση διανυσμάτων λέξεων, που αναπτύχθηκε από τους Pennington, et al. στο Στάνφορντ. Το μοντέλο αυτό παράγει διανυσματικές αναπαραστάσεις ενσωματώνοντας συνολική (global) πληροφορία.

Ένα μειονέκτημα του αλγορίθμου word2vec είναι η μη ενσωμάτωση πληροφορίας, στις διανυσματικές αναπαραστάσεις, για τα συνολικά στατιστικά στοιχεία του σώματος κειμένου. Στα παραπάνω κεφάλαια, μεταξύ άλλων είδαμε και δύο γενικούς τρόπους εξαγωγής διανυσματικών αναπαραστάσεων, όπου ο πρώτος εκμεταλλεύεται τα συνολικά στατιστικά στοιχεία του σώματος κειμένου και εμπλέκει συνήθως μεθόδους παραγοντοποίησης του co-occurrence πίνακα 4.2.2.4, ενώ ο δεύτερος εκμεταλλεύεται τοπική πληροφορία με τη χρήση παραθύρων κειμένου 4.2.2.5.

Οι κλασικές αναπαραστάσεις λέξεων μοντέλων διανυσματικού χώρου, αναπτύχθηκαν χρησιμοποιώντας τεχνικές παραγοντοποίησης διανύσματος όπως η Latent Semantic Analysis (LSA) που είδαμε στο 4.2.2.4, οι οποίες κάνουν καλή δουλειά στη χρήση global στατιστικών κειμένου (όπως είναι ο co-occurrence πίνακας), αλλά δεν είναι τόσο καλές όσο οι

εκπαιδευμένες μέθοδοι, όπως το word2vec, στο να συλλαμβάνουν την σημασιολογία και να την επιδεικνύουν σε εργασίες όπως ο υπολογισμός αναλογιών μεταξύ λέξεων. Τέτοιες εργασίες είναι για παράδειγμα η αφαίρεση του «αντρικού» στοιχείου από το «βασιλιά» και η προσθήκη του «γυναικείου» που οδηγεί στη λέξη «βασίλισσα», καταγράφοντας την αναλογία «ο βασιλιάς είναι στη βασίλισσα όπως ο άντρας στη γυναίκα». Μια άλλη τέτοιου είδους αναλογία μπορεί να είναι μεταξύ των χρόνων ενός ρήματος: για παράδειγμα το "dance is to dancing as fly is to flying".

Το GloVe είναι μια προσέγγιση για να συνδιαστούν και τα δύο παραπάνω στοιχεία: τα **global στατιστικά** στοιχεία των τεχνικών παραγοντοποίησης πίνακα όπως το LSA, με την τοπική εκμάθηση βάσει-του-περιβάλλοντος **(context-based)** του word2vec.

Ο αλγόριθμος Glove λειτουργεί ως εξής: αντί να εξαγάγουμε τις ενσωματώσεις από ένα νευρωνικό δίκτυο που έχει σχεδιαστεί για να εκτελεί μια διαφορετική εργασία όπως η πρόβλεψη γειτονικών λέξεων (CBOW) ή η πρόβλεψη της λέξης-στόχου (Skip-Gram), οι ενσωματώσεις βελτιστοποιούνται απευθείας, έτσι ώστε το εσωτερικό γινόμενο δύο διανυσμάτων λέξεων να ισούται με τον λογάριθμο του αριθμού των φορών που οι δύο λέξεις εμφανίζονται η μία κοντά στην άλλη. Αυτό αναγκάζει το μοντέλο να κωδικοποιεί την κατανομή συχνότητας των λέξεων που εμφανίζονται κοντά μεταξύ τους σε ένα πιο συνολικό πλαίσιο (global context).

Παρακάτω παρουσιάζονται κάποια λογικά βήματα για την κατανόηση της ιδέας αυτού το αλγορίθμου και του τρόπου που αυτός δημιουργήθηκε:

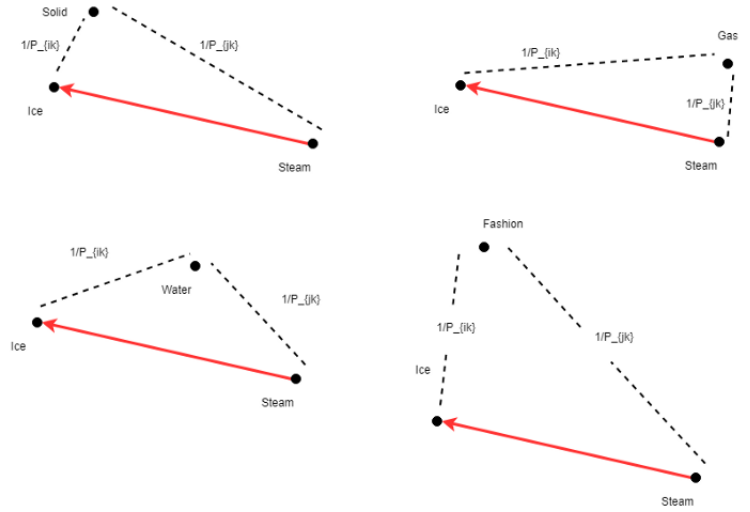
- Αρχικά, κατασκευάζουμε τον πίνακα συνύπαρξης (co-occurrence matrix) X ενός πλαισίου-κειμένου. Ο πίνακας αυτός κατασκευάζεται όπως είδαμε στο 4.3, με την διαφορά ότι εδώ μπορούμε να ορίσουμε context-windows με βάση τα οποία θα υπολογιστούν οι συνυπάρξεις του πίνακα. Δηλαδή, η συχνότητα εμφάνισης της ακολουθίας δύο λέξεων, δεν είναι απαραίτητο να υπολογιστεί μόνο για την περίπτωση που οι δύο λέξεις είναι γειτονικές κατά μία θέση, αλλά μπορεί να υπολογιστεί για μεγαλύτερες γειτονιές/παράθυρα περιβάλλοντος, όπως αυτά που είδαμε στο 4.4 και 4.5. Βέβαια, η επιλογή του πόσο μεγάλο θα είναι το παράθυρο περιβάλλοντος αλλά και του αν θα είναι συμμετρικό ή όχι (αν θα πρέπει να διαχωριστεί το αριστερό περιβάλλον από το δεξί), είναι υπερπαράμετροι του μοντέλου και τίθενται προς διερεύνηση.

- Στην συνέχεια, από τον πίνακα συνύπαρξης, μπορεί να υπολογιστεί μια μετρική η οποία δείχνει την σημασιολογική ομοιότητα ανάμεσα στις λέξεις. Αυτή η μετρική είναι το $P_{ik} = X_{ik}/X_i$, όπου P_{ik} είναι η πιθανότητα η λέξη i να είναι κοντά με την λέξη k και υπολογίζεται διαιρώντας τον αριθμό των φορών που η λέξη i και η λέξη k εμφανίστηκαν μαζί X_{ik}, με το συνολικό αριθμό των φορών που η λέξη i εμφανίστηκε στο πλαίσιο X_i.

- Αυτό που μας ενδιαφέρει είναι δεδομένων τριών λέξεων i,j και k, να υπολογιστεί η έκφραση P_{ik}/P_{jk}, η οποία δίνει μια αναλογία της πιθανότητας να εμφανιστεί μαζί η λέξη i με την k, σε σχέση με την πιθανότητα να εμφανιστεί μαζί η λέξη j με την k. Αυτό πρακτικά σημαίνει πως αν η λέξη k είναι πιο όμοια με την i από ότι με την j, τότε ο λόγος θα δίνει τιμή >1, αν είναι πιο όμοια με την j από ότι με την i, τότε ο λόγος θα δίνει τιμή <1 και αν είναι το ίδιο συγγενική και με τις δύο τότε θα δίνει τιμή κοντά στο 1.

- Η παραπάνω έκφραση για να οριστεί με την μορφή ισότητας δίνεται ως εξής: $F(w_i, w_j, u_k) = P_{ik}/P_{jk}$, όπου w και u είναι δύο ξεχωριστά επίπεδα ενσωμάτωσης. Διαφορετικά, η ισότητα μπορεί να θεωρηθεί και ως εξής: $F(w_i - w_j, u_k) = P_{ik}/P_{jk}$. Αυτή η μορφή, που εμπεριέχει την διαφορά $w_i - w_j$, έχει νόημα γιατί τα διανύσματα λέξεων είναι γραμμικά συστήματα και έτσι μπορεί να εκτελεστεί αριθμητική στο χώρο ενσωμάτωσης, αλλά

και επειδή η απόσταση μεταξύ διανυσμάτων λέξεων είναι μια σημαντική πληροφορία. Από την άλλη η απόσταση αυτή μπορεί να συνδεθεί με τις πιθανότητες P_{ik} και P_{jk} με τον τρόπο που φαίνεται στο παρακάτω σχήμα [93]:



Σχήμα 4.40: Συμπεριφορά των αποστάσεων των διανυσμάτων σε σχέση με μία λέξη

όπου βλέπουμε πως καθώς η λέξη k αλλάζει, αλλάζουν και οι αποστάσεις της από τις δύο άλλες σταθερές λέξεις j και i. Οι αποστάσεις αυτές ορίζονται ως το αντίστροφο της πιθανότητας $1/P_{ik}$ και $1/P_{jk}$, καθώς όσο πιο μεγάλη είναι η πιθανότητα οι λέξεις να βρίσκονται μαζί, τόσο πιο όμοιες είναι και άρα τόσο πιο μικρή είναι η απόστασή τους στον χώρο. Η απόσταση όμως μεταξύ των σταθερών λέξεων i,j $(w_i - w_j)$ παραμένει ίδια.

- Στην συνέχεια, ένα κύριο μέλημα είναι η μετατροπή του διανύσματος στο αριστερό μέρος της ισότητας, σε βαθμίδα (vector to scalar) όπως είναι και το δεξί μέρος της ισότητας. Αυτή η μετατροπή γίνεται εύκολα χρησιμοποιώντας transpose και εσωτερικό γινόμενο μεταξύ δύο οντοτήτων: $F((w_i - w_j)^*.u_k) = P_{ik}/P_{jk}$.

- Στην συνέχεια, γίνεται η υπόθεση ότι η F είναι ομοιόμορφη συνάρτηση και καταλήγουμε στις παρακάτω συνεπαγωγές: $F(w_i^* u_k - w_j^* u_k) = F(w_i * u_k)/F(w_j^* u_k) = P_{ik}/P_{jk}$.

- Με μία πιο αφαιρετική θεώρηση θα μπορούσε να υποτεθεί ότι $F(w_i^* u_k) = P_{ik}$ και αν F=exp τότε καταλήγουμε στα εξής: $\exp(w_i^* u_k) = P_{ik} = X_{ik}/X_i \implies w_i^* u_k = \log(X_{ik}) - \log(X_i) \implies w_i^* u_k + \log(X_i) = \log(X_{ik})$.

  Τώρα δεδομένου ότι δεν έχουμε ακόμα όρους bias στην εξίσωση, η εξίσωση θα μπορούσε να αναδιαμορφωθεί ως εξής:

$$w_i^* u_k + bw_i + bu_k = log(X_{ik}) \tag{4.32}$$

  με τα bw και bu να είναι biases του νευρωνικού.

Από την σχέση (4.32), φαίνεται τελικά ότι το εσωτερικό γινόμενο δύο διανυσμάτων λέξεων απαιτείται να ισούται με τον λογάριθμο του αριθμού των φορών που οι δύο λέξεις εμφανίζονται η μία κοντά στην άλλη, όπως είχαμε πει και παραπάνω.

Επομένως, η συνάρτηση κόστους του νευρωνικού σε αυτήν την περίπτωση ορίζεται ως εξής:

$$J = \sum_{i=1}^{V}\sum_{j=1}^{V} f(X_{ij})(w_i^T u_j + bw_i + bu_j - log(X_{ij}))^2 \tag{4.33}$$

όπου το σφάλμα που είδαμε παραπάνω (4.32) έχει υψωθεί στο τετράγωνο (MSE-mean squared error), και έχει πολλαπλασιαστεί με το $\boldsymbol{f(X_{ij})}$ που είναι μια συνάρτηση στάθμισης (weighting function). Στο τέλος αθροίζονται τα σφάλματα όλων των ζευγών λέξεων για την εκπαίδευση του νευρωνικού. Η συνάρτηση στάθμισης f χρησιμοποιείται με σκοπό να αποτραπεί το πρόβλημα που θα εμφανιζόταν σε περίπτωση που $X_{ij} = 0$ και ορίζεται ως εξής:

$$\begin{cases} f(x) = (\frac{x}{x_{max}})^a & , if \quad x < x_{max} \\ 1 & , otherwise \end{cases}$$

Μια πιο εκτενής ανάλυση του αλγορίθμου Glove καθώς και του παραλληλισμού του με άλλα μοντέλα όπως το skip-gram μπορεί να βρεθεί εδώ [94].

4.2.2.7 FastText

Το **FastText** είναι μια βιβλιοθήκη ανοιχτού κώδικα, που δημιουργήθηκε από την ομάδα έρευνας τεχνητής νοημοσύνης του Facebook (Bojanowski et al) το 2016, για αποτελεσματική εκμάθηση αναπαράστασης λέξεων και ταξινόμησης προτάσεων [95]. Είναι μια επέκταση του word2vec, η οποία χρησιμοποιεί τα n-grams των λέξεων (4.2.2.2).

Αντί να τροφοδοτεί μεμονωμένες λέξεις στο Νευρωνικό Δίκτυο, το FastText διασπά τις λέξεις σε αρκετά n-grams (υπο-λέξεις). Για παράδειγμα, τα 3-grams για τη λέξη "artificial" είναι <ar, art, rti, tif, ifi, fic, ici, ial, al> (όπου οι γωνιακές αγκύλες υποδεικνύουν την αρχή και το τέλος της λέξης). Ενώ αυτό προσθέτει πολλούς επιπλέον υπολογισμούς στην εκπαίδευση, επιτρέπει στις ενσωματώσεις λέξεων να κωδικοποιούν πληροφορίες υπο-λέξεων. Αυτό βοηθά στη σύλληψη της σημασίας των μικρότερων λέξεων και επιτρέπει στις ενσωματώσεις να κατανοήσουν τα επιθέματα και τα προθέματα.

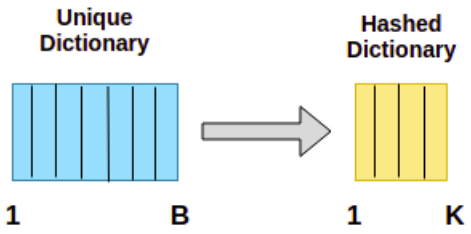
Μόλις αναπαριστάται η λέξη χρησιμοποιώντας χαρακτήρες n-grams, ένα μοντέλο skip-gram εκπαιδεύεται για να μάθει τις ενσωματώσεις. Αυτό το μοντέλο θεωρείται ως μοντέλο σάκου λέξεων με συρόμενο παράθυρο πάνω από μια λέξη, επειδή δεν λαμβάνεται υπόψη η εσωτερική δομή της λέξης. Όσο οι χαρακτήρες βρίσκονται μέσα σε αυτό το παράθυρο, η σειρά των n-grams δεν έχει σημασία.

Πιο αναλυτικά η δομή και η εκπαίδευση του νευρωνικού αυτού, παρουσιάζεται στα επόμενα βήματα:

1. Αρχικά η λέξη-στόχος που θα τροφοδοτηθεί στο νευρωνικό skip-gram ως είσοδος, διασπάται σε n-grams. Έστω ότι έχουμε την λέξη eating και την διασπάμε σε 3-grams. Έτσι παίρνουμε την παρακάτω λίστα χαρακτήρων 3-grams της λέξης [96]:

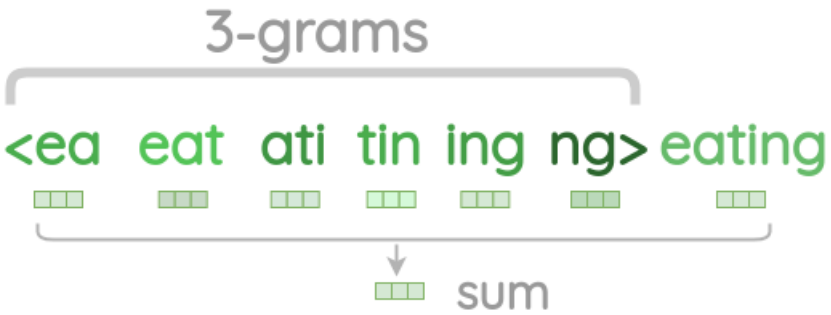
Σχήμα 4.41: 3-grams της λέξης "eating"

2. Στην συνέχεια εφαρμόζεται ένας κατακερματισμός (hashing). Αντί να μαθαίνουμε μια ενσωμάτωση για κάθε μοναδικό n-gram, μαθαίνουμε συνολικές ενσωματώσεις K όπου το K υποδηλώνει το μέγεθος του κάδου (bucket size). Πιο συγκεκριμένα, για να περιοριστούν οι απαιτήσεις μνήμης του μοντέλου, χρησιμοποιείται μια συνάρτηση κατακερματισμού που αντιστοιχίζει τα n-grams σε ακέραιους αριθμούς 1 έως K. Οι ακολουθίες χαρακτήρων κατακερματίζονται χρησιμοποιώντας τη λειτουργία κατακερματισμού Fowler-Noll-Vo (συγκεκριμένα την παραλλαγή FNV-1a) [99]. Τελικά, μια λέξη αντιπροσωπεύεται από τον δείκτη της στο λεξικό λέξεων και το σύνολο κατακερματισμένων n-grams που περιέχει.



Σχήμα 4.42: Κατακερματισμός

3. Σε αυτό το σημείο, υπολογίζεται η ενσωμάτωση για την λέξη στόχο, δηλαδή το διάνυσμά της, αθροίζοντας τα διανύσματα όλων των n-gram της καθώς και της ίδιας [96]:



Σχήμα 4.43: Ενσωμάτωση της Λέξης Στόχου

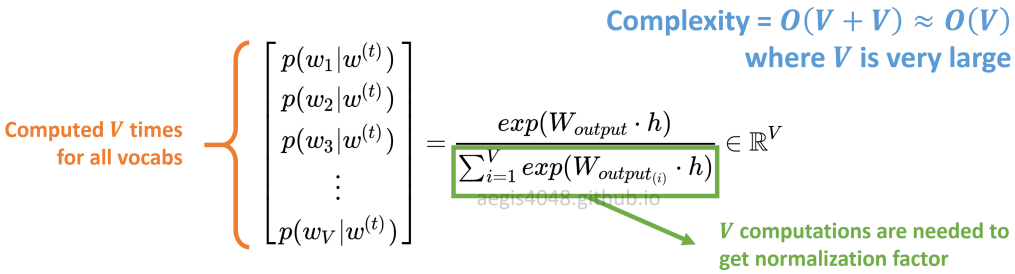
4. Για τις context λέξεις οι οποίες καθορίζονται όπως έχουμε δει από ένα context παράθυρο, λαμβάνονται απευθείας τα ίδια τα διανύσματά τους χωρίς την προσθήκη n-grams [96]:

Σχήμα 4.44: Context λέξεις

5. Ο αλγόριθμος skip-gram που παρουσιάστηκε στο κεφάλαιο 4.2.2.5 ονομάζεται **vanilla Skip-Gram** και ερμηνεύεται ως ένα πρόβλημα πολλαπλών κλάσεων (multiclassifica-

tion task) και τα διανύσματα εξόδου είναι διάστασης V, όσος και ο συνολικός αριθμός λέξεων στο λεξικό. Τα διανύσματα αυτά παρουσιάζουν την πιθανότητα που προκύπτει για κάθε λέξη του λεξικού έπειτα από την εφαρμογή της συνάρτησης softmax στην έξοδο του νευρωνικού, η οποία στην περίπτωση των λέξεων που ανήκουν πραγματικά στο περιβάλλον (context), επιδιώκεται να είναι ίση με 1. Ουσιαστικά, είναι μια ταξινόμηση μεταξύ V κλάσεων και τα προκύπτοντα διανύσματα περιέχουν τις πιθανότητες, η λέξη πρόβλεψης να ανήκει στην κάθε κλάση.

Η συνάρτηση ενεργοποίησης softmax είναι υπολογιστικά πολύ ακριβή, καθώς απαιτεί σάρωση σε ολόκληρο τον πίνακα ενσωμάτωσης εξόδου (Woutput) για τον υπολογισμό της κατανομής πιθανότητας όλων των λέξεων V, όπου το V μπορεί να είναι εκατομμύρια ή περισσότερα. Οι συνολικοί υπολογισμοί που γίνονται ώστε να προκύψει το διάνυσμα εξόδου είναι O(V+V), αφού χρειάζονται V υπολογισμοί, ένας για κάθε λέξη του λεξικού, συν V για τον υπολογισμό του συντελεστή κανονικοποίησης του παρονομαστή, ο οποίος υπολογίζεται μία φορά και είναι ίδιος για όλες τις λέξεις [97]:



Σχήμα 4.45: Πολυπλοκότητα του αλγορίθμου Vanilla Skip-Gram

Λόγω αυτής της υπολογιστικής αναποτελεσματικότητας, το softmax δεν χρησιμοποιείται στις περισσότερες εφαρμογές του Skip-Gram, ούτε και στον αλγόριθμο FastText. Αντ ᾽αυτού, χρησιμοποιούμε μια εναλλακτική που ονομάζεται αρνητική δειγματοληψία (negative sampling) με τη συνάρτηση σιγμοειδούς (sigmoid function), η οποία αναδιατυπώνει το πρόβλημα σε ένα σύνολο ανεξάρτητων εργασιών δυαδικής ταξινόμησης.

Πιο συγκεκριμένα, στην αναδιατύπωση του προβλήματος ως συνόλου ανεξάρτητων εργασιών δυαδικής ταξινόμησης, ο στόχος είναι να προβλεφθεί ανεξάρτητα η παρουσία (ή απουσία) των λέξεων περιβάλλοντος (context words). Επομένως, για μια λέξη-στόχο, θεωρούμε όλες τις λέξεις περιβάλλοντος ως θετικά παραδείγματα και όλες τις άλλες λέξεις, ως αρνητικά. Αντί να προσπαθούμε να προβλέψουμε την πιθανότητα να είναι μια λέξη κοντινή για όλες τις λέξεις στο λεξιλόγιο, προσπαθούμε να προβλέψουμε την πιθανότητα του αν τα δείγματα λέξεων εκπαίδευσης είναι γείτονες ή όχι. Για παράδειγμα, έχοντας ως λέξη στόχο την λέξη "πορτοκάλι" και ως context λέξη, την λέξη "χυμός", δεν προσπαθούμε να προβλέψουμε την πιθανότητα ο χυμός να είναι μια κοντινή λέξη, δηλαδή P (χυμός | πορτοκάλι), προσπαθούμε να προβλέψουμε εάν οι λέξεις (πορτοκάλι, χυμός) είναι κοντινές λέξεις ή όχι με τον υπολογισμό του P (1 | <πορτοκάλι, χυμός>). Έτσι, αντί να έχουμε ένα γιγαντιαίο softmax - ταξινόμηση μεταξύ V κλάσεων, το έχουμε μετατρέψει σε V δυαδικά προβλήματα ταξινόμησης.

Ο νέος στόχος είναι να προβλεφθεί, για κάθε δεδομένο ζεύγος περιβάλλοντος-λέξης (w, c), εάν η λέξη (c) βρίσκεται στο παράθυρο περιβάλλοντος της κεντρικής λέξης (w) ή όχι. Δεδομένου ότι ο στόχος είναι να προσδιοριστεί μια δεδομένη λέξη ως True (θετική, D = 1) ή False (αρνητική, D = 0), χρησιμοποιούμε τη συνάρτηση σιγμοειδούς

(sigmoid function) αντί για τη συνάρτηση softmax. Η πιθανότητα μιας λέξης (c) να εμφανίζεται στο πλαίσιο της κεντρικής λέξης (w) μπορεί να οριστεί ως εξής:

$$p(D = 1|w, c; \theta) = \frac{1}{1 + exp(-\bar{c}_{output_{(j)}} \cdot w)} \in \mathbb{R}^1 \tag{4.34}$$

όπου c είναι η λέξη που θέλουμε να μάθουμε εάν ανήκει στο παράθυρο περιβάλλοντος ή όχι, w είναι η λέξη εισαγωγής (κέντρο) και θ είναι ο πίνακας βαρών που περνά στο μοντέλο. Σημειώνεται ότι το w είναι ισοδύναμο με το κρυφό επίπεδο (h) και το $\bar{c}_{output_{(j)}}$ είναι το διάνυσμα λέξης από τον πίνακα βαρών εξόδου (Woutput).

Σε αυτό το σημείο εφαρμόζεται μια ακόμη μέθοδος που δεν έχει αναφερθεί προηγουμένως στο skipgram. Η μέθοδος αυτή ονομάζεται αρνητική δειγματοληψία (negative sampling). Η αρνητική δειγματοληψία μας επιτρέπει να τροποποιούμε μόνο ένα μικρό ποσοστό των βαρών, και όχι όλα τα βάρη για κάθε δείγμα εκπαίδευσης. Έτσι, απλοποιούμε περαιτέρω το πρόβλημα επιλέγοντας τυχαία έναν μικρό αριθμό «αρνητικών» λέξεων k (μια υπερ-παράμετρος, ας πούμε 5) για τις οποίες να ενημερώσουμε τα βάρη (μια "αρνητική" λέξη είναι αυτή για την οποία θέλουμε το δίκτυο να εξάγει 0, δηλαδή να μην ανήκει στο context window). Για το εκπαιδευτικό μας δείγμα (πορτοκάλι, χυμός), θα πάρουμε πέντε λέξεις, ας πούμε μήλο, δείπνο, σκύλο, καρέκλα, σπίτι και θα τις χρησιμοποιήσουμε ως αρνητικά δείγματα. Για αυτήν τη συγκεκριμένη επανάληψη θα υπολογίσουμε μόνο τις πιθανότητες για χυμό, μήλο, δείπνο, σκύλο, καρέκλα, σπίτι. Ως εκ τούτου, η απώλεια θα μεταδοθεί μόνο για αυτές και επομένως θα ενημερωθούν μόνο τα βάρη που αντιστοιχούν σε αυτές.

Οι συνολικοί υπολογισμοί που γίνονται, πλέον, ώστε να προκύψει το διάνυσμα εξόδου είναι O(K+1), αφού υπολογίζονται οι πιθανότητες μόνο για την εκάστοτε context λέξη για τις αντίστοιχες K αρνητικές που έχουν επιλεχθεί:

$$\begin{bmatrix} p(D = 1|w, c_{pos}) \\ p(D = 1|w, c_{neg,1}) \\ p(D = 1|w, c_{neg,2}) \\ p(D = 1|w, c_{neg,3}) \\ \vdots \\ p(D = 1|w, c_{neg,K}) \end{bmatrix} = \frac{1}{1 + exp(-(\{\bar{c}_{pos}\} \cup \bar{W}_{neg}) \cdot h)} \in \mathbb{R}^{K+1} \tag{4.35}$$

Η νέα αντικειμενική συνάρτηση για skip-gram με αρνητική δειγματοληψία δίνεται από την παρακάτω έκφραση:

$$J(\theta; w, c_{pos}) = -log\, \sigma(\bar{c}_{pos} \cdot h) - \sum_{c_{neg} \in W_{neg}} log\, \sigma(-\bar{c}_{neg} \cdot h) \tag{4.36}$$

όπου θ είναι ο πίνακας βαρών που περνά στο μοντέλο, w είναι η κεντρική λέξη-στόχος, c_{pos} είναι η θετική λέξη-δείγμα, δηλαδή αυτή που ανήκει στο περιβάλλον της w, σ είναι η σιγμοειδής συνάρτηση, $\bar{c}_{pos}$ είναι το διάνυσμα της θετικής λέξης c_{pos} από τον πίνακα βαρών εξόδου (Woutput), h είναι το διάνυσμα του κρυφού επιπέδου, και $\bar{c}_{neg}$ είναι το διάνυσμα από τον πίνακα βαρών εξόδου (Woutput) της αρνητικής λέξης c_{neg}, που ανήκει στο σύνολο αρνητικών λέξεων W_{neg} που έχουμε δειγματοληπτήσει.

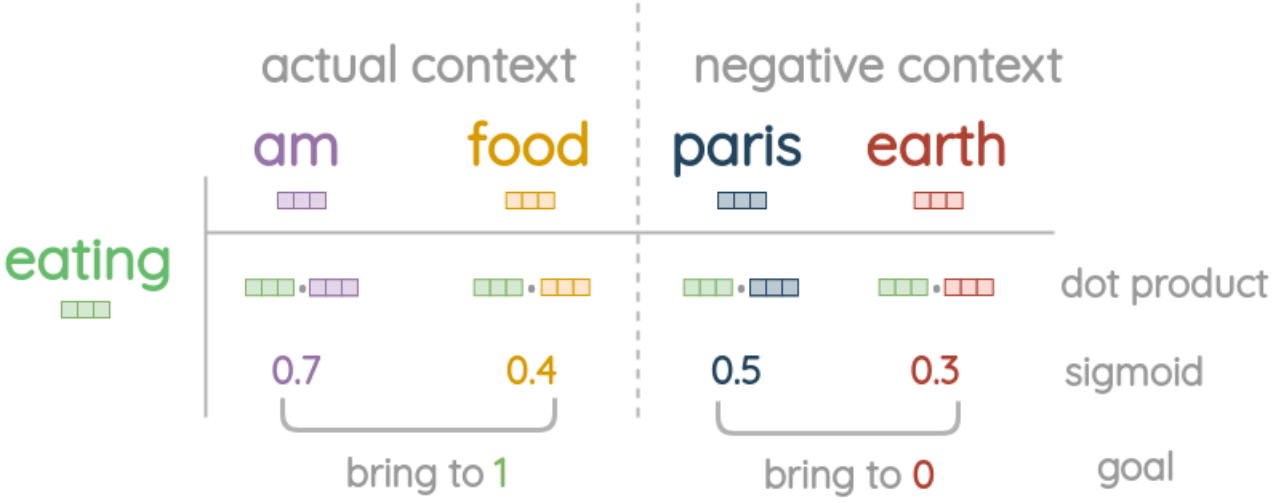
Με την ελαχιστοποίηση της παραπάνω συνάρτησης κόστους, ο πρώτος όρος προσπαθεί να μεγιστοποιήσει την πιθανότητα εμφάνισης για λέξεις που βρίσκονται πραγματικά στο παράθυρο περιβάλλοντος, δηλαδή συνυπάρχουν, ενώ ο δεύτερος όρος, προσπαθεί να μαξέψει κάποιες τυχαίες λέξεις j που δεν βρίσκονται στο παράθυρο και να ελαχιστοποιήσει την πιθανότητα συνύπαρξής τους με την κεντρική λέξη.

Για το παράδειγμα που έχουμε δει ως τώρα, με κεντρική λέξη την "eating" και context λέξεις τις "am" και "food", έστω ότι δειγματοληπτούμε δύο αρνητικές λέξεις [96]:

Σχήμα 4.46: Αρνητικά δείγματα

6. Στην συνέχεια, αφού έχουν εισαχθεί στο νευρωνικό τα word vectors των ngrams της κεντρικής λέξης "eating", μαζί με την ίδια την λέξη και έχει υπολογιστεί ο μέσος όρος τους, στην συνέχεια ο μέσος όρος αυτός που αποτελεί το διάνυσμα h του hidden layer, πολλαπλασιάζεται με εσωτερικό γινόμενο με τα βάρη Woutput.

7. Από το προκύπτον διάνυσμα μας ενδιαφέρουν μόνο οι σειρές που αντιστοιχούν στα K+1 στοιχεία, δηλαδή στην context λέξη και στα K αρνητικά δείγματα. Για αυτές τις σειρές του προκύπτοντος διανύσματος, υπολογίζεται η σιγμοειδής συνάρτηση, ώστε να προκύψει για την καθεμία από αυτές μια πιθανότητα. Η πιθανότητα αυτή αντικατοπτρίζει το πόσο πιθανό είναι οι συγκεκριμένες λέξεις να εμφανίζονται στο context. Στην περίπτωσή μας, όπου οι context λέξεις είναι οι "am" και "food" και τα αρνητικά δείγματα είναι τα "paris" , "earth" η διαδικασία αυτή φαίνεται παρακάτω [96]:



Σχήμα 4.47: Στόχος του FastText

Όπως φαίνεται και στο σχήμα, μετά την εφαρμογή της σιγμοειδούς συνάρτησης, στόχος είναι οι πιθανότητες τον πραγματικών context λέξεων να έρθουν όσο πιο κοντά γίνεται στο 1, δηλαδή να ταξινομηθούν με όσο το δυνατόν μεγαλύτερη πιθανότητα γίνεται, ως θετικά δείγματα, ενώ οι πιθανότητες των αρνητικών δειγμάτων

να έρθουν όσο πιο κοντά στο 0 γίνεται, δηλαδή να ταξινομηθούν με όσο το δυνατόν μικρότερη πιθανότητα γίνεται ως θετικά δείγματα.

Ένα τεράστιο πλεονέκτημα του FastText είναι ότι μπορεί να εξάγει διανυσματικές αναπαραστάσεις για λέξεις που δεν υπάρχουν στο λεξικό, δηλαδή που δεν έχει δει ο αλγόριθμος κατά την εκπαίδευση. Αυτό επιτυγχάνεται μέσω των n-grams που χρησιμοποιούνται, αφού η αναπαράσταση μιας λέξης είναι το άθροισμα των αναπαραστάσεων των n-grams της και ακόμα και αν η λέξη δεν υπάρχει αυτούσια, μπορεί να βρεθεί στο λεξικό κάποιο απ τα n-grams της. Με λίγα λόγια, αυτό το απλό μοντέλο επιτρέπει την κοινή χρήση των αναπαραστάσεων μεταξύ λέξεων, βοηθώντας έτσι την εκμάθηση αξιόπιστης αναπαράστασης για σπάνιες λέξεις. Το φαινόμενο αυτό δεν παρουσιάζεται στα μοντέλα word2vec και glove, τα οποία όταν μια λέξη δεν υπάρχει στο λεξικό δίνουν out-of-vocabulary (OOV).

Το FastText τα πηγαίνει επίσης καλύτερα σε γλώσσες που είναι μορφολογικά πλούσιες, όπως είναι τα γερμανικά όπου πολλές λέξεις είναι σύνθετες. Για παράδειγμα, η ονομαστική φράση "table tennis" γράφεται με μία λέξη ως "Tischtennis". Αξιοποιώντας τις ομοιότητες σε επίπεδο χαρακτήρων μεταξύ «Tischtennis» και «Tennis», το μοντέλο δεν αντιμετωπίζει τις δύο λέξεις ως εντελώς διαφορετικές λέξεις, αλλά μπορεί να καταλάβει ότι υπάρχει ένας συγγενικός βαθμός αφού έχουν κάποια κοινά n-grams.

Μια λέξη η οποία έχει κάποιο ορθογραφικό λάθος και κατ᾽ επέκτασιν ο ομιλητής δεν την έχει προφέρει σωστά, δεν αποτελεί πρόβλημα για το μοντέλο αυτό, καθώς μέσω των n-grams μπορεί να την συνδέσει με την σωστή -σωστά ορθογραφημένη- εκδοχή της. Αυτό είναι ένα μεγάλο πλεονέκτημα σε εργασίες που αφορούν την ομιλία (speech), όπως συμβαίνει στην παρούσα διπλωματική, καθώς είναι πολύ πιθανό να υπάρχουν λανθασμένες προφορές λέξεων οι οποίες δεν θα έπρεπε να οδηγούν σε λάθος σημασιολογία.

Το FastText συνδυάζει πληροφορίες από την εσωτερική δομή μιας λέξης (n-grams) αλλά και από τις λέξεις περιβάλλοντος που την περικλείουν (context words). Αυτό του δίνει την δυνατότητα να συγκρίνει τις λέξεις με βάση όχι μόνο τις λεκτικές παραλλαγές, όπως κάνουν οι παραδοσιακές μέθοδοι υπολογισμού απόστασης λέξεων, αλλά και τις σημασιολογικά σχετικές λέξεις.

Στην παρούσα διπλωματική εργασία, δοκιμάστηκε μεταξύ άλλων και το μοντέλο FastText για να εξαχθούν οι ενσωματώσεις των λέξεων ή αλλιώς τα χαρακτηριστικά τους. Πιο συγκεκριμένα, χρησιμοποιήσαμε ένα προ-εκπαιδευμένο μοντέλο FastText το οποίο έχει εκπαιδευτεί σε δεδομένα της Wikipedia 2017 [95] [98].

Το προεκπαιδευμένο μοντέλο αυτό έχει εκπαιδευτεί με τα εξής χαρακτηριστικά:

- Έχει χρησιμοποιηθεί η αρχιτεκτονική skip-gram (και όχι CBOW).
- Τα διανύσματα λέξεων έχουν διάσταση = 300. Αυτό σημαίνει ότι το κρυφό στρώμα (hidden layer) αποτελείται από το διάνυσμα h που έχει διάσταση 300.
- Για κάθε θετικό δείγμα, δηλαδή δείγμα που ανήκει στο context window, έχουν επιλεχθεί 5 αρνητικά δείγματα. Ο τρόπος που δειγματοληπτήθηκαν αυτά τα δείγματα είναι με πιθανότητα ανάλογη της τετραγωνικής ρίζας της uni-gram συχνότητας και η κατανομή αυτή ονομάζεται **κατανομή θορύβου (noise distribution)**. Η μαθηματική έκφραση της κατανομής δίνεται ως εξής:

$$P_n(w) = \left(\frac{U(w)}{Z} \right)^{\alpha} \tag{4.37}$$

είναι ο αριθμός των φορών που η λέξη (w) εμφανίζεται μέσα στο σώμα (corpus), διαιρεμένος με έναν συντελεστή ομαλοποίησης Z έτσι ώστε η κατανομή να γίνει κατανομή πιθανότητας εύρους

$$0, 1$$

και να αθροίζει στο 1. Στην περίπτωση του προ-εκπαιδευμένου FastText μοντέλου το α ισούται με $\frac{1}{2}$.

- Το παράθυρο περιβάλλοντος (context window) είναι μεγέθους c, όπου το c δειγματοληπτείται ομοιόμορφα μεταξύ του 1 και του 5.
- Οι πιο συχνές λέξεις υφίστανται subsampling με όριο απόρριψης το 10^{-4}. Το subsampling είναι μια τεχνική κατά την οποία περιορίζεται ο αριθμός των δειγμάτων για μια λέξη, περιορίζοντας τη συχνότητα εμφάνισής της.
- Χρησιμοποιείται συνάρτηση κατακερματισμού Fowler-Noll-Vo που αντιστοιχίζει τα n-grams σε ακέραιους αριθμούς 1 έως K, με K = 2.10^6.
- Κατά τη δημιουργία του λεξικού λέξεων, διατηρούνται οι λέξεις που εμφανίζονται τουλάχιστον 5 φορές στο σύνολο εκπαίδευσης.
- Το μέγεθος βήματος γ_0 ορίζεται στα 0,05 για το μοντέλο.

Για την εξαγωγή χαρακτηριστικών κειμένου στο σύστημά μας, δοκιμάστηκε η χρήση όλων των διανυσμάτων (embeddings), του συγκεκριμένου προ-εκπαιδευμένου λεξικού, αλλά και η χρήση μόνο των πρώτων 5.10^5 διανυσμάτων λέξεων. Αυτό έγινε για λόγους μνήμης, καθώς για να φορτωθεί ολόκληρο το προ-εκπαιδευμένο μοντέλο Fasttext, χρειάστηκαν πολύ μεγάλοι πόροι σε ο,τι αφορά την μνήμη ram και τα συμβατικά μηχανήματα, αδυνατούσαν να το φορτώσουν. Για αυτόν τον λόγο, δοκιμάστηκε και αυτή η μικρή περικοπή που δεν φάνηκε να κοστίζει αρκετά σε απόδοση, μια και οι λέξεις στο λεξικό είναι ταξινομημένες με βάση την συχνότητα εμφάνισης και επομένως οι πρώτες 5.10^5 είναι οι πιο συχνά εμφανιζόμενες. Βέβαια, σε αυτό το σημείο πρέπει να σημειωθεί ότι όταν φορτώνεται ένα μέρος του προ-εκπαιδευμένου Fasttext, χάνεται η δυνατότητα εξαγωγής διανυσμάτων λέξεων με βάση τα n-grams και ως εκ τούτου υπάρχουν περισσότερες πιθανότητες εμφάνισης λέξεων εκτός λεξικού (out-of-vocabulary words).

Συμπερασματικά, όπως έχουμε ήδη πει, τα χαρακτηριστικά σε επίπεδο τμήματος, εξάγονται μετά την τμηματοποίηση της πληροφορίας. Στην συγκεκριμένη περίπτωση, αφού το κείμενο έχει διασπαστεί σε τμήματα με κάποιον από τους τρόπους που παρουσιάστηκαν στο 4.2.1, εξάγονται οι αναπαραστάσεις λέξεων για κάθε τμήμα ξεχωριστά. Αυτό πρακτικά σημαίνει πως για κάθε λέξη ενός τμήματος, εξάγονται 300 χαρακτηριστικά από το παραπάνω λεξικό, ή αλλιώς ένα διάνυσμα διάστασης 300. Για την αναπαράσταση του συνολικού τμήματος, υπολογίζεται ένας μέσος όρος αυτών των διανυσμάτων από όλες τις λέξεις που περιέχονται στο τμήμα. Τελικά, καταλήγουμε με 300 χαρακτηριστικά ανά τμήμα/πρόταση.

Ένα επιπλέον βήμα το οποίο δεν έχει αναφερθεί ως τώρα είναι η **προεπεξεργασία (preprocessing)** του κειμένου. Αφού έχουμε διασπάσει το κείμενο σε τμήματα, εφαρμόζουμε μια επεξεργασία στο κάθε τμήμα, με την οποία αφαιρούμε τους ειδικούς χαρακτήρες και τα σημεία στίξης, μετατρέπουμε τους αριθμούς σε λέξεις/ολογράμματα και κάνουμε όλα τα γράμματα από κεφαλαία, σε μικρά. Αυτή η προεπεξεργασία γίνεται για να βοηθήσει στην ανίχνευση των λεκτικών αναπαραστάσεων. Για παράδειγμα, η λέξη "apple" και η λέξη "Apple", θέλουμε να δίνει την ίδια διανυσματική αναπαράσταση, η οποία δεν πρέπει να αλλοιώνεται εξαιτίας του κεφαλαίου γράμματος. Το ίδιο συμβαίνει και στην περίπτωση των σημείων στίξης όπως για παράδειγμα "apple" με "apple.". Φυσικά, υπάρχουν πολλές ακόμα μέθοδοι προεπεξεργασίας κειμένου στον τομέα της επεξεργασίας φυσικής γλώσσας, όπως είναι το stemming, το lemmatization η αφαίρεση των stopwords κ.α. Οι μέθοδοι αυτοί, αλλάζουν την μορφή της λέξης, για παράδειγμα απομονώνουν την ρίζα της (π.χ. από troubled σε troubl με stemming ή σε trouble με lemmatization) ή στην περίπτωση του stopwords removal, αφαιρούν τις συχνά χρησιμοποιούμενες λέξεις όπως "is" "the" κλπ.

Σε εργασίες αναγνώρισης συναισθήματος, όμως, τέτοιες παραποιήσεις μπορεί να αποβούν μοιραίες καθώς, για παράδειγμα, το "do" από το "donnot" μπορεί να δείχνει διαφορετικό συναίσθημα ή έκφραση, η οποία θα αναγνωριστεί ως αυτούσια αν το "donnot" μετατραπεί σε "do" ή θα χαθεί εντελώς η διαχωριστική πληροφορία αν οι λέξεις αφαιρεθούν από το κείμενο. Για τον λόγο αυτό, αποφεύχθηκαν τέτοιες μέθοδοι προεπεξεργασίας για την παρούσα εργασία.

Παρά τα μεγάλα πλεονεκτήματα των μοντέλων Fasttext, η χρήση τους εξακολούθησε να μας δίνει λέξεις εκτός λεξικού, αναγκάζοντάς μας έτσι να απορρίπτουμε εντελώς τις λέξεις αυτές, μια και δεν ήταν δυνατή η αναπαράστασή τους. Αυτό το γεγονός, σε συνδυασμό με το γεγονός ότι θέλαμε να δοκιμάσουμε και άλλες μεθόδους εξαγωγής αναπαραστάσεων λέξεων, με απώτερο στόχο την βελτίωση της απόδοσης των ταξινομητών τμημάτων, μας οδήγησε στην δοκιμή και χρήση των bert embeddings.

Τα bert embeddings, εξάγονται ουσιαστικά από την αρχιτεκτονική μετασχηματιστών (2.6), ένα είδος τεχνητών νευρωνικών δικτύων που εμφανίζεται πολύ δυνατό στον τομέα της επεξεργασίας φυσικής γλώσσας και δεν θα μπορούσαμε να παραβλέψουμε στην παρούσα διπλωματική εργασία. Η ακριβής λειτουργία τους δίνεται στο παρακάτω κεφάλαιο.

4.2.2.8 BERT

Το **BERT (Bidirectional Encoder Representations from Transformers)** είναι μια πρόσφατη εργασία που δημοσιεύθηκε από ερευνητές της Google AI. Προκάλεσε αναταραχή στην κοινότητα Μηχανικής Μάθησης παρουσιάζοντας αποτελέσματα τελευταίας τεχνολογίας σε μια ευρεία ποικιλία εργασιών NLP, όπως Question Answering, Natural Language Inference και άλλα.

Η βασική τεχνική καινοτομία του BERT είναι η εφαρμογή της αμφίδρομης εκπαίδευσης του Transformer (βλέπε 2.6), ενός δημοφιλούς μοντέλου προσοχής, στη μοντελοποίηση γλωσσών. Αυτό έρχεται σε αντίθεση με τις προηγούμενες προσπάθειες που εξέταζαν μια ακολουθία κειμένου είτε από αριστερά προς δεξιά είτε με συνδυασμένη εκπαίδευση από αριστερά προς δεξιά και από δεξιά προς αριστερά. Τα αποτελέσματα της δημοσίευσης [100] δείχνουν ότι ένα γλωσσικό μοντέλο το οποίο είναι διπλής κατεύθυνσης εκπαιδευμένο μπορεί να έχει μια βαθύτερη αίσθηση του γλωσσικού περιβάλλοντος και της ροής από τα μονόπλευρα γλωσσικά μοντέλα. Στην εργασία, οι ερευνητές περιγράφουν λεπτομερώς μια νέα τεχνική που ονομάζεται Masked LM (MLM) που επιτρέπει αμφίδρομη εκπαίδευση σε μοντέλα στα οποία ήταν προηγουμένως αδύνατο.

Ο BERT χρησιμοποιεί τον Transformer, έναν μηχανισμό προσοχής που μαθαίνει τις σχέσεις μεταξύ των λέξεων (ή υπο-λέξεων) σε ένα κείμενο. Όπως έχουμε δει και στο κεφάλαιο 2.6 της θεωρίας, οι transformers αποτελούνται από δύο δομικά μέρη: τους κωδικοποιητές και τους αποκωδικοποιητές. Οι μεν πρώτοι διαβάζουν την κειμενική είσοδο και οι δε δεύτεροι παράγουν μια πρόβλεψη για μια συγκεκριμένη εργασία. Δεδομένου ότι ο στόχος του BERT είναι να δημιουργήσει ένα μοντέλο γλώσσας, απαιτείται μόνο ο μηχανισμός κωδικοποίησης.

Σε αντίθεση με τα κατευθυντικά μοντέλα, τα οποία διαβάζουν διαδοχικά την εισαγωγή κειμένου (από αριστερά προς δεξιά ή από δεξιά προς τα αριστερά), ο κωδικοποιητής Transformer διαβάζει ταυτόχρονα ολόκληρη την ακολουθία λέξεων. Επομένως, θεωρείται αμφίδρομο μοντέλο (bidirectional), αν και θα ήταν πιο ακριβές να πούμε ότι είναι μη-κατευθυντικό (non-directional). Αυτό το χαρακτηριστικό επιτρέπει στο μοντέλο να μάθει το περιεχόμενο μιας λέξης με βάση όλα το περιβάλλον της (αριστερά και δεξιά της λέξης).

Στρατηγικές Εκπαίδευσης

Το ερώτημα εδώ είναι ποίος θα είναι ο στόχος πρόβλεψης του BERT μοντέλου. Τα περισσότερα γλωσσικά μοντέλα προβλέπουν την επόμενη λέξη της ακολουθίας ή κάποιες

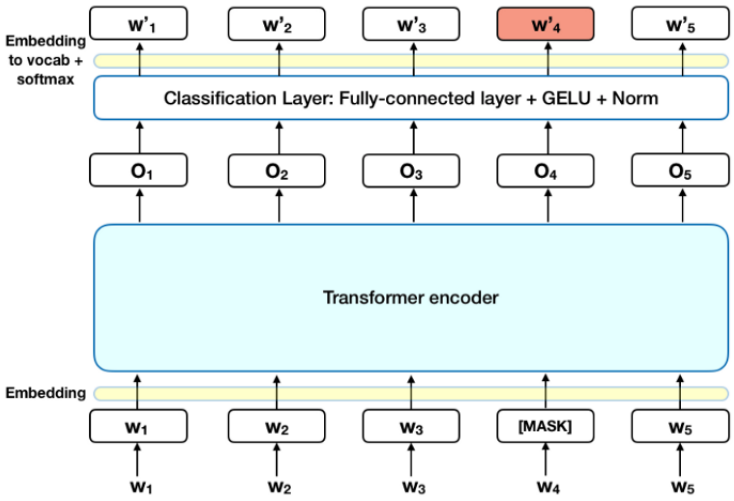
γύρω λέξεις όπως είναι το Word2Vec και το Fasttext. Το Bert, χρησιμοποιεί δύο στρατηγικές εκπαίδευσης:

1. **Masked LM (MLM)**
   Πριν τροφοδοτηθούν οι ακολουθίες λέξεων στο μοντέλο, το 15% των λέξεων σε κάθε ακολουθία, αντικαθίσταται από μία ένδειξη (token) [MASK]. Έτσι, το μοντέλο προσπαθεί να προβλέψει τις αρχικές τιμές αυτών των λέξεων, οι οποίες αποκρύφτηκαν μέσω της μάσκας, βασιζόμενο στο περιβάλλον που σχηματίζεται από τις υπόλοιπες λέξεις της ακολουθίας.

   Για να γίνει αυτή η πρόβλεψη, προστίθεται ένα στρώμα ταξινόμησης στην έξοδο του κωδικοποιητή, το οποίο είναι ουσιστικά ένα πλήρως συνδεδομένο στρώμα με μίασυνάρτηση ενεργοποίησης gelu και ομαλοποίηση. Στην συνέχεια, τα διανύσματα εξόδου πολλαπλασιάζονται με τον πίνακα ενσωμάτωσης ώστε να αποκτήσουν την διάσταση του λεξιλογίου και τέλος, περνούν από την συνάρτηση ενεργοποίησης softmax για να δοθεί μια πιθανότητα για κάθε λέξη του λεξικού.

   Η συνάρτηση κόστους, λαμβάνει υπόψιν μόνο τις προβλέψεις των λέξεων με μάσκα και αγνοεί τις προβλέψεις των λέξεων χωρίς μάσκα.

   Στο παρακάτω σχήμα φαίνεται αυτή η αρχιτεκτονική [101]:



Σχήμα 4.48: Αρχιτεκτονική Bert με Masked LM στρατηγική

   Παρόλο που οι ενδείξεις [MASK], μας επιτρέπουν να αποκτήσουμε ένα αμφίδρομο προ-εκπαιδευμένο μοντέλο, οι δημιουργοί του μοντέλου, επισημαίνουν πως το μειονέκτημα είναι ότι δημιουργείται αναντιστοιχία μεταξύ της προ-εκπαίδευσης και του fine-tuning, που είναι η χρήση του προ-εκπαιδευμένου μοντέλου για άλλα classification tasks, καθώς το token [MASK] δεν εμφανίζεται κατά τη διάρκεια του fine-tuning. Για να το μετριάσουμε αυτό, δεν αντικαθιστούμε πάντα τις «καλυμμένες» λέξεις με το token [MASK]. Η γεννήτρια δεδομένων εκπαίδευσης επιλέγει τυχαία το 15% των tokens για πρόβλεψη. Εάν επιλεγεί το i-th token, αντικαθιστούμε το i-th token με (1) το [MASK] token στο 80% των περιπτώσεων (2) ένα τυχαίο token στο 10% των περιπτώσεων και (3) το αμετάβλητο i-th token στο 10% των περιπτώσεων.

2. **Next Sentence Prediction (NSP)**
   Η επόμενη στρατηγική για την εκπαίδευση του μοντέλου είναι η πρόβλεψη επόμενης ακολουθίας. Ουσιαστικά, το μοντέλο τροφοδοτείται με ζευγάρια ακολουθιών και προσπαθεί να προβλέψει αν η δεύτερη ακολουθία βρίσκεται αμέσως μετά την

πρώτη στο αρχικό κείμενο. Κατά την διαδικασία εκπαίδευσης το 50% των ζευγαριών αποτελείται από διαδοχικές ακολουθίες στο κείμενο, ενώ το υπόλοιπο 50% αποτελείται από ζευγάρια τυχαία επιλεγμένων ακολουθιών.

Και σε αυτήν την περίπτωση, για να γίνει η πρόβλεψη προστίθεται ένα στρώμα ταξινόμησης στην έξοδο του κωδικοποιητή, προσπαθώντας να ταξινομήσει τις εξόδους στις κλάσεις IsNext, NotNext, επιστρέφοντας δηλαδή ένα διάνυσμα διάστασης 2 στο οποίο στην συνέχεια εφαρμόζεται πάλι τη συνάρτηση softmax για να δώσει πιθανότητες.

Fine-Tuning

Οι παραπάνω δύο στρατηγικές χρησιμοποιούνται στην διαδικασία της προ-εκπαίδευσης του μοντέλου BERT. Όταν όμως το μοντέλο αυτό χρειάζεται να χρησιμοποιηθεί σε κάποια συγκεκριμένη εφαρμογή, για παράδειγμα σε κάποιο συγκεκριμένο classification task όπως είναι η ανάλυση συναισθήματος ή σε ένα Question Answering task, τότε προστίθεται στο τέλος του μοντέλου ένας αποκωδικοποιητής ή ένας ταξινομητής (ρηχό πλήρως συνδεδεμένο δίκτυο). Η διαδικασία αυτή ονομάζεται **Fine-Tuning** και κατά την εφαρμογή της, μόνο τα επιπρόσθετα επίπεδα ανανεώνουν τις παραμέτρους τους και ίσως μερικά από τα τελευταία επίπεδα του δικτύου, ενώ οι περισσότερες από τις παραμέτρους του συστήματος δεν επηρεάζονται.

Αναπαράσταση Εισόδου

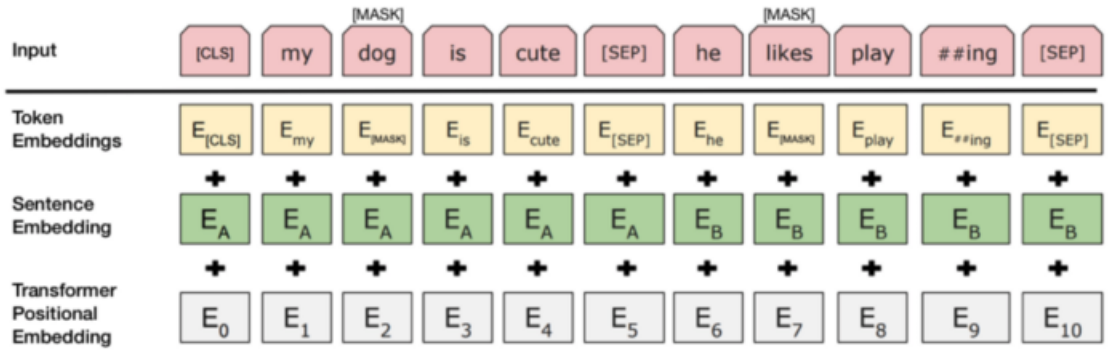
Το BERT αναμένει μια συγκεκριμένη αναπαράσταση της εισόδου. Τα βήματα που ακολουθούνται για την προ-επεξεργασία της εισόδου είναι τα εξής:

- Αρχικά, ένα δείγμα-πρόταση μετατρέπεται σε μία ακολουθία από μονάδες (tokenization). Το BERT διαθέτει δικό του tokenizer, ο οποίος, δεδομένης μιας πρότασης στην είσοδό του, θα κατασκευάσει στην έξοδο μια ακολουθία από τις λέξεις (tokens) που θα βρει στο λεξιλόγιο. Συγκεκριμένα, χρησιμοποιεί την τεχνική WordPiece, η οποία στην ουσία χωρίζει ένα token όπως το "playing" σε "play" και "ing". Αυτό γίνεται κυρίως για την κάλυψη ενός ευρύτερου φάσματος λέξεων εκτός λεξιλογίου (Out-Of-Vocabulary / OOV). Για τις λέξεις της εισόδου που δεν αναγνωρίζονται στο λεξιλόγιο ο tokenizer θα προσπαθήσει να τις διασπάσει στους όρους του λεξιλογίου με μέγιστο αριθμό χαρακτήρων. Στη χειρότερη περίπτωση θα διασπάσει μια λέξη στους μεμονωμένους χαρακτήρες που την απαρτίζουν. Σε περίπτωση διάσπασης μιας λέξης ο πρώτος όρος θα εμφανιστεί στην ακολουθία ως έχει, ενώ στους υπόλοιπους όρους θα προστεθεί στην αρχή τους το διπλό σύμβολο # ώστε να αναγνωρίσει το μοντέλο ότι πρόκειται για όρους στους οποίους έχει προηγηθεί διάσπαση.

- Στην συνέχεια, προστίθεται στην αρχή κάθε ακολουθίας δηλαδή κάθε δείγματος το token [CLS]. Είναι ένα token ταξινόμησης που χρησιμοποιείται συνήθως σε συνδυασμό με ένα επίπεδο softmax για εργασίες ταξινόμησης. Για οτιδήποτε άλλο, μπορεί να αγνοηθεί με ασφάλεια.

- Προστίθεται, επίσης, το token [SEP], το οποίο εισάγεται στο τέλος κάθε ακολουθίας ως ένα διαχωριστικό token. Χρησιμοποιείται κατά την προ-εκπαίδευση για εργασίες ζεύγους ακολουθιών, δηλαδή πρόβλεψης επόμενης πρότασης, όπως είναι η Next Sentence Prediction που είδαμε. Όταν χρησιμοποιείται μία μόνο ακολουθία, προσαρτάται στο τέλος.

Μετά την παραπάνω διαδικασία προ-επεξεργασίας, το BERT αναπαριστά κάθε ακολουθία από τρία διανύσματα όπως φαίνεται στο παρακάτω σχήμα [101]:



Σχήμα 4.49: Αναπαράσταση εισόδου BERT

Αρχικά, έχουμε την είσοδο (input) από τα tokens που δημιουργήσαμε με τον τρόπο που αναλύθηκε παραπάνω. Σημειώνεται ότι στα tokens αυτά μπορεί να συμπεριλαμβάνεται το token [MASK], αν μιλάμε για προ-εκπαίδευση με στρατηγική Masked LM. Ύστερα, δημιουργούνται τα τρία διανύσματα: Token embeddings, Sentence Embeddings, Transformer Positional Embedding. Τα token embeddings είναι τα IDs του κάθε token στο λεξιλόγιο. Τα sentence embeddings είναι μια δυαδική τιμή για τον διαχωρισμό των προτάσεων. Τέλος, τα tranformer positional embeddings υποδεικνύουν τη θέση κάθε λέξης στην ακολουθία. Έτσι, για ένα δεδομένο token, η αναπαράσταση εισόδου του κατασκευάζεται αθροίζοντας τις αντίστοιχες 3 ενσωματώσεις: token, sentence και positional.

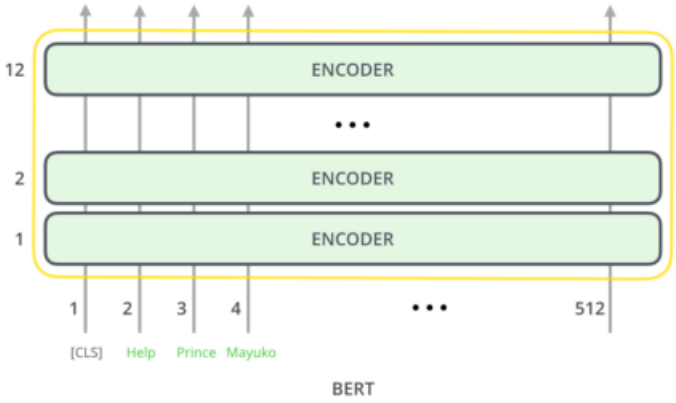
Εξαγωγή Bert Embeddings

Στην παρούσα διπλωματική εργασία, αυτό που μας ενδιαφέρει είναι να χρησιμοποιήσουμε ένα προ-εκπαιδευμένο μοντέλο BERT, με σκοπό να εξάγουμε αναπαραστάσεις λέξεων και κατ' επέκτασιν ολόκληρων δειγμάτων, δηλαδή ενσωματώσεις (embeddings), που στην συνέχεια θα χρησιμοποιηθούν ως χαρακτηριστικά από τους ταξινομητές τμημάτων, οι οποίοι καλούνται να ταξινομήσουν τα δείγματα κειμένου.

Η Google παρέχει δύο ειδών μοντέλα: το base και το large. Η διαφορά τους έγκειται στο μέγεθος. Το μεν base έχει τα εξής χαρακτηριστικά: L=12, H=768, A=12, Total Parameters=110M, δηλαδή 12 επίπεδα με μέγεθος κρυφού επιπέδου (hidden layer size) ίσο με 768 και 110 εκατομμύρια συνολικές παραμέτρους. Το δε large, έχει τα εξής χαρακτηριστικά: L=24, H=1024, A=16, Total Parameters=340M, δηλαδή 24 επίπεδα, κρυφό μέγεθος επιπέδου ίσο με 1024 και 340 εκατομμύρια συνολικές παραμέτρους. Επίσης, παρέχει τις εξής δύο εκδοχές: case και uncased. Στην περίπτωση uncased, οι λέξεις της εισόδου αναμένεται να είναι στα αγγλικά χωρίς κεφαλαίους χαρακτήρες, ενώ στην περίπτωση case, οι λέξεις μπορούν να περιέχουν και κεφαλαίους χαρακτήρες. Τα δεδομένα προεκπαίδευσης που χρησιμοποίησαν είναι από το BooksCorpus (800 εκατομμύρια λέξεις) (Zhu et al., 2015) και την αγγλική Wikipedia (2.500 εκατομμύρια λέξεις).

Στην εργασία αυτή χρησιμοποιήσαμε την έκδοση bert base cased. Η αρχιτεκτονική αυτού του μοντέλου εσωτερικά αποτελείται από 12 πανομοιότυπα επίπεδα μεγέθους 768, όπως αναφέρθηκε, όπου κάθε επίπεδο προσομοιώνει έναν κωδικοποιητή. Ο κάθε κωδικοποιητής αποτελείται από διάφορα τμήματα επεξεργασίας των διανυσματικών ενσωματώσεων όπως έχουμε δει και στην θεωρία των Transformers (2.6). Συγκεκριμένα, το πρώτο τμήμα είναι ένας μηχανισμός προσοχής multihead-attention με 12 σημεία εστίασης. Το επόμενο τμήμα είναι ένα επίπεδο κανονικοποίησης (normalization layer) ακολουθούμενο από ένα feed-forward δίκτυο, δομημένο από δύο πλήρως συνδεδεμένα επίπεδα νευρώνων με συνάρτηση ενεργοποίησης την συνάρτηση relu. Τελευταίο τμήμα του κωδικοποιητή είναι ο μηχανισμός κωδικοποίησης θέσης εισάγοντας διανύσματα ενσωμάτωσης θέσης παράλληλα με τα διανύσματα ενσωμάτωσης των λέξεων.

Η εσωτερική δομή του μοντέλου BERT φαίνεται στο παρακάτω σχήμα [101]:



Σχήμα 4.50: Εσωτερική δομή του BERT μοντέλου

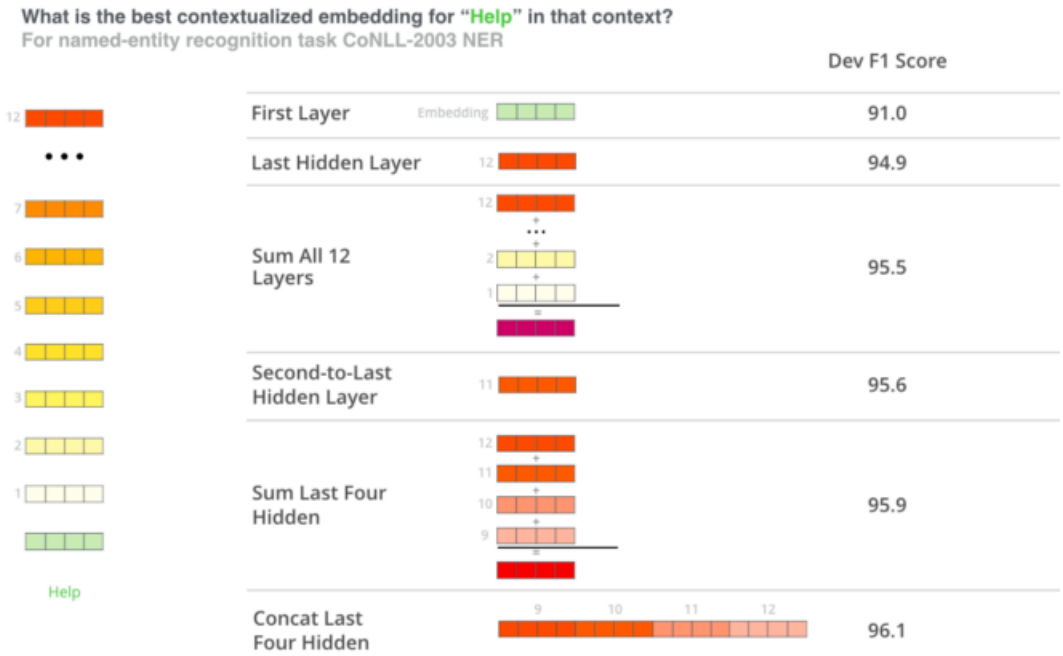
Επομένως, τα βήματα που ακολουθήθηκαν για την απόκτηση των διανυσματικών ενσωματώσεων των δειγμάτων μας είναι τα εξής:

- Πρώτα χωρίζεται το κάθε δείγμα στα tokens από τα οποία αποτελείται μέσω του ειδικού tokenizer του bert.
- Στην συνέχεια, προσθέτουμε στην αρχή κάθε ακολουθίας από tokens, το token [CLS] και στο τέλος το token [SEP], καθώς ακόμα και αν ένα δείγμα αποτελείται από περισσότερες από μία προτάσεις, εμείς το θεωρούμε ως ένα συνολικό δείγμα δηλαδή μια πρόταση, μια και το [SEP] διαχωρισμού των προτάσεων είναι χρήσιμο μόνο κατά την διάρκεια της προ-εκπαίδευσης.
- Το μοντέλο, όπως έχουμε ήδη δει, αναμένει τρία παράλληλα διανύσματα με σταθερό μήκος για την αναπαράσταση κάθε ακολουθίας στην είσοδο. Δεδομένου πως τα δείγματά μας έχουν διαφορετικό μήκος, μια και έχουν διαφορετικό αριθμό λέξεων το καθένα, επιλέγουμε ως σταθερό μήκος το μέγιστο μήκος από όλα τα δείγματα που ορίζεται από το μέγιστο αριθμό λέξεων μέσα στο δείγμα. Έτσι, τα δείγματα που αποτελούνται από αριθμό tokens, μικρότερο του μέγιστου μήκους, τα συμπληρώνουμε με το token [PAD] και για όσα αποτελούνται από αριθμό tokens μεγαλύτερο του μέγιστου μήκους, αποκόπτουμε τους περίσσιους όρους.
- Τα διανύσματα που τελικά κατασκευάζουμε είναι το διάνυσμα tokens_ids από τα αναγνωριστικά IDs για κάθε όρο της ακολουθίας, όπως παρέχεται από το λεξιλόγιο του μοντέλου και το διάνυσμα attention_mask το οποίο αποτελείται από άσσους στις n πρώτες θέσεις, όπου n είναι το μήκος της ακολουθίας εισόδου, και από μηδενικά στις υπόλοιπες max_lengthn θέσεις. Μια και δεν χρησιμοποιούμε περισσότερες των μία προτάσεων μέσα στο δείγμα, το διάνυσμα sentence_ids παραλείπεται και δεν θα το χρησιμοποιήσουμε.

Τελικά περνάμε τα δύο παραπάνω διανύσματα στο προ-εκπαιδευμένο μοντέλο ανά batch και παίρνουμε τα embeddings του κάθε token, αθροίζοντας τα διανύσματα εξόδου από τα τέσσερα τελευταία επίπεδα. Αυτό που εμείς ζητάμε είναι να βρούμε την αναπαράσταση ολόκληρου του δείγματος. Αυτήν την αναπαράσταση την υπολογίζουμε από τον μέσο όρο των embeddings των tokens από τα οποία αποτελείται το δείγμα.

Ο λόγος για τον οποίο επιλέγουμε να αθροίσουμε τις ενσωματώσεις από τα 4 τελευταία επίπεδα είναι επειδή εμπειρικά οι εμπνευστές του BERT προτείνουν πως αυτή είναι μια καλή τακτική για υψηλή απόδοση.

Στην παρακάτω εικόνα, φαίνονται ενδεικτικά διάφοροι συνδυασμοί αυτών των διανυσμάτων, από διαφορετικά επίπεδα και τα f1 scores τους, όπως αυτοί εξετάστηκαν στην δημοσίευση [100]. Φυσικά, η επιλογή του κατάλληλου συνδυασμού εξαρτάται κάθε φορά από το πρόβλημα [103]:



Σχήμα 4.51: Συνδυασμοί διανυσμάτων από διάφορα επίπεδα του bert

Συμπερασματικά, μετά την τμηματοποίηση της ομιλίας, θα καταλήξουμε τελικά με ένα διάνυσμα διάστασης 768 που αποτελεί την αναπαράσταση του τμήματος-πρότασης και το οποίο θα χρησιμοποιηθεί ως χαρακτηριστικά για τους ταξινομητές τμημάτων κειμένου που θα δούμε στο επόμενο κεφάλαιο.

Τα διανύσματα που παράγονται με την μεθοδολογία BERT υπερτερούν έναντι των προηγούμενων μεθόδων επειδή η αναπαράσταση κάθε λέξης δεν γίνεται στατικά, όπως για παράδειγμα με τη μέθοδο Word2Vec και Fasttext, όπου η αναπαράσταση είναι ανεξάρτητη από τα συμφραζόμενα, αλλά κάθε αναπαράσταση ενημερώνεται δυναμικά από τις περιβάλλουσες λέξεις.

4.2.3 Ταξινομητές Τμημάτων - Αναγνώριση Συναισθήματος

Οι ταξινομητές τμημάτων κειμένου είναι οι ίδιοι με τους αντίστοιχους ταξινομητές τμημάτων ήχου που είδαμε στο κεφάλαιο 4.1.4. Όπως φαίνεται και στο σχήμα 4.15, χρησιμοποιούνται τρεις ταξινομητές τμημάτων. Ένας ταξινομητής ο οποίος ταξινομεί τα κείμενα σε τέσσερα **συναισθήματα (emotion)** ή αλλιώς κατηγορηματικά χαρακτηριστικά: θυμός, λύπη, ουδετερότητα, χαρά.Ένας ταξινομητής ο οποίος ταξινομεί τα κείμενα ανάλογα με το **σθένος (valence)**: θετικό, μέτριο, αρνητικό και ένας ο οποίος τα ταξινομεί με βάση την **διέγερση (arousal)**: υψηλή, μέτρια, χαμηλή. Το σθένος και η διέγερση,όπως έχουμε δει, αποτελούν ένα δισδιάστατο χώρο χαρακτηριστικών συναισθήματος.

4.2.4 Σύνολα Δεδομένων

Για την εκπαίδευση των τριών αυτών μοντέλων ταξινομητών χρησιμοποιήθηκε μόνο το σύνολο δεδομένων iemocap που παρουσιάστηκε στο κεφάλαιο 4.1.5, καθώς είναι το

μοναδικό που περιέχει πληροφορίες κειμένου (transcriptions).

Για τα συναισθήματα, διατηρήθηκαν και πάλι μόνο τα δείγματα τα οποία ανήκουν στα 4 βασικά συναισθήματα που εξετάζονται στην παρούσα εργασία (λύπη, θυμός, ουδετερότητα, χαρά) και τα δείγματα που ανήκουν στην κλάση "ενθουσιασμός", θεωρήθηκαν δείγματα "χαράς" (πίνακας 4.1).

Για το σθένος (valence) και την διέγερση (arousal), καθώς όπως έχουμε δει η βάση iemocap περιέχει συνεχείς τιμές, δημιουργήθηκαν 6 κλάσεις συναισθημάτων (3 για σθένος και 3 για διέγερση), με τον ίδιο τρόπο όπως και στην περίπτωση του ήχου. Οι κλάσεις χωρίστηκαν δηλαδή σε negative,neutral,positive για το σθένος και σε low, neutral, high για την διέγερση με βάση τα κατώφλια τιμών που φαίνονται στον πίνακα 4.2.

Τέλος, μετά την νέα διαμόρφωση των δεδομένων, κάθε δείγμα κειμένου ακολουθεί την διαδικασία που περιγράφηκε στα προηγούμενα υποκεφάλαια. Το πρώτο βήμα είναι η διάσπαση σε τμήματα. Συγκεκριμένα για τα πειράματά μας χρησιμοποιήθηκε η διάσπαση ανά πρόταση, επειδή μια πρόταση θεωρείται ένα ικανά μεγάλο πλαίσιο μέσα στο οποίο προλαβαίνει να αναπτυχθεί ένα συναίσθημα, αλλά και ικανά μικρό για να μην εμπλέκει περισσότερα του ενός συναισθήματα/χαρακτηριστικά. Το επόμενο βήμα είναι η εξαγωγή χαρακτηριστικών για κάθε τμήμα και το τελευταίο βήμα είναι η τροφοδότησή τους στους ταξινομητές τμημάτων ώστε κάθε τμήμα να χαρακτηριστεί με μία ετικέτα συναισθήματος, σθένους και διέγερσης. Σημειώνεται πως για την διαδικασία της εκπαίδευσης, τα δεδομένα κειμένου θεωρούνται προ-τμηματοποιημένα (presegmented), καθώς στο σύνολο δεδομένων iemocap το κάθε δείγμα αποτελείται από μία και μόνο ετικέτα, ακόμα και αν εμπεριέχει περισσότερες της μίας πρότασης.

4.2.5 Πειράματα και Αποτελέσματα

Και στα τρία classification tasks εκτελέστηκαν πειράματα τόσο με fasttext embeddings, όσο και με bert embeddings. Κατά την εκπαίδευση και την αξιολόγηση, εφαρμόστηκε το ίδιο pipeline που είδαμε στο κεφάλαιο 4.1.6. Συγκεκριμένα, χρησιμοποιήθηκε SMOTE-Tomek για την διαχείριση της πιθανής ανισορροπίας, StandardScaler, VarianceThreshold και PCA. Επιπλέον, εφαρμόστηκε gridsearch με Repeated-StratifiedKFold cross validation το οποίο χρησιμοποεί 5 folds και 3 επαναλήψεις. Σε αυτό το gridsearch γίνεται hyper-parameter tuning στον αριθμό των components του pca και για τα μοντέλα SVM, στις παραμέτρους C και gamma. Τα αποτελέσματα των πειραμάτων κατά την διάρκεια του validation φαίνονται στον παρακάτω πίνακα. Σημειώνεται πως σε αυτήν την περίπτωση δεν υπάρχει ανάγκη αξιολόγησης σε ξεχωριστό test set, καθώς πρόκειται για κείμενα τα οποία δεν είναι εξαρτώμενα από κάποια άλλη παράμετρο όπως τον ομιλητή. Αντιθέτως, οι αξιολογήσεις που προέκυψαν από το cross validation είναι πολύ πιο ρεαλιστικές καθώς έχουν βασιστεί σε πολλές διαφορετικές διασπάσεις και όχι σε ένα μοναδικό test set.

	Iemocap			
Classification Task	Train/Val SVM with Full FastText Embeddings	Train/Val SVM with 500k FastText Embeddings	Train/Val Xgboost with Bert Embeddings	Train/Val SVM with Bert Embeddings
Emotion	66.5 (+/-1)	65.2 (+/-1.3)	64 (+/-1.7)	**69.5 (+/-1.4)**
Valence	61.5 (+/-1)	60.4 (+/-0.9)	59.4 (+/-0.9)	**63.8 (+/-1)**
Arousal	48.8 (+/-1)	48.1 (+/-0.9)	48.2 (+/-1)	**51 (+/-1.1)**

Table 4.10: Iemocap Evaluation

Σημειώνεται πως οι τιμες + που παρουσιάζονται εντός των παρενθέσεων είναι η τυπική απόκλιση των αποδόσεων στα διαφορετικά test folds.

Από τον παραπάνω πίνακα μπορούμε αρχικά να παρατηρήσουμε πως όταν χρησιμοποιούμε ένα κομμάτι των fasttext embeddings και όχι ολόκληρο το προ-εκπαιδευμένο μοντέλο, τότε χάνουμε περίπου 1% σε απόδοση σε σχέση με την χρήση ολόκληρου του

fasttext. Αυτό ήταν κάτι αναμενόμενο, καθώς περικόπτοντας το fasttext έχουμε λιγότερες λέξεις μέσα στο λεξικό, ενώ ταυτόχρονα χάνουμε και το πλεονέκτημα να βρίσκουμε αναπαραστάσεις λέξεων μέσω των ngrams τους. Ως εκ τούτου, το πλήθος των λέξεων εκτός λεξιλογίου (OOV) αυξάνεται.

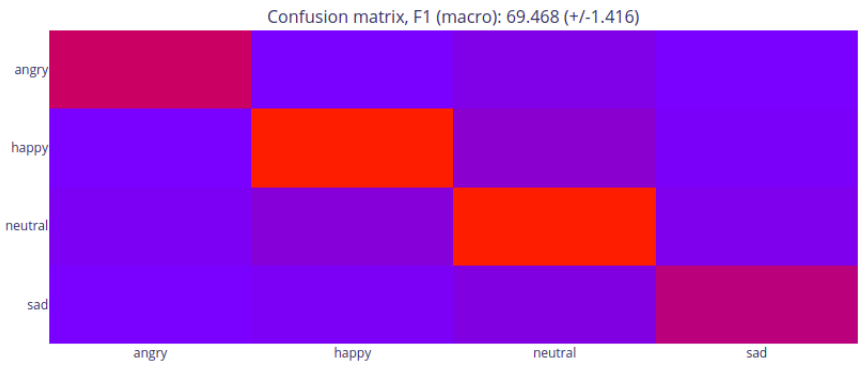
Σε αντιπαράθεση με το fasttext, δοκιμάζουμε και τα bert embeddings. Αυτό μας δίνει το πλεονέκτημα να μην έχουμε καμία λέξη εκτός λεξιλογίου αλλά και ταυτόχρονα να εκμεταλλευτούμε όλο το περιβάλλον μίας λέξης για να παράξουμε την αναπαράστασή της, κάτι που δεν συμβαίνει με το fasttext. Έτσι εξηγείται και η αύξηση της απόδοσης που παρατηρούμε στην τελευταία στήλη που αφορά την εκπαίδευση ενός svm με χρήση bert embeddings. Η αύξηση αυτή ανέρχεται στο 3-4% για το emotion και το valence και στο 2% για το arousal. Τέλος, κάνουμε και μία σύγκριση του μοντέλο SVM με έναν xgboost (πάλι με bert embeddings) και παρατηρούμε ότι ο SVM υπερτερεί. Επομένως, αυτός θα είναι και το τελικό μας μοντέλο που θα χρησιμοποιηθεί από το σύστημα και για τα τρία classification tasks.

Οι καλύτερες τιμές υπερπαραμέτρων που έδωσαν το καλύτερο μοντέλο φαίνονται παρακάτω:

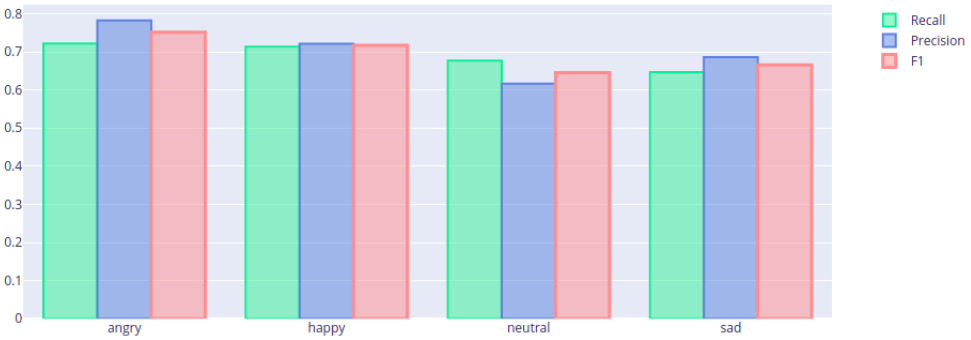
- **Emotion**:
  Parameters of best svm model: 'SVM C': 5.0, 'SVM gamma': 'auto', 'pca n_components': 0.98
- **Valence**:
  Parameters of best svm model: 'SVM C': 1.0, 'SVM gamma': 'auto', 'pca n_components': 'mle'
- **Arousal**:
  Parameters of best svm model: 'SVM C': 0.5, 'SVM gamma': 'auto', 'pca n_components': None

Παρακάτω μπορούμε να δούμε πιο αναλυτικά τα confusion matrices για τα μοντέλα της τελευταίας στήλης, καθώς επίσης και τις μετρικές απόδοσης ανά κλάση. Σημειώνεται πως το confusion matrix εδώ υπολογίζεται προσθέτοντας όλα τα ταξινομημένα δείγματα από όλα τα test folds και από όλες τις επαναλήψεις. Επομένως, έχουν προστεθεί τα δείγματα από τις 5*3 = 15 διαφορετικές ταξινομήσεις που έγιναν κατά την διάρκεια του repeated stratified cross validation. Αυτό πρακτικά σημαίνει πως το άθροισμα των δειγμάτων όλου του πίνακα δεν αντιστοιχεί στον πραγματικό συνολικό αριθμό δειγμάτων, δηλαδή οι απόλυτες τιμές δεν είναι αντιπροσωπευτικές, όμως οι normalized τιμές μπορούν να δείξουν σωστά τις αναλογίες.

Emotion

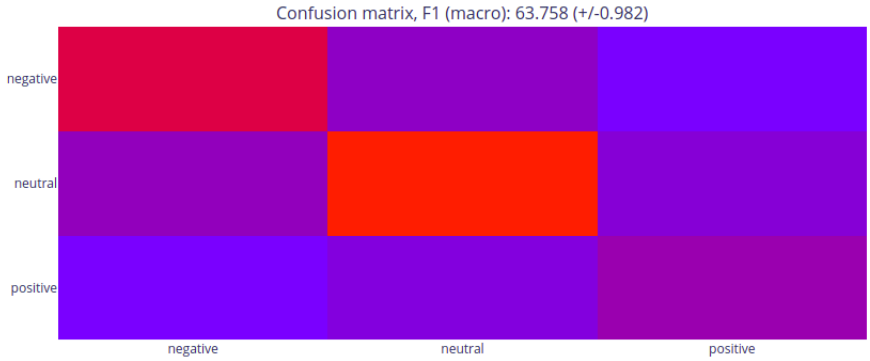


Σχήμα 4.52: Confusion Matrix Συναισθήματος

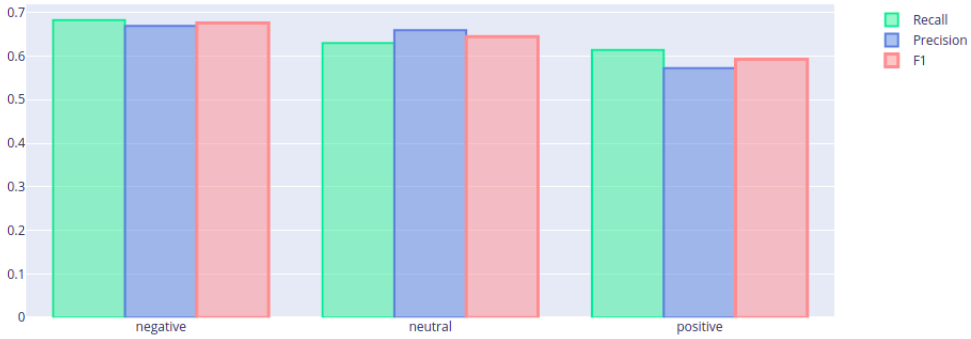


Σχήμα 4.53: Μετρικές Απόδοσης ανά Κλάση Συναισθήματος

Valence

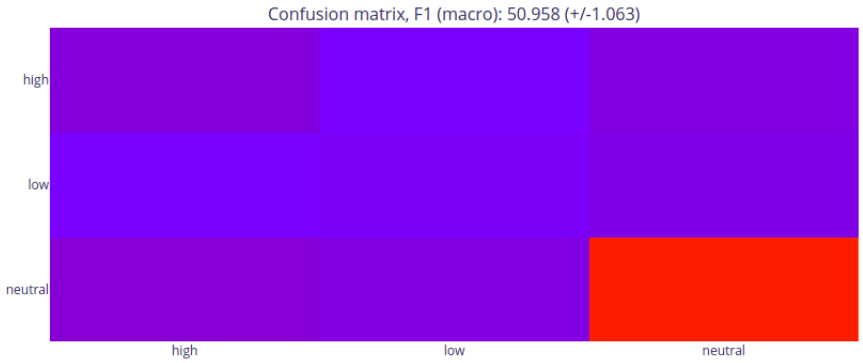


Σχήμα 4.54: Confusion Matrix Σθένους

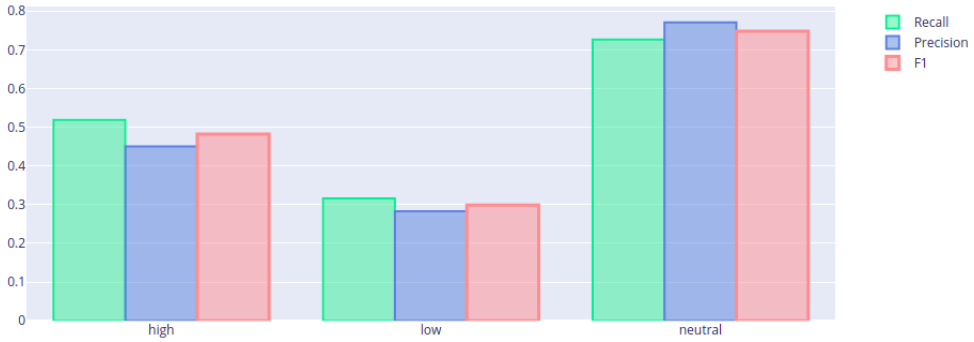


Σχήμα 4.55: Μετρικές Απόδοσης ανά Κλάση Σθένους

Arousal



Σχήμα 4.56: Confusion Matrix Διέγερσης



Σχήμα 4.57: Μετρικές Απόδοσης ανά Κλάση Διέγερσης

Από τις παραπάνω εικόνες παρατηρούμε πως τόσο στο emotion όσο και στο valence, οι αποδόσεις ανά κλάση είναι παρόμοιες. Απο το emotion task, η κλάση neutral είναι αυτή που παρουσιάζει το μικρότερο precision, ενώ η κλάση sad το μικρότερο recall. Όλες οι κλάσεις όμως παρουσιάζουν f1 score μεταξύ του εύρος 65% και 75%. Από την άλλη, στο valence task φαίνεται η κλάση positive να είναι αυτή που τα πηγαίνει χειρότερα σε σχέση με τις άλλες δύο πράγμα που γίνεται αντιληπτό τόσο από το confusion matrix, όσο και από το χαμηλότερο precision και recall που παρουσιάζει στην γραφική παράσταση των μετρικών απόδοσης ανά κλάση. Και σε αυτήν την περίπτωση όμως, οι διαφορές μεταξύ των κλάσεων δεν είναι μεγάλες και το f1 score κυμαίνεται στο εύρος 59% με 68%.

Σε ο,τι αφορά το arousal task, εμφανίζεται μια χειρότερη εικόνα, καθώς φαίνεται η κλάση neutral να υπερτερεί σε σχέση με τις άλλες δύο (73% recall και 77% precision). Εν αντιθέσει, η κλάση high έχει περίπου 52% recall και 45% precision, ενώ η κλάση low τα πηγαίνει ακόμα χειρότερα αφού παρουσιάζει 32% recall και 28% precision. Επομένως, το μοντέλο τα πηγαίνει πολύ καλά στο να διαχωρίζει/ταξινομεί την κλάση neutral και μέτρια στην κλάση high, ενώ την κλάση low δεν μπορεί να την διαχωρίσει καθόλου, αφού η απόδοσή του στην κλάση αυτή είναι ισοδύναμη με έναν random classifier, αν όχι και χειρότερη.

5 Ανάλυση σε Επίπεδο Εγγραφής

Έχοντας αναλύσει τα κομμάτια του συστήματος που αφορούν την τμηματική ανάλυση (segment-level analysis), δηλαδή αυτά που χρησιμοποιούν ταξινομητές τμημάτων, τόσο για τον ήχο όσο και για το κείμενο, το επόμενο βήμα είναι να περάσουμε στην ανάλυση σε επίπεδο εγγραφής (recording-level analysis). Ονομάζεται έτσι γιατί αφορά πλέον την συνολική εγγραφή, δηλαδή την συνολική ομιλία και όχι μόνο κάποια κομμάτια της. Ο στόχος μας από την αρχή ήταν να αξιολογήσουμε την ποιότητα της ομιλίας, επομένως, θα χρησιμοποιήσουμε την πληροφορία που εξαγάγαμε από τα τμήματα, στα προηγούμενα κεφάλαια, για να πετύχουμε αυτόν τον σκοπό, προσθέτοντας όμως και κάποια επιπλέον πληροφορία/χαρακτηριστικά.

Ουσιαστικά, στο κεφάλαιο αυτό θα αναλυθεί το υπόλοιπο κομμάτι της αρχιτεκτονικής και πάλι ξεχωριστά για τον ήχο και ξεχωριστά για το κείμενο (τα πράσινα στοιχεία στα σχήματα 3.1 και 3.2).

5.1 Ανάλυση Ήχου

Στα παρακάτω δύο υποκεφάλαια θα παρουσιαστούν τα χαρακτηριστικά που χρησιμοποιούνται για την συνολική εγγραφή/ομιλία.

5.1.1 Aggregation of Class Posteriors

Το ηχητικό σήμα της ομιλίας, έχει περάσει πρώτα από την ανάλυση σε επίπεδο τμήματος, δηλαδή έχει διασπαστεί σε τμήματα και για καθένα από τα τμήματα αυτά έχουν παραχθεί τα class posteriors από τα μοντέλα που επιλέξαμε στο κεφάλαιο 4.1.6. Δηλαδή, έχουμε τρεις ετικέτες για κάθε τμήμα που αφορούν το συναίσθημα (emotion), το σθένος (valence) και την διέγερση (arousal) αντίστοιχα. Εμάς όμως μας ενδιαφέρει να χαρακτηρίσουμε το συνολικό σήμα, δηλαδή την συνολική ομιλία. Επομένως, κάνουμε μία **συνάθροιση (aggregation)** των class posteriors. Η συνάθροιση αυτή υπολογίζεται παίρνοντας τον μέσο όρο για κάθε ετικέτα. Δηλαδή, υπολογίζεται ο αριθμός των τμημάτων που ταξινομήθηκαν στην ετικέτα αυτή και διαιρείται με τον συνολικό αριθμό των τμημάτων. Έτσι, καταλήγουμε με ένα ποσοστό ανά ετικέτα, που στην ουσία είναι η πιθανότητα η ομιλία αυτή να ανήκει στην εκάστοτε ετικέτα.

Για παράδειγμα, ένα ηχητικό σήμα μπορεί να χαρακτηριστεί με τα εξής ποσοστά/πιθανότητες:

- Ως προς το συναίσθημα:
  p(emotion=sad)=30%, p(emotion=neutral)=40%, p(emotion=happy)=10%, p(emotion=angry)=20%
- Ως προς το σθένος:
  p(valence=positive)=60%, p(valence=neutral)=40%, p(valence=negative)=0%
- Ως προς την διέγερση:
  p(arousal=high)=10%, p(arousal=neutral)=90%, p(arousal=low)=0%

Οι παραπάνω πιθανότητες χρησιμοποιούνται ως χαρακτηριστικά της συνολικής ομιλίας, τα οποία θα βοηθήσουν στην συνέχεια τους ταξινομητές να αξιολογήσουν την ποιότητα της ομιλίας. Επομένως, καταλήγουμε με 10 χαρακτηριστικά (όσες και οι συνολικές ετικέτες), που έχουν προκύψει από τους ταξινομητές τμημάτων, για το συνολικό αρχείο ήχου.

Πέρα από αυτά τα χαρακτηριστικά, όμως, προστίθεται και κάποια ακόμα χαρακτηριστικά συνολικής ομιλίας που αναλύονται στο επόμενο υποκεφάλαιο.

5.1.2 High Level Features

Τα high level χαρακτηριστικά, όπως τα ονομάζουμε, προκύπτουν έπειτα από τον εντοπισμό των τμημάτων ομιλίας μέσα στο σήμα ήχου. Το σήμα αυτό, περιέχει τόσο τμήματα ησυχίας (silence), όσο και τμήματα ομιλίας. Ο πιο απλός τρόπος διαχωρισμού της ομιλίας από την ησυχία βασίζεται στην ενέργεια του σήματος, καθώς είναι γνωστό πως η ησυχία έχει χαμηλή ενέργεια ενώ η φωνή/ομιλία έχει υψηλή.

Ανίχνευση Δραστηριότητας Φωνής

Ακολουθούμε μια διαδικασία εντοπισμού των τμημάτων ομιλίας με την βοήθεια της βιβλιοθήκης pyAudioAnalysis [56]. Τα βήματα για τον εντοπισμό των τμημάτων αυτών έχουν ως εξής:

- Χωρίζουμε το σήμα ήχου σε βραχυπρόθεσμα παράθυρα όπως ακριβώς έχουμε δει στο κεφάλαιο 4.1.1.
- Για το καθένα από τα παράθυρα αυτά εξάγουμε τα 68 ηχητικά χαρακτηριστικά που έχουμε ξαναδεί (4.1.3).
- Από τα χαρακτηριστικά αυτά επιλέγουμε την ενέργεια, για όλα τα δείγματα/παράθυρα. Κάνουμε μια ταξινόμηση του χαρακτηριστικού αυτού κατά αύξουσα τιμή. Στην συνέχεια, υπολογίζουμε τον αριθμό του 10% των συνολικών παραθύρων. Παίρνουμε αυτόν τον αριθμό παραθύρων χαμηλότερης ενέργειας και υπολογίζουμε τον μέσο όρο της ενέργειας στα παράθυρα αυτά. Κάνουμε το αντίστοιχο και για αυτόν τον αριθμό παραθύρων υψηλότερης ενέργειας. Έτσι, καταλήγουμε με δυο τιμές ενέργειας, μια που έχει προκύψει από τα χαμηλότερης ενέργειας παράθυρα και μια από τα υψηλότερης ενέργειας παράθυρα, οι οποίες αποτελούν το κατώφλι χαμηλής και υψηλής ενέργειας αντίστοιχα.
- Στην συνέχεια, παίρνουμε όλα τα χαρακτηριστικά (όχι μόνο την ενέργεια), για τα δείγματα/παράθυρα που έχουν τιμή ενέργειας κάτω από το κατώφλι χαμηλής ενέργειας και αντίστοιχα για τα δείγματα/παράθυρα που έχουν τιμή ενέργειας πάνω από το κατώφλι υψηλής ενέργειας.
- Από το προηγούμενο βήμα, καταλήγουμε με δυο ειδών δείγματα: τα δείγματα χαμηλής ενέργειας και τα δείγματα υψηλής ενέργειας, ή αλλιώς τα δείγματα ησυχίας/σιωπής και τα δείγματα ομιλίας/φωνής αντίστοιχα. Επομένως, έχουμε δείγματα δύο κλάσεων με τα αντίστοιχα χαρακτηριστικά τους τα οποία μπορούμε να τροφοδοτήσουμε σε ένα μοντέλο για να μάθει να τα διαχωρίζει.
- Κάνοντας μια κανονικοποίηση των παραπάνω χαρακτηριστικών, εκπαιδεύουμε έναν γραμμικό svm ταξινομητή με τα δείγματα των δύο κλάσεων του προηγούμενου βήματος.
- Μετά την εκπαίδευση του ταξινομητή, τον χρησιμοποιούμε για να προβλέψουμε τις πιθανότητες για κάθε δείγμα/παράθυρο. Αυτό σημαίνει πως θα παράξουμε μια πιθανότητα το κάθε δείγμα να ανήκει στην θετική κλάση, που στην περίπτωσή μας είναι η κλάση "high energy" ή αλλιώς η ομιλία.
- Ταξινομούμε τις πιθανότητες αυτές με αύξουσα σειρά και στην συνέχεια υπολογίζουμε τον μέσο όρο του 10% του συνολικού αριθμού των πιθανοτήτων, που έχουν τις μικρότερες τιμές και τον μέσο όρο του 10% του συνολικού αριθμού των πιθανοτήτων που έχουν τις μεγαλύτερες τιμές. Προσθέτουμε τους δύο αυτούς μέσους όρους με βάρος 0.5 στον καθένα. Αυτό σημαίνει πως οι δυο μέσοι όροι έχουν ισότιμη επιρροή

στο άθροισμα. Το άθροισμα αυτό, αποτελεί τελικά το κατώφλι πιθανότητας, πάνω από το οποίο τα δείγματα θεωρούνται ομιλία.

- Τελικά από όλες τις πιθανότητες, όλων των παραθύρων του ήχου, επιλέγουμε αυτές με τιμή μεγαλύτερη από το κατώφλι του προηγούμενου βήματος. Αντίστοιχα επιλέγουμε τα παράθυρα που αντιστοιχούν σε αυτές τις πιθανότητες.

- Από τα παράθυρα αυτά, "πετάμε" εκείνα που είναι πολύ μικρά και συγκεκριμένα που έχουν διάρκεια μικρότερη ή ίση των 0.2 δευτερολέπτων. Έτσι τελικά, καταλήγουμε με μια λίστα από βραχυπρόθεσμα παράθυρα του αρχικού σήματος, που έχουν αναγνωριστεί ως παράθυρα ομιλίας. Δηλαδή, έχουν αφαιρεθεί από το σήμα τα παράθυρα ησυχίας/σιωπής.

Εάν το αρχικό σήμα είναι μικρότερο των 6 δευτερολέπτων, τότε δεν εφαρμόζουμε την παραπάνω διαδικασία, αλλά θεωρούμε ολόκληρο το σήμα ως "ομιλία".

Την διαδικασία που περιγράφηκε παραπάνω, την εφαρμόζουμε για 2 διαφορετικές τιμές βραχυπρόθεσμων παραθύρων: παράθυρα μήκους 0.5 και βήματος 0.25, παράθυρα μήκους 1.0 και βήματος 0.25. Με αυτόν τον τρόπο, χωρίζουμε τον ήχο τόσο σε μικρά παράθυρα όσο και σε μεγαλύτερα καθώς οι δυο αυτές τεχνικές μπορούν να μας δώσουν διαφορετικές πληροφορίες. Για παράδειγμα, ένα μεγαλύτερο παράθυρο που μπορεί να ταξινομηθεί ως "ομιλία", υπάρχει πιθανότητα να εμπεριέχει μικρότερες παύσεις σιωπής, οι οποίες αντίθετα θα εντοπιστούν όταν ο ήχος χωρίζεται σε μικρότερα παράθυρα.

Από τα παραπάνω, καταλήγουμε με δύο λίστες από τμήματα ομιλίας: μία λίστα μικρότερης διάρκειας τμημάτων (short segments) και μια λίστα μεγαλύτερης διάρκειας τμημάτων (long segments). Ο σκοπός μας, όμως, είναι να εξάγουμε κάποια "υψηλού επιπέδου (high level)" χαρακτηριστικά για το συνολικό σήμα ήχου.

Χαρακτηριστικά από την Ανίχνευση Φωνής Θα χρησιμοποιήσουμε την πληροφορία των λιστών που εξαγάγαμε παραπάνω, για να δημιουργήσουμε τα παρακάτω χαρακτηριστικά:

1. Μέσος όρος διάρκειας σιωπής (average silence duration), η οποία υπολογίζεται από τον μέσο όρο διάρκειας των διαστημάτων σιωπής δηλαδή των διαστημάτων που παρεμβάλλοτναι ανάμεσα στα διαστήματα ομιλίας.

2. Αριθμός παύσεων/σιωπών ανά λεπτό (silence segment per minute), ο οποίος υπολογίζεται διαιρώντας τον συνολικό αριθμό παύσεων με το λεπτό.

3. Η τυπική απόκλιση της διάρκειας σιωπής (std of silence duration).

4. Η αναλογία ομιλίας (speech ratio), η οποία υπολογίζεται από την συνολική διάρκεια ομιλίας διαιρούμενη με την συνολική διάρκεια όλου του σήματος ήχου.

5. Ο ρυθμός λέξεων μέσα στην ομιλία (word rate in speech), ο οποίος υπολογίζεται από τον αριθμό των τμημάτων ομιλίας διαιρούμενο με την συνολική διάρκεια ομιλίας. Εδώ θεωρούμε ότι τα τμήματα ομιλίας είναι τόσο μικρά που προσομοιάζει μία λέξη το καθένα.

Τα παραπάνω χαρακτηριστικά όπως είπαμε, υπολογίζονται δύο φορές: μία για τα τμήματα ομιλίας που έχουν προκύψει από μικρότερης διάρκειας βραχυπρόθεσμα παράθυρα και μία για τα τμήματα ομιλίας που έχουν προκύψει από μεγαλύτερης διάρκειας βραχυπρόθεσμα παράθυρα. Επομένως, καταλήγουμε με 10 υψηλού επιπέδου χαρακτηριστικά για κάθε αρχείο ήχου.

5.2 Ανάλυση Κειμένου

Και πάλι στα παρακάτω δύο υποκεφάλαια θα παρουσιαστούν τα χαρακτηριστικά που χρησιμοποιούνται για την συνολική εγγραφή/ομιλία, αυτή την φορά από πλευράς κειμένου.

5.2.1 Aggregation of Class Posteriors

Αφού έχουμε μετατρέψει την ομιλία σε κείμενο μέσω speech to text, όπως έχουμε ξαναπεί, το κείμενο αυτό της ομιλίας, έχει περάσει πρώτα από την ανάλυση σε επίπεδο τμήματος, δηλαδή έχει διασπαστεί σε τμήματα (στην περίπτωσή μας προτάσεις) και για καθένα από τα τμήματα αυτά έχουν παραχθεί τα class posteriors από τα μοντέλα που επιλέξαμε στο κεφάλαιο 4.2.5. Δηλαδή, έχουμε τρεις ετικέτες για κάθε τμήμα που αφορούν το συναίσθημα (emotion), το σθένος (valence) και την διέγερση (arousal) αντίστοιχα. Εμάς όμως μας ενδιαφέρει να χαρακτηρίσουμε το συνολικό κείμενο της ομιλίας. Επομένως, κάνουμε μία **συνάθροιση (aggregation)** των class posteriors. Η συνάθροιση αυτή υπολογίζεται παίρνοντας τον μέσο όρο για κάθε ετικέτα. Δηλαδή, υπολογίζεται ο αριθμός των τμημάτων που ταξινομήθηκαν στην ετικέτα αυτή και διαιρείται με τον συνολικό αριθμό των τμημάτων. Έτσι, καταλήγουμε με ένα ποσοστό ανά ετικέτα, που στην ουσία είναι η πιθανότητα το κείμενο να ανήκει στην εκάστοτε ετικέτα.

Για παράδειγμα, ένα κείμενο ομιλίας μπορεί να χαρακτηριστεί με τα εξής ποσοστά/πιθανότητες:

- Ως προς το συναίσθημα:
  p(emotion=sad)=30%, p(emotion=neutral)=40%, p(emotion=happy)=10%, p(emotion=angry)=20%
- Ως προς το σθένος:
  p(valence=positive)=60%, p(valence=neutral)=40%, p(valence=negative)=0%
- Ως προς την διέγερση:
  p(arousal=high)=10%, p(arousal=neutral)=90%, p(arousal=low)=0%

Οι παραπάνω πιθανότητες χρησιμοποιούνται ως χαρακτηριστικά του συνολικού κειμένου ομιλίας, τα οποία θα βοηθήσουν στην συνέχεια τους ταξινομητές να αξιολογήσουν την ποιότητα της ομιλίας. Επομένως, καταλήγουμε με 10 χαρακτηριστικά (όσες και οι συνολικές ετικέτες), που έχουν προκύψει από τους ταξινομητές τμημάτων, για το συνολικό κείμενο.

Πέρα από αυτά τα χαρακτηριστικά, όμως, προστίθεται και κάποια ακόμα χαρακτηριστικά συνολικού κειμένου που αναλύονται στο επόμενο υποκεφάλαιο.

5.2.2 High Level Features

Τα υψηλού επιπέδου χαρακτηριστικά, όπως τα ονομάζουμε, που αφορούν το συνολικό κείμενο, είναι τα παρακάτω:

1. Ο ρυθμός λέξεων (word rate), ο οποίος υπολογίζεται διαιρώντας των αριθμό λέξεων ολόκληρου του κειμένου με την διάρκεια σε λεπτά (words/minute).
2. Ο ρυθμός μοναδικών λέξεων (unique word rate). Ο αριθμός μοναδικών λέξεων υπολογίζεται παίρνοντας την εμφάνιση κάθε λέξης μία φορά, δηλαδή δεν συνυπολογίζονται διπλές ή παραπάνω εμφανίσεις της ίδιας λέξης. Διαιρούμε αυτόν τον αριθμό με την διάρκεια και έχουμε τον ρυθμό μοναδικών λέξεων.
3. Ιστόγραμμα 10 σημείων των συχνοτήτων των λέξεων (10-bin histogram of word frequencies). Για τον υπολογισμό αυτού του χαρακτηριστικού, μετράμε για κάθε λέξη του κειμένου, πόσες εμφανίσεις έχει, δηλαδή την συχνότητά της μέσα στο κείμενο. Στην συνέχεια, κάνουμε μια κανονικοποίηση των συχνοτήτων αυτών, διαιρώντας με

τον συνολικό αριθμό λέξεων στο κείμενο. Τέλος, δημιουργούμε ένα ιστόγραμμα 10 σημείων και εύρους από 0 έως 0.1 για αυτές τις κανονικοποιημένες συχνότητες λέξεων. Η κάθε τιμή του ιστογράμματος κανονικοποιείται με το άθροισμα όλων των τιμών. Επομένως, καταλήγουμε με 10 χαρακτηριστικά που αντιστοιχούν σε 10 εύρη τιμών, καθένα από τα οποία δείχνει την πιθανότητα ή αλλιώς το ποσοστό των λέξεων που έχει συχνότητα εντός του εκάστοτε εύρους τιμών.

Για παράδειγμα, αν έχουμε το κείμενο:
This is a text. The text is short. We are beautiful. Let me explain. Happy birthday to you. Yes, I understand.

Οι κανονικοποιημένες συχνότητες ανά λέξη έχουν ως εξής:
This: 0.048, is: 0.095, a: 0.048, text: 0.095, The: 0.048 , short: 0.048, We: 0.048 , are: 0.048, beautiful: 0.048 , Let: 0.048, me: 0.048, explain: 0.048. Happy: 0.048 , birthday: 0.048, to: 0.048 , you: 0.048, Yes: 0.048, I: 0.048, understand: 0.048.

Το κανονικοποιημένο ιστόγραμμα 10 σημείων έχει ως εξής:
0,...,0.01 : 0.
0.01,...,0.02: 0.
0.02,...,0.03: 0.
0.03,...,0.04: 0.
0.04,...,0.05: 0.89
0.05,...,0.06: 0.
0.06,...,0.07: 0.
0.07,...,0.08: 0.
0.08,...,0.09: 0.
0.09,...,0.1 : 0.11

Έχουμε επιλέξει εύρος συχνοτήτων από 0 έως 0.1 καθώς το μέγιστο 0.1 σημαίνει ότι μία λέξη θα εμφανίζεται 1 στις 10 φορές, μια συχνότητα που είναι αρκετή.

Από τα παραπάνω καταλήγουμε με 12 νέα χαρακτηριστικά συνολικού κειμένου που θα προστεθούν στα 10 χαρακτηριστικά που προέκυψαν από τους ταξινομητές τμημάτων, στο προηγούμενο υποκεφάλαιο, για να χρησιμοποιηθούν από τα μοντέλα αξιολόγησης της ομιλίας.

Χαρακτηριστικά βάσει Κειμένου Αναφοράς

Σε αυτό το σημείο, είναι χρήσιμο να αναφερθεί και η προαιρετική προσθήκη κάποιων ακόμα χαρακτηριστικών. Εάν η ομιλία βασίζεται σε κάποιο κείμενο αναφοράς, δηλαδή ο ομιλητής καλούνταν να διαβάσει/απαγγείλει ένα προκαθορισμένο κείμενο, τότε μπορούν να προκύψουν κάποια ακόμα χαρακτηριστικά από την σύγκριση του κειμένου της ομιλίας, δηλαδή αυτού που τελικά είπε ο ομιλητής, με το κείμενο αναφοράς.

Πιο συγκεκριμένα, από μία τέτοια σύγκριση μπορούν να προκύψουν τρία χαρακτηριστικά:

- Recall
  Το recall σε αυτήν την περίπτωση αντικατοπτρίζει το πόσα είπε ο χρήστης από όσα υπήρχαν μέσα στο κείμενο αναφοράς. Αυτή η μετρική λαμβάνει υπόψιν και προσμετρά πιθανές παραλείψεις που μπορεί να έκανε ο χρήστης.

- Precision
  Το precision σε αυτήν την περίπτωση αντικατοπτρίζει το πόσα από αυτά που είπε

ο χρήστης υπήρχαν μέσα στο κείμενο. Αυτή η μετρική λαμβάνει υπόψιν πιθανές προσθήκες που μπορεί να έκανε ο χρήστης.

- F1 score
  Το σκορ F1 που όπως έχουμε δει και στην θεωρία υπολογίζεται από τα recall και precision ως εξής: $2 * precision * recall/(precision + recall)$.

Για να υπολογιστούν οι παραπάνω μετρικές/χαρακτηριστικά, θα πρέπει πρώτα να γίνει μια ευθυγράμμιση του κειμένου ομιλίας με το κείμενο αναφοράς. Αυτό πρακτικά σημαίνει ότι πρέπει να ξέρουμε κάθε στιγμή, αυτό που είπε ο χρήστης σε ποια λέξη αντιστοιχούσε στο κείμενο αναφοράς, ακόμα και αν δεν το είπε ακριβώς όπως εμφανίζεται στο κείμενο αναφοράς, ακόμα και αν παρέλειψε κάποιες λέξεις σε κάποια σημεία ή προσέθεσε δικές του. Δυο παραδείγματα ευθυγράμμισης παρουσιάζονται παρακάτω:

Example 1:
Reference text: You are very pretty
Speech text: You - very pretty

Example 2:
Reference text: You have - eyes
Speech text: You have pretty eyes

Στο πρώτο παράδειγμα μπορούμε να δούμε πως ο ομιλητής παρέλειψε την λέξη "are" και στο δεύτερο ότι πρόσθεσε μόνος του την λέξη "pretty", ενώ δεν υπήρχε στο αυθεντικό κείμενο αναφοράς. Έτσι, βλέπουμε πως η ευθυγράμμιση των δύο κειμένων πρέπει να γίνεται σωστά, ακόμα και για ιδιόμορφες περιπτώσεις, όπως στα παραπάνω παραδείγματα.

Η ευθυγράμμιση αυτή, μπορεί να πραγματοποιηθεί υπολογίζοντας όλους τους πιθανούς συνδυασμούς, δηλαδή όλα τα πιθανά ταιριάσματα λέξεων ανάμεσα στο κείμενο ομιλίας και στο κείμενο αναφοράς και επιλέγοντας τελικά τον συνδυασμό, ο οποίος είναι πιο αποδοτικός, δηλαδή δίνει το μεγαλύτερο σκορ. Ως σκορ εδώ ορίζουμε το συνολικό σκορ από το ταίριασμα όλων των λέξεων. Πιο συγκεκριμένα, όταν δύο λέξεις είναι αυτούσιες και τις ταιριάζουμε μαζί, τότε αυτό το ταίριασμα δίνει σκορ = 1 (matching words: score=1). Αντίθετα, όταν δύο λέξεις δεν είναι αυτούσιες, τότε υπολογίζεται το σκορ με βάση το Leveshtein ratio (mismatching words: score = Leveshtein ratio). Το Levenshtein ratio είναι μια μετρική απλής απόστασης λέξεων, η οποία συγκρίνει τις δυο λέξεις γράμμα-γράμμα. Για παράδειγμα, αν έχουμε τις λέξεις ab και ac τότε το Leveshtein ratio θα δώσει 0.5 σκορ αφού τα μισά γράμματα των δύο λέξεων είναι ίδια (το a με το a) και τα άλλα μισά όχι (το b με το c).

Η εύρεση του καλύτερου συνδυασμού, δηλαδή αυτού με το μέγιστο σκορ, γίνεται με την χρήση ενός δυναμικού αλγορίθμου, ο οποίος υπολογίζει έναν πίνακα δύο διαστάσεων. Ο πίνακας αυτός έχει τόσες σειρές όσος ο αριθμός λέξεων του κειμένου ομιλίας και τόσες στήλες όσος ο αριθμός λέξεων του κειμένου αναφοράς. Έτσι διατρέχουμε τον πίνακα για κάθε λέξη του κειμένου ομιλίας που μπορεί να συνδυαστεί με κάθε λέξη του κειμένου αναφοράς. Σε κάθε επανάληψη εξετάζουμε τρεις περιπτώσεις, είτε να πάρουμε το σκορ ταιριάζοντας τις δυο λέξεις της επανάληψης αυτής, είτε να πάρουμε το σκορ βάζοντας κενό στο κείμενο ομιλίας (παράλειψη λέξης), είτε βάζοντας κενό στο κείμενο αναφοράς (προσθήκη λέξης από τον ομιλητή). Σε καθεμία από τις περιπτώσεις αυτές προσθέτουμε το σκορ στο σκορ που έχουμε συλλέξει με τους μέχρι τώρα συνδυασμούς και επιλέγουμε τελικά την περίπτωση που δίνει το μέγιστο τελικό σκορ για να ανανεώσουμε τον πίνακα στην θέση της συγκεκριμένης επανάληψης. Στο τέλος, η τελευταία θέση του πίνακα αυτού, δίνει το μέγιστο συνολικό σκορ, δηλαδή το σκορ που έχει προκύψει από τον βέλτιστο συνδυασμό όλων των λέξεων. Επιλέγεται τελικά η ευθυγράμμιση που αντιστοιχεί

σε αυτό το σκορ, και με βάση αυτήν την ευθυγράμμιση υπολογίζονται οι τρεις μετρικές που αναφέραμε παραπάνω (recall, precision και f1).

Τα τρία αυτά χαρακτηριστικά μπορούν να προστεθούν στα υπόλοιπα χαρακτηριστικά συνολικού κειμένου, ώστε να χρησιμοποιηθούν από τα τελικά μοντέλα. Παρ᾽ όλα αυτά, στην συγκεκριμένη διπλωματική εργασία εξετάζουμε την ποιότητα της ελεύθερης ομιλίας. Έτσι, δεν θα ήταν χρήσιμο να συγκρίνουμε την ομιλία αυτή με ένα προκαθορισμένο κείμενο. Επομένως, δεν θα χρησιμοποιήσουμε τα τρία αυτά επιπλέον χαρακτηριστικά. Υπάρχουν, όμως, σαν επιλογή στο σύστημα, ώστε να δίνουν την δυνατότητα χρήσης σε άλλες διεργασίες ανάλυσης ομιλίας όπως για παράδειγμα θα μπορούσε να είναι ο εντοπισμός/αξιολόγηση της δυσλεξίας, όπου ο χρήστης/ομιλητής καλείται να διαβάσει προκαθορισμένο κείμενο και στην αξιολόγηση παίζει σημαντικό ρόλο η εύρεση των λαθών που έκανε με βάση το κείμενο αναφοράς.

6 Συλλογή Δεδομένων και Επισημείωση

Με την ολοκλήρωση του προηγούμενου κεφαλαίου, έχουμε πλέον συλλέξει όλα τα χαρακτηριστικά επιπέδου εγγραφής, είτε αυτά προήλθαν από τους ταξινομητές τμημάτων, είτε προστέθηκαν στην συνέχεια ως υψηλού επιπέδου χαρακτηριστικά. Το σύνολο αυτών των χαρακτηριστικών έχει ως τελικό στόχο να χρησιμοποιηθεί από ταξινομητές επιπέδου εγγραφής, οι οποίοι θα μπορούν να αξιολογούν την ποιότητα της συνολικής ομιλίας.

Για την εκπαίδευση αλλά και την αξιολόγηση αυτών των ταξινομητών, ακολουθήθηκε μια διαδικασία συλλογής, επισημείωσης και συμφωνίας δεδομένων. Εφόσον δεν υπάρχουν ανοιχτού τύπου δεδομένα στο διαδίκτυο, που να αφορούν την ποιότητα της ομιλίας και που να είναι επισημειωμένα (επιβλεπόμενη μάθηση), η διαδικασία αυτή της συλλογής των δεδομένων εκτελέστηκε από άκρο σε άκρο, στα πλαίσια της έρευνας για την παρούσα διπλωματική εργασία.

Στα παρακάτω υποκεφάλαια παρουσιάζονται όλα τα βήματα που ακολουθήθηκαν για την εξ' ολοκλήρου δημιουργία ενός επισημειωμένου συνόλου δεδομένων ομιλίας.

6.1 Συλλογή Δεδομένων

Για τον σκοπό της συλλογής των δεδομένων και της δημιουργίας μίας βάση αυτών, δημιουργήθηκε ένας διαδικτυακός ιστότοπος με χρήση των γλωσσών προγραμματισμού Javascript/html/css σε ο,τι αφορά το front-end κομμάτι, καθώς επίσης και nodejs σε ο,τι αφορά την επικοινωνία με τον server, δηλαδή το back-end κομμάτι. Ο ιστότοπος αυτός ανέβηκε σε έναν online server στην υπηρεσία cloud του okeanos.

Ο ιστότοπος εμπεριείχε πληροφορίες προς ενημέρωση των χρηστών, δηλαδή των υποψήφιων ομιλητών, κάποια δημογραφικά χαρακτηριστικά που οι χρήστες καλούνταν να συμπληρώσουν και στην συνέχεια την κύρια διεργασία της ηχογράφησης ομιλίας.

Ως δημογραφικά χαρακτηριστικά υπήρχαν τα εξής:

- Το όνομα χρήστη.
- Το φύλο.
- Η ηλικία.
- Το επίπεδο αγγλικών.
- Το πόσο συχνά ο χρήστης μιλάει στα αγγλικά ή διαβάζει κάποιο αγγλικό κείμενο.
- Εάν αισθάνεται άγχος όταν μιλάει παρουσία κοινού.
- Εάν είναι εσωστρεφής ή εξωστρεφής σε ο,τι αφορά το να εξασκεί τις επικοινωνιακές δεξιότητες.
- Ποιο είναι το επίπεδο της αίσθηση του χιούμορ που έχει.
- Πόσο εύκολο είναι για αυτόν να βρίσκει παραδείγματα/να επιχειρηματολογεί/να αιτιολογεί.
- Ποια είναι η συμπεριφορά του απέναντι σε διαπληκτισμούς. Εάν δηλαδή τους αποφεύγει τους αποδέχεται ή τους προκαλεί.

Όλες οι παραπάνω ερωτήσεις έχουν ως στόχο να δημιουργήσουν ένα προφίλ του ομιλητή, δίνοντας κάποιες επιπλέον πληροφορίες που ίσως θα μπορούσαν στην συνέχεια να βοηθήσουν στην αιτιολόγηση των αποτελεσμάτων ή να συμβάλλουν στην διαδικασία παραγωγής τους.

Σε ο,τι αφορά την κύρια διεργασία ηχογράφησης ομιλίας, συλλέχθηκαν 40 σύντομα αγγλικά κείμενα (4-5 γραμμές το καθένα) από το διαδίκτυο και κυρίως από το medium, πάνω σε διαφορετικά θέματα (όπως η πολιτική, τα βιβλία, η μηχανική μάθηση κ.α.) τα οποία εμπνέουν μια έντονη ανάγνωση/ύφος στον ομιλητή. Επιλέχθηκαν, επίσης, 20 γενικού τύπου ερωτήσεις που απαντώνται σύντομα, όπως "ποια είναι η καλύτερη εποχή;", "ποια πιστεύεις πως θα είναι η μεγαλύτερη πρόκληση για την ανθρωπότητα τις επόμενες δεκαετίες;", "τι σου αρέσει περισσότερο στην τωρινή σου εργασία;" κ.α.

Ο λόγος για τον οποίο δημιουργήθηκαν δύο ειδών κατηγορίες, αυτή των προκαθορισμένων κειμένων και αυτή των ελεύθερων ερωτήσεων, είναι από την μία για να δοθεί η ευχέρεια στους χρήστες να διαβάσουν ένα συγκεκριμένο κείμενο, παρέχοντάς τους έτσι μεγαλύτερη ευκολία, προσαρμόζοντάς το όμως στην ανάλογη εκφραστικότητα, τον τόνο και την προφορά και από την άλλη για να εξεταστεί η ελεύθερη ομιλία, η οποία δεν εξαρτάται από κάποιο κείμενο αναφοράς, αλλά είναι αυθόρμητη και στιγμιαία. Αυτοί οι δύο άξονες δίνουν και την δυνατότητα σύγκρισης, καθώς αναμένονται διαφορετικά αποτελέσματα στο ελεύθερο κείμενο από ότι στο προκαθορισμένο, κυρίως σε ό,τι αφορά την πληροφορία που παρέχεται από το κειμενικό κομμάτι του συστήματος (3.2).

Το σύνολο αυτών των 60 κειμένων/ερωτήσεων εμφανίζονται στον χρήστη με τυχαία σειρά και ο ίδιος καλείται να διαβάσει στην περίπτωση των κειμένων ή να απαντήσει στην περίπτωση των ερωτήσεων, ηχογραφώντας τον εαυτό του. Στην συνέχεια, έχει την ευκαιρία να ακούσει τις ηχογραφήσεις του και να διαλέξει να ανεβάσει (αποθηκεύσει στον server), αυτή που θεωρεί πως ήταν η καλύτερη ως προς την ποιότητα της ομιλίας του.

Ο ιστότοπος αυτός κοινοποιήθηκε τόσο σε άντρες όσο και σε γυναίκες, καθώς επίσης και τόσο σε άτομα ελληνικής καταγωγής όσο και σε άτομα άλλων εθνικοτήτων, ώστε να υπάρχει μια ποικιλομορφία ως προς την προφορά και τα δεδομένα να είναι όσο το δυνατόν πιο γενικευμένα.

Συγκεκριμένα, συλλέχθηκαν 695 ηχογραφήσεις/ομιλίες συνολικά, από 42 διαφορετικά άτομα εκ των οποίων τα 26 ήταν γυναικείου φύλου και τα 16 ανδρικού.

6.2 Κλάσεις Δεδομένων

Ύστερα από την συλλογή των δεδομένων που περιγράφηκε παραπάνω, είναι απαραίτητη και παρεπόμενη η διαδικασία επισημείωσής τους, δηλαδή εφαρμογής ετικετών (labels) σε καθένα δείγμα. Εφόσον χρησιμοποιούμε επιβλεπόμενη μάθηση, τα δεδομένα πρέπει να συνοδεύονται από τις ετικέτες τους, ώστε τα μοντέλα που θα εκπαιδευτούν να μπορούν να μάθουν να ταξινομούν τα δείγματα στην ανάλογη ετικέτα-κλάση, παρατηρώντας κάποια μοτίβα από τα χαρακτηριστικά τους.

Τρία είναι τα tasks που επιλέχθηκαν προς επισημείωση: η εκφραστικότητα, η ευκολία παρακολούθησης της ομιλίας και το πόσο απολαυστική είναι αυτή.

Πιο συγκεκριμένα, ως εκφραστικότητα ορίζεται το πόσο ενεργητική, συναισθηματική ή παθιασμένη είναι η ομιλία, ανεξάρτητα από το περιεχόμενό της. Η επισημείωση στην συγκεκριμένη κλάση γίνεται με χρήση 5 ετικετών:

1. Καθόλου εκφραστική
2. Όχι τόσο εκφραστική
3. Κάπως εκφραστική
4. Λίγο εκφραστική
5. Πολύ εκφραστική

Ως ευκολία παρακολούθησης, ορίζεται η αξιολόγηση της λεκτικής σαφήνειας, της ευχέρειας και του ρυθμού ομιλίας, για το συγκεκριμένο περιεχόμενο που περιγράφεται.

Σημειώνεται ότι η ευχέρεια, η σαφήνεια και ο ρυθμός μπορούν να διασυνδεθούν και να συσχετιστούν, π.χ. ένας ομιλητής μπορεί να κάνει την ομιλία του πολύ εύκολη στην παρακολούθηση, παρά το γεγονός ότι μιλά πολύ γρήγορα, ενώ ένας άλλος ομιλητής μπορεί να μιλήσει αργά και να μην κάνει την ομιλία του εύκολη στην παρακολούθηση. Η επισημείωση της συγκεκριμένης κλάσης γίνεται και πάλι με χρήση 5 ετικετών:

1. Πολύ δύσκολο στην παρακολούθηση
2. Λίγο δύσκολο στην παρακολούθηση
3. Σχετικά εύκολο στην παρακολούθηση
4. Κατά κόρον εύκολο στην παρακολούθηση
5. Πολύ εύκολο στην παρακολούθηση

Ως απόλαυση, ορίζεται η προσωπική άποψη του ακροατή/σχολιαστή ως προς το αν η ομιλία ήταν συναρπαστική, διασκεδαστική ή παρακινητική. Και σε αυτήν την περίπτωση εξετάζονται 5 ετικέτες:

1. Την μίσησα
2. Δεν μου πολυάρεσε
3. Ήταν εντάξει
4. Μου άρεσε λίγο
5. Την αγάπησα

Τα τρία παραπάνω tasks επιλέχθηκαν με τρόπο ώστε να είναι ανεξάρτητα μεταξύ τους, δηλαδή ο σχολιαστής/ακροατής να βαθμολογεί το καθένα ανεξάρτητα, χωρίς το ένα να συνδέεται με το άλλο. Αυτό πρακτικά σημαίνει πως μια ομιλία θα μπορούσε να είναι εκφραστική αλλά όχι διασκεδαστική, ή θα μπορούσε να είναι διασκεδαστική αλλά όχι τόσο εύκολη στην παρακολούθηση. Επομένως, μιλάμε για τρεις ανεξάρτητες διεργασίες οι οποίες θα εκτελεστούν από τρεις διαφορετικούς ταξινομητές.

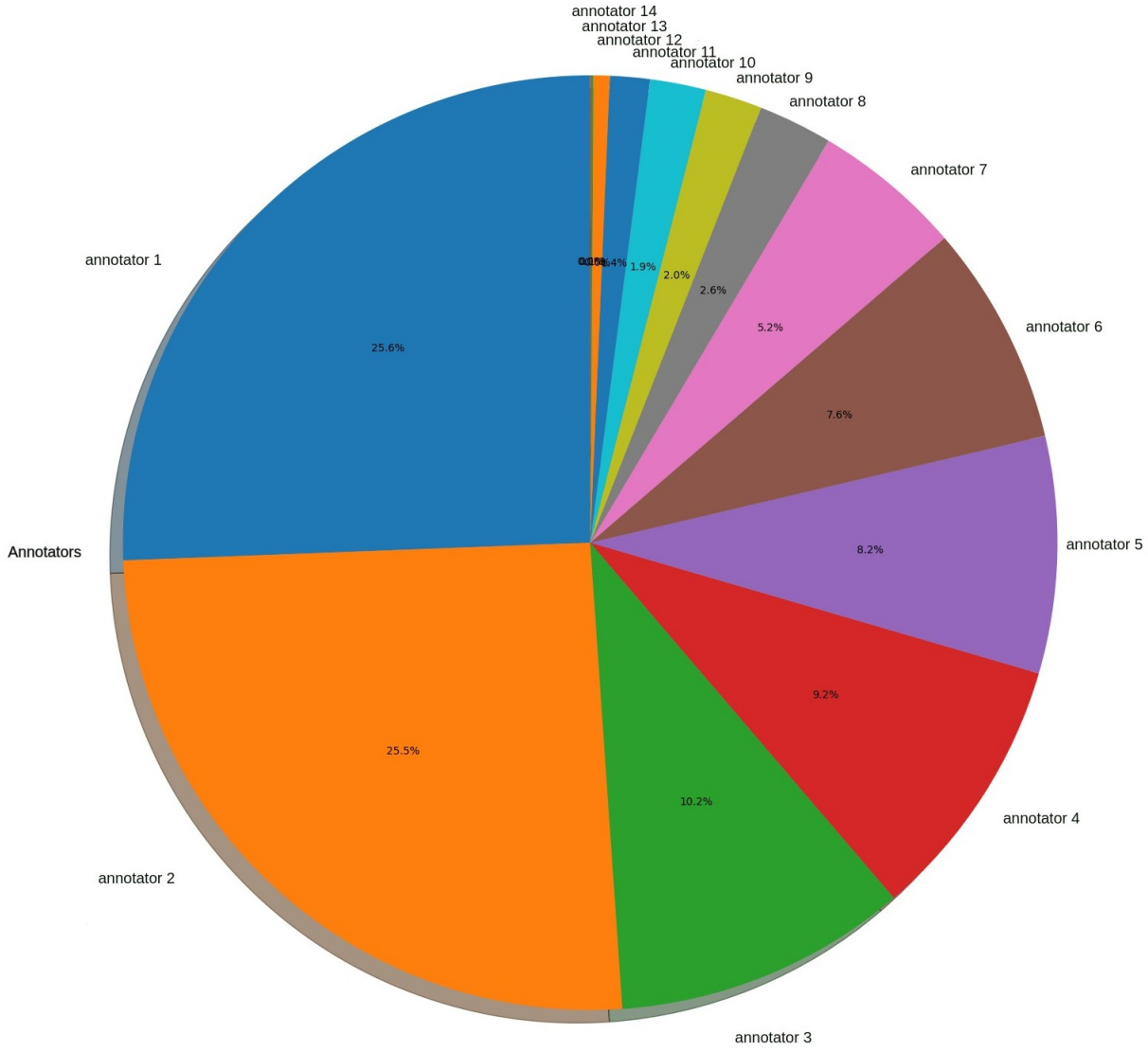
Πηγές: [104],[105]

6.3 Επισημείωση Δεδομένων

Έχοντας πλέον ορίσει τους τρεις άξονες πάνω στους οποίους θα αξιολογηθεί η κάθε ομιλία, δημιουργήθηκε ακόμα ένα UI (user interface), αυτήν την φορά με την βοήθεια της python/html και του flask. Ανέβηκε επίσης σε έναν online server στην υπηρεσία cloud του okeanos και στην συνέχεια ζητήθηκε από διάφορους ακροατές/σχολιαστές να αξιολογήσουν μέσω αυτής της πλατφόρμας/ιστοτόπου, τις ομιλίες που συλλέχθηκαν πρωτύτερα.

Πιο συγκεκριμένα, ο κάθε χρήστης έκανε εγγραφή στον ιστότοπο, στην συνέχεια ξεκινούσε την διαδικασία επισημείωσης όπου εμφανίζονταν ομιλίες με τυχαία σειρά, τις οποίες καλούνταν να ακούσει και τέλος να διαλέξει μια ετικέτα που αρμόζει στην εκάστοτε ομιλία για καθένα από τα τρία tasks (εκφραστικότητα,ευκολία παρακολούθησης,απόλαυση).

Συνολικά, από τα 695 δείγματα, επισημειώθηκαν με ετικέτες τα 689. Επίσης, οι συνολικές επισημειώσεις που έγιναν από όλους τους χρήστες ήταν 2687 (για καθένα από τα 3 tasks), ενώ ο αριθμός των χρηστών/σχολιαστών ήταν 14. Στο παρακάτω σχήμα φαίνεται πιο αναλυτικά ο αριθμός επισημειώσεων ανά χρήστη:



Σχήμα 6.1: Κατανομή επισημειώσεων ανά χρήστη

6.4 Συμφωνία Επισημειώσεων και Παραγωγή Τελικών Ετικετών

Μετά την διαδικασία της επισημείωσης των δεδομένων, ακολουθεί η συνάθροιση και η συμφωνία αυτών. Κάθε δείγμα θα πρέπει να καταλήξει με μία και μοναδική ετικέτα για το κάθε task. Για να συμβεί αυτό, θα πρέπει με κάποιον τρόπο να επέλθει μια συμφωνία μεταξύ των ετικετών που έθεσαν οι χρήστες/σχολιαστές.

Για τον σκοπό της συνάθροισης των επισημειώσεων, εκτελέστηκε η διαδικασία ξεχωριστά για τα δείγματα/ομιλίες που είχαν γυναίκα ομιλητή και ξεχωριστά για αυτά με άντρα ομιλητή. Αυτό συνέβη διότι, όπως είδαμε παραπάνω, οι γυναίκες ομιλητές υπερτερούν σε αριθμό από τους άντρες και ως εκ τούτου τα δείγματα γυναικών είναι πολύ περισσότερα από αυτά των αντρών. Έτσι, για να μην γίνει το μοντέλο πολύ bias ως προς το φύλο, δηλαδή να αποφευχθεί η αναγωγή του προβλήματος σε gender recognition, αποφασίστηκε να εξεταστούν τα δύο φύλα ξεχωριστά.

Παραγωγή Δυαδικών Ετικετών

Παρ᾽ ότι οι ετικέτες είναι διακριτές (5 ετικέτες/τιμές ανά task), έχουν μια συνεχή, κλιμακωτή μορφή καθώς η μικρότερη ετικέτα (1) αντιστοιχεί στην χειρότερη αξιολόγηση και η

μεγαλύτερη ετικέτα (5) αντιστοιχεί στην καλύτερη. Ως εκ τούτου, αντί να συναθροιστούν οι ετικέτες ενός δείγματος με βάση την πλειοψηφία (majority vote), θα χρησιμοποιηθεί μέσος όρος (averaging).

Πιο συγκεκριμένα, για κάθε task (εκφραστικότητα, ευκολία παρακολούθησης, απόλαυση) και για κάθε φύλο (γυναικείο, αντρικό), θα εξαχθούν 2 κλάσεις/ετικέτες : μία negative και μια positive, αντί για τις 5 αρχικές. Αυτό συμβαίνει λόγω έλλειψης δεδομένων, καθώς δεν υπάρχουν επαρκή δείγματα για όλες τις ετικέτες. Η εξαγωγή αυτών των δύο κλάσεων προκύπτει έπειτα από τους παρακάτω περιορισμούς/βήματα:

1. Ο πρώτος περιορισμός είναι πως διατηρούνται μόνο τα δείγματα τα οποία έχουν από τρεις σχολιαστές (annotators) και πάνω. Αυτό συμβαίνει διότι για τα δείγματα με 2 ή ακόμα χειρότερα με μόνο 1 ακροατή/αξιολογητή δεν μπορούν να εξαχθούν έγκυρα συμπεράσματα ως προς την τελική ετικέτα, καθώς αυτή θα βασίζεται στην προσωπική οπτική ενός ή δύο προσώπων.

2. Ένας άλλος περιορισμός είναι τα κατώφλια της μέσης τιμής. Υπολογίζεται η μέση τιμή των επισημειώσεων που έχουν δοθεί από διαφορετικούς σχολιαστές στο κάθε δείγμα, και στην συνέχεια αν αυτή η τιμή είναι κάτω από ένα συγκεκριμένο κατώφλι, τότε το δείγμα θεωρείται αρνητικό (negative) και επισημειώνεται αναλόγως. Αντίθετα, αν η μέση τιμή είναι πάνω από ένα συγκεκριμένο κατώφλι, τότε το δείγμα θεωρείται θετικό (positive) και επισημειώνεται αναλόγως. Στην περίπτωση που η μέση τιμή, βρίσκεται ενδιάμεσα από τις τιμές των κατωφλίων, το δείγμα αυτό απορρίπτεται και δεν θα χρησιμοποιηθεί στην τελική διεργασία. Για παράδειγμα, αν έχουμε ως κατώτερο κατώφλι την τιμή 2.0 και ως ανώτερο την τιμή 4.0, αυτό σημαίνει πως ένα δείγμα με μέση τιμή επισημειώσεων μικρότερη ή ίση του 2, θα λάβει ως τελική ετικέτα το "negative", ενώ ένα δείγμα με μέση τιμή επισημειώσεων μεγαλύτερη ή ίση του 4, θα λάβει ως τελική ετικέτα το "positive". Έτσι, αν για παράδειγμα 3 χρήστες έχουν αξιολογήσει το συγκεκριμένο δείγμα στο task της εκφραστικότητας, ως "όχι τόσο εκφραστικό" (τιμή 2),"όχι τόσο εκφραστικό" (τιμή 2), "λίγο εκφραστικό" (τιμή 4), η μέση τιμή αυτών των επισημειώσεων είναι $(2 + 2 + 4)/3 = 2.7$ και το δείγμα θα απορριφθεί καθώς δεν ικανοποιεί καμία από τις δυο ανισότητες των κατωφλίων ($<=2$ (negative) ή $>=4$ (positive)).

3. Εκτός από την μέση τιμή, εξετάζεται και ένα κατώφλι απόκλισης. Συγκεκριμένα, υπολογίζεται η διάμεση απόλυτη απόκλιση των επισημειώσεων ενός δείγματος, που είναι η διάμεσος έναντι των απόλυτων αποκλίσεων από την διάμεσο. Πρακτικά, είναι η μέση απόκλιση που προκύπτει από τις αποκλίσεις της κάθε επισημείωσης από τον μέσο όρο. Έτσι, για κάθε δείγμα απαιτείται η τιμή αυτή να είναι μικρότερη ή ίση από ένα προκαθορισμένο κατώφλι. Αν αυτή η ανισότητα δεν ικανοποιείται τότε το δείγμα απορρίπτεται. Για παράδειγμα, αν 5 χρήστες έχουν αξιολογήσει ένα δείγμα στο task της ευκολίας παρακολούθησης ως "κατά κόρον εύκολο" (τιμή 4),"κατά κόρον εύκολο" (τιμή 4),"κατά κόρον εύκολο" (τιμή 4),"κατά κόρον εύκολο" (τιμή 4),"πολύ εύκολο" (τιμή 5), τότε η μέση τιμή των επισημειώσεων αυτών είναι 4.2 και η απόκλιση 0.32. Για να διατηρηθεί το συγκεκριμένο δείγμα, θα πρέπει το 0.32 να είναι μικρότερο ή ίσο από το κατώφλι απόκλισης, καθώς επίσης και η μέση τιμή 4.2 να ικανοποιεί μία από τις δυο ανισότητες των κατωφλίων του προηγούμενου βήματος.

Συμφωνία Επισημειώσεων

Ως μέση διαφωνία των χρηστών/αξιολογητών ορίζεται η μέση τιμή των διάμεσων απόλυτων αποκλίσεων από όλα τα δείγματα. Πρακτικά, υπολογίζεται η διαφωνία των χρηστών για κάθε δείγμα ξεχωριστά, παίρνοντας την διάμεση απόλυτη απόκλιση των επισημειώσεων

που περιγράψαμε παραπάνω, και στην συνέχεια υπολογίζεται η μέση τιμή όλων αυτών των διαφωνιών για να προκύψει μια μέση συνολική διαφωνία πάνω σε όλο το dataset και τις επισημειώσεις. Με αυτόν τον τρόπο, έχουμε μια εικόνα του πόσο καλά επισημειωμένα μπορεί να είναι τελικώς τα δείγματα. Αν η μέση διαφωνία είναι υψηλή τότε οι χρήστες είχαν διαφορετικές απόψεις ως προς την αξιολόγηση και επομένως, τα δείγματα δεν θα έχουν μια "καθαρή", έγκυρη ετικέτα. Αντίθετα, αν η μέση διαφωνία είναι χαμηλή, αυτό σημαίνει πως στα περισσότερα δείγματα οι αξιολογήσεις των χρηστών συμφωνούσαν και επομένως οι τελικές ετικέτες είναι καλύτερα ορισμένες.

Πέρα από την συνολική διαφωνία για όλα τα δεδομένα, υπολογίζεται επίσης και η μέση διαφωνία για κάθε χρήστη ξεχωριστά. Έτσι, για κάθε δείγμα που ο χρήστης αξιολόγησε, υπολογίζεται η απόκλιση της ετικέτας/επισημείωσης που έθεσε, από την μέση τιμή όλων των επισημειώσεων του δείγματος αυτού. Στην συνέχεια, παίρνοντας την μέση τιμή των αποκλίσεων αυτών, έχουμε την μέση διαφωνία του χρήστη. Αυτό πρακτικά είναι ένας δείκτης εγκυρότητας του χρηστή, ο οποίος μπορεί να μας δείξει πόσο κοντά βρίσκονται οι αξιολογήσεις του συγκεκριμένου χρήστη, σε σχέση με το υπόλοιπο σύνολο, δηλαδή πόσο καλή είναι η κρίση του. Όταν η μέση διαφωνία ενός χρήστη είναι πολύ υψηλή σε σχέση με των υπολοίπων, τότε οι αξιολογήσεις του συγκεκριμένου χρήστη απορρίπτονται και δεν λαμβάνονται υπόψιν στην διαδικασία της παραγωγής τελικών ετικετών.

6.5 Τελικά Datasets

Με βάση τα παραπάνω, θα δούμε πως τελικά διαμορφώθηκαν τα δεδομένα με τις ετικέτες τους, για κάθε task ξεχωριστά.

6.5.1 Expressiveness

Στον παρακάτω πίνακα, παρουσιάζονται οι τιμές κατωφλίων που χρησιμοποιήθηκαν για το task της εκφραστικότητας, καθώς επίσης και ο αριθμός δειγμάτων ανά κλάση, μετά την εφαρμογή κάθε περιορισμού. Δηλαδή, αρχικά εφαρμόζονται τα κατώφλια μέσης τιμής, στην συνέχεια εφαρμόζεται το κατώφλι της απόκλισης και τέλος εφαρμόζεται ο περιορισμός του ελάχιστου αριθμού σχολιαστών ανά δείγμα (3 στην προκειμένη περίπτωση). Τέλος, στον πίνακα εμφανίζεται και η μέση διαφωνία των σχολιαστών. Τα αποτελέσματα αυτά παρουσιάζονται για κάθε φύλο ξεχωριστά:

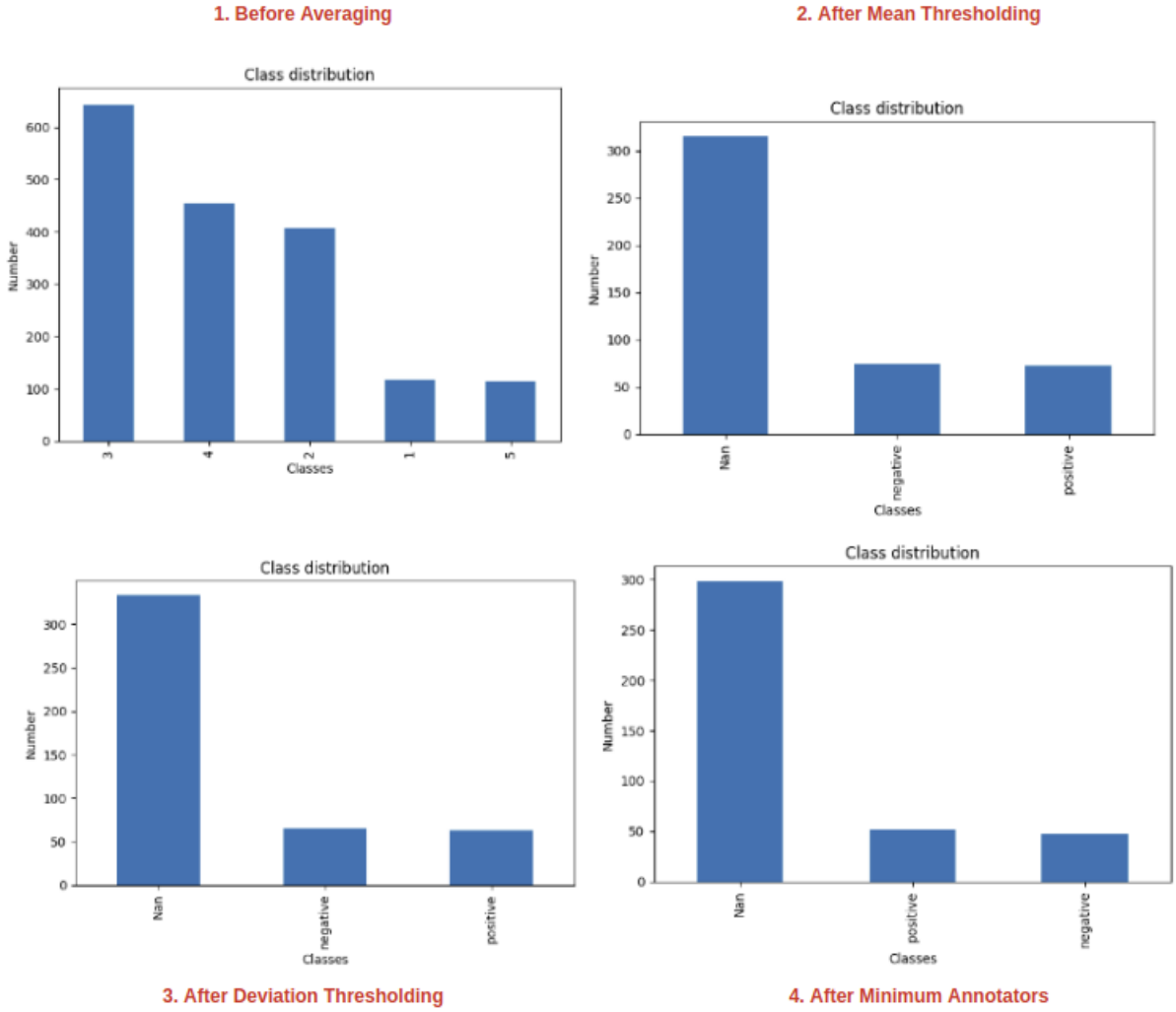
Expressiveness				
	Female		**Male**	
	Positive	**Negative**	**Positve**	**Negative**
Mean Threshlod	μ>=4	μ<=2	μ>=3.1	μ<=2
Number of samples after Mean Thresholding	72	80	70	67
Deviation Threshold	σ<0.75	σ<0.75	σ<0.75	σ<0.75
Number of samples after Deviation Thresholding	63	71	48	60
Number of samples after Minimum Annotators	52	53	41	50
Average Disagreement	**0.52**		**0.53**	

Table 6.1: Καθορισμός του Dataset Εκφραστικότητας

Στις παρακάτω εικόνες, φαίνεται πιο αναλυτικά η κατανομή του αριθμού των δειγμάτων

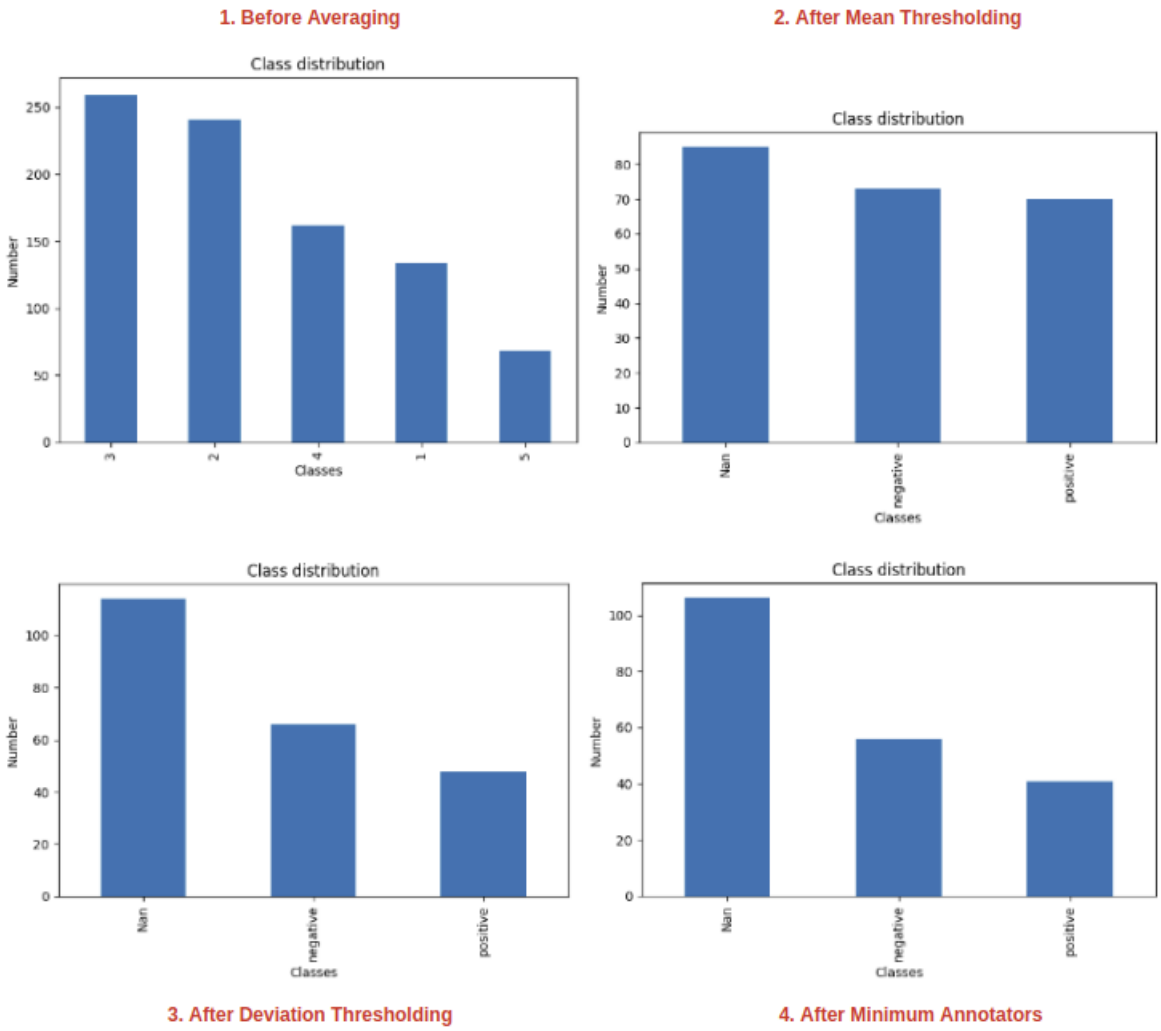
ανά κλάση, 1. πριν εφαρμοστεί μέσος όρος επισημειώσεων, 2. μετά από την εφαρμογή κατωφλίου μέσης τιμής, 3. μετά την εφαρμογή κατωφλίου απόκλισης και τέλος 4. μετά την εφαρμογή ελάχιστου αριθμού σχολιαστών.

Female



Σχήμα 6.2: Κατανομή αριθμού δειγμάτων ανά κλάση | Female Expressiveness

Male



Σχήμα 6.3: Κατανομή αριθμού δειγμάτων ανά κλάση | Male Expressiveness

Σημειώνεται πως στα παραπάνω αποτελέσματα, για τα female, απορρίφθηκε ένας σχολιαστής ο οποίος είχε μέση διαφωνία 1.03, ενώ για τα male απορρίφθηκαν τρεις σχολιαστές που είχαν τιμές διαφωνιών 1.5,0.79 και 0.73 αντίστοιχα. Αυτό συνέβη διότι οι διαφωνίες αυτές ήταν υψηλές σε σχέση με των υπολοίπων σχολιαστών και έτσι οι επισημειώσεις των συγκεκριμένων χρηστών θεωρήθηκαν μη έγκυρες.

Σε ο,τι αφορά το κατώφλι της απόκλισης (σ), παρατηρείται πως έχει επιλεγεί πάντα η ίδια τιμή (0.75), καθώς θεωρήθηκε σκόπιμο να διατηρηθεί σε αυστηρά όρια (χωρίς να αυξάνεται), ώστε να "επιζήσουν" μόνο τα δείγματα στα οποία οι χρήστες/σχολιαστές συμφωνούσαν ως έναν εύλογο βαθμό.

Παρατηρείται, επίσης, πως για τα female το κατώφλι μέσης τιμής για την κλάση positive είναι πιο αυστηρό από ότι για τα male. Αυτό συμβαίνει διότι τα δείγματα των αντρών ήταν λιγότερα και προκειμένου να δημιουργηθεί ένα dataset το οποίο να έχει επαρκή δείγματα αλλά να είναι και ισορροπημένο ως προς τις δύο κλάσεις, θα έπρεπε να διευρυνθεί το εύρος τιμών που γίνεται αποδεκτό για την κλάση positive. Πρακτικά, ένα πιο χαλαρό κατώφλι σημαίνει πως τα δείγματα ίσως να μην είναι τόσο διαχωρίσιμα στις δύο κλάσεις καθώς η μία κλάση πλησιάζει πιο κοντά στην άλλη.

Οπότε τελικά, για τα **female** καταλήγουμε με 52 δείγματα στην positive κλάση και

με 53 στην negative. Ακούγοντας αυτές τις τελικές ταξινομήσεις, αφαιρέθηκαν 5 ακόμα δείγματα από την κλάση positive τα οποία δεν φαινόντουσαν να είναι αρκετά καλά ως προς την εκφραστικότητα. Επομένως, ο τελικός αριθμός δειγμάτων ανέρχεται στα **47** δείγματα στην κλάση **positive** και **53** στην κλάση **negative**.

Αντίστοιχα για τα **male**, καταλήγουμε με 41 δείγματα στην κλάση positive και 50 στην κλάση negative. Με ένα άκουσμα αφαιρέθηκαν 2 επιπλέον δείγματα από την κλάση negative, τα οποία δεν φαινόντουσαν να είναι αρκετά κακά ως προς την εκφραστικότητα. Έτσι, καταλήξαμε με **41** δείγματα στην κλάση **positive** και **48** στην κλάση **negative**.

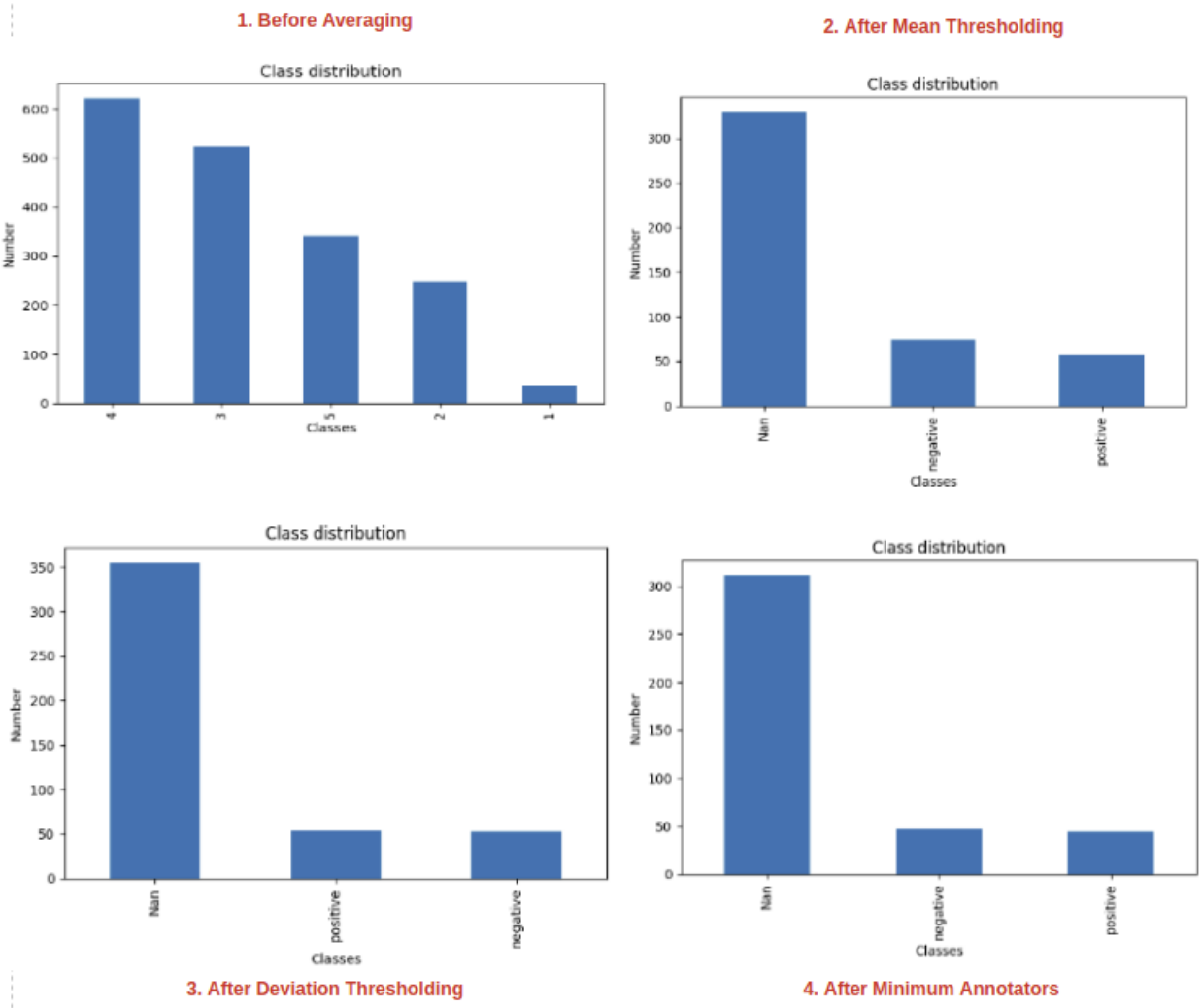
6.5.2 Ease of Following

Ease of Following				
	Female		**Male**	
	Positive	**Negative**	**Positve**	**Negative**
Mean Threshlod	μ>=4.5	μ<=2.8	μ>=3.6	μ<=2.2
Number of samples after Mean Thresholding	57	82	66	49
Deviation Threshold	σ<0.75	σ<0.75	σ<0.75	σ<0.75
Number of samples after Deviation Thresholding	54	60	49	38
Number of samples after Minimum Annotators	44	54	43	33
Average Disagreement	**0.57**		**0.58**	

Table 6.2: Καθορισμός του Dataset Ευκολίας Παρακολούθησης

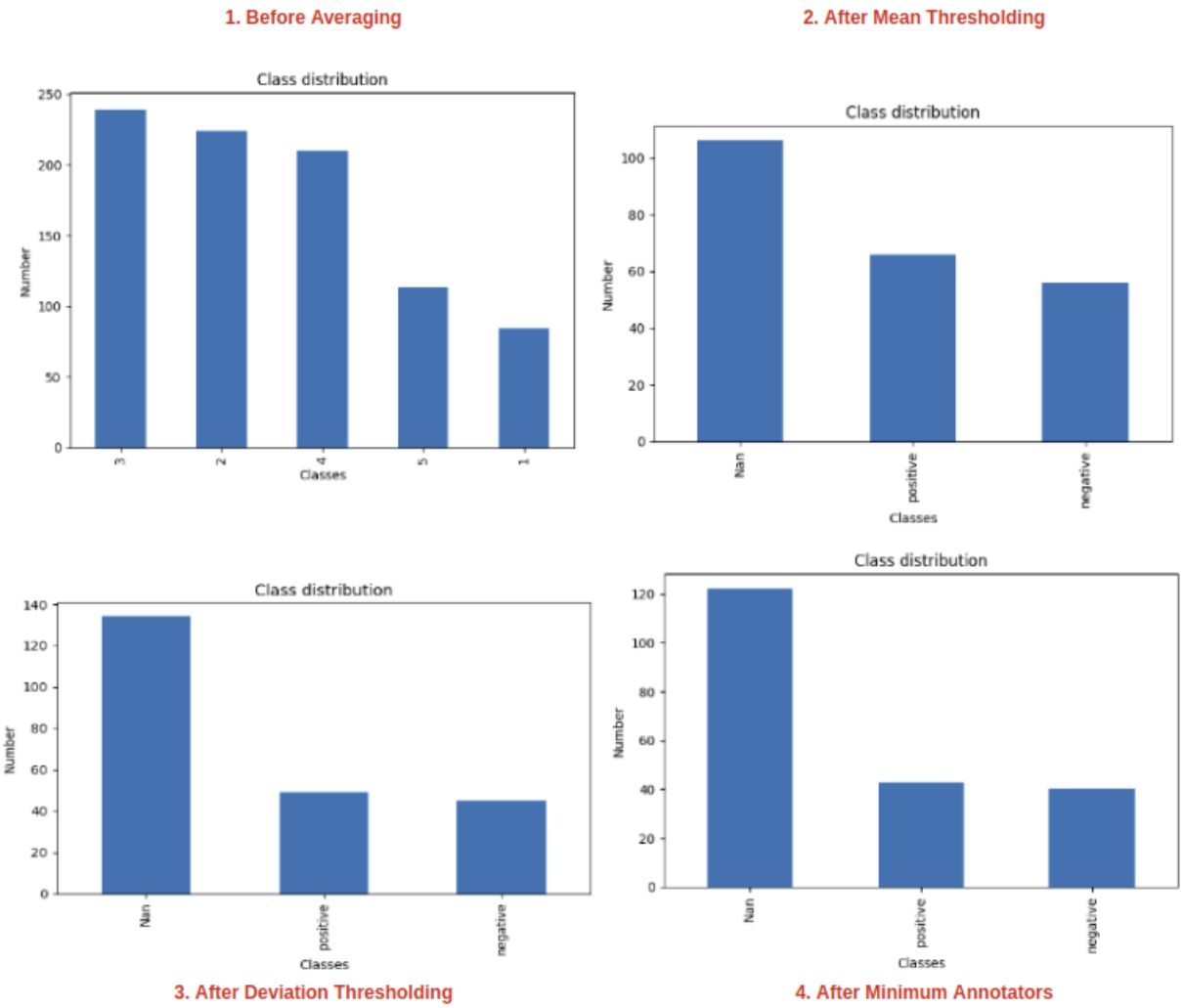
Στις παρακάτω εικόνες, φαίνεται πιο αναλυτικά η κατανομή του αριθμού των δειγμάτων ανά κλάση, 1. πριν εφαρμοστεί μέσος όρος επισημειώσεων, 2. μετά από την εφαρμογή κατωφλίου μέσης τιμής, 3. μετά την εφαρμογή κατωφλίου απόκλισης και τέλος 4. μετά την εφαρμογή ελάχιστου αριθμού σχολιαστών.

Female



Σχήμα 6.4: Κατανομή αριθμού δειγμάτων ανά κλάση | Female Ease of Following

Male



Σχήμα 6.5: Κατανομή αριθμού δειγμάτων ανά κλάση | Male Ease of Following

Σημειώνεται πως στα παραπάνω αποτελέσματα, για τα female, απορρίφθηκε ένας σχολιαστής ο οποίος είχε μέση διαφωνία 1.08, ενώ για τα male απορρίφθηκαν δύο σχολιαστές που είχαν τιμές διαφωνιών 0.76 και 0.75.

Το κατώφλι της απόκλισης (σ) παρέμεινε και σε αυτήν την περίπτωση σταθερό (0.75), ενώ σε ο,τι αφορά τα κατώφλια μέσης τιμής, παρατηρείται μια πολύ πιο χαλαρή τιμή στην κλάση positive των male, απ' ότι στην κλάση positive των female, ενώ η κλάση negative των male φέρει μια λίγο πιο αυστηρή τιμή σε σχέση με την κλάση negative των female. Αυτό μας δείχνει ότι οι γυναίκες έχουν περισσότερα θετικά δείγματα ως προς την ευκολία παρακολούθησης.

Οπότε τελικά, για τα female καταλήγουμε με 44 δείγματα στην positive κλάση και με 54 στην negative. Ακούγοντας αυτές τις τελικές ταξινομήσεις, αφαιρέθηκαν 6 ακόμα δείγματα από την κλάση negative τα οποία δεν φαινόντουσαν να είναι αρκετά κακά ως προς την ευκολία παρακολούθησης. Επομένως, ο τελικός αριθμός δειγμάτων ανέρχεται στα 44 δείγματα στην κλάση positive και 48 στην κλάση negative.

Για τα male, τα δείγματα διατηρήθηκαν όπως ακριβώς προέκυψαν από τους παραπάνω περιορισμούς, δηλαδή 43 δείγματα για την κλάση positive και 33 για την κλάση negative.

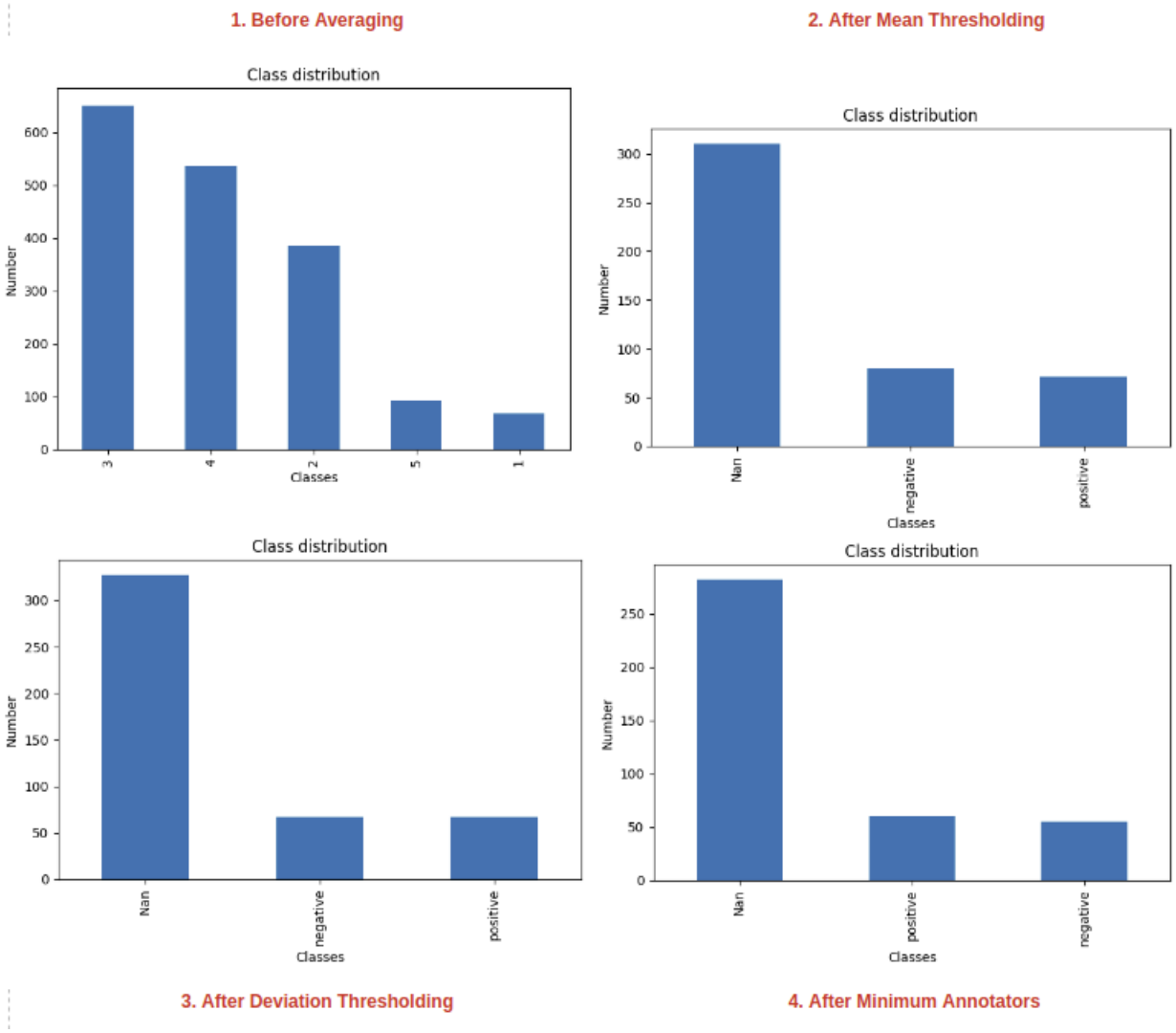
6.5.3 Enjoyment

Enjoyment				
	Female		Male	
	Positive	Negative	Positve	Negative
Mean Threshlod	μ>=4.0	μ<=2.4	μ>=3.2	μ<=2.0
Number of samples after Mean Thresholding	71	86	72	64
Deviation Threshold	σ<0.75	σ<0.75	σ<0.75	σ<0.75
Number of samples after Deviation Thresholding	67	73	54	58
Number of samples after Minimum Annotators	60	61	50	51
Average Disagreement	0.5		0.54	

Table 6.3: Καθορισμός του Dataset Απόλαυσης

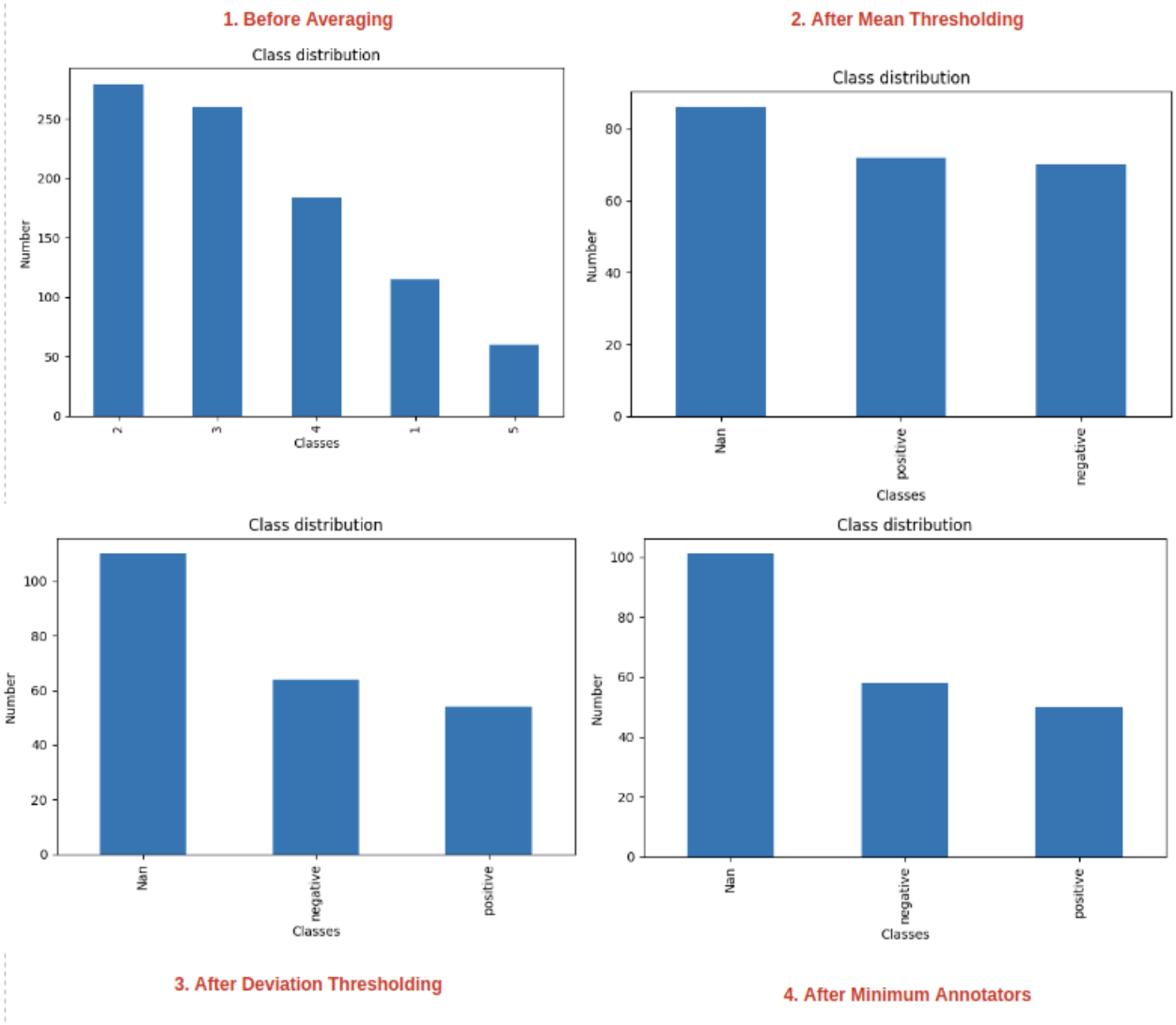
Στις παρακάτω εικόνες, φαίνεται πιο αναλυτικά η κατανομή του αριθμού των δειγμάτων ανά κλάση, 1. πριν εφαρμοστεί μέσος όρος επισημειώσεων, 2. μετά από την εφαρμογή κατωφλίου μέσης τιμής, 3. μετά την εφαρμογή κατωφλίου απόκλισης και τέλος 4. μετά την εφαρμογή ελάχιστου αριθμού σχολιαστών.

Female



Σχήμα 6.6: Κατανομή αριθμού δειγμάτων ανά κλάση | Female Enjoyment

Male



Σχήμα 6.7: Κατανομή αριθμού δειγμάτων ανά κλάση | Male Enjoyment

Σημειώνεται πως στα παραπάνω αποτελέσματα, για τα female, απορρίφθηκε ένας σχολιαστής ο οποίος είχε μέση διαφωνία 0.79 και για τα male ένας ο οποίος είχε διαφωνία 0.72.

Το κατώφλι της απόκλισης παραμένει και εδώ σταθερό στην τιμή 0.75, ενώ σε ότι αφορά τα κατώφλια της μέσης τιμής, παρατηρείται η ίδια συμπεριφορά με το προηγούμενο task της ευκολίας παρακολούθησης.

Τελικά καταλήγουμε για το female με 60 δείγματα στην κλάση positive και 61 στην κλάση negative, ενώ για το male με 50 δείγματα στην κλάση positive και 51 στην κλάση negative.

6.5.4 Free Text

Δημιουργήθηκαν ακόμα δύο μικρά-ενδεικτικά σύνολα δεδομένων, τα οποία συμπεριλαμβάνουν μόνο δείγματα που είναι ανεξάρτητα από κάποιο κείμενο αναφοράς, δηλαδή μόνο τις ομιλίες που προέκυψαν από απαντήσεις σε ερωτήσεις και όχι από κάποιο προκαθορισμένο κείμενο το οποίο ο ομιλητής διάβασε.

Ο λόγος για τον οποίο παράχθηκαν και αυτά τα σύνολα δεδομένων, είναι για να εξεταστούν πιθανές διαφορές στο κομμάτι του συστήματος που αφορά στην ανάλυση κειμένου. Είναι προφανές, πως τα κειμενικά χαρακτηριστικά που εξάγονται από διάφορες, ελεύθερου τύπου, ομιλίες, συσσωρεύουν πολύ περισσότερη πληροφορία, απ' ότι τα χαρακτηριστικά που εξάγονται από ομιλίες του ίδιου κειμένου αναφοράς.

Τα σύνολα δεδομένων αυτά δημιουργήθηκαν πάνω στα δύο tasks της εκφραστικότητας και της απόλαυσης.

Από το task της εκφραστικότητας (expressiveness), συνενώθηκαν τα female και τα male και τελικά επιλέχθηκαν μόνο τα δείγματα εξ αυτών, που προέρχονται από ελεύθερη ομιλία. Ο λόγος για τον οποίο έγινε συνένωση των δειγμάτων από τα δύο φύλα, είναι ο ελλιπής αριθμός δεδομένων.

Τελικά, από τα 88 (47 female, 41 male) δείγματα της κλάσης positive και τα 101 (53 female, 48 male) της κλάσης negative, αυτά που παρέμειναν καθώς ήταν ελεύθερες ομιλίες, ήταν **22** δείγματα στην κλάση **positive** και **16** στην κλάση **negative**. Φυσικά, είναι προφανές πως αυτός ο αριθμός δειγμάτων είναι μικρός για την εκπαίδευση ενός μοντέλου, παρ' όλα αυτά μπορεί να χρησιμοποιηθεί ως μια ένδειξη.

Αντίστοιχα, από το task της απόλαυσης (enjoyment), συνενώθηκαν και πάλι τα δείγματα από female και male που προέρχονται από ελεύθερη ομιλία.

Τελικά, από τα 110 (60 female, 50 male) δείγματα της κλάσης positive και τα 112 (61 female, 51 male) της κλάσης negative, αυτά που παρέμειναν καθώς ήταν ελεύθερες ομιλίες, ήταν **34** δείγματα στην κλάση **positive** και **21** στην κλάση **negative**.

Στον παρακάτω πίνακα φαίνεται η κατανομή δειγμάτων ανα κλάση, για αυτά τα δύο καινούρια σύνολα δεδομένων:

Free Text		
	Positive	**Negative**
Expressiveness	22	16
Enjoyment	34	21

Table 6.4: Κατανομή Δειγμάτων στα Free Text Datasets

7 Πειράματα και Αποτελέσματα

7.1 Πειράματα

Για τα τελικά πειράματα επιπέδου εγγραφής επιλέχθηκαν τα σύνολα δεδομένων της εκφραστικότητας και της απόλαυσης καθώς ήταν αυτά που είχαν την μικρότερη ασυμφωνία στις ετικέτες.

Για καθένα από τα tasks, διεξήχθησαν 8 διαφορετικών ειδών πειράματα, ως προς τον τύπο χαρακτηριστικών τα οποία χρησιμοποιήθηκαν στο καθένα. Τα είδη αυτά απαριθμούνται παρακάτω:

1. **Meta Audio (MA)**:
   Στα πειράματα αυτά χρησιμοποιήθηκαν τα χαρακτηριστικά ήχου τα οποία προέκυψαν από τους ταξινομητές τμημάτων (βλέπε 4.1.6), καθώς επίσης και τα υψηλού επιπέδου χαρακτηριστικά ήχου. Ο συνολικός αριθμός των χαρακτηριστικών αυτών είναι 20 και έχουν αναλυθεί στο κεφάλαιο 5.1. Σημειώνεται πως για την εξαγωγή χαρακτηριστικών τα οποία τροφοδοτούνται στα εκπαιδευμένα μοντέλα-ταξινομητές τμημάτων, χρησιμοποιήθηκαν μεσοπρόθεσμα παράθυρα (mid-term windows) των 3ων δευτερολέπτων, με βήμα 3 και βραχυπρόθεσμα παράθυρα (short-term windows) των 0.05 δευτερολέπτων, με βήμα 0.05, όπως ακριβώς έγινε και κατά την εκπαίδευση των μοντέλων αυτών.

2. **Text (T)**:
   Στα πειράματα αυτά χρησιμοποιήθηκαν τα χαρακτηριστικά κειμένου τα οποία προέκυψαν από τους ταξινομητές τμημάτων (βλέπε 4.2.5), καθώς επίσης και τα υψηλού επιπέδου χαρακτηριστικά κειμένου. Ο συνολικός αριθμός των χαρακτηριστικών αυτών είναι 22 και έχουν αναλυθεί στο κεφάλαιο 5.2. Σημειώνεται πως για την εξαγωγή χαρακτηριστικών τα οποία τροφοδοτούνται στα εκπαιδευμένα μοντέλα-ταξινομητές τμημάτων, χρησιμοποιήθηκαν ως τμήματα, προκαθορισμένης διάρκειας παράθυρα και συγκεκριμένα παράθυρα 3ων δευτερολέπτων. Η καλύτερη μέθοδος τμηματοποίησης που θα μπορούσε να εφαρμοστεί στο κείμενο, ίσως να ήταν η τμηματοποίηση ανά πρόταση, καθώς κάθε πρόταση είναι αυτοτελής και μπορεί να περιέχει την δική της πληροφορία. Παρ' όλα αυτά παρατηρήθηκε πως στα κείμενα που παράγονται από το google speech to text api, σπανίως εντοπίζονται σημεία στίξης και προτάσεις. Επομένως, η τμηματοποίηση που επιλέχθηκε εδώ ήταν τα χρονικά παράθυρα.

3. **Low Level Audio (LLA)**:
   Στα πειράματα αυτά χρησιμοποιήθηκαν χαμηλού επιπέδου χαρακτηριστικά ήχου. Πιο συγκεκριμένα, ως χαμηλού επιπέδου, ορίζονται τα χαρακτηριστικά εκείνα που χρησιμοποιήθηκαν για του ταξινομητές τμημάτων, δηλαδή αυτά που παρουσιάστηκαν στο κεφάλαιο 4.1.3. Πρόκειται για 136 χαρακτηριστικά ανά μεσοπρόθεσμο παράθυρο (μέσος όρος και τυπική απόκλιση των χαρακτηριστικών στα βραχυπρόθεσμα παράθυρα). Φυσικά, εφόσον οι τελικοί ταξινομητές αφορούν τον συνολικό ήχο/ομιλία, υπολογίζεται ο μακροπρόθεσμος μέσος όρος (long-term averaging), των χαρακτηριστικών στα μεσοπρόθεσμα παράθυρα και καταλήγουμε με 136 χαρακτηριστικά επιπέδου εγγραφής.

   Τα παράθυρα που χρησιμοποιήθηκαν και σε αυτήν την περίπτωση ήταν μεσοπρόθεσμα παράθυρα 3ων δευτερολέπτων, με βήμα 3 και βραχυπρόθεσμα παράθυρα (short-term windows) των 0.05 δευτερολέπτων, με βήμα 0.05.

   Ο λόγος για τον οποίο εκτελούνται αυτού του είδους τα πειράματα, είναι διότι η σύγκριση των αποτελεσμάτων τους με τα αποτελέσματα των πειραμάτων του βήματος 1 που παρουσιάστηκαν παραπάνω, θα είναι πολύ ενδιαφέρουσα και σημαντική. Είναι ιδιαίτερης σημασίας να δούμε κατά πόσο είναι ωφέλιμη η χρήση ταξ-

ινομητών τμημάτων και υψηλού επιπέδου χαρακτηριστικών έναντι της απευθείας χρήσης χαμηλού επιπέδου χαρακτηριστικών, για την ανάλυση της ομιλίας.

4. **MA + T**:
   Στα πειράματα αυτά χρησιμοποιούνται τα χαρακτηριστικά των βημάτων 1 και 2 μαζί, δηλαδή συνενωμένα. Κατ' ουσίαν, αυτού του είδους τα πειράματα συνδυάζουν τα δύο συστήματα (ήχου και κειμένου) (3.3), έτσι ώστε να αξιοποιηθεί η συνολική πληροφορία από τους ταξινομητές επιπέδου εγγραφής, για την εκτίμηση της ποιότητας της ομιλίας.

5. **Early Fusion of MA and LLA**:
   Στα πειράματα αυτά, γίνεται μια σύμπτυξη/ένωση των χαρακτηριστικών των βημάτων 1 και 3. Δηλαδή χρησιμοποιείται ένα διάνυσμα χαρακτηριστικών που περιέχει τόσα τα υψηλού επιπέδου χαρακτηριστικά όσο και τα χαμηλού επιπέδου χαρακτηριστικά ήχου.

6. **Late Fusion of MA and LLA**:
   Στην περίπτωση αυτή χρησιμοποιούνται και πάλι τα χαρακτηριστικά των βημάτων 1 και 3, αλλά αυτήν την φορά δεν συνενώνονται σε ένα κοινό διάνυσμα χαρακτηριστικών (early fusion). Αυτό που γίνεται είναι η χρήση ξεχωριστού ταξινομητή για τα χαρακτηριστικά τύπου 1 και ξεχωριστού ταξινομητή για τα χαρακτηριστικά τύπου 3 και στην συνέχεια η συνάθροιση των αποφάσεών τους. Πιο συγκεκριμένα, υπολογίζεται ο μέσος όρος από τις πιθανότητες που παράγουν οι δυο ταξινομητές και στην συνέχει επιλέγεται η κλάση της οποίας η μέση πιθανότητα είναι μεγαλύτερη.

7. **Early Fusion of MA+T and LLA**:
   Στην περίπτωση αυτή γίνεται μια συνένωση των χαρακτηριστικών των βημάτων 1,2 και 3 ή αλλιώς των βημάτων 3 και 4, σε ένα κοινό διάνυσμα χαρακτηριστικών (early fusion).

8. **Late Fusion of MA+T and LLA**:
   Σε αυτήν την περίπτωση χρησιμοποιούνται τα χαρακτηριστικά του βήματος 4 για την εξαγωγή απόφασης από έναν ταξινομητή, τα χαρακτηριστικά του βήματος 3 για την εξαγωγή απόφασης από έναν άλλο ταξινομητή και στην συνέχεια συναθροίζονται οι δυο αποφάσεις αυτές παίρνοντας τον μέσο όρο των πιθανοτήτων σε κάθε κλάση και επιλέγοντας την κλάση με την μεγαλύτερη μέση πιθανότητα.

Για την εκπαίδευση των μοντέλων, χρησιμοποιείται gridsearch ώστε να βρεθούν οι καλύτερες παράμετροι. Για τον αλγόριθμο svm rbf εξετάζονται οι εξής τιμές παραμέτρων: 'gamma': ['auto', 'scale'] και 'C': [0.001, 0.01, 0.5, 1.0, 5.0, 10.0, 20.0]. Για τον αλγόριθμο Gaussian Naive Bayes εξετάζονται οι εξής τιμές παραμέτρων: 'var smoothing': 100 ίσα διαχωρισμένα σημεία στο διάστημα $[1,10^{-9}]$ και τέλος για τον αλγόριθμο logistic regression οι εξής: 'penalty' : ['l1', 'l2'], 'C' : 20 ίσα διαχωρισμένα σημεία στο διάστημα $[10^{-4}, 10^{4}]$ και 'solver' : ['liblinear'].

Στο gridsearch εφαρμόζεται standardscaler για κανονικοποίηση των χαρακτηριστικών, ενώ ως cross validation μέθοδος χρησιμοποιείται το Leave One Group Out. Η συγκεκριμένη μέθοδος, χρησιμοποιεί ομάδες (groups) δειγμάτων που είναι ήδη χωρισμένες και σε κάθε επανάληψη, εκπαιδεύει το μοντέλο σε όλα τα groups εκτός από ένα, το οποίο χρησιμοποιείται για testing. Τα groups που χρησιμοποιούμε εμείς είναι χωρισμένα με βάση τον ομιλητή. Δηλαδή, κάθε group περιέχει τα δείγματα/ομιλίες ενός μόνο ομιλητή. Έτσι, ο συνολικός αριθμός διασπάσεων στο cross validation είναι ίσος με τον αριθμό των ομιλητών, αφού σε κάθε διάσπαση ένα καινούριο group-ομιλητής χρησιμοποιείται ως test set. Κατ' αυτόν τον τρόπο, γίνεται testing σε δείγματα από ομιλητές που το μοντέλο δεν έχει προηγουμένως ξαναδεί, κατά την εκπαίδευση. Καταφέρνουμε δηλαδή να υλοποιήσουμε μια

speaker independent αξιολόγηση, η οποία μας δίνει πιο ρεαλιστικά και σωστά αποτελέσματα, καθώς όπως έχουμε ξαναπεί η ομιλία είναι άμεσα εξαρτώμενη από τον ομιλητή και αν το μοντέλο τεσταριστεί σε γνώριμους ομιλητές, μπορεί να δώσει παραπλανητικά αποτελέσματα.

Για την επιλογή του καλύτερου μοντέλου, δηλαδή των παραμέτρων που δίνουν την καλύτερη απόδοση, υπολογίζουμε τον συγκεντρωτικό (aggregated) confusion matrix. Ως aggregated confusion matrix ορίζεται το confusion matrix που έχει προκύψει από όλες τις διασπάσεις του cross validation, δηλαδή αθροίζοντας τις ταξινομήσεις από κάθε test set. Έπειτα υπολογίζεται η f1 macro μετρική, αυτού του confusion matrix και επιλέγονται τελικά οι παράμετροι που έδωσαν την μέγιστη αυτή μετρική.

Τέλος, εφαρμόζεται ένα επιπλέον βήμα για τον υπολογισμό της μετρικής auc (area under the roc curve), καθώς και της γραφικής αναπαράστασης του roc curve. Συγκεκριμένα, αφού έχουν βρεθεί οι βέλτιστοι παράμετροι, εκτελείται ένα cross validation με σταθερές αυτές τις παραμέτρους, χρησιμοποιώντας και πάλι leave one group out, που στην περίπτωσή μας σημαίνει leave one speaker out, και συγκεντρώνονται όλες οι προβλεπόμενες πιθανότητες (predicted probabilities) , από κάθε test set. Από τις συγκεντρωμένες πιθανότητες αυτές, υπολογίζεται η μετρική auc και δημιουργείται το roc curve.

Στους παρακάτω πίνακες φαίνονται αναλυτικά κάποια πειράματα που έγιναν με χρήση διαφορετικών αλγορίθμων-ταξινομητών, από τις αποδόσεις των οποίων, επιλέχθηκε τελικά ο καλύτερος αλγόριθμος κατά περίπτωση. Σημειώνεται πως ως f1,cm και auc παρουσιάζονται τα aggregated f1, confusion matrix και auc που έχουν προκύψει από την συνάθροιση των test sets, όπως περιγράφηκε παραπάνω. Επίσης, στο confusion matrix ως 0 συμβολίζεται η κλάση negative και ως 1 η κλάση positive.

Female Expressiveness						
	SVM RBF		Gaussian Naive Bayes		Logistic Regression	
	MA	LLA	MA	LLA	MA	LLA
f1	53.9	51	57.8	56.3	**67**	**71**
auc	41.2	47.6	54.3	49.9	**71**	**77.3**
cm	0 1 0 29 24 1 22 25	0 1 0 24 29 1 20 27	0 1 0 21 32 1 9 38	0 1 0 22 31 1 12 35	**0 1 0 34 19 1 14 33**	**0 1 0 37 16 1 13 34**

Table 7.1: Female Expressiveness

Male Expressiveness						
	SVM RBF		Gaussian Naive Bayes		Logistic Regression	
	MA	LLA	MA	LLA	MA	LLA
f1	60.6	66.3	**74.7**	54.7	69.5	**76.4**
auc	57	63.5	**69.4**	67.6	74.1	**71**
cm	0 1 0 29 19 1 16 25	0 1 0 31 17 1 13 28	**0 1 0 40 8 1 14 27**	0 1 0 33 15 1 24 17	0 1 0 34 14 1 13 28	**0 1 0 35 13 1 8 33**

Table 7.2: Male Expressiveness

Female Enjoyment						
	SVM RBF		Gaussian Naive Bayes		Logistic Regression	
	MA	LLA	MA	LLA	MA	LLA
f1	56.2	54.7	50.6	44.5	**52**	**55.8**
auc	19.6	22.9	44.5	31.9	**44.3**	**57**
cm	0 1 0 21 40 1 10 50	0 1 0 23 38 1 15 45	0 1 0 19 42 1 15 45	0 1 0 14 47 1 16 44	**0 1 0 29 32 1 26 43**	**0 1 0 51 10 1 40 20**

Table 7.3: Female Enjoyment

Male Enjoyment						
	SVM RBF		Gaussian Naive Bayes		Logistic Regression	
	MA	LLA	MA	LLA	MA	LLA
f1	65.3	64.3	53.9	65.1	**59.3**	**64.3**
auc	18.9	17.6	56.3	61.5	**57.8**	**70.4**
cm	0 1 0 35 16 1 19 31	0 1 0 35 16 1 20 30	0 1 0 33 18 1 28 22	0 1 0 43 8 1 26 24	**0 1 0 32 19 1 22 28**	**0 1 0 30 21 1 15 35**

Table 7.4: Male Enjoyment

Από τα παραπάνω αποτελέσματα, θα επιλέξουμε τα μοντέλα που δίνουν την μεγαλύτερη απόδοση σε ένα συνδυασμό των μετρικών f1 και auc.

Με βάση τα παραπάνω, παρατηρούμε πως ο αλγόριθμος logistic regression είναι αυτός που δίνει στις περισσότερες περιπτώσεις (tasks/gender), τις καλύτερες αποδόσεις, τόσο για τα πειράματα MA όσο και για τα LLA. Σημειώνεται πως κατ' εξαίρεση για το task του male expressiveness για την περίπτωση του MA επιλέγεται ο αλγόριθμος Gaussian Naive Bayes, καθώς μπορεί να δίνει λίγο μικρότερο auc σε σχέση με το logistic regression, αλλά έχει καλύτερο f1. Φαίνεται πως άλλοι αλγόριθμοι όπως ο svm rbf είναι πολύ πολύπλοκοι για το πρόβλημα που πάμε να λύσουμε και έτσι χρειάζεται ένας πιο απλός συμβατικός αλγόριθμος. Επομένως, για τα επόμενα πιο αναλυτικά πειράματα επιλέγουμε να πορευθούμε με τους αλγορίθμους που είχαν καλύτερη απόδοση ανάλογα το task και τον τύπο χαρακτηριστικών (σε όλα logistic regression, εκτός από το male expressiveness MA που θα είναι Gaussian Naive Bayes).

Σημειώνεται πως για τις late fusion μεθόδους, ακολουθήθηκε και πάλι gridsearch με leave one speaker out, ξεχωριστά για τα χαρακτηριστικά MA ή MA + T και ξεχωριστά για τα χαρακτηριστικά LLA. Και για τις δύο παραπάνω περιπτώσεις χαρακτηριστικών, βρέθηκαν οι καλύτεροι παράμετροι για τον αλγόριθμο logistic regression. Στην συνέχεια, ακολουθήθηκε ένα cross validation με leave one speaker out, όπου σε κάθε διάσπαση (split) εκπαιδευόταν στο ίδιο train set τόσο το μοντέλο του MA (ή MA + T) όσο και το μοντέλο του LLA, με προκαθορισμένες τις καλύτερες παραμέτρους που βρέθηκαν προηγουμένως. Στο ίδιο test set, γινόταν επίσης πρόβλεψη και από τα δυο μοντέλα. Από τις προβλέψεις/πιθανότητες αυτές, των δύο μοντέλων, υπολογιζόταν ο μέσος όρος για κάθε δείγμα και στην συνέχεια επιλεγόταν η κλάση με την μέγιστη μέση πιθανότητα πρόβλεψης. Όλες αυτές οι τελικές προβλέψεις/κλάσεις, από όλες τις διασπάσεις/test sets, συναθροίστηκαν σε ένα κοινό (aggregated) confusion matrix. Επίσης, οι μέσες πιθανότητες, από όλα τα test sets, συνενώθηκαν μαζί σε ένα διάνυσμα, από το οποίο και προέκυψε η συγκεντρωτική τιμή auc (area under the curve), όπως και οι γραφικές παραστάσεις roc curves.

Παρακάτω παρουσιάζονται αναλυτικά οι μετρικές f1, area under the curve και confusion matrix, υπολογισμένες με τον τρόπο που περιγράφηκε παραπάνω, για κάθε task και για κάθε μέθοδο (τύπο χαρακτηριστικών) ξεχωριστά:

Female Expressiveness								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	67	56	71	65	67	69	66	68
auc	71	36.8	77.3	65.6	77	76	74.6	74.9
cm	0 1 0 34 19 1 14 33	0 1 0 45 8 1 32 15	0 1 0 37 16 1 13 34	0 1 0 34 19 1 16 31	0 1 0 35 18 1 15 32	0 1 0 35 18 1 13 34	0 1 0 35 18 1 16 31	0 1 0 35 18 1 14 33

Table 7.5: Female Expressiveness

Male Expressiveness								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	74.7	49.6	76.4	62.9	73.7	-	66.3	-
auc	69.4	40.6	71	71	74.9	-	78.9	-
cm	0 1 0 40 8 1 14 27	0 1 0 32 16 1 27 14	0 1 0 35 13 1 8 33	0 1 0 40 8 1 23 18	0 1 0 39 9 1 14 27	-	0 1 0 42 6 1 22 19	-

Table 7.6: Male Expressiveness

Ο λόγος για τον οποίο παρουσιάζονται πάβλες στον παραπάνω πίνακα στα πειράματα που αφορούν το early fusion, είναι επειδή χρησιμοποιούνται διαφορετικοί αλγόριθμοι-μοντέλα για το MA και το LLA αντίστοιχα, επομένως δεν έχει νόημα η συνένωση (fusion) των χαρακτηριστικών αυτών με την χρήση ενός κοινού αλγορίθμου.

Female Enjoyment								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	52	63.3	55.8	57.1	51	55.8	56.4	56.3
auc	44.3	64.8	57	50.7	43.8	57	51.2	62.3
cm	0 1 0 29 32 1 26 34	0 1 0 40 21 1 23 26	0 1 0 51 10 1 40 20	0 1 0 29 32 1 19 40	0 1 0 27 34 1 25 35	0 1 0 51 10 1 40 20	0 1 0 29 32 1 20 39	0 1 0 51 10 1 39 20

Table 7.7: Female Enjoyment

Male Enjoyment								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	58.2	59.9	64.3	61.9	59.3	64.3	66.3	65.1
auc	57.4	48.3	70.4	59.9	64.3	70.5	74	66.1
cm	0 1 0 33 18 1 24 26	0 1 0 19 32 1 6 44	0 1 0 30 21 1 15 35	0 1 0 37 14 1 24 26	0 1 0 32 19 1 22 28	0 1 0 30 21 1 15 35	0 1 0 36 15 1 19 31	0 1 0 34 17 1 21 29

Table 7.8: Male Enjoyment

Free Text								
Expressiveness								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	67.6	67.6	80.9	84	81.3	78.4	86.6	80.9
auc	74.7	64.8	86.7	91.2	86.7	84.4	93.5	86.1
cm	0 1 0 10 6 1 6 16	0 1 0 10 6 1 6 16	0 1 0 12 4 1 3 19	0 1 0 14 2 1 4 18	0 1 0 13 3 1 4 18	0 1 0 12 4 1 4 18	0 1 0 14 2 1 3 19	0 1 0 12 4 1 3 19

Table 7.9: Free Text Expressiveness

Free Text								
Enjoyment								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	62.1	57.7	47.7	81.1	65.3	65.3	55.3	70.1
auc	71.2	57.3	55.7	86.1	65.6	61.8	74.5	66.5
cm	0 1 0 12 9 1 11 23	0 1 0 13 8 1 14 20	0 1 0 6 15 1 11 23	0 1 0 17 4 1 6 28	0 1 0 12 9 1 9 25	0 1 0 11 10 1 10 24	0 1 0 9 12 1 11 23	0 1 0 14 7 1 11 23

Table 7.10: Free Text Enjoyment

7.2 Τελικά Αποτελέσματα

Εδώ θα γίνει μια σύνοψη των αποτελεσμάτων που είδαμε στους παραπάνω πίνακες, για όλα τα tasks μαζί, χρησιμοποιώντας μόνο την μετρική auc, ώστε να καταστεί πιο εύκολος ο σχολιασμός τους. Επιλέγουμε να βασιστούμε σε αυτήν την μετρική και όχι στο f1, καθώς τα δεδομένα είναι λίγα και το f1, μπορεί να μεταβάλλεται εύκολα, ακόμα και αν ένα από τα δείγματα ταξινομηθεί διαφορετικά. Αντίθετα, η μετρική auc είναι μια πιο ρεαλιστική τιμή που μας δείχνει πως τα πηγαίνουν τα μοντέλα και σε διαφορετικά operation points, δηλαδή διαφορετικά κατώφλια πιθανότητας. Επομένως, με αυτήν την μετρική έχουμε μια εικόνα για την απόδοση των ταξινομητών ανεξάρτητα από το συγκεκριμένο σημείο λειτουργίας (operation point).

	Individual Modalities			Fusion Methods				
	Meta Audio MA	Text T	Low Level Audio LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
Female Expressiveness	71	37	**77**	66	**77**	76	75	75
Male Expressiveness	69	41	**71**	71	75	-	**79**	-
Female Enjoyment	44	**65**	57	51	44	57	51	**62**
Male Enjoyment	57	48	**70**	60	64	70	**74**	66
Free Text Expressiveness	75	65	**87**	91	87	84	**93**	86
Free Text Enjoyment	**71**	57	56	**86**	66	62	75	67

Table 7.11: Τελικές Αποδόσεις (μετρική auc)

7.2.1 Σχολιασμός

Στον παραπάνω πίνακα εμφανίζονται αριστερά τα ατομικά πειράματα, δηλαδή τα πειράματα τα οποία χρησιμοποιούν έναν τύπο χαρακτηριστικών, χωρίς να γίνεται κάποια συνένωση, ενώ δεξιά εμφανίζονται τα πειράματα με όλες τις μεθόδους συγχώνευσης χαρακτηριστικών που παρουσιάστηκαν παραπάνω. Και στις δυο περιπτώσεις, σκιαγραφείται με πιο έντονα γράμματα (bold), η μεγαλύτερη απόδοση.

Αρχικά, βλέπουμε πως σε όλες τις περιπτώσεις (σειρές), η καλύτερη fusion μέθοδος, έχει αυξήσει την απόδοση σε σχέση με την καλύτερη individual μέθοδο ή στην χειρότερη περίπτωση την έχει διατηρήσει ίδια. Εξαίρεση αποτελεί μόνο το Female Enjoyment στο οποίο υπάρχει μια μικρή επιδείνωση των 3ων μονάδων (65-62=-3), η οποία όμως μπορεί να θεωρηθεί αμελητέα. Αντίθετα, στο male expressiveness βλέπουμε πως η απόδοση έχει ανέβει κατά 8 μονάδες, με την χρήση σύμπτυξης (fusion), στο male enjoyment κατά 4 και στο free text expressiveness κατά 6, ενώ στο free text enjoyment έχει αυξηθεί κατά 15 μονάδες, μια πολύ σημαντική διαφορά.

Μια άλλη σημαντική παρατήρηση είναι πως η συνένωση των meta audio με τα text χαρακτηριστικά (MA + T), φαίνεται να έχει αυξήσει την απόδοση σε σχέση με τα individual MA και individual T, στα tasks του male expressiveness και male enjoyment. Αυτή η βελτίωση γίνεται πολύ πιο έντονη στα free text tasks, όπου βλέπουμε η συνένωση MA + T, να έχει αυξήσει την απόδοση κατά 16 στο Expressiveness (σε σχέση με το ατομικό MA) και κατά 15 στο Enjoyment (σε σχέση με το ατομικό MA). Το γεγονός αυτό, δείχνει

πως η κειμενική πληροφορία μπορεί να βοηθήσει σημαντικά το μοντέλο στο να διακρίνει τις δύο κλάσεις (negative, positive), αφού με μια απλή συνένωση των δυο τύπων χαρακτηριστικών (audio + text), η απόδοση αυτομάτως αυξήθηκε. Φυσικά, ήταν αναμενόμενο το φαινόμενο αυτό να παρουσιαστεί κυρίως στα πειράματα που αφορούν το ελεύθερο κείμενο, καθώς τότε το κείμενο διαφοροποιείται από δείγμα σε δείγμα και δεν είναι πλέον προκαθορισμένο.

Επιπροσθέτως, παρατηρείται στις περισσότερες περιπτώσεις (4 από τις 6 περιπτώσεις), η συνένωση της πληροφορίας από όλα τα χαρακτηριστικά (meta audio, text και LLA) να είναι αυτή που δίνει το βέλτιστο αποτέλεσμα. Επίσης, τις περισσότερες φορές (4 στις 6 φορές), το late fusion τα πηγαίνει καλύτερα από το early fusion. Αυτό δείχνει πως το late fusion εισάγει έναν παράγοντα κανονικοποίησης (normalization), ο οποίος λείπει από την μέθοδο early fusion. Για παράδειγμα, το πλήθος των χαρακτηριστικών είναι κάτι που μπορεί να επηρεάσει την αξιοποίησή τους. Έτσι, αν η LLA έχει πολύ παραπάνω χαρακτηριστικά από την MA, αυτό πιθανόν να σημαίνει πως κάποια καλά χαρακτηριστικά από τα MA θα "χαθούν" μέσα στον "συνωστισμό" και δεν θα αξιοποιηθούν σωστά. Αντίθετα, ένα τέτοιο γεγονός μπορεί να αποφευχθεί με το late fusion, όπου οι δυο τύποι χαρακτηριστικών αξιοποιούνται από ξεχωριστά μοντέλα και η συνένωση των αποφάσεων γίνεται στο τέλος (late).

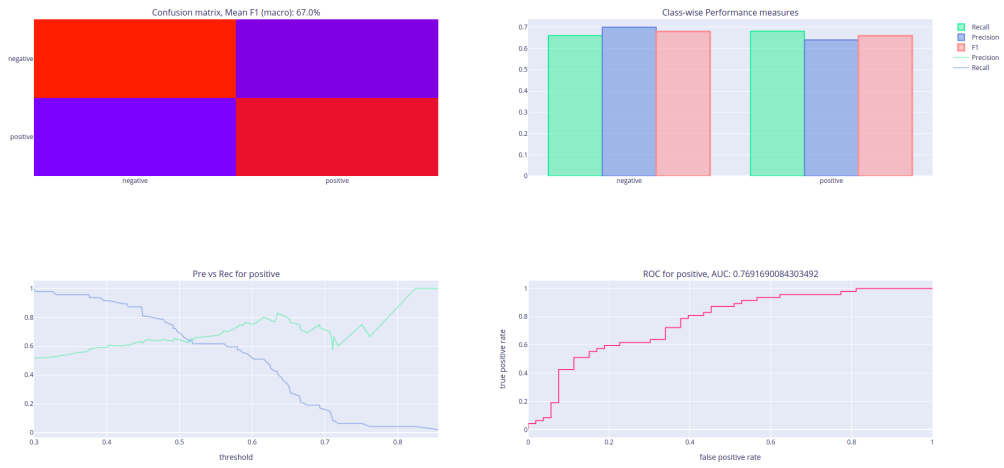
Μια τελευταία παρατήρηση που θα μπορούσε να αναφερθεί είναι πως στο free text enjoyment task, τα χαρακτηριστικά MA + T, τα πηγαίνουν καλύτερα από οποιαδήποτε άλλη συνένωση περιέχει και την LLA πληροφορία. Έτσι, φαίνεται τα low level χαρακτηριστικά ήχου (LLA) όπως είναι τα MFCCs ή το Chroma, να μην δίνουν πληροφορία σε ό,τι αφορά την ποιότητα μιας ομιλίας, σε αντίθεση με τα high level χαρακτηριστικά ήχου και κειμένου, τα οποία έχουν προκύψει από εσωτερικούς ταξινομητές συναισθήματος, που φαίνεται να είναι πολύ σημαντικά στην ανάλυση της ομιλίας.

7.2.2 Γραφικές Παραστάσεις

Εδώ παρουσιάζονται οι γραφικές παραστάσεις των τελικών αποτελεσμάτων, οι οποίες περιλαμβάνουν το συγκεντρωτικό confusion matrix, την απόδοση ανά κλάση, το precision vs recall και το roc curve για την positive κλάση, δίνοντας μια πιο ολοκληρωμένη οπτική.

Female Expressiveness

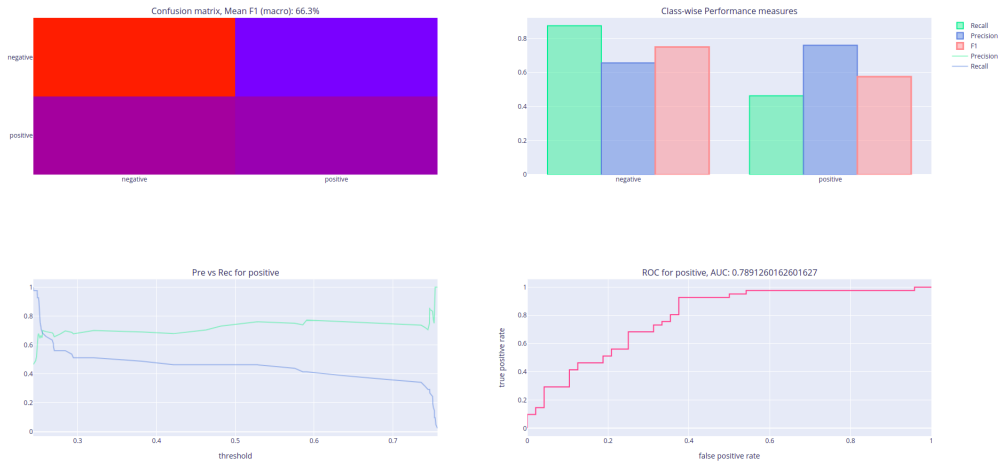
Σε αυτό το task το καλύτερο αποτέλεσμα φάνηκε να είναι στην μέθοδο **MA and LLA Late Fusion** (ίδιο με το ατομικό LLA). Επομένως, παρουσιάζονται οι γραφικές απεικονίσεις από αυτήν την μέθοδο:



Σχήμα 7.1: Female Expressiveness | Καλύτερα Αποτελέσματα

Male Expressiveness

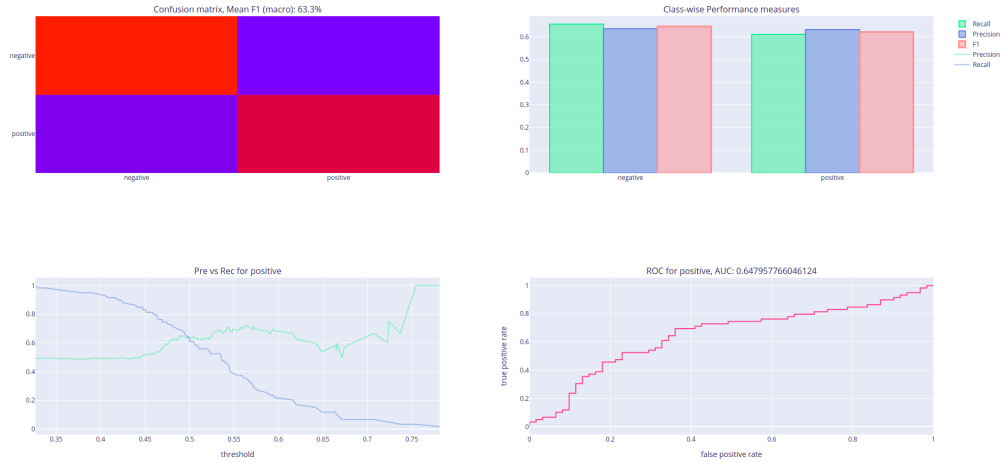
Σε αυτό το task το καλύτερο αποτέλεσμα φάνηκε να είναι στην μέθοδο **MA + T and LLA Late Fusion**. Επομένως, παρουσιάζονται οι γραφικές απεικονίσεις από αυτήν την μέθοδο:



Σχήμα 7.2: Male Expressiveness | Καλύτερα Αποτελέσματα

Female Enjoyment

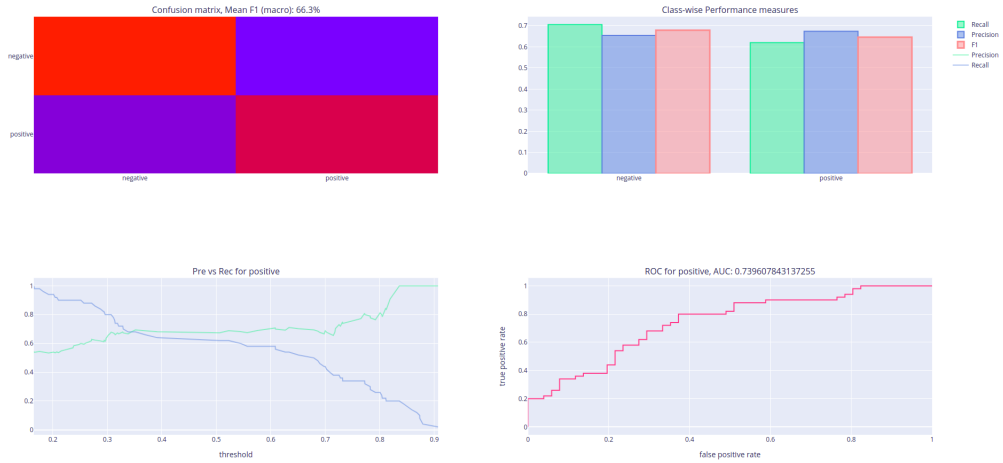
Σε αυτό το task το καλύτερο αποτέλεσμα φάνηκε να είναι στην μέθοδο **T**. Επομένως, παρουσιάζονται οι γραφικές απεικονίσεις από αυτήν την μέθοδο:



Σχήμα 7.3: Female Enjoyment | Καλύτερα Αποτελέσματα

Male Enjoyment

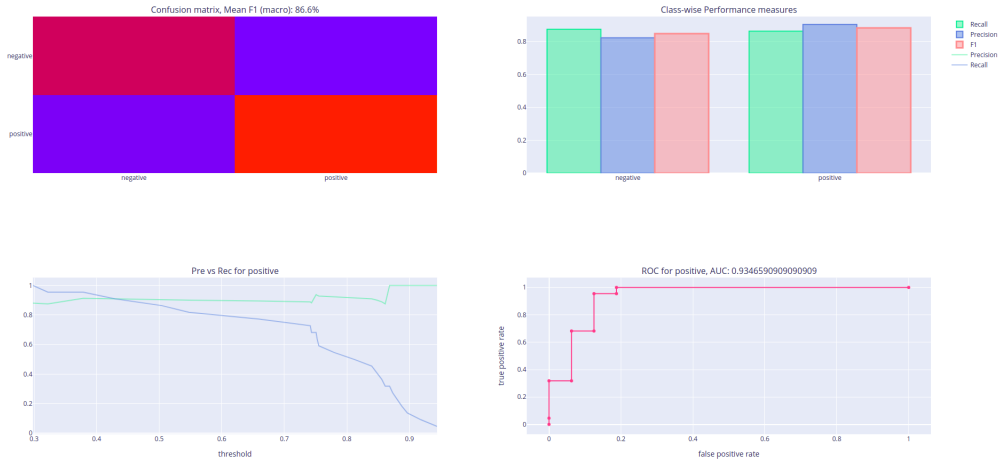
Σε αυτό το task το καλύτερο αποτέλεσμα φάνηκε να είναι στην μέθοδο **MA + T and LLA Late Fusion**. Επομένως, παρουσιάζονται οι γραφικές απεικονίσεις από αυτήν την μέθοδο:



Σχήμα 7.4: Male Enjoyment | Καλύτερα Αποτελέσματα

Free Text Expressiveness

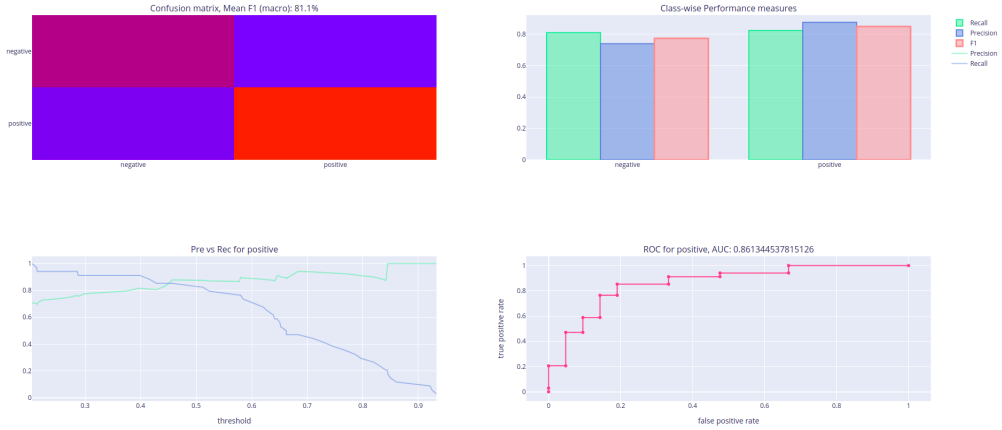
Σε αυτό το task το καλύτερο αποτέλεσμα φάνηκε να είναι στην μέθοδο **MA + T and LLA Late Fusion**. Επομένως, παρουσιάζονται οι γραφικές απεικονίσεις από αυτήν την μέθοδο:



Σχήμα 7.5: Free Text Expressiveness | Καλύτερα Αποτελέσματα

Free Text Enjoyment

Σε αυτό το task το καλύτερο αποτέλεσμα φάνηκε να είναι στην μέθοδο **MA + T**. Επομένως, παρουσιάζονται οι γραφικές απεικονίσεις από αυτήν την μέθοδο:



Σχήμα 7.6: Free Text Enjoyment | Καλύτερα Αποτελέσματα

8 Συμπεράσματα

8.1 Ανακεφαλαίωση Συστήματος

Ανακεφαλαιώνοντας, δημιουργήθηκε ένα σύστημα ανάλυσης ομιλίας. Το σύστημα αυτό χωρίστηκε σε δύο επίπεδα: την ανάλυση σε επίπεδο τμήματος και την ανάλυση σε επίπεδο εγγραφής. Τα δύο επίπεδα αυτά εφαρμόστηκαν τόσο στην πληροφορία ήχου όσο και στην πληροφορία κειμένου της ομιλίας. Για το πρώτο επίπεδο, εκπαιδεύτηκαν 3 ταξινομητές τμημάτων με άξονες το συναίσθημα, την διέγερση και το σθένος αντίστοιχα. Τα χαρακτηριστικά που χρησιμοποιήθηκαν για τους ταξινομητές αυτούς είναι φασματικά και χρονικά χαρακτηριστικά στην περίπτωση του ήχου και bert embeddings στην περίπτωση του κειμένου. Οι αποφάσεις που εξάγουν αυτοί οι ταξινομητές είναι τμηματικές, δηλαδή αφορούν κάθε παράθυρο/τμήμα ξεχωριστά. Για την ανάλυση σε επίπεδο εγγραφής, οι προαναφερθείσες αποφάσεις συμπτύχθηκαν με χρήση μέσου όρου ανά κλάση και έτσι δημιουργήθηκαν τα aggregated posteriors, που στην πράξη είναι οι πιθανότητες ή αλλιώς τα ποσοστά που υποδηλώνουν κατά πόσο η συνολική ομιλία ανήκει στην κάθε κλάση (συναισθήματος, διέγερσης και σθένους). Αυτές οι συνεπτυγμένες αποφάσεις, μαζί με την προσθήκη κάποιων ακόμα high level χαρακτηριστικών της συνολικής ομιλίας, χρησιμοποιήθηκαν για την εκπαίδευση των τελικών μοντέλων εκτίμησης ποιότητας.

Για τα τελικά μοντέλα, ακολουθήθηκε διαδικασία συλλογής και επισημείωσης δεδομένων, καθώς επίσης και συμφωνίας των επισημειώσεων. Συλλέχθηκαν δύο ειδών ομιλίες, αυτές που βασίζονται σε προκαθορισμένο κείμενο και αυτές με ελεύθερο κείμενο. Στην συνέχεια, έγινε επισημείωση αυτών, από διαφορετικούς σχολιαστές πάνω σε τρεις άξονες: την εκφραστικότητα, την ευκολία παρακολούθησης και την απόλαυση. Ακολούθησε η συμφωνία των επισημειώσεων με τρεις περιορισμούς: την μέση τιμή, την απόκλιση και τον αριθμό σχολιαστών ανά δείγμα. Από την συμφωνία αυτή προέκυψαν δύο κλάσεις για καθέναν από τους τρεις άξονες: negative και positive, ενώ παράχθηκαν κάποια στατιστικά όπως είναι η μέση διαφωνία. Εν τέλη, επιλέχθηκε να διατηρηθούν μόνο οι άξονες της εκφραστικότητας και της απόλαυσης, καθώς ο άξονας της ευκολίας παρακολούθησης είχε πιο μεγάλη μέση διαφωνία σχολιαστών.

Τελικά, διενεργήθηκαν πειράματα, με τα παραπάνω δεδομένα, πάνω σε 3 ατομικούς τύπους χαρακτηριστικών: τα χαρακτηριστικά ήχου επιπέδου εγγραφής όπως περιγράφηκαν παραπάνω (MA), τα αντίστοιχα χαρακτηριστικά κειμένου επιπέδου εγγραφής (T) και τέλος κάποια low level χαρακτηριστικά ήχου (LLA) τα οποία είναι τα ίδια φασματικά και χρονικά χαρακτηριστικά ήχου που χρησιμοποιήθηκαν για τους ταξινομητές τμημάτων, υπολογισμένα όμως αυτήν την φορά κατά μέσο όρο για όλη την ομιλία (long term averaging). Επιπλέον, διεξήχθησαν πειράματα με συνδυασμούς των παραπάνω τύπων χρησιμοποιώντας είτε early fusion, δηλαδή πρώιμη συνένωση χαρακτηριστικών, είτε late fusion δηλαδή μεταγενέστερη συνένωση αποφάσεων από ξεχωριστά μοντέλα.

8.2 Σχολιασμός και Συμπεράσματα

Το σύστημα που αναπτύχθηκε για την εκτίμηση της ποιότητας της ομιλίας, φαίνεται να είναι λειτουργικό. Τα αποτελέσματα που είδαμε στο προηγούμενο κεφάλαιο είναι αρκετά ικανοποιητικά, ενώ οι προτεινόμενες fusion μέθοδοι, φαίνεται να είναι μια καινοτομία η οποία βελτιώνει αρκετά το αποτέλεσμα.

Σημαντικό είναι να αναφέρουμε πως το σύστημα αυτό μπορεί να χρησιμοποιηθεί για οποιαδήποτε διεργασία ανάλυσης της ομιλίας, πέρα από την εκτίμηση της ποιότητάς της, όπως είναι για παράδειγμα η αναγνώριση μαθησιακών δυσκολιών, δυσλεξίας ή αυτισμού, η ανάκτηση σημάτων από τηλεφωνικά κέντρα κ.α.

Παρά το γεγονός ότι η ανάλυση της ομιλία γενικότερα, αλλά και η αναγνώριση του συναισθήματος από την ομιλία ειδικότερα, είναι ένα δύσκολο πρόβλημα, καθώς εξαρτάται

από πάρα πολλούς παράγοντες όπως είναι οι συνθήκες, ο ομιλητής, το φύλο, το σύνολο δεδομένων κ.α., το σύστημα που αναπτύχθηκε στην παρούσα διπλωματική εργασία και τα πειράματα που εκτελέστηκαν, έγιναν σε όσο το δυνατόν πιο αμερόληπτες και ανεξάρτητες συνθήκες. Έτσι, τα αποτελέσματα που παρουσιάστηκαν είναι ρεαλιστικά.

Αξιοσημείωτο είναι το γεγονός ότι τα υψηλού επιπέδου χαρακτηριστικά, όπως αυτά που εξήχθησαν από τους ταξινομητές τμημάτων και αφορούν το συναίσθημα, συνήθως έχουν περισσότερο νόημα και προσδίδουν περισσότερη πληροφορία σε σχέση με πιο απλά, χαμηλού επιπέδου χαρακτηριστικά όπως είναι για παράδειγμα τα MFCCs. Επιπλέον, τα χαρακτηριστικά αυτά δίνουν την δυνατότητα επεξηγηματικότητας (explainability), καθώς είναι ικανά να εξηγήσουν το αποτέλεσμα. Για παράδειγμα, αν κάποιος ομιλητής παρουσιάζει χαμηλή εκφραστικότητα, αυτό πιθανώς να συνδέεται με το μεγάλο ποσοστό "χαμηλής διέγερσης" που παρουσίασε μέσα στην ομιλία του. Έτσι, μπορεί να υπάρξει αιτιολόγηση των αποτελεσμάτων και με αυτόν τον τρόπο, ο ομιλητής να καταλάβει τι πρέπει να βελτιώσει ή σε άλλη περίπτωση να κατανοήσει τι είναι αυτό που κάνει καλά και του προσδίδει καλή ποιότητα ομιλίας. Κάτι τέτοιο δεν μπορεί να συμβεί με χαμηλού επιπέδου χαρακτηριστικά τα οποία δεν είναι επεξηγηματικά και άμεσα εκμεταλλεύσιμα.

Τέλος, σημαντικό κομμάτι έρευνας είναι και η δημιουργία συνόλου δεδομένων από ομιλίες, καθώς επίσης και η επισημείωση αυτών, η οποία διενεργήθηκε με αμερόληπτο και αντικειμενικό τρόπο. Η παρασκευή ενός επιβλεπόμενου συνόλου δεδομένων (που περιέχει ετικέτες), είναι ένα εξίσου σημαντικό κομμάτι της επιστήμης των δεδομένων και της μηχανικής μάθησης.

8.3 Εκτέλεση

Ο ανοιχτός κώδικας του συστήματος της παρούσας διπλωματικής εργασίας, δηλαδή ενός speech analytics python εργαλείου για αξιολόγηση ποιότητας ομιλίας, μπορεί να βρεθεί στο github [106].

8.4 Μελλοντικές Επεκτάσεις

Μελλοντικές επεκτάσεις και αλλαγές που θα μπορούσαν να γίνουν στο σύστημα της παρούσας εργασίας είναι:

- Η προσαρμογή τομέα (domain adaptation), ως προς τα tasks του συναισθήματος, της διέγερσης και του σθένους. Τα μοντέλα αυτά που εκπαιδεύτηκαν στο σύστημά μας, χρησιμοποιήσαν ανοιχτού τύπου δεδομένα χωρίς να έχουν έρθει σε επαφή με παρόμοια δεδομένα με αυτά που τελικά χρησιμοποιούνται για την αξιολόγηση της ποιότητας της ομιλίας (αυτά που προέκυψαν από την συλλογή των δεδομένων). Για αυτόν τον λόγο μπορεί να γίνει μια προσαρμογή στον τομέα των συγκεκριμένων δεδομένων, είτε με το να εισάγουμε στην εκπαίδευση των μοντέλων αυτών, κάποια παρόμοια δείγματα με αυτά που τελικά χρησιμοποιούμε, είτε να ακολουθήσουμε μη-πιβλεπόμενη μέθοδο, όπου εισάγουμε την πληροφορία των δεδομένων αυτών σε επίπεδο χαρακτηριστικών, ώστε η κανονικοποίηση των χαρακτηριστικών να λαμβάνει υπόψιν το πραγματικό εύρος. Με αυτόν τον τρόπο θα μπορούσαν να αποφευχθούν φαινόμενα όπως το να αναπαριστάται ένα δείγμα θετικής εκφραστικότητας ως "θυμός" ως προς το συναίσθημα ή ως "αρνητικό" ως προς το σθένος, λόγω παρερμήνευσης των χαρακτηριστικών του.

- Η εισαγωγή πιο ισχυρών μοντέλων-ταξινομητών τμημάτων, τα οποία μπορούν να αυξήσουν την απόδοση. Αναφερόμαστε σε μεθόδους βαθιάς μηχανικής μάθησης, όπως είναι τα CNNs, τα LSTMs και οι Transformers. Επιπλέον, σημαντική είναι η μελέτη που μπορεί να γίνει για το πως τελικά η βελτίωση της απόδοσης

των ταξινομητών τμημάτων, επηρεάζει την απόδοση των τελικών μοντέλων επιπέδου εγγραφής (long-term recording level models). Για παράδειγμα αν ένας ταξινομητής συναισθήματος αυξήσει την απόδοσή του κατά 5% στο επίπεδο τμημάτων, πόσο καλύτερα αποτελέσματα θα δώσει στο επίπεδο εγγραφής.

- Η χρήση νευρωνικού δικτύου (LSTM ή άλλου ακολουθιακού μοντέλου) στο κομμάτι του επιπέδου εγγραφής, δηλαδή έπειτα από τα posteriors των ταξινομητών τμημάτων. Με αυτόν τον τρόπο θα μπορούσαμε να εκμεταλλευτούμε την διαδοχική πληροφορία, καθώς δουλεύουμε με τμήματα/παράθυρα τα οποία έχουν μια χρονική διαδοχή.

- Η επέκταση των πειραμάτων επιπέδου εγγραφής σε περισσότερα επισημειωμένα δεδομένα, για μια καλύτερη γενίκευση του αποτελέσματος, καθώς διατηρήθηκαν λίγα δεδομένα έπειτα από την συνάθροιση/συμφωνία των επισημειώσεων.

- Η εφαρμογή μεταφοράς γνώσης από μη επιβλεπόμενα χρονικά (temporal) μοντέλα. Σε αυτήν την περίπτωση αποθηκεύονται γνώσεις που αποκτήθηκαν κατά την επίλυση προβλημάτων σε μη επισημειωμένα δεδομένα/δείγματα, με χρήση διαδοχικής πληροφορίας και αντίστοιχων μοντέλων (π.χ. LSTM) και εφαρμόζονται οι γνώσεις αυτές στο συγκεκριμένο πρόβλημα που επιλύουμε.

- Η εφαρμογή μεθόδων μάθησης που λαμβάνουν υπόψιν τους την εμπιστοσύνη των επισημειώσεων (annotation confidence), δηλαδή το πόσο έγκυρη είναι η επισημείωση του κάθε δείγματος, η οποία μπορεί να οριστεί και ως η συμφωνία μεταξύ των επισημειώσεων.

CHAPTER 1

Introduction

In this Chapter, the meaning and significance of public speaking is underlined. The motivation for the diploma thesis is discussed. Moreover, the objective and the outline of the different chapters of this thesis are presented.

1.1 Speech

Public speaking (also called oratory or oration) is giving speech face to face to live audience. However, due to the evolution of public speaking, it is modernly viewed as any form of speaking (formally and informally) between an audience and the speaker. Traditionally, public speaking was considered to be a part of the art of persuasion. The act can accomplish particular purposes including information, persuasion, and entertainment. Additionally, differing methods, structures, and rules can be utilized according to the speaking situation.

So when we talk about public speaking we are not only referring to the traditional form between audience and speaker but also to the general everyday human speech that includes 2 or more interlocutors.

Public speaking was developed in Rome and Greece. Prominent thinkers from these lands influenced the development and evolutionary history of public speaking. Currently, technology continues to transform the art of public speaking through newly available technology such as videoconferencing, multimedia presentations, and other nontraditional forms. Knowing when speech is most effective and how it is done properly are key to understanding the importance of it.

Speech for business and commercial events is often done by professionals. These speakers can be contracted independently, through representation by a speakers bureau, or by other means. Speech plays a large role in the professional world. In fact, it is believed that 70 percent of all jobs involve some form of public speaking. [1]

Oral communication is recognized as one of the most highly valued skills for success in the workplace (cf. Kyllonen, 2012) and school. Speech is widely known to be the most feared form of oral communication (Pull, 2012) but is often overlooked in educational assessment. A successful speaking performance is distributed across multiple axes — e.g., the speech content, voice and intonation, facial expressions, head poses, hand gestures, and body postures.

While most current rubrics for evaluating speech performance attend to both verbal and non-verbal aspects, virtually all existing assessments in practice require human rating (Ward,

2013; Schreiber, Paul and Shibley, 2012; Carlson and Smith-Howell, 1995). [2]

1.2 Motivation

We, as young electrical and computer engineers, are obligated to keep track of the technological tendencies taking place in our field and train ourselves to be able to contribute toward state-of-the-art solutions.

Machine learning is a large field, quite up-to-date in the modern technological age, which could not be more suited to our field of study.

The urgent need to use our speech in our daily lives, is the one that requires its evaluation and its improvement. This evaluation becomes easier and faster if it is done by a machine and not by the person himself. Thus the evaluation of the quality of speech through machine learning is the only way.

Speech is everywhere and the way we speak is just as important as what we say. Thus, the speech analytics and even the multimodal speech analytics, are an important process that falls into various types of problems, not only in assessing its quality, but also in the identification of learning disabilities (dyslexia, autism) , in the retrieval of signals from telephone exchanges, etc.

The aforementioned reasons motivated the research and writing down of this diploma thesis. It seems of value to young professionals to familiarize themselves with concepts that appear to be the future in the fields of technology, programming and automation.

1.3 Objective

The aim of this dissertation is first and foremost the familiarization and in-depth study of both machine learning and deep learning techniques, the use of these techniques in real issues and data and the emergence of beneficial results. In addition, this thesis aims to highlight solutions that could be used to solve other problems that require similar techniques, that is, other speech signal processing applications.

More specifically, in this dissertation, the issue of speech evaluation is investigated, so that the machine understands when a speaker has good speech skills and when he lags behind and has room for improvement. The importance of assessing the quality of speech becomes apparent, a process that is habitually carried out by humans and whose automation through machine learning methods is essential.

It is of course important to mention that speech signal analysis, such as recognizing the emotion of speech, is a difficult task, as speech is directly dependent on many factors such as the recording conditions, the content, the data and the way in which it was annotated, the speakers, the gender etc. These dependencies are called to be dealt with in the present dissertation.

The techniques or otherwise the system developed in the present dissertation, could be also applied in other issues of speech analysis, appart from evaluating its quality, as mentioned above.

1.4 Outline

This diploma thesis is organized as follows. **Chapter 2** provides a detailed theoretical background necessary for understanding the field of artificial intelligence and machine learning, as well as the methods to be used . **Chapter 3** will analyze the system architecture, both in terms of audio and text, as well as their combination. **Chapter 4** describes segment-level analysis for both audio and text, which includes segmentation, feature extraction,

and training of segment-level models . **Chapter 5** the recording level is analyzed, both in the audio and text part, which includes the recording level features. **Chapter 6** describes the data collection and annotation process. **Chapter 7** lists the final experiments for the recording level models and their results. Finally, **Chapter 8** presents the conclusions of this dissertation.

CHAPTER 2

Theoretical Background

In this Chapter, an extensive theoretical background on artificial intelligence and machine learning will be carried out. In essence, this will provide a helpful tool for the reader to understand the methods used in this diploma thesis.

2.1 Artificial Intelligence

2.1.1 Definition

Artificial Intelligence (AI) is a field of information technology designing and developing systems that mimic human behavior. Artificial intelligence is the ability of machines to think like the human brain. The difference between the latter and the former is that the human brain contains consciousness and emotions. Usually, the term "artificial intelligence" is used to describe machines (or computers) that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving". Machines understand their environment and solve problems, acting in order to achieve a specific goal.

Artificial Intelligence is continuously evolving to benefit many different industries. Machines are wired using a cross-disciplinary approach based on mathematics, computer science, linguistics, psychology, and more.

The image most people have in mind when they think about artificial intelligence is robots that look like humans in shape and construction. But the reality is far from it.

The term artificial intelligence does not necessarily refer to robots but to machines that learn to mimic human intelligence and perform tasks.

2.1.2 Applications

Artificial intelligence can be applied to many different areas such as the healthcare industry for drug dosing and surgery.

Other known examples of the use of artificial intelligence is the computers that play chess and the motor cars. Each of these machines must be performed taking into account the consequences of its actions in order to achieve the desired result. For example, the desired result in chess is to win the game, while in cars any collision must be avoided and the necessary regulations (such as the traffic code) must be observed, with the ultimate goal of the machine reaching its destination.

Artificial intelligence also has applications in the financial industry, where it is used to detect and flag activity in banking and finance such as unusual debit card usage and large account deposits—all of which help a bank's fraud department. Applications for AI are also being used to help streamline and make trading easier. This is done by making supply, demand, and pricing of securities easier to estimate.

2.1.3 Categorization

Artificial intelligence is divided into two different categories: weak and strong. Weak artificial intelligence refers to systems designed to perform a specific function. Such systems are video games such as the example of chess mentioned above, the personal-electronic assistants that you ask a question and they answer you, image analysis software, search engines, face and speech recognition systems, etc.

Strong artificial intelligence refers to systems that perform functions that are considered to be human-like. These functions are more difficult and complicated. The systems are programmed to handle situations and solve problems without human intervention. Such systems are robots, autonomous cars, drones, the Internet of Things, etc.

2.1.4 Moral Issues

A common thought regarding artificial intelligence is that machines will evolve to such an extent that humans cannot keep up and that they will redesign themselves exponentially.

Another issue is that machines can violate people's privacy, as well as how many and what rights a machine can have. In this case we are talking about ethical issues. For example, in the case of the self-driving car, if the car is faced with a dilemma where it has to decide who has to live and who has to die, what would be its choice? Imagine this scenario: the brakes fail in a self-driving car as it leads to a busy pedestrian crossing. A homeless man and a criminal pass in front of the car. Two cats are in the opposite lane. Should the car spin to kill the cats or collide with the two people?

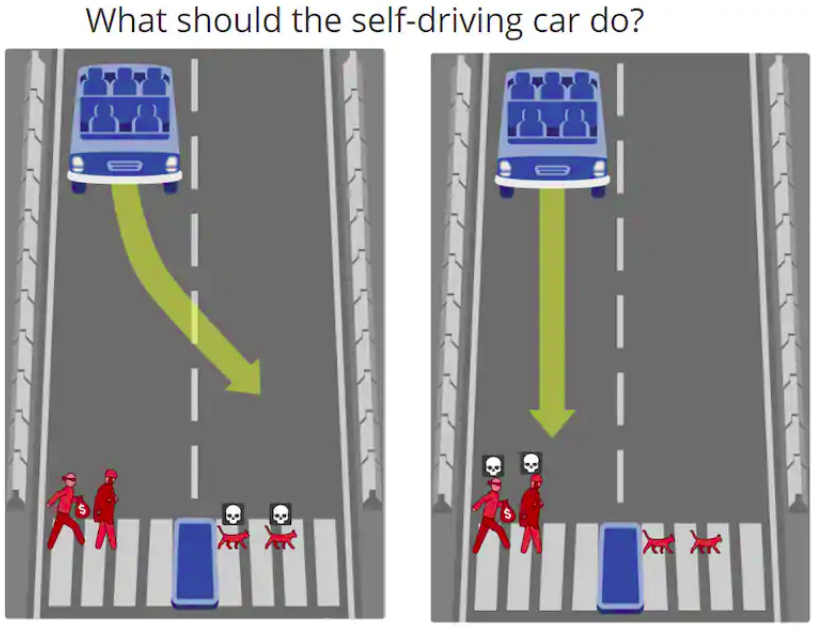


Figure 2.1: Example of an ethical issue of Artificial Intelligence. What should the self driving car do?

An online platform, Moral Machine, was created by Iyad Rahwan's Scalable Cooperation

group at the Massachusetts Institute of Technology, to investigate people's views on such ethical issues that artificial intelligence can address. It creates ethical dilemmas and gathers information about the decisions people make between two destructive outcomes. The [4] platform is the idea of Iyad Rahwan and social psychologists Azim Shariff and Jean-François Bonnefon, who conceived of the idea about the ethics of self-driving cars. The key contributors to building the platform were MIT Media Lab graduate students Edmond Awad and Sohan Dsouza [5]. The thought experiments posed by the researcher's Moral Machine website went viral, with their pictorial quiz taken by several million people in 233 countries or territories. The study, which included 40 million responses to different dilemmas, provide a fascinating snapshot of global public opinion as the era of self-driving cars looms large in the imagination, a vision of future convenience. The published study, identified a few preferences that were strongest: People opt to save people over pets, to spare the many over the few and to save children and pregnant women over older people. But it also found other preferences for sparing women over men, athletes over obese people and higher status people, such as executives, instead of homeless people or criminals. There were also cultural differences in the degree, for example, that people would prefer to save younger people over the elderly in a cluster of mostly Asian countries.

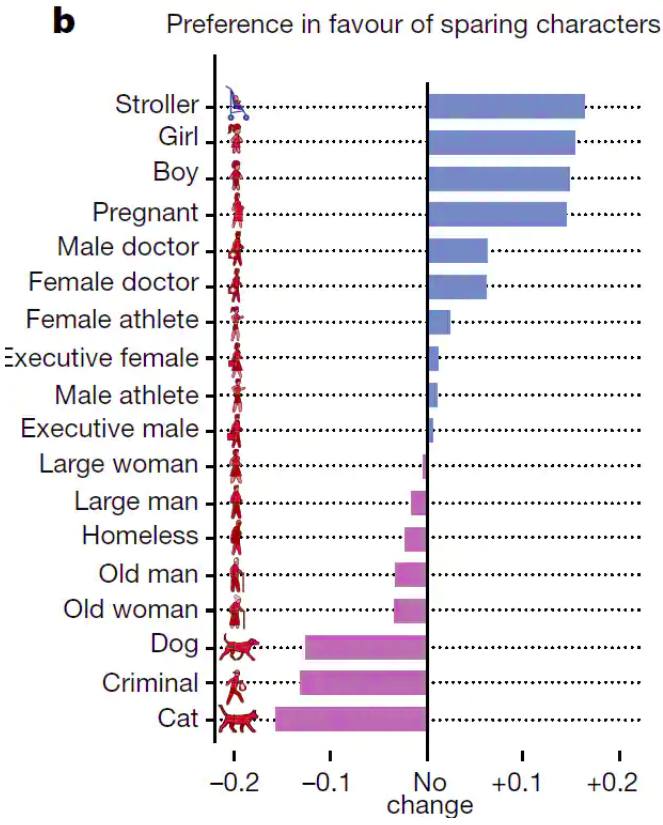


Figure 2.2: Results of "Moral Machine" survey.

Another controversial issue is the fear that artificial intelligence could lead to high levels of unemployment as machines will take over jobs that were always done by human hands. For example, motor cars can eliminate the need for taxis and car stock programs, while manufacturers can easily replace human labor with machines, making people's skills more obsolete.

Here it would be important to note that a survey conducted in 2017 by the European Commission, and the Directorate-General for Communications Networks, Content and Technology, on attitudes towards the impact of digitisation and automation on daily life,

shows that 61 percent of Europeans favor artificial intelligence and robots, while 88 percent believe that artificial intelligence technologies require careful management (Eurobarometer 2017, EU28). [6]

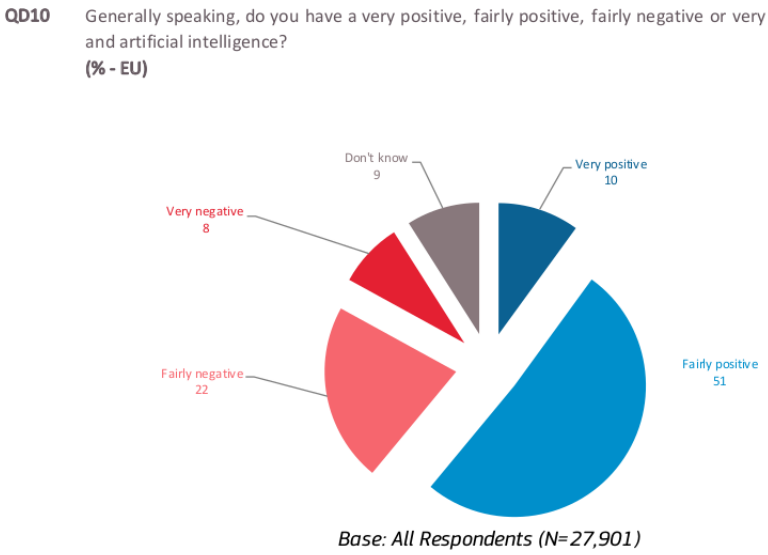


Figure 2.3: Over six in ten respondents (61%) have a positive view of robots and artificial intelligence.

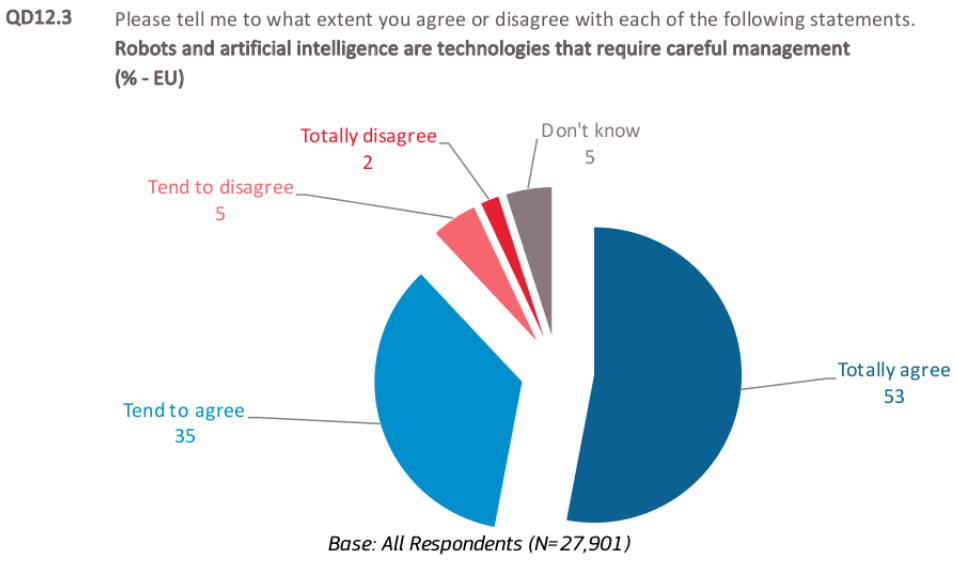


Figure 2.4: Almost nine in ten respondents (88%) agree robots and artificial intelligence are technologies that require careful management.

2.1.5 Fields of Research

Artificial intelligence has various fields of research such as natural language processing (NTLK), robotics (Robotics), computer vision (Computer Vision) etc.

The fields that will be used in this thesis are machine learning (Machine Learning) and deep machine learning (Deep Learning). The following subsections analyze and explain these fields.

2.2 Machine Learning

2.2.1 Definition

Machine learning is considered part of artificial intelligence. It belongs to the field of computer science and has to do with learning algorithms that are trained in data samples, creating models that can make predictions or make decisions on their own. These statistical models are considered to be somewhat used to solve a practical problem.

In 1959, Arthur Samuel, a pioneer in the field of machine learning (ML) defined it as the "field of study that gives computers the ability to learn without being explicitly programmed".

ML can be understood as computational methods that use experience to improve performance or to make accurate predictions. In this case, experience refers to past information or data that is available to us, which has been labelled and categorized. As with any computational exercise, the quality and amount of the data will be crucial to the accuracy of the predictions that will be made.

Looking through this lens, ML seems to be a lot like statistical modelling. In statistical modelling, we collect data, verify that it is clean — in other words, that we have completed, corrected, or deleted any incomplete, incorrect, or irrelevant parts of the data — and then use this clean dataset to test hypotheses and make predictions and forecasts. The idea behind statistical modelling is the attempt to represent complex issues in relatively generalizable terms, which is to say, terms that explain most events studied. Effectively, we program the algorithm to perform certain functions based on the data we submit. Put differently, the algorithm is static. It needs a programmer to tell it what to do when it is fed with data. So far, it makes indeed sense to use this approach when activated by the programmer.

Below are represented five different definitions of the term, from reliable sources:

- "Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world." – Nvidia [8]
- "Machine learning is the science of getting computers to act without being explicitly programmed." – Stanford [9]
- "Machine learning is based on algorithms that can learn from data without relying on rules-based programming."- McKinsey & Co. [10]
- "Machine learning algorithms can figure out how to perform important tasks by generalizing from examples." – University of Washington [11]
- "The field of Machine Learning seeks to answer the question "How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?" – Carnegie Mellon University [12]

2.2.2 Types of Learning

Machine learning is divided into four categories: **Supervised**, **Unsupervised**, **Semi-supervised** and **Reinforcement**.

1. **Supervised Learning**
   In supervised learning or otherwise supervised learning, the set of data used by the algorithm is a collection of samples labeled with a $\{(x_i, y_i)\}_{i=1}^{N}$ label. Each element x_i between N is called a feature vector. The feature vector has dimensions $j = 1, ..., D$, where each j reflects an feature. For example, if the sample x_1 concerns a person then

$x_1^{(1)}$ could be his height, $x_1^{(2)}$ his age, $x_1^{(3)}$ his sex and so on. Each sample x_i has the same attribute type in position j as the other samples. That is, the second person x_2, again would have in position $x_2^{(1)}$ his height. The y_i tag can be either an element that belongs to a finite set of classes $\{1, 2, ..., C\}$, or real number, or a more complex structure, such as a vector, array, tree, or graph. The label can be defined as a category to which the specific sample belongs. If for example our samples were e-mails then each sample could have a tag belonging to the set of classes {spam, non-spam}.
For the first two cases of tags (a. Finite set of classes and b. Real number) we have corresponding **classification** problems and **regression** problems.

(a) **Classification**
In classification problems, label values belong to a finite set of classes, i.e. labels express a qualitative characteristic. The algorithm aims to classify the input samples into the category to which they belong. For example, a classification problem is what was mentioned above regarding the classification of e-mails as spam and non-spam.

(b) **Regression**
In regression problems, the label values take on continuous values, i.e. the labels express a quantitative characteristic. The algorithm in this case aims to predict a value for the input samples. For example, a regression problem is predicting a person's age.

2. **Unsupervised Learning**
In non-supervised learning or otherwise unsupervised learning, the data set used by the algorithm are samples that haven't any label $\{x_i\}_{i=1}^N$. In this case, for each x_i feature vector, the algorithm generates at its output either a new feature vector or a value. For example, the new feature vector could be smaller in size than the input so that the algorithm meets the purpose of reducing dimensions. Accordingly, the output value could be a real number that functions as an identifier of each input sample vector and shows that this differs from the other samples in the set.

3. **Semi-Supervised Learning**
In semi-supervised learning,the data set used by the algorithm contains both labeled and unlabeled samples $(x_i, y_i)_{i=1}^N$, $x_i{}_{i=N+1}^K$. Unlabeled samples usually outnumber labeled samples. The goal of the algorithm in this case is the same as the goal of supervised learning. The hope here is that using many unlabeled examples can help the learning algorithm produce a better model.

4. **Reinforcement Learning**
In reinforcement learning, all the data used is not labeled. The machine "lives" in an environment and is able to perceive the state of that environment as a feature vector. The machine can handle any situation (for any input vector). Each action of the machine can result in different rewards or bring the machine to a new state. The goal of the machine is to create a model-function, which given a situation, will select an action that contributes to the maximum expected average reward. More specifically, an agent is rewarded or punished for the actions who chooses to do in an environment trying to maximize a cumulative reward. Examples of the use of such models are automotive cars and programs that act as chess players. In these cases we can observe that the machine adapts to its environment according to the new data/situations it receives (obstacles, roads and traffic lights for cars or new opponent moves for chess), aiming at the long-term reward (to reach to his destination or to win at chess).

In the present dissertation the category of machine learning used is supervised learning and more specifically the classification, as the aim is to classify the speeches (samples) in poor or good quality.

2.2.3 Fundamental Algorithms

2.2.3.1 Linear Regression

Linear regression is a well-known regression algorithm that takes as input a collection of labeled samples $\{x_i\}_{i=1}^N$, which are feature D-dimensional vectors and foresees in output, linear combinations of the features of these samples $\{y_i\}_{i=1}^N$.

In this way, the aim is to produce a model as the linear combination of the characteristics of the sample x. The linear equation that describes this model is shown below:

$$f_{w,b}(x) = wx + b \tag{2.1}$$

where **w** is a vector of D-dimensional parameters and **b** is a real number.

We will use this model to predict the unknown y tag for the given sample x. What we are interested in is making a prediction that will get as close as possible to the actual value of the y tag. To do this it is enough to find the optimal value of the parameters (w *, b *) that will give us the most accurate predictions.

The above equation-model could be graphically represented as a line which, given the x-samples, tries to approach the y-points in the best way. Below is shown such a representation for one-dimensional samples x[13]:

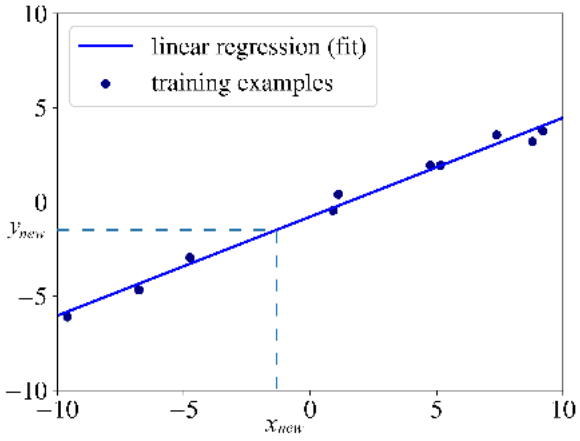


Figure 2.5: Linear Regression για δείγματα μίας διάστασης.

This line is practically the tool to predict the y_{new} tag, of a new sample x_{new}.

If our examples are D-dimensional characteristic characteristic vectors (for D> 1), the only difference from the one-dimensional case is that the regression model is not linear but flat (for two dimensions) or hyperplane (for D> 2).

The necessary condition that must be adhere to here, which is the key to solve our problem and find the required model, is that this line or the hyperplane must be as close as possible to all the training data y_i. Otherwise, the predictions y_{new} would be less likely to be correct. This condition is mathematically expressed as the minimization of the **Cost**

Function which in this case is the **Average Squared Error (MSE)**:

$$min \quad \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2 \tag{2.2}$$

where the expression $(f_{w,b}(x_i) - y_i)^2$, is called **Loss Function** and is the difference between the predicted label value and the actual label value for sample i. In fact, it serves as a penalty measure for incorrect classification of example i.

From the above mathematical expression we observe that to get the average square error, which we ultimately want to minimize, we add the loss function for all the samples and divide this value by the total number of samples.

The procedure followed to find the optimal values of the parameters w * and b * and consequently the optimal model through the above mathematical expression, is called optimization procedure.

The reason the square of the difference $(f_{w,b}(x_i) - y_i)^2$ was used and not the absolute value is because the absolute value has no continuous derivative, which would make the even. Functions that are not even create unnecessary difficulties when using linear algebra to find closed-form solutions to optimization problems. Intuitively, square penalties are better because they exaggerate the difference between the actual goal and the predicted one based on the value of that difference. We can also use forces 3 or 4, but their derivatives are more complicated to work with. The question here is which optimization method will be used to find the values w * and b * that satisfy (2.2). The answer to this question is given by the following algorithm.

Gradient Descent Algorithm

The reason we are interested in the derivatives of the function (2.2), as mentioned above, is because by derivating the expression we want to minimize and setting this derivative to 0, we can find the solution in a system of equations that will give us the optimal value of w * and b *. The derivation is done by calculating the partial derivatives of w and of b accordingly.

The idea is to start with some values for w and b and then change these values repeatedly to reduce costs. The abrupt descent algorithm, or derivative convergence algorithm, helps us to change these values.

To better understand the operation of this algorithm, we can imagine that we are at the top of a slope. To reach the lowest point (minimum), we will follow a specific route. On this route we can go with smaller steps-pace, with the result that we arrive later or go with bigger steps, with the risk of overcoming the lower point. In the abrupt descent algorithm, the size of the steps followed is the learning rate. This decides how fast the algorithm converges to a minimum. The following graphic shows this script [15]:

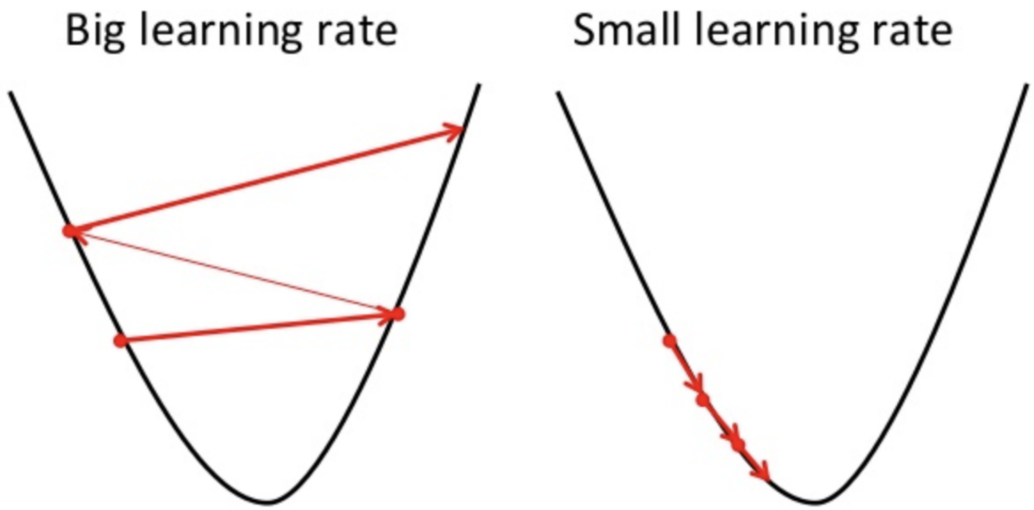


Figure 2.6: Ποσοστό εκμάθησης αλγορίθμου απότομης καθόδου.

Below are the mathematical operations performed:

$$J = \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2$$

$$J = \frac{1}{N}\sum_{i=1}^{N}(wx_i + b - y_i)^2$$

$$\frac{\partial J}{\partial w} = \frac{2}{N}\sum_{i=1}^{N}(wx_i + b - y_i)x_i \implies \frac{\partial J}{\partial w} = \frac{2}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)x_i$$

$$\frac{\partial J}{\partial b} = \frac{2}{N}\sum_{i=1}^{N}(wx_i + b - y_i) \implies \frac{\partial J}{\partial b} = \frac{2}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)$$

$$w^* = w - \alpha\frac{2}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)x_i \tag{2.3}$$

$$b^* = b - \alpha\frac{2}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i) \tag{2.4}$$

where **a** is the learning rate, a parameter that is determined by us and **w** * and **b** * are the optimal parameters. A lower learning rate could bring us closer to the minimum, but it takes longer to get there. A higher rate of learning converges earlier, but there is a chance that we will exceed the minimum.

2.2.3.2 Logistic Regression

The logistic regression algorithm does not refer to a regression problem but essentially to a classification problem. Its name is due to the fact that the mathematical configuration of the accounting regression is similar to that of the linear regression we saw above. We will study the algorithm for the binary problem, however it can be extended to more classification problems.

We will define a negative label as 0 and a positive label as 1. What remains to be done is to find a simple continuous function with a value field in space [0,1]. Thus, for each x_i input sample, the model function will return a value, which if it is closer to 0, then we assign a negative tag to x_i. Otherwise, we classify the sample as positive. One function that has such a property is the **sigmoid function**, which is presented below:

$$f(x) = \frac{1}{1 + e^{-x}}$$

where **e** is the basis of the natural logarithm (also called the Euler number))[16].

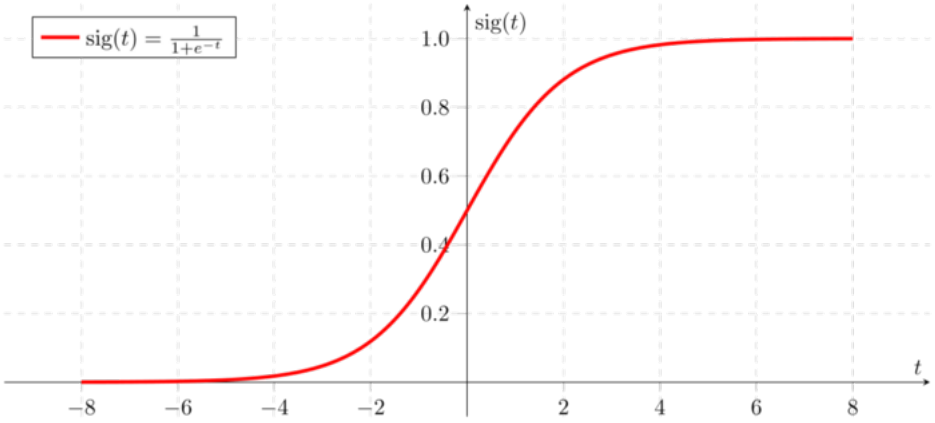


Figure 2.7: (sigmoid function)

So finally the accounting regression model will be the following:

$$f_{w,b}(x_i) = \frac{1}{1 + e^{-(wx_i+b)}} \tag{2.5}$$

where **w** and **b** are the parameters to be optimized. Here we see the well-known $wx_i + b$ extrusion from linear regression. All that remains is to find the way to calculate the optimal w * and b *.

In the case of linear regression we wanted to minimize the mean square error (MSE). In this case the optimization criterion is **maximum likelihood**:

$$max \quad \prod_{i=1}^{N} f_{w,b}(x_i)^{y_i}(1 - f_{w,b}(x_i))^{(1-y_i)} \tag{2.6}$$

Basically, what we want to maximize is the likelihood that education data will be classified correctly. If we look more closely at $f_{w,b}(x_i)$ is the output of our model which gives the probability that the sample x_i is positive. The To $1 - f_{w,b}(x_i)$ gives the probability that the sample x_i is negative. If the actual value of the sample label is 1, i.e. $y_i = 1$, then the above expression will be as follows:

$$f_{w,b}(x_i)^{y_i}(1 - f_{w,b}(x_i))^{(1-y_i)} = f_{w,b}(x_i)$$

where we are finally asked to maximize the probability that the sample x_i will be sorted with a label of 1 (which is the real one). Accordingly if the actual value of the sample label is 0, i.e. $y_i = 0$, then the above expression will be as follows:

$$f_{w,b}(x_i)^{y_i}(1 - f_{w,b}(x_i))^{(1-y_i)} = 1 - f_{w,b}(x_i)$$

where finally we are called to maximize the probability $1 - f_{w,b}(x_i)$, i.e. the probability that the sample x_i is sorted with a label 0 (which is the real one).

The reason we used the product $\prod$ here and not the sum of $\sum$ is because the probability of observing N tags for N samples is the product of the probabilities of each observation for each sample, just as in probability theory with the multiplication of probabilities results in a series of independent experiments.

Due to the function of the exponential function (exp) used in the model, in practice, it is more convenient to maximize the logarithmic probability (log-likelihood) instead of the probability. The ln is a strictly increasing function and maximizing of this is the same as maximizing its definition. The new expression to be maximized is defined as follows:

$$max \quad \sum_{i=1}^{N} y_i \ln f_{w,b}(x_i) + (1 - y_i) \ln (1 - f_{w,b}(x_i)) \tag{2.7}$$

Because by definition the cost function is something that has a negative effect and we would like to minimize it, we will turn the above maximization problem into minimization:

$$min \quad -\sum_{i=1}^{N} y_i \ln f_{w,b}(x_i) + (1 - y_i) \ln (1 - f_{w,b}(x_i)) \tag{2.8}$$

Basically , we made the expression from positive, negative and so instead of maximizing it, we want to minimize it. The new expression to minimize is called **cross entropy**.

In conclusion, to find the optimal parameters w * and b * that satisfy (2.8), we apply again the gradient descent algorithm:

$$J = -\sum_{i=1}^{N} y_i \ln f_{w,b}(x_i) + (1 - y_i) \ln (1 - f_{w,b}(x_i))$$

$$J = -\sum_{i=1}^{N} y_i \ln \frac{1}{1 + e^{-(wx_i+b)}} + (1 - y_i) \ln (1 - \frac{1}{1 + e^{-(wx_i+b)}})$$

$$w^* = w - \alpha \frac{\partial J}{\partial w} \tag{2.9}$$

$$b^* = b - \alpha \frac{\partial J}{\partial b} \tag{2.10}$$

where $\frac{\partial J}{\partial w} = \sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)x_i$, $\frac{\partial J}{\partial b} = \sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)$ και $\boldsymbol{\alpha}$ the learning rate.

The above two algorithms (linear regression, logistic regression) will not be used directly in this dissertation, but will help in the better and easier understanding of the algorithms presented in the following chapters.

2.2.3.3 Support Vector Machine (SVM)

The "Support Vectors Algorithm" (SVM) algorithm sees sample feature vectors as points in a multi-dimensional (N-dimensional) space. Thus, by placing the feature vectors on an imaginary graph, the algorithm draws a line (a hyper-level) (N-dimensional), which separates the samples with positive labels from the samples with negative labels (classify). The boundary that separates samples of different classes is called **decision boundary**.

To break up samples into classes, there are many hyperlevels, we could design. Our goal is to find a level that has the maximum margin, i.e. the maximum distance between the data points of both categories. Maximizing the margin distance provides some support so that future data points can be classified with more confidence [17].

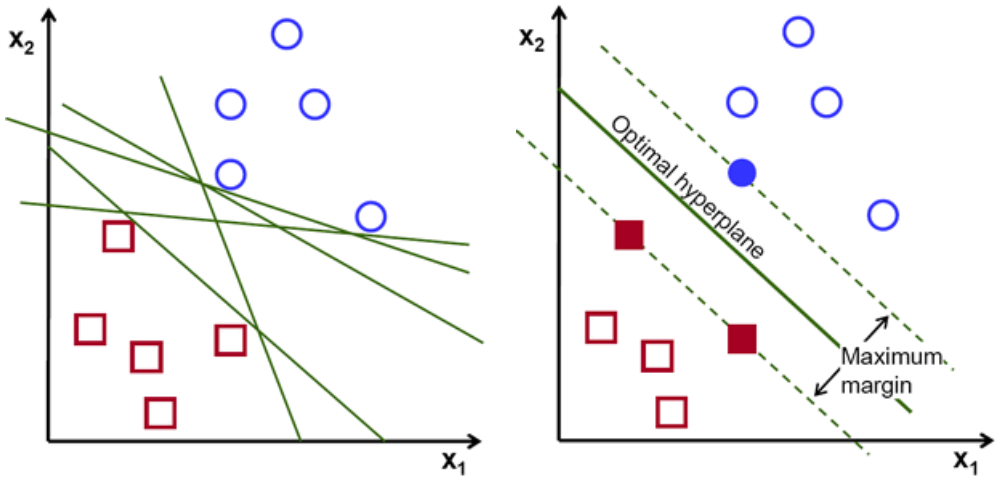


Figure 2.8: Possible upper-level limits

Below are graphs for two-dimensional and three-dimensional space respectively. That is, for a 2-dimensional feature vector and a feature vector of 3-dimensional [17].

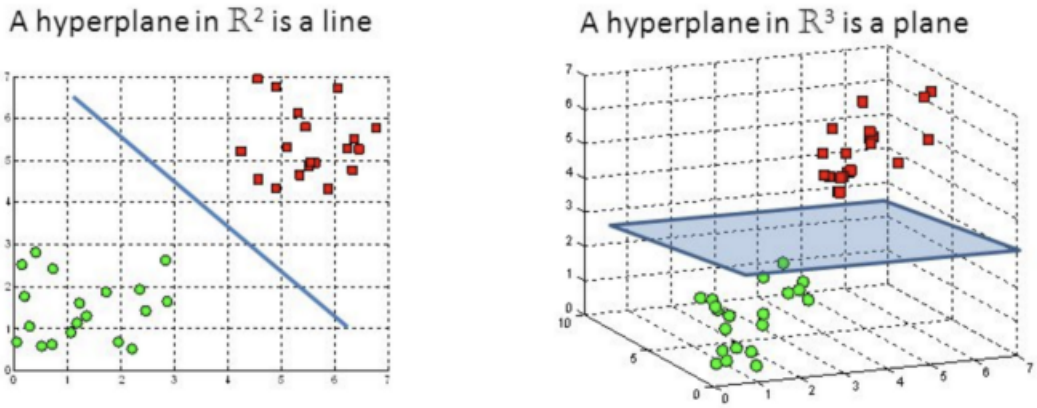


Figure 2.9: Hyperplanes in space of 2 and 3 dimensions

The model in this case is defined as follows:

$$f_{w,b}(x_i) = sign(wx_i + b) \tag{2.11}$$

where $\boldsymbol{wx_i + b}$ is the hyper-level line which we explained above and $\boldsymbol{sign}$ is the signum function which takes as input any real number and returns +1 if this number is positive, -1

if the number is negative and 0 if the number is equal to 0. The mathematical definition of this function is given below:

$$sign(x) = \begin{cases} -1, x < 0 \\ 0, x = 0 \\ 1, x > 0 \end{cases}$$

The aim is to find the optimal values w * and b * of the parameters w and b, as we saw in the previous algorithms.

In other words, if the sample x_i is below the hyperplane $w^* x_i + b^*$ then it will be classified as negative (label -1), while if it is above the hyper-level then it will be classified as positive (label +1).

Hard-Margin

In case the data is linearly separable, to find the optimal parameters w * and b *, we will again solve an optimization problem with the following constraints:

- $wx_i - b \geq 1$, if $y_i = +1$
- $wx_i - b \leq -1$, if $y_i = -1$
- Maximum margin between the hyperplane and the samples of the two classes.

The above optimization problem is mathematically defined as follows:

$$min \quad ||w|| \quad \text{subject to} \quad y_i(wx_i - b) \geq 1 \tag{2.12}$$

where $\boldsymbol{||w|| = \sqrt{\sum_{j=1}^{D}(w^{(j)})^2}}$ is the margin of the sample and $\boldsymbol{y_i(wx_i - b) \geq 1}$ expresses the first two constraints, since if the actual y_i label of the sample $y_i(wx_i - b) \geq 1 \implies (wx_i - b) \geq 1$ and if the real label y_i is -1 then $y_i(wx_i - b) \geq 1 \implies (wx_i - b) \leq -1$

The equations $wx_i - b = 1$ and $wx_i - b = -1$ define two parallel hyper-levels at our decision limit $wx_i - b = 0$. The distance from these two hyper-levels is defined as $\frac{2}{||w||}$. Therefore, the smaller the norm $||w||$, the greater the distance between the two hyper-levels [13].

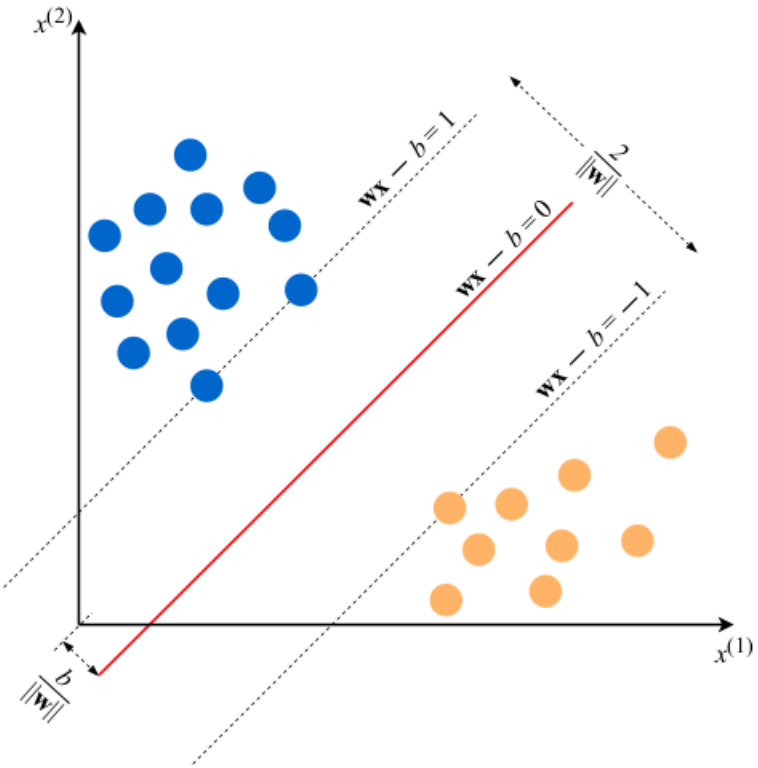


Figure 2.10: Hyperplane margin

The model described above is linear, as it uses as a decision limit a line (or a level or a hyper-level) and the data are linearly separable. However, there are cases where the data is not linearly separable into classes. This is because there may be **outliers**, i.e. samples that are far away in space, relative to the rest, or there may be noise in the data.

Soft-Margin

We will extend the algorithm in cases where the data is not linearly separable. For convenience we will symbolize the decision limit as:

$$wx_i - b = \beta_0 + \beta_1 x_{i1} + ... + \beta_k x_{ik} = \beta x_i$$

provided that the sample x_i is a vector of characteristic k-dimensions.

Here we will use the cost function **hinge loss**:

$$max(0, 1 - y_i(\beta x_i))$$

which is 0 if the first two constraints we saw above are satisfied. That is, if the sample x_i is positive and $\beta x_i \geq 1$ or if the sample x_i is negative and $\beta x_i \leq -1$. This means that this cost function is zeroed when βx_i is classified on the right side of the decision boundary.

The expression we finally want to minimize here is this:

$$min \quad C\|\beta\|^2 + \frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_i(\beta x_i)) \tag{2.13}$$

where $\|\beta\|$ is again the margin of the decision limit with the samples.

The C parameter determines the balance between the magnitude of the influence of the decision limit and ensuring that each sample x_i is on the right side of the decision

limit. Parameter C is usually selected experimentally. For fairly high C values, the second term in the cost function will become negligible, so the SVM algorithm will try to find the highest margin by completely ignoring the incorrect classification. As we reduce the value of C, classification errors become more costly, so the SVM algorithm will try to make fewer errors by sacrificing the margin size. A larger margin is better for generalization. Therefore, C regulates the exchange between good classification of training data (minimization of empirical risk) and good classification of future examples (generalization). We will talk about regularization later in chapter 2.2.7.

It should be noted that minimizing $||\beta||^2$ is the same as minimizing $||\beta||$.

What ultimately concerns us, is to find the optimal values of the parameters b. To find these, the gradient descent algorithm will be used again, as we saw in the previous chapters. Below, the partial derivative of the expression (2.13) is calculated for b:

$$\frac{\partial}{\partial\beta}(C||\beta||^2) = 2C\beta$$

$$\frac{\partial}{\partial\beta}(max(0, 1 - y_i(\beta x_i))) = \begin{cases} 0, & \text{if} \quad y_i(\beta x_i) \geq 1 \\ -y_i x_i, & \text{else} \end{cases}$$

$$\frac{\partial}{\partial\beta}(C||\beta||^2 + \frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_i(\beta x_i))) = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} 2C\beta, & \text{if} \quad y_i(\beta x_i) \geq 1 \\ 2C\beta - y_i x_i, & \text{else} \end{cases} \tag{2.14}$$

Optimal parameter value b:

$$\begin{cases} \beta^* = \beta - \alpha(2C\beta), & \text{No misclassification} \\ \beta^* = \beta + \alpha(y_i x_i - 2C\beta), & \text{Misclassification} \end{cases} \tag{2.15}$$

where ***alpha*** is the learning rate as we have seen in the previous chapters.

Kernel Trick

The above algorithm of the maximum margin of the hyper-level (decision limit), creates a linear classifier. However, the SVM can be adapted to work also in cases where the data cannot be separated from a hyper-level in their original space. If we manage to transform their original space into a space of more dimensions, then the data could be linearly separable in the new space. This method is called **kernel trick**.

Below is a case where in the original 2-dimensional space (left), the data is not linearly separable, but when the space is converted to 3-dimensional (right), then the data can be separated by a hyper-level [14].

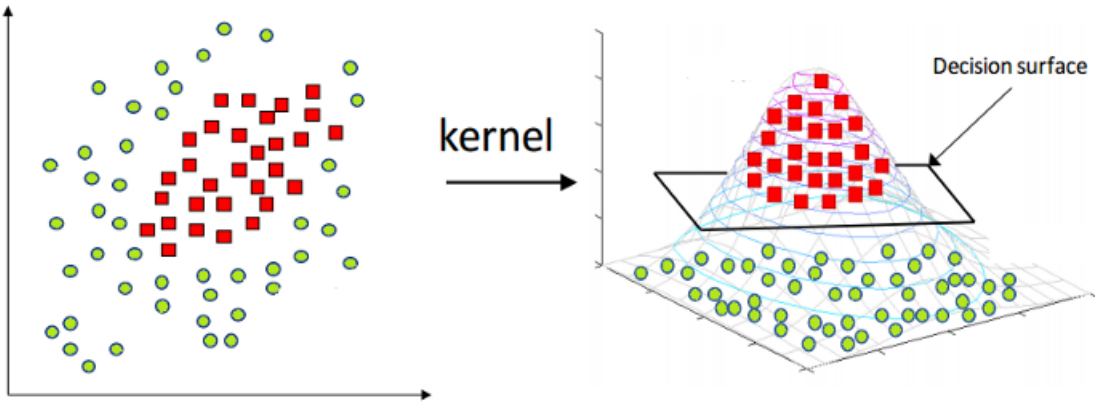


Figure 2.11: Kernel Trick

The resulting algorithm is typically similar, except that each internal product is replaced by a non-linear kernel function. This allows the algorithm to find the maximum hyper-level margin in a transformed feature space. To see in more detail how the algorithm works with the addition of kernels, we will first see how the following mathematical expression is solved:

$$min \quad \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(wx_i - b) - 1 \geq 0, \text{ i=1,...,N} \tag{2.16}$$

which is equivalent to (2.12) and defines the SVM algorithm for linearly separable data.

The Lagrange multipliers method is used to solve (2.16). Thus (2.16) is defined as:

$$\max_{a_i \ldots a_N} \sum_{i=1}^{N} a_i - \sum_{i=1}^{N}\sum_{k=1}^{N} y_i a_i (x_i x_k) y_k a_k \quad \text{subject to} \quad \sum_{i=1}^{N} a_i y_i = 0 \quad \text{and} \quad a_i \geq 0, \text{ i=1,...,N} \tag{2.17}$$

where $\boldsymbol{a_i}$ are the Lagrange multipliers.

Square Kernel (quadratic kernel)

The square kernel is defined as follows:

$$k(x_i, x_k) \stackrel{def}{=} (x_i x_k)^2 \tag{2.18}$$

In the relation (2.17), we observe that the only point where the attribute vectors exist is the interior product $(x_i x_k)$. If the original space was 2-dimensional and we wanted to elevate the problem to 3-dimensional then we would have to convert the attribute vectors $x_i = (p_i, q_i)$ and $x_k = (p_k, q_k)$, to (p_i, q_i) $implies({p_i}^2, \sqrt{2}p_i q_i, {q_i}^2)$ and $(p_k, q_k) \implies ({p_k}^2, \sqrt{2}p_k q_k, {q_k}^2)$ accordingly and then calculate the interior product between the new vectors. Instead, taking the interior product of the original vectors (2-dimensional) and subtracting the square, we will have exactly the same result with less effort $x_i x_k \implies (p_i, q_i)(p_k, q_k) \implies (p_i p_k + q_i q_k) \underset{quadrate}{\implies}$ $({p_i}^2 {p_k}^2 + 2p_i p_k q_i q_k + {q_i}^2 {q_k}^2)$. So we used the square kernel.

At bottom, what the kernel trick does for us is provide a more efficient and less expensive way to convert data to higher dimensions. Thus, the implementation of the kernel trick is not limited to the SVM algorithm. Any calculations involving the internal products (x_i, y_i) can use the kernel trick.

polynomial kernel The polynomial kernel is the general category to which the square kernel belongs and extends to dimensions greater than 2 (d> 2):

$$k(x_i, x_k) \stackrel{def}{=} (x_i x_k + c)^d \tag{2.19}$$

where $\boldsymbol{c \geq 0}$ is a free parameter that exchanges the effect of higher-order versus to lower-order terms on the polynomial. When c = 0, the kernel is called a homogeneous kernel text.

RBF Kernel (radial basis function kernel)

The kernel rbf (radial base function) or Gaussian kernel is defined as follows:

$$k(x_i, x_k) \stackrel{def}{=} exp(-\frac{\|x_i - x_k\|^2}{2\sigma^2}) \tag{2.20}$$

where $\boldsymbol{\sigma}$ defines the shape of the decision boundary in the original dimension space (i.e. whether it will be smooth or have many angles) and $\|x_i - x_k\|^2$ is the square of **Euclidean distance** between the two feature vectors defined as follows:

$$\|x_i - x_k\| = d(x_i, x_k) \stackrel{def}{=} \sqrt{(x_i^{(1)} - x_k^{(1)})^2 + (x_i^{(2)} - x_k^{(2)})^2 + ... + (x_i^{(N)} - x_k^{(N)})^2} = \sqrt{\sum_{j=1}^{N} (x_i^{(j)} - x_k^{(j)})^2} \tag{2.21}$$

In this kernel we note that the vector space is of infinite dimensions (N), because it can be expanded from the Taylor series.

The above two kernels are the most commonly used in the field of machine learning. Although the RBF kernel is more popular in the SVM classification than the polynomial kernel, the latter is quite popular in Natural Language Processing (NLP). The most common grade is d = 2 (square), as higher grades tend to overfitting [2.2.7] in NLP problems.

2.2.3.4 Naive Bayes

Naive Bayes classifiers are a family of probabilistic models based on Bayes' theorem, considering that the features are independent of each other.

The well-known bayes theorem is given by the following equation:

$$p(A|B) = \frac{p(A)p(B|A)}{p(B)} \tag{2.22}$$

where A and B are events and p (B) is nonzero. Also, p(A|B) is the conditional probability, ie the probability that A will happen given that B happens, p(B|A) is respectively the probability that B will happen given that A occurs and p(A), p(B) are the probabilities of observing A and B, independent of each other, also known as marginal probabilities.

In the case of a classification process, where the purpose is to find the observation class, given some characteristic values, the above theorem can also be written as follows:

$$p(y_i|x_1, x_2, .., x_n) = \frac{p(x_1, x_2, ..., x_n|y_i)p(y_i)}{p(x_1, x_2, ..., x_n)} \tag{2.23}$$

where $p(y_i|x_1, x_2, .., x_n)$ is the probability that the sample belongs to the class y_i, given the values of the features $x_1, x_2, ..., x_n$. Also, $p(x_1, x_2, ..., x_n|y_i)$ indicates the probability of displaying a specific combination of attributes, given the label of a class.

In the above relation (2.23), the denominator can be subtracted as all it does is normalize the value of the probability of the left-hand side of the equation. In addition, the probability $p(y_i)$ is calculated very simply as follows:

$$p(y_i) = \frac{number \quad of \quad observations \quad with \quad class \quad y_i}{number \quad of \quad all \quad observations} \tag{2.24}$$

Finally, as already mentioned, all Bayes classifiers assume that the value of a particular attribute is independent of the value of any other attribute, given the variable class, hence they are called naive. For example, a fruit can be considered an apple if it is red, round and about 10 cm in diameter. A naive Bayes classifier considers that each of these characteristics contributes independently to the likelihood that this fruit is an apple, regardless of any possible correlations between color, rounding, and diameter characteristics. In these terms, the probability $p(x_1, x_2, ..., x_n|y_i)$ of the relation (2.23) can be written as follows:

$$p(x_1, x_2, ..., x_n|y_i) = p(x_1|y_i)p(x_2|y_i)...p(x_n|y_i) \tag{2.25}$$

since it is known that for independent events A and B, the probability of their joint is equal to the product of their probabilities: $p(A \quad and \quad B) = p(A)p(B)$.

According to the above, the relation (2.23) is converted as follows:

$$p(y_i|x_1, x_2, .., x_n) = p(x_1|y_i)p(x_2|y_i)...p(x_n|yi)p(y_i) \implies$$

$$p(y_i|x_1, x_2, .., x_n) = p(y_i) \prod_{i=1}^{n} p(x_i|y) \tag{2.26}$$

From the relation (2.26), a decision function can finally be created, which determines the prediction of the model, ie the class in which the respective sample is classified. A common decision rule is to choose the case that is most likely. This rule is called maximum a posteriori or MAP decision rule. Based on this, the sample will be sorted on the y_k label as follows:

$$\hat{y} = \underset{k \in \{1,...,K\}}{\operatorname{argmax}} p(y_k) \prod_{i=1}^{n} p(x_i|y_k) \tag{2.27}$$

In short, it is selected the label-class, between the K in total, which has the highest probability of observation, given the specific values of the features presented by the sample to be classified.

Types of Naive Bayes Classifiers

The conditional probability for a particular attribute, given the class label (eg $p(x_1|y_i)$ of the relation (2.26)), is more easily calculated from the data. The algorithm needs to store the probability distributions of the attributes for each class-label separately. So if we have 5 classes and 10 attributes, 50 different probability distributions should be stored. To calculate the distribution parameters of an attribute, a distribution must be assumed. Feature distribution assumptions are called "event model" of Naive Bayes classifier.

The type of distribution depends on the characteristics:

- For binary features, such as whether or not a word appears in a text, the Bernoulli distribution is used. Thus the probability of a given features of a class becomes as follows:

$$p(x|y_k) = \prod_{i=1}^{n} p(x_i|y_k)^{x_i}(1 - p(x_i|y_k))^{(1-x_i)} \tag{2.28}$$

- For distinct features, such as the frequency with which words appear in a text, the Multinomial distribution is used. Thus the probability of a given feature of a class becomes as follows:

$$p(x|y_k) = \frac{(\sum_{i=1}^{n} x_i)!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} p(x_i|y_k)^{x_i} \tag{2.29}$$

- For continuous features, the Gaussian / Normal Distribution is used. Thus, if the set of training includes the feature x, the features are first separated by class and then the mean value and the deviation of x in each class are calculated. If μ_k is the mean value of the attribute x which in the class y_k and ${\sigma_k}^2$ is the deviation of the values of the feature x and assuming that there is a observing value "υ" of this feature, then the probability of observing this value given the class y_k is given by the following relation:

$$p(x = \upsilon|y_k) = \frac{1}{\sqrt{2\pi{\sigma_k}^2}} e^{-\frac{(\upsilon-\mu_k)^2}{2{\sigma_k}^2}} \tag{2.30}$$

In conclusion, the notion that all features are independent of each other makes naive bayes classifiers very fast in computation and training, but usually does not correspond to reality, making this algorithm less efficient than more complex algorithms. Of course, how efficient an algorithm can be depends on the problem under consideration and can vary from process to process.

2.2.3.5 k-Nearest Neighbors (kNN)

The kNN algorithm is based on the assumption that things that are similar are close by. So to determine the class-label of a sample, we search at the classes of k of its nearest neighbors and take the prevailing class, that is, the one that has the majority of neighbors (classification problem). Accordingly, to determine the true value of a sample label, we search the values of k of its nearest neighbors and take their average (regression problem).

In the image below we notice that most of the time, similar data points are close to each other [18]:

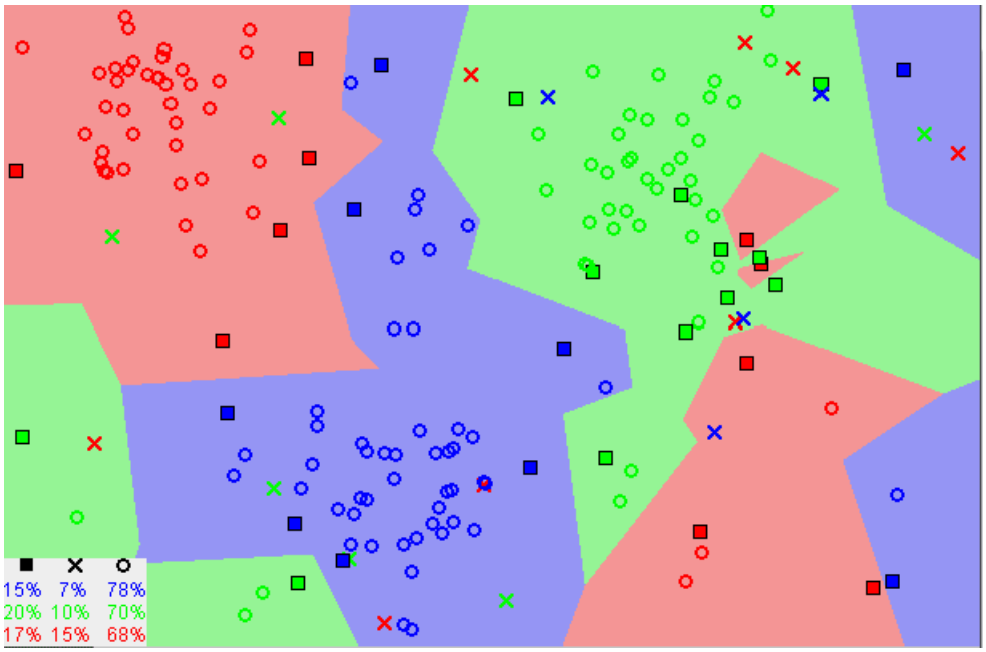


Figure 2.12: Similar neighbors

The proximity of two sample points is given by a distance function. There are many different distance functions that can be used here.
The **Euclidean distance** mentioned above is the most commonly used:

$$d(p,q) \overset{def}{=} \sqrt{(p-q)^2} \quad \text{1-dimension} \tag{2.31}$$

$$d(p,q) \overset{def}{=} \sqrt{(p-q)^2} \quad \text{1-dimension} \tag{2.32}$$

$$d(p,q) \overset{def}{=} \sqrt{(q_1-p_1)^2+(q_2-p_2)^2} \quad \text{2-dimensions} \tag{2.33}$$

$$d(q,p) \overset{def}{=} \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+...+(q_i-p_i)^2+...+(q_n-p_n)^2} \quad \text{N-dimensions} \tag{2.34}$$

Another popular distance function option is
textbfcosine similarity:

$$CoS(p,q) \overset{def}{=} \cos(\angle(p,q)) = \frac{\sqrt{\sum_{i=1}^{n} p_i q_i}}{\sqrt{\sum_{i=1}^{n} {p_i}^2}\sqrt{\sum_{i=1}^{n} {q_i}^2}} \tag{2.35}$$

where the angle between two vectors p and q is calculated and then its cosine. This measurement ranges between -1 and 1. If CoS is close to 1, this indicates that the angle between the two vectors is close to zero and is therefore similar (neighbors).

Other famous distance metrics are the Chebychev distance and the Hamming distance.

The similarity of cosine of two samples is equivalent to their internal product after normalization. Under these assumptions, kNN makes a local linear classification with a coefficient vector defined as follows:

$$w_x = \sum_{(x',y') \in R_k(x)} y' x' \tag{2.36}$$

where $\boldsymbol{w_x}$ are the coefficients for the input sample x, $\boldsymbol{R_k(x)}$ is the set of k nearest neighbors, $\boldsymbol{y'}$ and $\boldsymbol{x'}$ is the class and the vector of the neighbor x' attributes accordingly. The output of the classifier, i.e. the prediction of the sample class x, is obtained from the internal product $w_x x$ (which in the case of the normalized feature vectors is equal to the cosine similarity between w_x and x) when this exceeds a threshold.

In the case for example we have a binary classification problem with classes (0,1), w_x represents the centroids of the neighbors belonging to class 1 (since for $y' = 0 \implies y'x' = 0$) and the interior product $w_x x$ represents how close this centroid is to sample x, that is, how close sample x is to its class 1 neighbors. If this proximity is greater than a certain threshold, then sample x it is also classified in class 1.

Finally, it is important to define the **cost function** on which the **optimization method** runs in order to generate the optimal coefficients w_x. Cost function:

$$L = - \sum_{(x',y') \in R_k(x)} y' x' w_x + \frac{1}{2} \|w_x\|^2 \tag{2.37}$$

$$min \quad L \tag{2.38}$$

According to (2.38), the **optimization criterion** is to minimize the cost function, as usual, defined by (2.37).

In this function, the first term $- \sum_{(x',y') \in R_k(x)} y' x' w_x$ reflects the error of the training set, while the second term $\frac{1}{2} \|w_x\|^2$ is a sentence of complexity, just like in the logic we encountered in SVM (2.13). Therefore, the loss of the training-set in a single example of learning is:

$$l = \begin{cases} -y' x' w_x, & \text{when} \quad y' = 1 \quad \text{and} \quad x' \in kNN(x) \\ 0, & \text{Otherwise} \end{cases} \tag{2.39}$$

where $\boldsymbol{-y'x'w_x}$ expresses the distance of the neighbor x ' labeled 1, from the centroid of all the neighbors of x labeled 1. If we look more closely at this accosiation, the thing that the algorithm tries to do, is to bring the neighbors that have the same label, that is, which belong to the same category-class, closer to each other. That is, to maximize their similarity or otherwise to minimize their distance (min $-y'x'w_x$).

Finally we will set as 0 the first derivative of the right term of equality (2.37), to find the optimal values of the coefficients w_x.

The above analysis of the cost function of the kNN algorithm is based on [19].

2.2.3.6 Decision Tree

The decision tree algorithm works exactly as its name implies. A sample comes as an input, for which we want to predict its class. The algorithm then asks a series of questions. After each answer, the next question follows, until at the end we come to a class. The algorithm is in the form of a decision tree, consisting of nodes and edges. Nodes are divided into three types:

- Root (root node)
  The root is the primary node, which has no incoming edge, and may have 0 or more outgoing edges.

- internal Internal nodes
  Internal nodes have exactly one inbound edge, and may have 2 or more outbound edges.

- Leaves (leaf nodes)
  The leaves have exactly one incoming edge and no outgoing edge. Each sheet is marked with the prediction tag class.

There are different algorithms that can be used in decision trees based on the format of the attributes and consequently the optimization criteria used:**ID3** → extension ID3, **C4.5** → successor ID3, **CART** → Classification And Regression Tree, **CHAID** → Chi-square automatic interaction detection. Performs multi-level splits when computing classification trees), **MARS** → multi custom multivariate adaptive regression splines.

We will study the function of ID3 for classification problems. ID3 is a greedy algorithm, which does a top-down search, always making the choice that seems to be the best at the moment, without taking into account the previous course.
This algorithm works as follows:

1. The original root node, contains the set S consisting of all the highlighted samples $S \overset{def}{=} \{(xi, yi)\}_{i=1}^{N}$.

2. In each iteration of the algorithm, all combinations of parameters (j, t) are calculated, where j is the position of a specific attribute from the attribute vector (j = 1,..., D) and t is the threshold.

3. For each combination of the above, break the set S into two subsets $S_- \overset{def}{=} \{(x, y)|(x, y) \in S, x^{(j)} < t\}$ και $S_+ \overset{def}{=} \{(x, y)|(x, y) \in S, x^{(j)} \geq t\}$

4. From the above combinations, we select the combination of parameters (j, t) that gives the best split of the set. The best split is the one that gives the minimum cross entropy.

5. O The algorithm continues to be repeated from step 2 for each subset, taking into account only the features that have not been selected before.

6. The algorithm stops when all the samples in the leaves have been sorted correctly or when there is no other attribute on the basis of which to make a new split or when the depth of the tree reaches a maximum value d or when the entropy becomes lower from a value of ϵ.

The model that applies to each set S, i.e. to each leaf of the tree that makes a prediction, can be defined as follows:

$$f_{ID3}^{S} = \frac{1}{|S|} \sum_{(x,y)\in S} y \tag{2.40}$$

The prediction given by the above model will be the same for each input x. This prediction gives a probability of whether the input sample x belongs to class 1 Pr(y=1|x). The probability is calculated from the average of the node samples belonging to class 1.

The following is an example ID3 decision tree with 7 training samples:

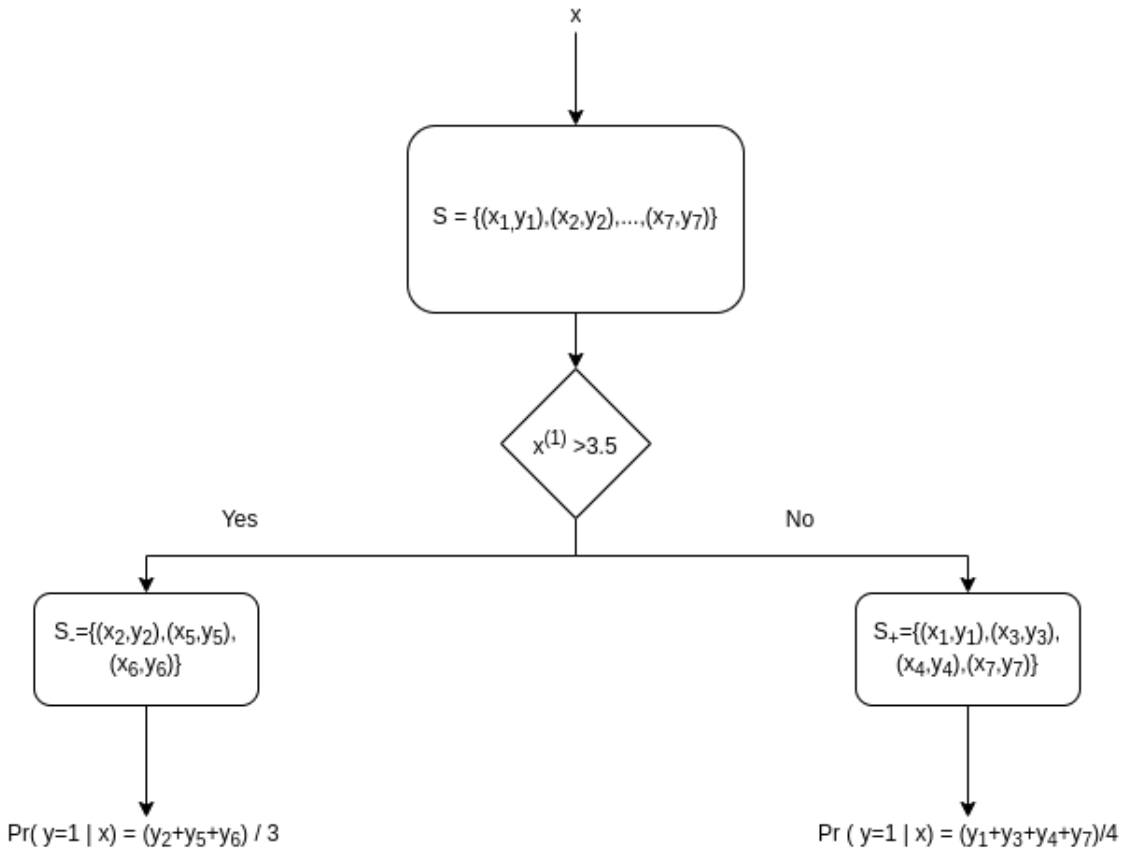


Figure 2.13: Example decision tree

The optimization criterion is the minimization of entropy as mentioned in 4. The transverse entropy is redefined in chapter 2.2.3.2, in the relation (2.8) for the logistic regression algorithm.
The entropy of a set of S samples is given by the following relation:

$$min \quad H(S) = -f_{ID3}^{S} \ln f_{ID3}^{S} - (1 - f_{ID3}^{S}) \ln (1 - f_{ID3}^{S})$$

where we want to maximize the logarithm probability (log-likelihood) that the samples of the set S of each node have been correctly sorted. In essence we want to maximize the probability that the samples in this node set belong to the same class or otherwise minimize the transverse entropy (negative logarithmic probability).

To better understand the role of entropy here, entropy is a measure of the randomness of the information being processed. The higher the entropy, the more difficult it is to draw conclusions from this information [20]:

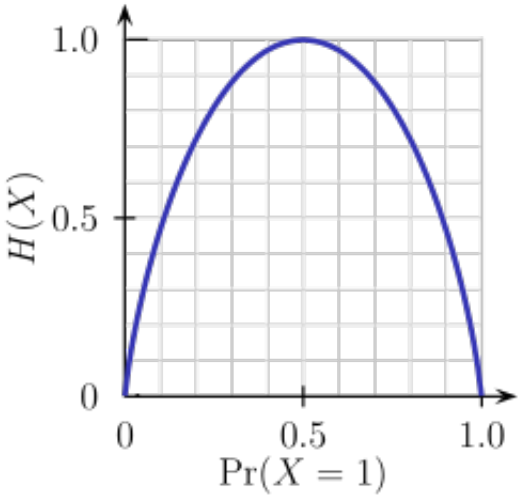


Figure 2.14: Εντροπία

From the above graph, it is quite obvious that the entropy H (X) is zero when the probability is either 0 or 1, i.e. in our case, when all the samples of the set S belong to class 0 or class 1 ($f^S_{ID3} = 0$ ή $f^S_{ID3} = 1$). In contrast, the entropy is maximum when the probability is 0.5 because it displays perfectly random in the data and there is no chance of perfectly determining the result, i.e. in our case, when half of the samples of the set S belong to class 0 and the other half to class 1 ($f^S_{ID3} = 0.5$).

Dividing the set S into two subsets S_- and S_+ based on a characteristic j and a threshold t, the entropy resulting from this split is defined as a weighted sum of the two entropies:

$$min \quad H(S_-, S_+) = \frac{|S_-|}{|S|} H(S_-) + \frac{|S_+|}{|S|} H(S_+) \tag{2.41}$$

Essentially, in each step of the ID3 algorithm, the relation (2.41) is used, in order to find the division of the set that gives the minimum entropy, i.e. to find the optimal combination (j, t) based on which the split takes place. In addition to entropy, other criteria can be used, based on which the best feature with the best threshold (j, t) is selected to perform the split. Other such criteria are: Information gain **(Information gain)** , text Gini **(Gini index)**, Gain Ratio **(Gain Ratio)**, Reduction in Variance **(Reduction in Variance)** and square-Chi **(Chi-Square)**.

Ensemble Learning

The algorithms that follow in the coming chapters belong to the category **ensemble learning** . The goal of algorithms in this category is to develop many models, which may not be as efficient, and by combining them to create a powerful meta-model. Low accuracy models are **weak learners** and are usually faster at learning and predicting than more powerful and complex models. The way in which a prediction is obtained from such models is by combining the predictions from each weak model, using a kind of weighted vote.

In collaborative learning there are two types of algorithms, **bagging (bootstrap aggregating)** and **boosting**.

In **bagging**,copies of the original training set are created, each of which is used by a weak student in order to create a weak model. These copies differ from each other, they may be different sizes of the original set, as well as they may contain some samples of the original set, multiple times. Specifically, it is estimated that if the new sets are equal in size to the original (n=n') then 63.2% will be unique samples and the rest duplicates. Finally, the weak models are combined to produce the collective meta-model.

The following is the bagging [21] :

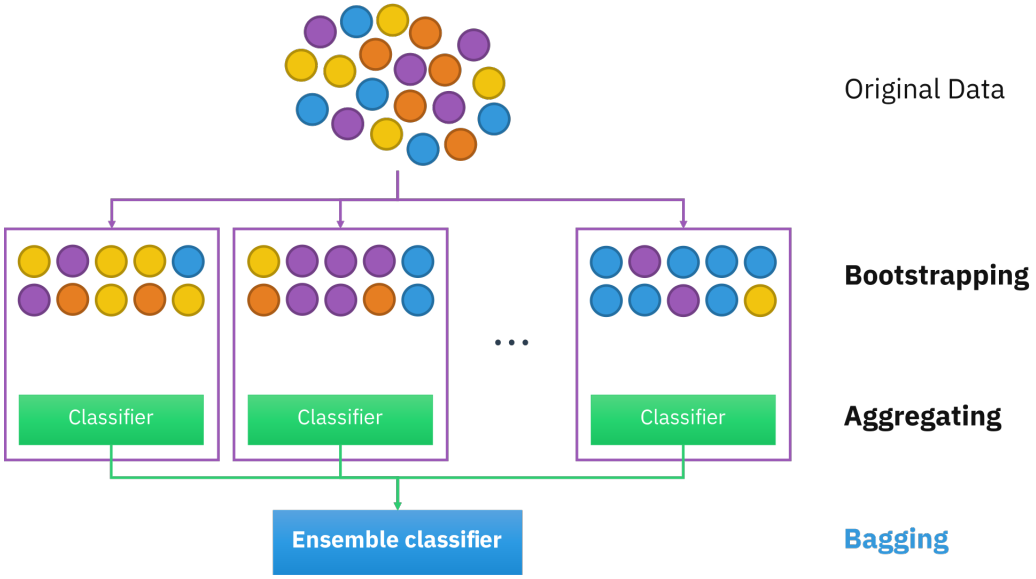


Figure 2.15: Collective learning with bagging

In **boosting**, There is an iterative process of learning weak classifiers, which are added to create a final strong classifier. After adding a new weak classifier, the data resets its weights (re-weighting) in such a way that incorrectly sorted data has heavier weights than the correct ones. Thus, prospective weak learners focus more on examples that previous weak learners misclassified.

The following is the boosting method [22]:

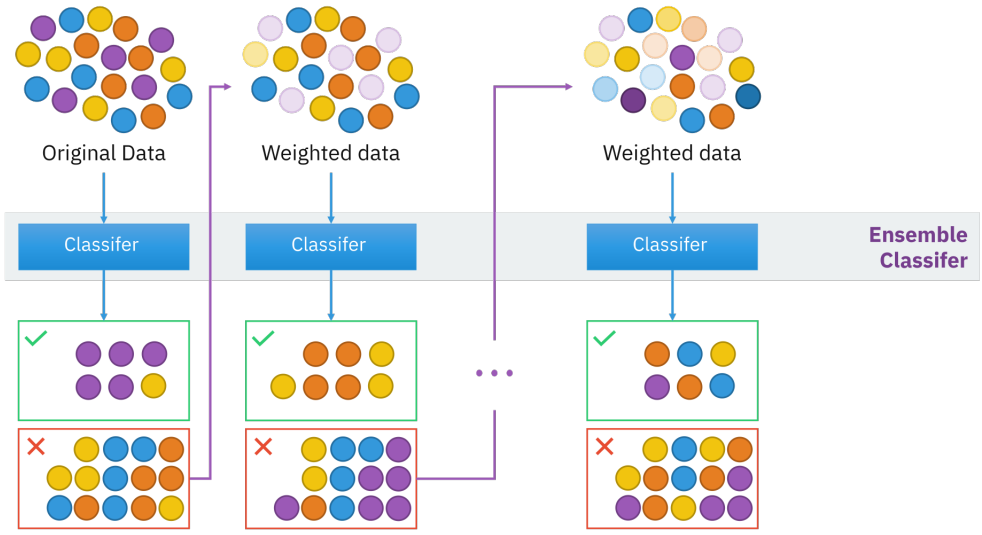


Figure 2.16: Colective learning with boosting

2.2.3.7 Random Forest

This algorithm uses bagging, having as weak students, decision trees which we saw in the chapter 2.2.3.5. More specifically, from an initial set of training data S, B sets are created with replacement (S_b,b=1,...,B). Replacement means we randomly select samples from the S

data set and copy them to the S_b data set, retaining the original samples in the original set. Then B decision trees are created, each of which uses one of the S_b as a training set. The final prediction for a new input sample x is taken from the average of the tree predictions, in the case of regression, or by the majority in the case of classification. In the case of regression, we have:

$$y = \frac{1}{B}\sum_{b=1}^{N} f_b(x) \tag{2.42}$$

where $\mathbf{y}$ is the prediction for the input sample $\mathbf{x}$ and $\boldsymbol{f_b(x)}$ is the prediction of each tree b.

An estimate of the prediction uncertainty can be defined as the standard deviation of the predictions from all the individual regression trees in the input sample x ':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - y')^2}{B-1}} \tag{2.43}$$

It is worth noting here that decision trees do not consider all the features in each split, as we have seen above, but a random subset of them. This is because if an attribute is more powerful-helpful to make a prediction, then that attribute will be selected from many different trees as a criterion for splitting the data set. As a result, many bad models (trees) tend to agree on some wrong predictions and thus affect the majority giving the wrong end result. This phenomenon is called correlation of the trees and can be avoided if the node split is based on a random subset of features at a time. From the above we understand that the name of the algorithm (random forest), derives both from the random sampling of training data sets during tree construction and from the random subsets of features that are taken into account when separating nodes.

The parameters that can be modified in the algorithm by changing the results are the number of B trees and the size of the random feature subsets. These parameters are set for investigation and experimentation, in order to find the best depending on the case (hyperparameter tuning 2.2.8.2).

The random forest algorithm is the most famous of the ensemble learning algorithms. The reason why it is so effective is the use of many different versions of the data set (training set), which have emerged with replacement. This function helps to eliminate phenomena such as noise in the data, extreme values (unusual) and similar examples that occur in too large or too small percentage. This boot process leads to better model performance because it reduces model variance without increasing bias or else without causing underfitting (2.2.6).

2.2.3.8 Extra Trees

The algorithm **extra trees** or else **extremely randomized trees classifier**, is similar to the random forest, changing only the way the decision trees of the forest are constructed. More specifically, there are two differences:

1. In the extra trees algorithm, each tree is trained using the entire training set and not a set that has resulted in replacement as we saw above.

2. In each split step, a random subset of the attributes is used as we noticed in the random forest, but the threshold on which the split will be based is selected randomly. Specifically, this value is selected from a uniform distribution within the empirical range of the attribute. On the other hand in the random forest algorithm, was calculated the

locally optimal breakpoint (threshold) for each feature under consideration. Then, from all the randomly generated splits, the split with the highest score is selected to split the node (just like in the decision tree and the random forest).

These differences encourage a reduction in both bias and variance. On the one hand, using the entire original sample set instead of the replacement copy will reduce bias. On the other hand, randomly selecting the point of separation of each node will reduce the variance. Extra Trees can sometimes generalize better than Random Forests, but it's hard to guess when this happens without trying the first two, as well as experimenting with parameters: number of trees, size of feature subsets, and minimal samples per set. .

In terms of computational cost, and therefore execution time, the Extra Trees algorithm is faster. This algorithm saves time because the whole process is the same, but randomly selects the breakpoint and does not calculate the optimal one. However, this algorithm usually produces a larger model than Random Forest which produces a more compact one.

2.2.3.9 Gradient Boosting

This algorithm uses boosting and usually decision trees like the above algorithms. Decision trees also function here as weak students, who combine to create an algorithm called **gradient boosted trees**. It can be applied to both regression and classification problems.

In the case of regression the original tree model is defined as follows:

$$f = f_0(x) \stackrel{def}{=} \frac{1}{N}\sum_{i=1}^{N} y_i \tag{2.44}$$

where the average of the sample labels is calculated.

The labels for each sample $x_i, i = 1, ...N$ of the training set are then falsified as follows:

$$\hat{y}_i \leftarrow y_i - f(x_i) \tag{2.45}$$

where $\hat{y}_i$ is called **residual** and reflects how well or poorly each sample of the training set is classified, since it is the difference between the actual label and the intended label of each sample. In short, it is the sizing / predicting error of each sample.

From these new tags a new training set is created, which will be used by the next model-tree to try to correct the mistakes of the previous one. The new model f_1 will be combined with the original f_0 giving the final **boosting model**: $f = f_0 + \alpha f_1$, where α is the learning rate we have seen in previous chapters. Then the process is repeated, creating a new data set from the new tags (2.45) and creating a new model f_2 which is called to correct f_1 errors. The amplification model is reshaped accordingly: $f = f_0 + \alpha f_1 + \alpha f_2$. The repetition will stop when we reach a maximum number of combined trees.

The name of the algorithm (gradient boosting), indicates some kind of relationship with the calculation of a derivative (gradient) as well as with the gradient descent algorithm, which was used in the chapter 2.2.3.1 (linear regression). More specifically, we saw that the abrupt descent algorithm helped us to determine the direction in which we should move the values of the parameters (weights) of the model, in order to reduce the mean squared error (MSE) to a minimum. This direction was calculated from the negative derivative of the error. The learning rate gave us respectively the size of the change of these parameters (small or big step). The gradient descent calculation was applied to each step-iteration of the algorithm, redefining it. In the case of gradient boosting, we do not have direct derivative use, but the relation (2.45), simulates this function through residuals, which show us how to adjust the model to reduce the error (the rest). Below we will study the application of this algorithm to

classification problems and specifically binary classification (prediction between 2 classes). While in the case of regression, the algorithm is similar to linear regression, in the case of classification it is similar to logistic regression.

Suppose we have M decision-regression trees. As in regression accounting, the final amplification model uses the sigmoid function:

$$Pr(y = 1|x, f) \stackrel{def}{=} \frac{1}{1 + e^{-f(x)}} \tag{2.46}$$

where **Pr** gives us a chance for a sample x to be classified as positive (label 1), $\boldsymbol{f(x) = \sum_{m=1}^{M} f_m(x)}$ is sum of the results given by each tree f_m and the sigmoid function $\frac{1}{1+e^{-f(x)}}$, converts this sum, of real values, into a value range [0,1], thus giving a probability.

The optimization criterion here is the likelihood maximization (likelihood) of all the data being sorted correctly or otherwise the log-likelihood maximization that allows us to get a sum versus product, for easier calculations:

$$max \quad L_f = \sum_{i=1}^{N} (y_i \log (Pr(y_i = 1|x, f)) + (1 - y_i) \log (1 - Pr(y_i = 1|x, f))) \tag{2.47}$$

The algorithm starts again by initializing a model-tree $f_0 = \frac{p}{1-p}$, where $p = \frac{1}{N} \sum_{i=1}^{N} y_i$ is prediction of the decision tree as we have seen in previous chapters. Then for each new f_m tree added to the booster model, the derivative of the loss function L_f is calculated in order to define the new y_i values for the new training sets. The derivative is calculated as follows

$$g_i = \frac{\partial L_f}{\partial f} \tag{2.48}$$

This derivative is calculated for each i (i = 1, ... N), i.e. for each sample.

Then we replace the y_i tag values with the g_i tags and the new training set that is created, we use it for the next decision tree.

To combine all the trees together, we need to calculate the ρ_m parameter for each of them as follows:

$$\rho_m = \underset{\rho}{\operatorname{argmax}} L_{f+\rho f_m} \tag{2.49}$$

where $\boldsymbol{\rho_m}$ takes the value for which the log-likelihood $\boldsymbol{L_{f+\rho f_m}}$ of the total booster model $\boldsymbol{f + \rho f_m}$ is maximized. In essence ρ_m functions as a weight that determines how much effect the m-tree model will have on the overall forest model.

After adding model m, the overall model (ensemble model) is updated as follows:

$$f \leftarrow f + \alpha \rho_m f_m \tag{2.50}$$

where the new model has been added and **a** which is the learning rate.

The parameters that can be experimented and searched in this algorithm are the number of trees M, the learning rate a and the depth of the trees. These parameters can affect the performance of the model and therefore their values should be investigated (hyperparameter tuning 2.2.8.2). The depth of the trees, in addition to performance, also affects the speed of training and forecasting. The greater the depth, the slower the algorithm. Through this algorithm the difference between bagging and boosting can be detected. Boosting reduces bias or underfitting, as it tries to correct errors related to the incorrect classification of the whole

training set, but at the risk of increasing the variance and leading to overfitting (2.2.6) . In contrast, bagging reduces model variability by using randomness in terms of training data. Nevertheless, overfitting can be avoided in the case of boosting by finding the appropriate parameters mentioned in the previous paragraph.

The gradient boosting algorithm usually does better than random forest, but it takes more time to train as the trees are sequential (there must have be finished the training of the first tree to start training the next).

2.2.3.10 XGBoost

The XGBoost (eXtreme Gradient Boosting) algorithm is a scalable tree boosting system that uses gradient boosting and was developed by Tianqi Chen and Carlos Guestrin as a research project at the University of Washington.

XGBoost and Gradient Boosting Machines (GBMs) are two collective tree methods that apply the principle of CARTs-classification and regression trees using the gradient descent architecture. However, XGBoost improves the basic GBM framework through system optimization and algorithmic enhancements.

System Optimazations:

- **Parallelism**
  Tree learning needs data in a formatted form according to attribute values, so that the algorithm can visit the data to collect the gradient statistics and list all possible separations for the continuous attributes. To reduce the sorting cost, the data is divided into compressed blocks (each column with a corresponding attribute value). XGBoost sorts each block in parallel using all available CPU cores / threads. This optimization is valuable as a large number of nodes are often created in a tree. In short, XGBoost parallels the sequential tree creation process.

- **Offline kernel calculation**
  Allows out-of-core computing for very large data sets that do not fit in memory and to optimize available disk space.

- **Cache Aware**
  With cache-aware optimization, gradient statistics (direction and value) are stored for each separator node in an internal buffer of each thread and accumulated in a mini-batch manner. This helps reduce the hassle of instant read / write tasks and also avoids cache loss. Cache awareness is achieved by selecting the optimal block size (generally 2^{16}).

Algorithmic enhancements:

- **Column (feature) subsampling**
  XGBoost uses some of the features every time, just like in Random Forest. The use of column sub-sampling prevents overfitting even more than traditional series sub-sampling (which is also supported). The use of column sub-samples also speeds up the calculations of the parallel algorithm.

- **Loss Function**
  While Gradient Boosting follows negative gradients to optimize the loss function, XGBoost uses the Taylor extension to calculate the value of the loss function for different base students.

- **Tree pruning**
  Pruning is a machine learning technique to reduce the size of regression trees by replacing nodes that do not help improve leaf sorting. The idea of pruning a regression tree is to avoid overfitting of the training data. The most effective method for pruning is Cost Complexity or Weakest Link Pruning which internally uses mean square error, k-fold cross-validation and learning rate. XGBoost creates knots / splits to the specified maximum depth (max_depth) and starts pruning backwards until the loss is below a threshold. For example, in a split that has a loss of -3 and the next node has a loss of +7, XGBoost will not remove the split just by looking at one of the negative losses. It will calculate the total loss (-3 + 7 = +4) and if it proves positive it will keep both.

- **Sparsity awareness**
  It is very common for the data we collect to be sparse (many missing or empty values) or to become sparse after mechanical data execution (attribute coding). In each tree, to know the patterns of dilutions in the data, a default direction is assigned. XGBoost handles missing data by assigning it in the default direction and finding the best attribution value to minimize training loss. Othewise, it automatically learns the best value missing depending on the loss of education and handles different types of sparse data patterns more effectively.

- **Regularization**
  It imposes sanctions on more complex models through the normalization of both LASSO (L1) and Ridge (L2) to avoid overfitting. These methods are presented in Chapter 2.2.7.

- **Weighted Quantile Sketch**
  XGBoost uses the distributed weighted Quantile Sketch algorithm to effectively find the optimal separation points between weighted datasets.

- **Approximate Split Algorithm**
  XGBoost enables, in addition to the exact greedy algorithm used by Gradient Boosting to find the separations, to use an approximate algorithm that first proposes the weighted separation points according to the percentages of the feature distribution (weighted quantile sketch).Subsequently,it maps the continuous characteristics into buckets separated by these candidate points, compiles the statistics and finds the best solution between the proposals based on the aggregate statistics.

- **Cross-Validation**
  The algorithm is accompanied by a built-in cross-validation method in each iteration, removing the need for explicit programming of this search and determining the exact number of iterations required in a single execution. More information about this method can be found in Chapter 2.2.8.2.

Thanks to the above improvements, the XGBoost algorithm is faster to execute than other gradient boosting applications. It is also more efficient in terms of memory.

More information about this algorithm can be found here [23].

2.2.4 Scaling of Features

As mentioned above, each input sample x_i, is a vector of characteristic D dimensions. Each attribute is denoted by $x^{(j)}$ με j=1,...,D . These characteristics identify the sample in such a way as to give information to the model, so that it can make a prediction. For example, for an animal the characteristics could be its height, its weight, its color, its fur, etc. These characteristics could be assigned to each input sample, i.e. to each animal, and enter a model that predicts the identity of each input sample, i.e. whether it is a dog, cat, etc. Features

that involve a great deal of information **(informative features)**, also called **features with high predictive power)**.

2.2.4.1 Normalization

Normalization is a scaling technique in which the actual value range of a feature is converted to a specified value range e.g. [0,1] or [-1,1], is also known as **Min-Max scaling** . This is achieved by subtracting from each attribute the minimum value it gets between the training data and dividing by the range of values it gets in the training data (i.e. the maximum minus the minimum value):

$$\bar{x}^{(j)} = \frac{x^{(j)} - min^{(j)}}{max^{(j)} - min^{(j)}} \tag{2.51}$$

From the mathematical definition of normalization results the following three cases:

- When the value of $x^{(j)}$ is the minimum value in the column (the minimum value of the attribute among all samples), the numerator will be 0 and therefore $\bar{x}^{(j)}$ is 0.
- On the other hand, when the value of $x^{(j)}$ is the maximum value in the column, the numerator is equal to the denominator and therefore the value of $\bar{x}^{(j)}$ is 1.
- If the value of $x^{(j)}$ is between the minimum and maximum values, then the value of $\bar{x}^{(j)}$ is between 0 and 1.

If we consider the gradient descent algorithm, in which the partial derivative of the error with respect to the parameter-weights $w^(j)$ is calculated, we will see that if one attribute has a much larger value range than another, then the partial derivative of the former will dominate the update of the parameters. This means that the features with a larger price range will have more power and strength in the outcome of the model and its direction. Thus, setting a specific range for all features will help to avoid this phenomenon.

Normalization also helps to avoid operations between very large or very small numbers, which can lead to computational problems also known as **numerical overlflow**.

2.2.4.2 Standarization

A corresponding method with the normalization is the standardization or otherwise **z-score normalization**. Standardization is another scaling technique where values are centered around the average with a standard unit deviation. This means that the mean of the attribute becomes zero and the resulting distribution has a standard unit deviation (μ = 0, σ = 1). The standardization of features is defined as follows:

$$\hat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sigma^{(j)}} \tag{2.52}$$

where $\boldsymbol{\mu}$ is the average of the attribute values and $\boldsymbol{\sigma}$ is the standard deviation of these values. In this case the prices are not limited to a certain range. The reasonable question here is when we use normalization and when we standardize our data. There is no rule to this question that could be given aw answer. There are only a few indications: Normalization is better used when we know that data distribution does not follow a Gaussian distribution. This can be useful in algorithms that do not require data allocation such as K-Nearest Neighbors and Neural Networks.

Contrarily, standardization can be useful in cases where the data follows a Gaussian distribution. However, this does not necessarily have to be the case. Also, Contrary to

normalization, standardization has no marginal range. So, even if some data have outliers, they will not be affected by standardization, which is better. Also, standardization seems to be more beneficial in unsupervised learning algorithms.

Despite all the above considerations, the choice between standardization and normalization depends on the problem and the algorithm used. A good solution is to try both methods on the model and choose the one that gives the best performance.

Usually the scaler is trained in the training set and it is also applied to the test set. This will prevent any **data leakage** during the model testing process. Data leakage is the phenomenon in which the model receives information outside of the training set which allows him to learn something it would not have known before. As a result, the model becomes invalid or often too efficient, but this performance does not correspond to reality. For this reason, the same scaler used as the training set should be applied to the data set so that the test data does not provide any additional information. Finally, scaling does not apply to target values, i.e. sample labels.

2.2.5 Three Sets

The data sets used to construct and evaluate a model have been mentioned several times in previous chapters. Here we will have the opportunity to clarify the role of each and its importance.

Once we have received the data, the first process that takes place is to break it down into sets. We mix them and divide them into: training set, test set and validation set. The last two sets are also called **hold hold-out sets** because they are not used in model building-training. The ratio by which the sets will be divided is not specific, but there is the informal rule that the set of training will be the largest, since to train a model requires many samples (for example 70 % training, 15 % test, 15 % validation). The separation ratio also depends on the amount of data we have. If, for example, the data set consists of millions of samples, then an even higher percentage can be given to the training set (90 %) and the other two sets to take what is left (5 % and 5 %).

In more detail, the operation of each set:

- **Training Set**

  The training set is what is used to model training. It is basically a set of examples used to customize the parameters of the model. In practice, the training data set often consists of pairs of an input vector and the corresponding output vector. Model prediction is usually called goal (or tag). The current model is run with the training data set and produces a result-prediction, which is then compared to the target, for each sample entry in the training data set. The parameters of the model are adjusted, based on the result of the comparison and the specific learning algorithm used.

- **Validation Set**
  The validation set is used to evaluate the model, in order to help find the optimal values of the hyperparameters (hyperparameter tuning), as well as the optimal training algorithm for the specific problem.

  Initially, models are trained that use different hyperparameter values or different learning algorithms. The input vectors (x_i) of the validation set are inserted into the trained models, then the models make the corresponding predictions and these predictions are compared to the output vector of the set (y_i). As a result, a performance of each model is calculated on the validation data. This gives an idea of which hyperparameter values or which training algorithms work best for each problem.

Validation data sets can also be used during training of a model (stop training when the error in the validation data set increases, as this is an indication that the model is overfitting the training data set).

- **Test Set**
  The test set is used for the final evaluation of the model selected as the best among the tested models. It basically gives a representation of how well the model is doing in data it has never used before, that is, how well it can generalize. To do this, the final model is used to predict the labels of the examples of the set of tests. These predictions are compared with the true labels of the examples to evaluate the accuracy of the model.

In summary, if the most appropriate classifier is been looked for for a problem, the training data set is used to train the various candidate classifiers, the validation data set is used to compare their performance and decide who will be selected, and finally the test data set is used to obtain performance characteristics such as accuracy, sensitivity, F score and so on. If one wonders why not use a single set to do all the functions, the answer is simple. When we build a model, what we do not want is for the model to perform well only in predicting example tags that learning algorithms have already used. An algorithm that simply memorizes all the training examples and then uses the memory to "predict" their labels will not make mistakes when asked to predict the labels of the training examples, but such an algorithm would be useless in practice. What we really want is for our model to provide good examples that the learning algorithm has never used before. So we want good performance in hold-out sets. Finally, there is a separation between validation and test set. The reason why we use two separate such sets is because in addition to the training data, the model can also be overfitted when finding optimal hyperparameters, ie when using the validation set. For this reason we need to test the performance of the model with the optimal hyperparameters, in an independent set (test set). The phenomenon of overfitting will be analyzed in the next chapter.

2.2.6 Underfitting and Overfitting

When the model makes many mistakes in the training data (training set) then we say that it presents **underfitting** or otherwise that it has **high bias**. In this case, he is not able to correctly predict the data labels on which he was trained.

The reasons that can lead to this phenomenon are either that a very simple model has been chosen for the data we have, which is not able to classify it correctly (for example a linear model when the data is not linearly separable , or that the data characteristics do not provide enough information for the model to draw conclusions from it.

Underfitting is usually easily detected by a good metric performance. If this phenomenon is detected, the solution is to test alternative learning algorithms, or to test the features and enrich them on a case-by-case basis.

Overfitting

Conversely, when the model shows **overfitting** i.e. excessive adjustment **high variance**, then it haw very good results in the training data but not in the validation or test data. In this case, the model does not have the ability to generalize, that is to produce accurate predictions for data it has never used before. The essence of the over-adaptation is to have unknowingly extracted part of the remaining variant (i.e. noise) as if this variant represented the underlying structure of the model. In other words, the model remembers a huge number of examples instead of learning to observe features. This means that if training data had been sampled differently, learning would have led to a very different model. This is why the

overfitting model performs poorly on the test data: the test and training data are sampled from the data set independently of each other. The reasons that can lead to the phenomenon are either the model is too complex for the data (for example a large decision tree or a very deep neural network), or the features are many in relation to the small volume of training data.

Below are some solutions that can be given to reduce overfitting:

1. The use of a technique by which we re-sample data to evaluate the performance of the model. One such technique is cross-validation which will be presented in chapter 2.2.8.2.
2. The use of a validation set, which we described in 2.2.5, to evaluate the performance of models in new data that the model has never used before.
3. The test of simpler models (for example linear versus polynomial regression or SVM with linear kernel versus RBF kernel).
4. The reduction of sample dimensions in the data set. That is, the reduction of features.
5. Adding more training data.
6. The model regularization, which we will describe in the next chapter 2.2.7.

The following is an example of a linear regression problem with three cases [24]:

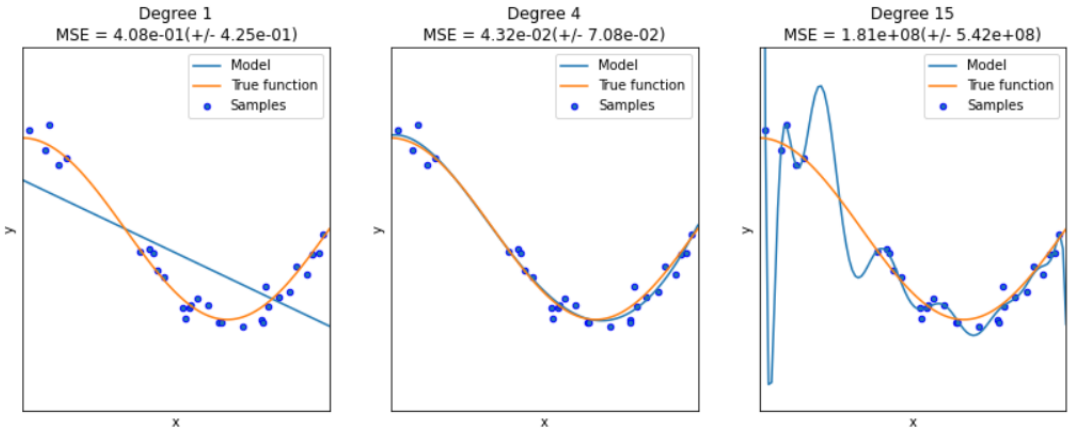


Figure 2.17: Example of overfitting-underfitting in linear regression

In the first graph left, a 1st degree polynomial has been used. As we can see, the model formed by this polynomial (blue line) does not approach the samples. It seems that it isn't able to make accurate predictions in the training data and thus presents **underfitting**.

In the second graph we see a polynomial model of degree 4. Here the model is quite close to the samples and is similar to the real function (orange line). In this case we assume that we have **"Best Fit"** or otherwise **"Good Model"**. In the third graph on the right, a 15th degree polynomial has been used and as we can see the model has become quite complex. It passes through almost every sample which shows excessive adaptation to the training data. In this case, if the model is tested on new data that does not resemble those of the training, then it is most likely to give a poor performance. Here the model features **overfitting**.

It is worth noting that the error (MSE) decreases, as the degrees of the polynomial increase, which is to be expected as the model adapts more and more to the training data. Thus it is obvious that a smaller error in the training data does not necessarily mean that

the model is better (since the right-hander model with the smallest error is not the best). For this reason, the other data sets (validation and test) are important in evaluating performance. By the term regularization we mean the methods that try to force the learning algorithm to create a less complex model, in order to avoid the risk of overfitting. This may increase the bias a bit but it will definitely reduce the variance. In other words, regularization is any modification one makes to a learning algorithm designed to reduce generalization error but not training error. This problem is also known as bias-variance tradeoff.

To create a normalized model, we modify the objective function / cost function by adding a punitive term whose value is higher when the model is more complex.

The most commonly used normalization techniques are L1 normalization **(L1 regularization)** and the normalization L2 **(L2 regularization)**. We will test these techniques in the problem of linear regression:

The cost function we want to minimize for linear regression is the following, as we have seen before:

$$RSS(w,b) = \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2$$

A linear regression problem that uses **L1 regularization** is called **regression LASSO (Least Absolute Shrinkage and Selection Operator)**. The objective function in this case is as follows:

2.2.7 Regularization

By the term regularization we mean the methods that try to force the learning algorithm to create a less complex model, in order to avoid the risk of overfitting. This may increase the bias a bit but it will definitely reduce the variance. In other words, regularization is any modification one makes to a learning algorithm designed to reduce generalization error but not training error. This problem is also known as bias-variance tradeoff.

To create a normalized model, we modify the objective function / cost function by adding a punitive term whose value is higher when the model is more complex.

The most commonly used normalization techniques are L1 normalization **(L1 regularization)** and the normalization L2 **(L2 regularization)**. We will test these techniques in the problem of linear regression:

The cost function we want to minimize for linear regression is the following, as we have seen before:

$$RSS(w,b) = \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2$$

A linear regression problem that uses **L1 regularization** is called **regression LASSO (Least Absolute Shrinkage and Selection Operator)**. The objective function in this case is as follows:

$$RSS_{LASSO}(w,b) = RSS(w,b) + \alpha\sum_{\{j=1\}}^{p}|w^{(j)}| \tag{2.53}$$

where the first term of the sum is the same as the cost function of the normal linear regression and the second term is equal to the sum of the weights of all the features on a variable α, which determines how much effect normalization has on the model .

If $\alpha = 0$ is set then there is no normalization and the problem is simple linear regression. Conversely, if parameter α gets a large value then the model will try to minimize the term $\sum_{\{j=1\}}^{p} |w^{(j)}|$, in order to minimize the whole cost function, that results in many weights taking small values or even seting equal to zero. Substantially, L1 regularization functions as a feature selector, which selects the features that are most important to predict, subtracting the rest. This automatically means that the model will become simpler and that can lead to underfitting. Therefore, parameter α is set to be investigated and an appropriate value must be found, which will reduce the overfitting, without greatly increasing the bias (underfitting). A linear regression problem that uses **L2 regularization** is called **RIDGE regression**. The objective function in this case is as follows:

$$RSS_{RIDGE}(w,b) = \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2 + \alpha \sum_{\{j=1\}}^{p} (w^{(j)})^2 \implies \tag{2.54}$$

$$RSS_{RIDGE}(w,b) = RSS + \alpha \sum_{\{j=1\}}^{p} (w^{(j)})^2 \tag{2.55}$$

where again the first term of the sum is the same as the cost function of the normal linear regression while the second term here is equal to the sum of the squares of the weights of all the features on a variable $\boldsymbol{\alpha}$, which determines how much effect has the normalization in the model.

The L2 regularization technique usually gives better results than the L1 in maximizing the performance of the model in the test data. Also, the factor L2 in the cost function is derivable, which means that the gradient descent algorithm can be used as an optimization method. Other normalization techniques used in neural networks are dropout, batch-normalization, early stopping, and data augmentation.

2.2.8 Model Performance Assessment

This chapter will present the metrics that can be used to evaluate a classification model.

2.2.8.1 Confusion Matrix

The confusion matrix is an NxN table that shows the successes and failures of the model in its predictions, by class. More specifically, the table has as rows the actual labels of the samples (N) and as columns the labels provided by the model (N). Considering a binary problem with two classes, the confusion table is 2x2 dimensions and contains the following 4 terms:

- **True Positives(TP)**
  The true positives are in position (0.0) of the table and are the number of samples that were positive (label 1) and predicted as positive by the model.
- **False Positives(FP)**
  The false positives are in position (0.1) and are the number of samples that were negative (label 0) and were predicted as positive by the model. item**False Negatives(FN)**
  The false negatives are in position (1.0) and are the number of samples that were positive (label 1) and predicted as negative by the model.
- **True Negatives(TN)**
  The true negatives are in position (1.1) and are the number of samples that were negative (label 0) and predicted as negative by the model.

The following is an example of a confusion table to better understand it:

Predicted classes / Actual classes	Cat	Non-Cat
Cat	**9(TP)**	4(FN)
Non-Cat	3(FP)	**4(TN)**

Table 2.1: Example of a Confusion Matrix in a binary problem

The above problem concerns the classification into cats and not cats. As can be seen from the table 9 samples were cats and predicted as cats, 3 samples were not cats and predicted as cats, 4 samples were cats and predicted as non-cats and 4 samples were not cats and predicted as non-cats. The total number of samples is 9+3+4+4=20, while the samples that were sorted-predicted correctly by the model are those presented in the diagonal, the true positives and true negatives => 9+4 = 13.

This metric can help us identify specific errors that the model tends to make. For example, if the problem was object recognition and we saw many of the samples in the table that are actually glasses to have been incorrectly classified as bottles, then this shows that the model has not learned to well distinguish the bottles from the glasses. In this case we could help the algorithm by adding more highlighted bottle samples or adding more sample features to help it better distinguish these items.

The following important metrics emerge from the confusion table.

2.2.8.1.1 Precision

The **precision** is also called **positive predictive value (PPV)** (accuracy or positive predictive value) and is the percentage of positive samples that were correctly predicted (i.e. were actually positive), of all samples predicted as positively. That is,according to the values of the confusion matrix,it is defined as follows:

$$Precision = PPV = \frac{TP}{TP + FP} \tag{2.56}$$

From (2.48), it appears that the precision is calculated by dividing the true positives, i.e. the number of positive samples predicted correctly, by the sum of the true positives and false positives, i.e. all the samples predicted as positive .

2.2.8.1.2 Negative Predictive Value

The **negative predictive value (NPV)** or otherwise the negative predictive value, is the equivalent of PPV but for negative samples. That is, the percentage of negative samples that were correctly predicted, out of all the samples that were predicted as negative:

$$NPV = \frac{TN}{TN + FN} \tag{2.57}$$

From (2.49), it appears that the NPV is calculated by dividing true negatives, i.e. the number of negative samples predicted correctly, by the sum of true negatives and false negatives, i.e. all samples predicted as negative .

2.2.8.1.3 Recall

The **recall** or otherwise **sensitivity** or otherwise **true positive rate (TPR)** (recall or sensitivity or true positive rate), is the percentage of true positive samples predicted correctly:

$$Recall = Sensitivity = TPR = \frac{TP}{TP + FN} \tag{2.58}$$

From (2.49), it appears that Recall is calculated by dividing the true positives, i.e. the number of positive samples correctly predicted, by the sum of the true positives and false negatives, i.e. all the real positive samples.

2.2.8.1.4 Specificity

The **specificity** or otherwise **true negative rate (TNR)** (specificity or true negative rate), is the equivalent of recall but for negative samples, i.e. is the percentage of true negative samples that were correctly predicted:

$$Specificity = TNR = \frac{TN}{TN + FP} \tag{2.59}$$

From (2.51), it appears that Specificity is calculated by dividing the true negatives, i.e. the number of negative samples predicted correctly, by the sum of the true negatives and false positives, i.e. all the real negative samples.
Usually, in practice, we are called to choose between a high precision or a high recall, as it is difficult to have both. The way we choose is proportional to the problem we have to face. For example, if we want to recognize spam messages from non-spam ones, then what we are most interested in is having a higher precision, at a cost to have a lower recall. This is because it is more important for us not to delete normally-important messages because they have been incorrectly predicted as spam (precision), than to accept spam messages which have been incorrectly considered normal (recall).

2.2.8.1.5 F1-score

Many times we try to achieve both good precision and good recall. This pursuit is summarized in a metric called **F1-score** and is the harmonic mean of the recall and precision values:

$$F1 = 2\frac{Precision \quad Recall}{Precision + Recall} \tag{2.60}$$

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{2.61}$$

The reason a harmonic means is used, as opposed to a numerical or geometric one, is because the harmonic means punishes the most extreme values.

There are, however, situations for which a data scientist would like to give more importance / weight to either precision or recall. For this reason there is the most general form of **F-score** or F_β shown below:

$$F_\beta = (1 + \beta^2)\frac{Precision \quad Recall}{(\beta^2 Precision) + Recall} \tag{2.62}$$

$$F_\beta = \frac{(1 + \beta^2)TP}{(1 + \beta^2)TP + \beta^2 FN + FP} \tag{2.63}$$

where **β** is a positive real number, which allows the metric to be adjusted to check the significance of the recall versus the precision. More specifically, β is chosen so that recall is considered β times more important than precision. For example, if we wanted to give more importance to precison we could set β=0.5 so that false positives (FP) have more weight than false negatives (FN), while if we wanted to give more importance to recall we could set β=2, so that false negatives (FN) have more weight than false positives (FP).

2.2.8.1.6 Accuracy

The accuracy **(accuracy-ACC)**, is the percentage of the total number of predictions that were correct. Regarding the symbols of the confusion table, the accuracy is defined as follows:

$$Accuracy = ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.64}$$

where in (2.56), the numerator is the sum of the number of correctly predicted samples, i.e. the true positive samples plus the true negative samples which were predicted correctly and the denominator is the sum of all samples (true positives and true negatives and false positives and false negatives).

This metric is useful when the prediction errors of all classes are equally significant. In the problem of spam detection, for example, which we mentioned above, where we are more interested in precision than recall, the metric of accuracy cannot give us a clear picture. In this case, however, the metric **cost-sensitive accuracy** can be used, which uses a function similar to the general form of f-score. In essence, we place a cost (positive number) on both types of errors (false positives and false negatives), before placing them in the mathematical expression (2.64).

The following table summarizes the metrics described so far:

Confusion Matrix		**Model**			
		Positive	Negative		
Target	Positive	TP	FN	**Recall (Sensitivity-TPR)**	TP/(TP+FN)
	Negative	FP	TN	**Specificity (TNR)**	TN/(TN+FP)
		Presicion (PPV) TP/(TP+FP)	**Negative Predictive Value (NPV)** TN/(TN+FN)	**Accuracy (ACC)** = (TP+TN)/(TP+TN+FP+FN)	

Table 2.2: Model-classifier evaluation metrics

2.2.8.1.7 AUC-ROC

This metric is called the area under the curve (AUC) or area under the ROC curve, where ROC comes from the "receiver operating characteristic", and means characteristic receiver operating curve. This metric uses two terms:

1. **True Positive Rate (TPR)**
   We used TPR above. It is the metric recall, which represents how many positive samples were correctly predicted.

$$TPR = \frac{TP}{TP + FN}$$

2. **False Positive Rate (FPR)**
   FPR represents how many of the negative samples were predicted incorrectly:

$$FPR = \frac{FP}{FP + TN}$$

   It is also equal to the 1-specificity (TNR) we used in 2.2.8.1.4.

This metric refers to learning algorithms that return a confidence score or probability. Such algorithms are logistic regression, decision trees, as well as all the collective learning algorithms we have seen, which use decision trees.

The curve is formed by a graph having FPR on the x-axis and TPR on the y-axis. These value pairs are calculated for different values of the decision threshold. In the case for example when the algorithm returns a probability between [0, 1], the threshold values can be divided by 0.1 in this interval (i.e. [0, 0.1, 0.2, 0.3, 0.4, 0.5 ,0.6, 0.7, 0.8, 0.9, 1]). The algorithm predicts the probability that the samples are positive and then for each of the threshold values, calculates the metric FPR and TPR. Then the value pairs of these metrics are represented in a graph.

An example is shown in the following graphic [25]:

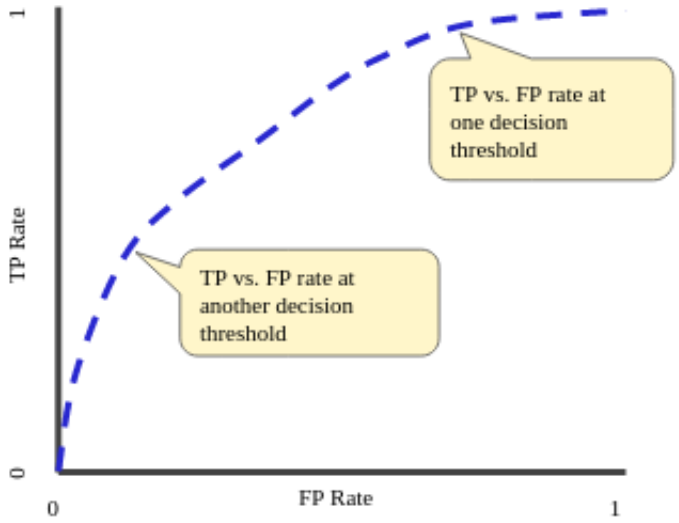


Figure 2.18: Metric AUC

If the threshold is equal to 1, then the samples will be considered positive for a probability greater than 1, so all samples will be considered negative and thus the terms FPR and TPR will be zeroed. This is because none of the negative samples will have been predicted incorrectly (hence FPR = 0) and none of the positive samples will have been predicted correctly (hence TPR = 0). Conversely, if the decision threshold is equal to 0, then the samples will be considered positive for a probability greater than 0, so all samples will be considered positive and so the terms FPR and TPR will become equal to 1. This is because all negative samples will have been predicted incorrectly (hence FPR = 1) and all positive samples will have been predicted correctly (hence TPR = 1).

Our goal is to have as many correctly sorted samples as possible, i.e. as higher TPR as possible and as few incorrectly sorted negative samples as possible, i.e. as lower FPR as possible. This is summed up in one goal, which is to have as much surface area as possible under the curve formed by the two metrics.

A random classifier can be considered to give AUC = 0.5, because in essence the more positive samples it classifies correctly, the more negative samples it classifies incorrectly (TPR = FPR). To have a better classifier than random we have to get AUC> 0.5, while if we get AUC <0.5, it means that our classifier is even worse than a random classifier, so something is wrong.

Below are various instances of the ROC curve, as well as the curve in the case of the random classifier [26]:

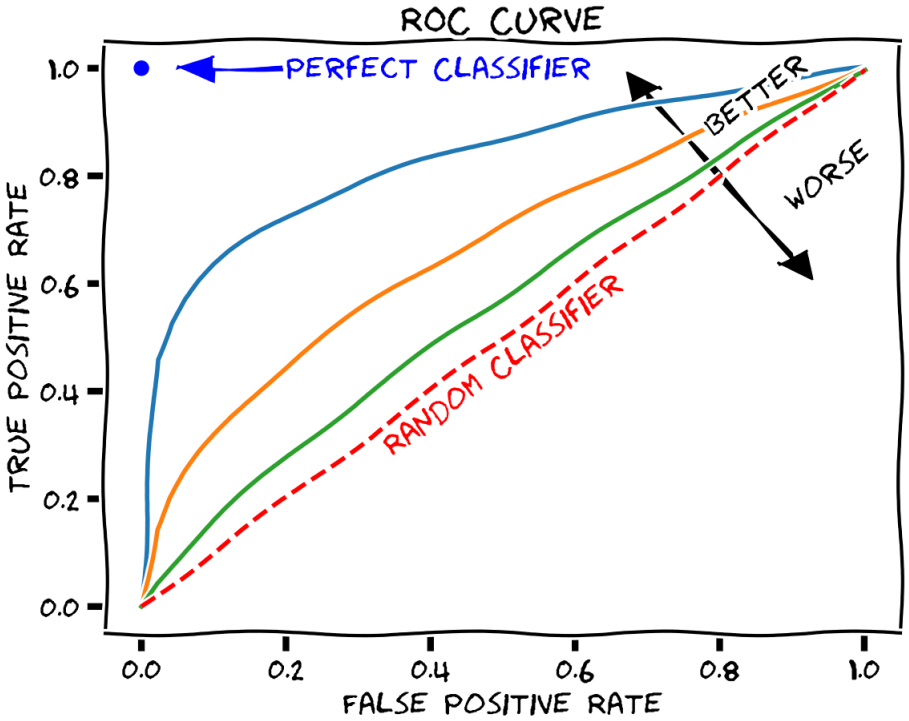


Figure 2.19: The ROC space for a "better" and "worse" classifier.

As we can see from the figure 2.19, the perfect classifier, i.e. the best case, is shown in the upper left corner where TPR is equal to 1, FPR is equal to 0 and the area below the curve will have the maximum value. In addition, the curves above the random classifier curve (AUC = 0.5) are getting better as we move away from it, as the area under the curve grows, while the curves below the random classifier curve are getting worse , as we move away from it, as the area under the curve decreases.

All of a classifier evaluation metrics presented in the previous subsections were in the case of a binary classifier. These metrics can easily be calculated for problems with more than two classes, if the class we are interested in is considered as positive and all the others are considered negative.

Finally, there are other similar metrics that can be derived from the terms of the confusion matrix. A summary of all metrics is presented below [27]:

		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Figure 2.20: Summary of metric evaluation metrics

2.2.8.2 Hyperparameter Tuning

Hyperparameter tuning or **hyperparameter optimization**, has been mentioned several times in previous chapters. It concerns the selection of the optimal hyperparameters for the learning algorithm. The term hyperparameters refers to the parameters used to control the learning process and not to the parameters that are learned during training, such as weights. Such hyperparameters are: the height of the tree in a decision tree, the parameter C in the soft margin SVM, the kernel in the SVM with kernel, the parameter α (learning rate) in the gradient descent algorithm, etc.

Grid Search

The method used to optimize the hyperparameters is called **grid search** and is a thorough search of parameter values, from a specific subset of the set of values of these parameters. In short, hyper-parameter combinations from a range of values are tested, training models and then the combination that gives the model the best performance is selected.

To fully understand, we will give an example. Suppose we have a soft margin SVM with a kernel. The hyperparameters that need to be optimized to give the model better performance are parameter C and the kernel to be used. So we choose two subsets of values for each of these hyperparameters as follows: $C \in \{10, 100, 1000\}$ and $kernel \in \{linear, rbf\}$. From these values result 6 combinations that should be tested. With each of these combinations we train a model (SVM). Here it is important to note that in case we do not know the possible range of values of the hyperparameters (for example parameter C), the most common phenomenon is to use a logarithmic scale.

Then each of the resulting models should be evaluated with one of the metrics mentioned in chapter 2.2.8. This evaluation can be done either in the validation data (validation set), if any, or in the training data (training set) with the method **cross validation**, which will be analyzed below.

After selecting the model with the combination of hyperparameters that gave the best performance, the overall performance of the model (regardless of the hyperparameters) is evaluated in the test set.

Apart from grid search, there are other hyperparameter optimization techniques such as: random search, bayesian hyperparameter optimization, gradient-based optimization, evolutionary optimization, population-based, early stopping-based etc. [28]. Grid search is a rather time consuming technique as it thoroughly tests all possible combinations. In contrast, other techniques are more time efficient, such as random search, which randomly selects hyperparametric values from a statistical distribution and creates as many combinations as

we specify.

Cross Validation

So, in case we do not have a validation set, since if the samples are too few, we cannot "waste" samples from the training set to make the validation set, the technique that can be used to evaluate its performance model is **cross validation**. Basically, we use cross validation in the training set to simulate a validation set.

A cross-validation cycle involves splitting a data set into complementary subsets, performing the analysis on one subset (called a training set), and validating the analysis on the other subset (called a validation set). To reduce variability, most methods run multiple rounds of cross-validation using different partitions and the validation results are combined (e.g. on average) in rounds to give an estimate of the predictive performance of the model.

k-Fold Cross Validation

The k-fold cross-validation formula works as follows: first, we specify the values of the hyperparameters we want to evaluate. Next, we divide the training set into several subsets of the same size. Each subset is called a fold. Usually, 5-fold cross-validation is used in practice. With 5-fold cross-validation, we randomly split the training data into five parts: $\{F_1, F_2, ..., F_5\}$. Each F_k, k = 1,. . . , 5 contains 20% of the training data. Next we train five models as follows: to train the first model, f_1, we use all the examples from the subsets F_1, F_3, F_4 και F_5 as a training set and the examples from F_1 as a set ratification. To train the second model, f_2, we use the examples from the F_1, F_3, F_4 and F_5 subsets as a training set and the examples from F_2 as a validation set. We continue to create models in this way and calculate the value of the metric we have chosen for evaluation, in each validation set, from F_1 to F_5. Next, we take the average of the five values of the metric to obtain the final performance of the model.

The above procedure concerns only one model, i.e. the testing of only a combination of hyperparameters. If we want to try many different combinations, we repeat the cross-validation for each combination. That is, we essentially use grid search with cross-validation. When we have finally selected the hyperparameters that gave us the best performance through cross-validation, we use the entire data set to retrain the model and then evaluate it in the test set.

The above procedure is summarized in the following figure [29]:

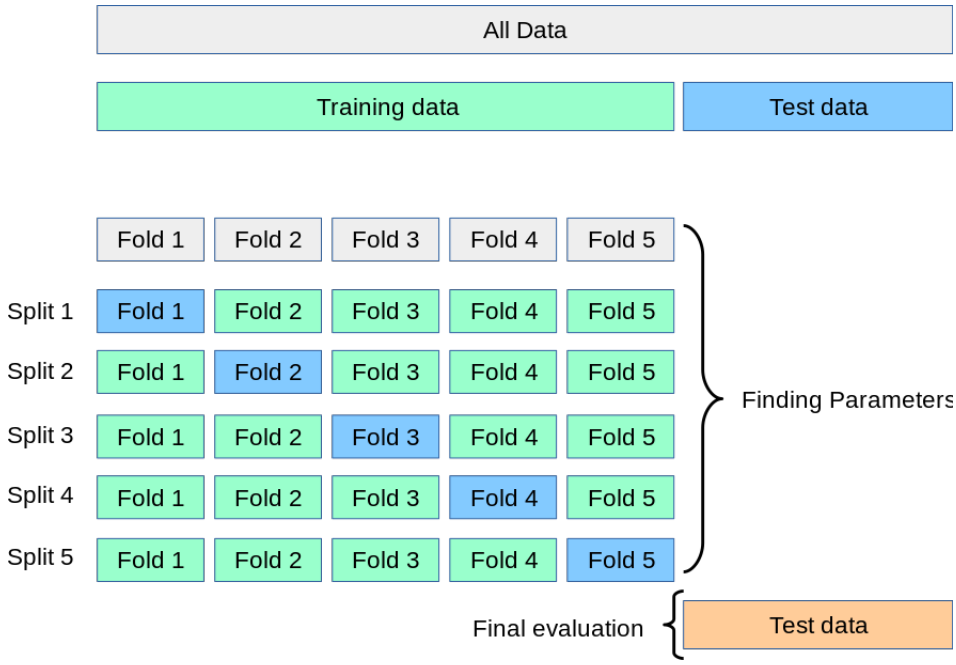


Figure 2.21: k-Fold Cross-Validation

There are different types of cross-validation. The one described above is the regular k-fold cross validation **(regular k-fold cross validation)** where the original data set is randomly split into k equal subsets. Another type is the stratified k-fold cross-validation **(stratified k-fold cross-validation)** where the parts into which we break the whole are selected so that the average response value is approximately equal to all parts. That is, there should be approximately an equal number of samples from each class, in all parts. If for example the problem was binary, then each part (fold) would contain approximately the same proportions of the two classes.

Both of the above types belong to the category **non-exhaustive cross-validation**. Not all possible ways to split the data set are calculated in this category, but a random split is selected. In contrast, **exhaustive cross-validation** includes cross-validation methods that learn and try all possible ways to split the original set into a training set and a validation set [30].

2.3 Artificial Neural Networks

This chapter will introduce you to artificial neural networks, so that the algorithms that will be used to extract features from text can be understood, as well as deep learning neural networks, such as the one described in next chapter. **artificial neural networks –(ANN)** or **neural networks (NN)**, resemble the neural networks of a biological brain, hence their name. They consist of layers, where each layer contains a number of nodes or artificial neurons that behave like biological neurons. Neurons exchange signals with each other, from one level to the next, through some synapses-edges. An artificial neuron that receives a signal then processes it and can signal neurons associated with it. The "signal" in a connection is a real number and the output of each neuron is calculated from some nonlinear function of the sum of its inputs. Each joint acne is usually accompanied by a weight, which adapts to the training. These weights reduce or increase the strength of the signal transmitted through this edge. Also, different layers can perform different transformations at their inputs. The signals travel from the first layer-layer (the input layer) to the last layer (the output layer), probably after passing the layers several times.

The way each layer works is none other than what we have mentioned in accounting regression. In essence, we have the mathematical expression $W_l x + b_l$, where W_l and b_l are the parameters for each layer l, which are set to be optimized during training, through the abrupt descent algorithm and the optimization criterion, as we have mentioned before. W_l consists of the weights $w_{l,u}$, where l is the each layer and u is the neuron-unit of this layer. Each $w_{l,u}$ has so many dimentions as the dimensions of the input samples x. That is, it is a matrix (number of samples x number of attributes). So each layer has as many weights as its neurons. B_l respectively consists of $b_{l,u}$ for each neuron u of layer l and is a vector of as many dimensions as the input samples x_i. This expression then passes through an activation function g_l, which is a scalar function, that is, it returns a prime number and not a vector, as in the case of accounting regression, in the relation (2.5), which had the sigmoid function. Finally, the output of layer l is $f_l(x) = g_l(W_l x + b_l)$, which is a vector of as many dimensions as the unit neurons in layer l.

Multilayer Perceptron

To better understand the neural network architecture described above, a specific type of architecture called **multilayer perceptron (MLP)** or **vanilla neural network** will be graphically represented [13]:

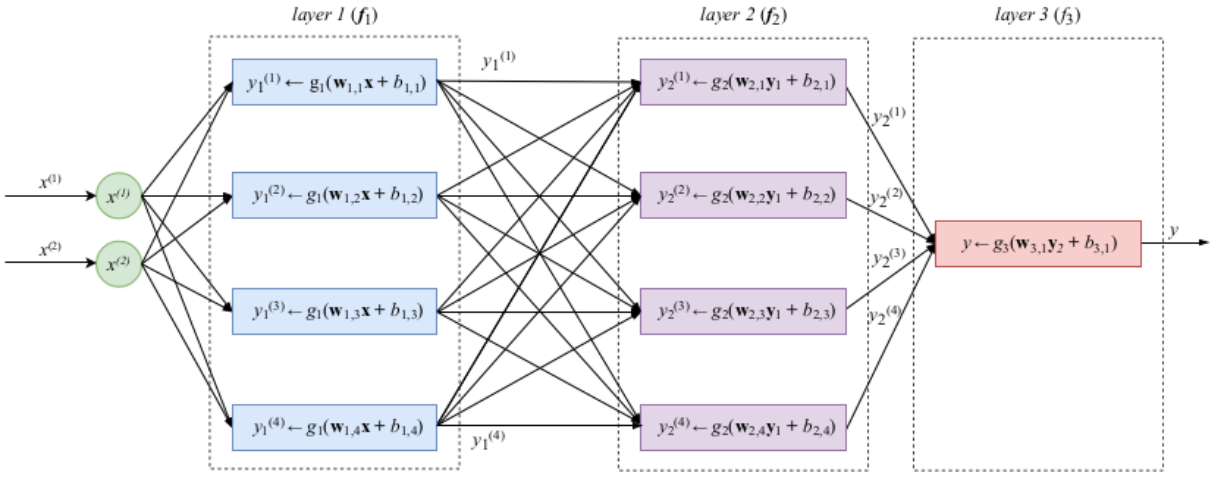


Figure 2.22: Example of a multi-layered perceptron with two layers

In the figure 2.22, we see an MLP, in which the input data x is 2 dimensional $x^{(1)}$ και $x^{(2)}$. Both samples are fed at the first level (layer 1) to each of the neurons (from 1 to 4). Each neuron first calculates the linear combination of the input samples $w_{l,u}x + b_{l,u}$ and then applies the activation function to it $g_l(w_{l,u}x + b_{l,u})$. From each neuron emerges the output $y_1^{(u)}$, where u is the index of the neuron. All these outputs are inserted into the neurons of the next level, where the new outputs $y_2^{(u)}$, calculated in the new weights $w_{2,u}$ and $\beta_{2,u}$, are calculated accordingly and the new activation functions g_2. Finally, the outputs of the 2nd level are inserted in the output level which contains a single neuron and with corresponding procedures extracts the output-prediction y.

It is important to mention the following:

- The number of neurons may vary from level to level. Here both levels happened to have 4 neurons.
- Intermediate levels that are neither input nor output are called hidden layers.
- The activation function is usually the same for all neurons in a level and may vary by level. But this is not the rule. It is possible to have different activation functions between neurons of the same level.
- There are many different activation functions that can be used, which are usually derivable so that the gradient descent algorithm can be applied to find the optimal values of the parameters w and b.
- The reason why activation functions are used is to have nonlinear components that allow the neural network to access nonlinear functions. Otherwise, the relations $w_{l,u}x + b_{l,u}$ would always lead to linear combinations.
- In the multilevel perceptron, all the outputs of one level are connected to each input of the next (with each neuron). These layers are called **fully connected layers**, and a neural network does not have to be made up of just such.
- To update the parameter values of a neural network, we use the training algorithm **backpropagation**, which consists of two steps: 1) feed forward 2) calculate the error and transmit it to previous levels. In essence, when running the gradient descent algorithm, the neural network parameters receive an update, proportional to the partial derivative of the cost function relative to the current parameter, at each iteration of

the training. In backpropagation, these derivatives are calculated from back to front (from the last level to the first), with the chain rule.

Activation Functions

The most commonly used activation functions are the following:

- **Softmax Function**
  The softmax function is applied to classification problems with multiple classes (more than two). In the case of multiple classes, in the output layer we mentioned above, there would be not just one neuron but as many neurons as the classes to predict. The mathematical expression of this function is as follows:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.65}$$

  where $\sigma(z)_i$ is the probability that the sample belongs to class i. In essence, z_i are the resulting scores for each class. Applying the standard exponential function to the score of each class i and dividing by the sum of the exponential scores of all the classes, we finally get a value in the interval [0,1] which reflects the probability. The sum of all $\sigma(z)_i$ for each i, i.e. for each class, is equal to 1. This means that the total probability of sorting the sample into classes is 1. A subcategory of the softmax function is the sigmoid or logistic activation function, which we mentioned in the case of the accounting regression. This function is used for binary sorting problems since we only have one output probability that indicates the probability that the sample will be sorted as positive:

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{2.66}$$

- **Tanh or Hyperbolic Tangent**
  The tangent activation function is similar to the sigmoid with the difference that it has a range (-1.1).

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.67}$$

  The following is the tanh function along with sigmoid/logistic [31]:

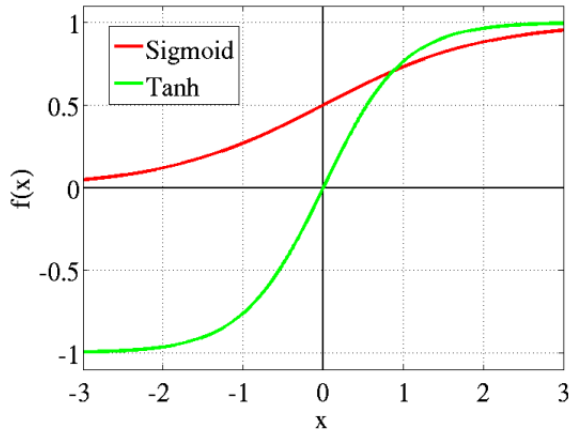


Figure 2.23: Tanh vs Logistic/Sigmoid

- **ReLU (Rectified Linear Unit)**
  The function of a corrected linear unit is equal to 0 when the score z is negative, otherwise it is equal to the score z:

$$relu(z) = \begin{cases} 0, & \text{if } z < 0 \\ z, & \text{otherwise} \end{cases} \tag{2.68}$$

The following is the function relu together with sigmoid/logistic [31]:

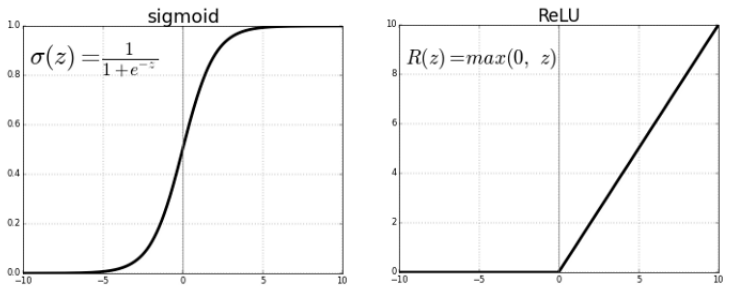


Figure 2.24: Relu vs Logistic / Sigmoid

2.4 Convolutional Neural Networks

Deep Learning refers to the training of networks that have more than 2 hidden layers. This is why these neural networks are called deep neural networks.

Convolutional Neural Networks (CNN) are a type of neural network that has found application in the fields of image and text analysis. So, because CNNs were invented with image analysis, they will be explained on an example of this kind of classification.

Edge Detection

The images are represented by pixels. Usually, the neighboring pixels refer to the same information, for example they represent a face, a cloud, water, etc. But there are also edges, i.e. the points where two different objects touch each other. So if the neural network learns to recognize both regions with the same information and edge regions, it will be a useful pathway that will allow it to predict the object represented in the image. [32]

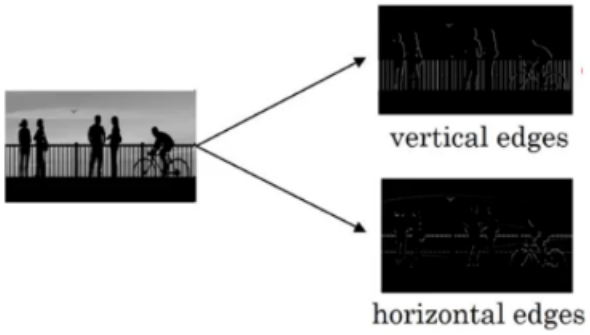


Figure 2.25: The edges of an image

In this way, the logic considering a CNN is that the first layers of a neural network locate edges of an image. Deeper layers may be able to detect the existence of objects and even deeper layers may detect the existence of complete objects (such as a person's face).

The edges can be horizontal or vertical and can be distinguished with the help of filters or otherwise cores (filter / kernel), which are in the form of tables. Such filters are shown below [32]:

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

Figure 2.26: Vertical and Horizontal Filter

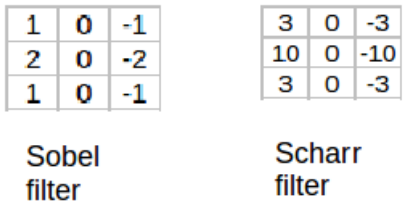


Figure 2.27: Sobel and Scharr Filter

The way these filters are applied to an image is by compiling the representation of the image in pixels with the respective filter. For example, if we have a 4 dimension image and a 2×2 filter, the result of the convolution will be a 3×3 dimension table. This table is obtained by first taking the upper left part of the image, dimension 2×2, multiplying it by the element-wise filter and adding the values of these products. From these operations only one number will emerge which will be the first element of the 3×3 result table. Then we repeat the same process, moving the filter one step to the right, above the original image, ie taking the immediately right part 2×2 of the image and applying the same operations. These iterations end when we have covered the full image, from right to left and from top to bottom, and a 3×3. times table has emerged. Each filter is also accompanied by a bias parameter, which is added to the result of the compilation of each step. An example of the above calculations with the dimensions we gave, is presented below [13]:

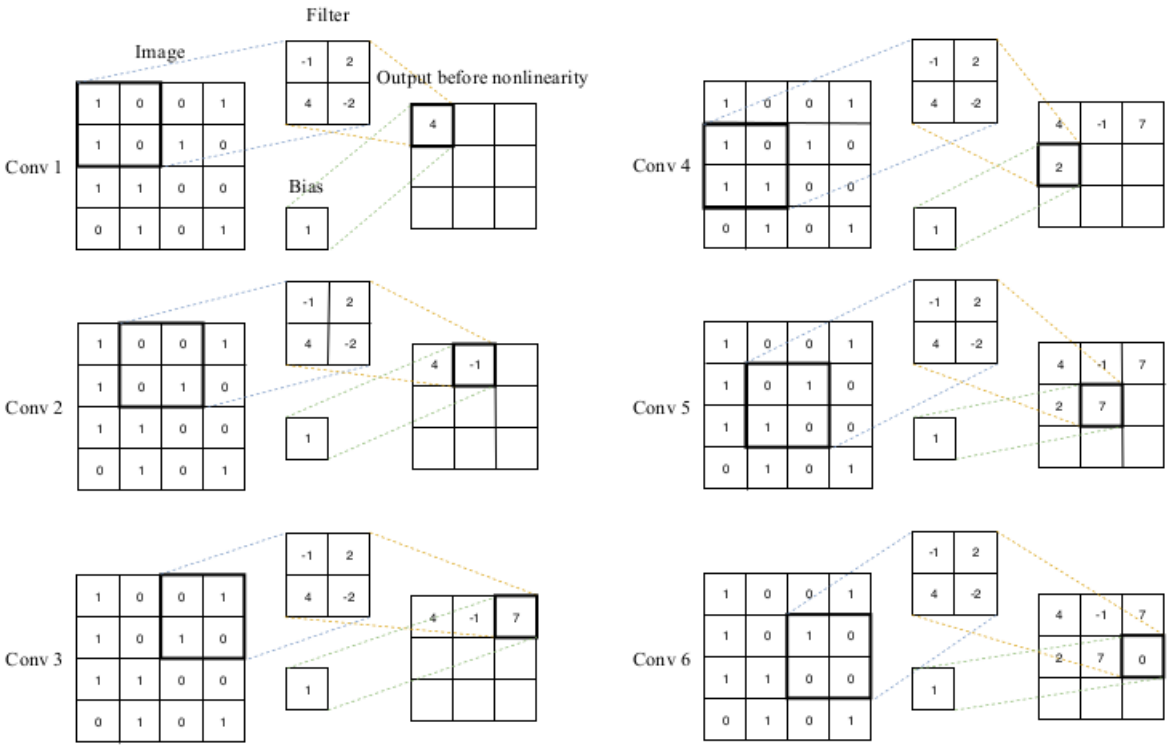


Figure 2.28: Rotate Filter along Image

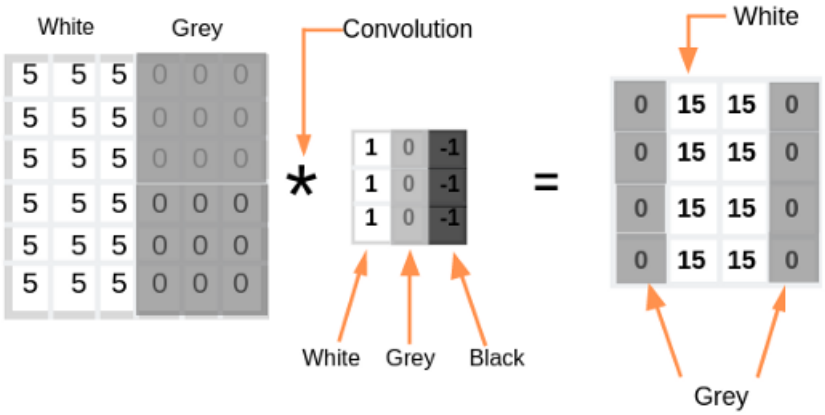


Figure 2.29: Edges in grayscale Image

Another example can be presented if at the entrance we had a grayscale image where the pixels with higher values are the brightest parts of the image and the pixels with lower values represent the darkest. In this case, it is shown in the following image that by applying a filter the vertical edges can be distinguished [33]:

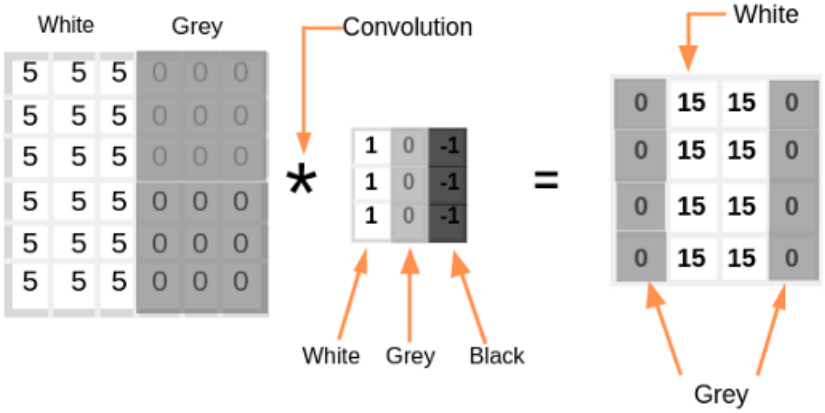


Figure 2.30: Edges in grayscale Image

From the above it follows that in the general case, if we have an image input $n \times n$ and a dimension filter $f \times f$ then the output dimension will be $(n - f + 1) \times (n - f + 1)$:

- **Input**: $n \times n$
- **Filter size**: $f \times f$
- **Output**: $(n - f + 1) \times (n - f + 1)$

Padding

Because images shrink by compression, i.e. they shrink in size, and pixels in the corners of an image are used less often during a convergence than central pixels, which can lead to loss information, the padding method is used. With this method, pixels are added around the edges of the image. So, for example, in an image size of 6×6, if a pixel padding is applied, the image will be converted to a size of 8×8 and applying a filter 3×3, the result

will give us a table of 6×6, as well as the size of the original image. These dimensions in the general case are calculated as follows:

- **Input**: $n \times n$
- **Padding**: p
- **Filter size**: $f \times f$
- **Output**: $(n + 2p - f + 1) \times (n + 2p - f + 1)$

The padding is optional and in some cases may not be applied at all, while in case we want the result table after the convolution to have the same dimensions as the original image, as we saw in the example above, then the padding that is applied is $p = (f - 1)/2$.

Strides

The stride is the size that indicates the number of steps by which we move the filter on the image, in each iteration of the convolution. These steps are appled to both the horizontal and vertical direction. The final dimensions after the application and the stride are calculated as follows:

- **Input**: $n \times n$
- **Padding**: p
- **Stride**: s
- **Filter size**: $f \times f$
- **Output**: $[(n + 2p - f)/s + 1] \times [(n + 2p - f)/s + 1]$

Convolution over Volume

In the examples we have seen so far, the images were 2 dimensional (grayscale). However, there are cases where the images are three-dimensional, with the third dimension representing the channel. Images are encoded in color channels. The most common display is RGB, which means Red, Blue and Green. The information contained in an image is the intensity of each channel color in the width and height of the image.

The size collection l (third dimension) of image boards (2 dimensions) is called **volume**.

So if for example we have an image of size $6 \times 6 \times 3$ (l = 3), then we will use a filter which will also have three dimensions e.g. $3 \times 3 \times 3$. The number of channels, i.e. the third dimension, should be the same for the input image and the filter. Basically the same filter is used 3 times, one for each channel separately.

In this example, the scoreboard will again be 4×4, as in the case where the image was two-dimensional. To calculate the first element of this table (row 1 and column 1), the element-wise product between the filter (3×3) and the 3×3 upper left part of the image is calculated, for each channel. This will result in 9 values for each channel, i.e. 27 values in total. Adding these values, we get the first element of the output table. The same procedure is followed by moving the filter from left to right and from top to bottom, on the image, as we already know. Essentially, the same calculations are made as before, only this time they are performed 3 times for each channel separately and the results are added together. This representation is shown below [32]:

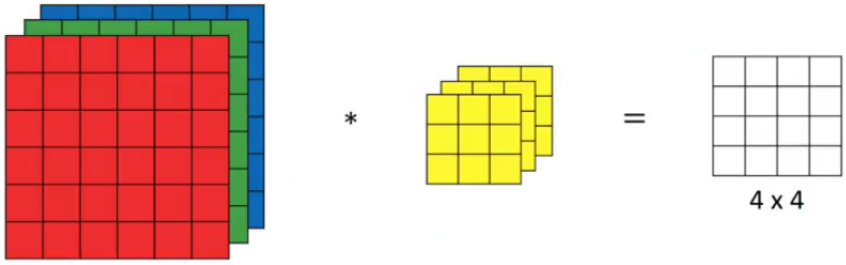


Figure 2.31: Convergence on Volume

Until now in our examples, we used a unique filter which in the above case is applied to each channel. But instead of using just one filter, there is the possibility of using more. For example, we can use 2 filters, one for horizontal end/ edge detection and one for vertical detection. In this case the output will be three dimensional with the third dimension being equal to the number of filters. So, in the example we saw above, the output instead of 4×4 will be $4 \times 4 \times 2$. This new representation is shown below [32]:

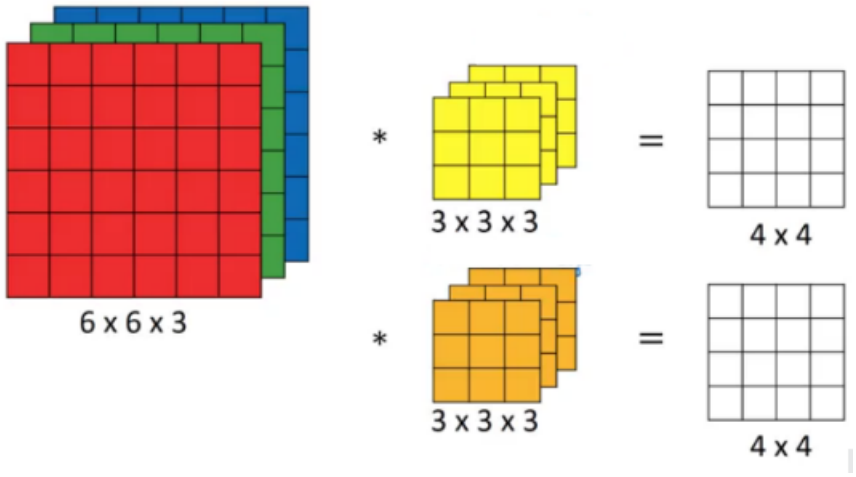


Figure 2.32: Volume Conversion with Two Filters

Finally, the generalized dimensions are given as follows:

- **Input**: $n \times n \times n_c$
- **Filter size**: $f \times f \times n_c$
- **Padding**: p
- **Stride**: s
- **Output**: $[(n + 2p - f)/s + 1] \times [(n + 2p - f)/s + 1] \times n'_c$

where $\boldsymbol{n_c}$ is the number of channels in the input image and $\boldsymbol{n'_c}$ is the number of filters.

CNN layers

The process described so far, of compiling an image with filters, is a single layer of the neural network called **convolutional layer**. Such layers can exist sequentially, one after the other, within a CNN and thus the output of one is the input of the next.

But beyond that kind of layers, on a CNN we can also find the layers called **pooling layers**, which are inserted between the convolutional layers. These layers are used to reduce the size of the input and thus increase the computing speed.

For example if we have a table 4×4 and apply max pooling size 2 and stride 2, then the result will be a table 2×2. Pooling size 2 and stepping 2, means that we apply a filter 2×2 on the input panel and move it with step 2 as it happens with filters / cores that we have mentioned. Max pooling means that for each 2×2 window of the input panel, over which we pass the pooling filter, we get the maximum value displayed in this window. One such example is shown below [32]:

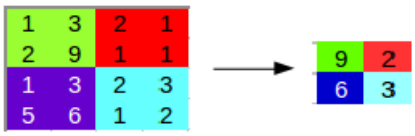


Figure 2.33: Max Pooling Size 2 and Step 2

In addition to max pooling, another method that can be applied is average pooling which instead of the maximum value within the window, calculates the average. As for the dimensions for the pooling filter inlet, the pooling filter itself and the outlet, we have the following:

- **Input:** $n_h \times n_w \times n_c$
- **Filter Size of Pooling:** $f \times f \times n_c$
- **Stride of Pooling:** s
- **Output:** $[(n_h - f)/s + 1] \times [(n_w - f)/s + 1] \times n_c$

After the convolutional and pooling layers of the neuron, a number of layers called fully connected layers are applied. Such layers are of the form we saw in artificial neural networks (2.3). The fully connected layer includes weights, bias parameters and neurons. It connects the neurons of one layer with the neurons of another layer. It is used to classify images between different categories, after its training. So, for example, if layer k contains 3 neurons and layer k + 1 4, then these two layers are connected to each other by a fully connected layer which is a table of weights of dimension 3×4. Note that the output of both convolutional and pooling layers is three-dimensional volumes, but a fully connected layer expects a 1D number vector. So we flatten the exit of the final pooling layer in a carrier and this becomes the entrance of the next fully connected layer. Leveling simply places the 3D volume of numbers in a 1D vector.

It is common to use multiple fully connected layers at the end of CNN, which increasingly reduce the dimension before the final layer which gives its output the final prediction, dimension as much as the classes.

After the fully connected layer, an activation function such as softmax for multiple classes or sigmoid for binary sorting problem can of course be applied, so that the predicted scores are converted into probabilities (2.3).

An example of a cnn with the layers it consists of can be seen below [34]:

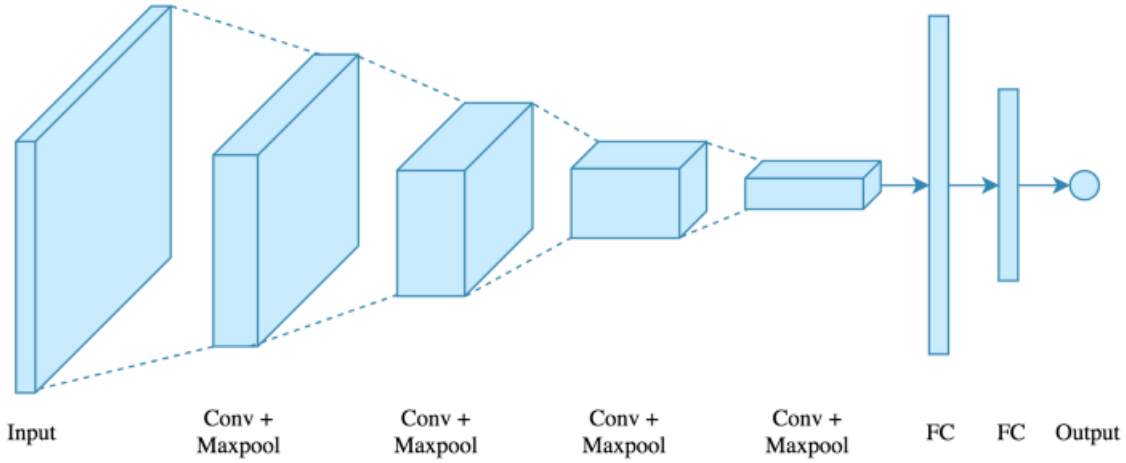


Figure 2.34: CNN Mattress Example

2.5 Recurrent Neural Networks

Recurrent neural networks (RNNs) are another class of artificial neural networks. Their difference from the other categories and at the same time their great qualification is the ability to manage sequential data, i.e. data that has a time sequential connection, such as the frames of a video and the words in a text. This is why RNNs are often used in word processing and speech.

The way these networks utilize sequential information is through the concept of memory. They are networks that use loops, thus allowing information to be retained over time. A repetitive neural network can be thought of as multiple copies of the same network, each of which transmits a message to a time successor. So, if we unroll the loop we get the form shown in the following image [35]:

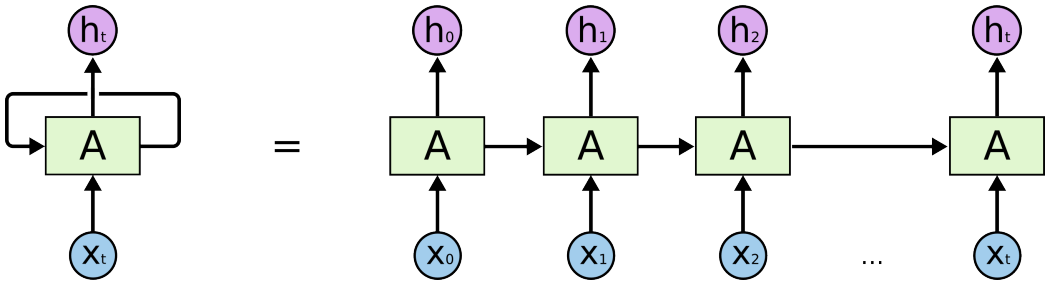


Figure 2.35: Loop on an RNN

This way, if we want to translate a text, for example, we can define each entry as a word in that text. The RNN transmits the information of the previous words to the next network that can use and process this information.

2.5.1 Mode operation

The concept of memory in the repetitive neural network is given by the "hidden state". Each network output depends not only on the input data at that time and the network parameters but also on a hidden state vector that represents all past network inputs. In addition, the calculation of the next hidden state depends on the input and the previous hidden state.

The following is the structure of a single network neuron, for different time points (x_{t-1}, x_t,x_{t+1}) [35]:

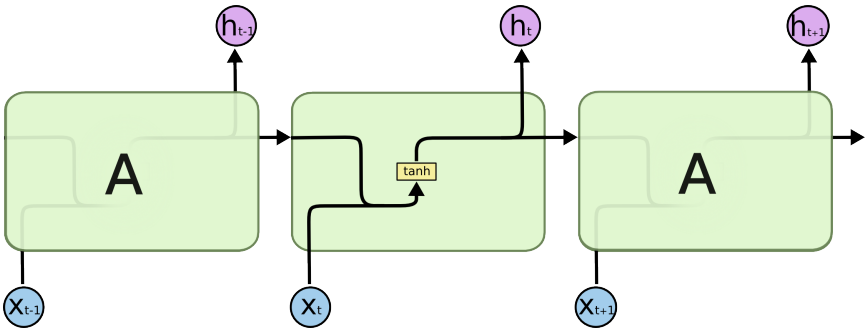


Figure 2.36: RNN Neuron Structure

The calculation of the latent state h_t but also the output y_t of the specific neuron for the time t is given by the following relations:

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{2.69}$$

$$y_t = \sigma_y(W_y h_t + b_y) \tag{2.70}$$

where $\boldsymbol{x_t}$ is the input, $\boldsymbol{\sigma_h}$ and $\boldsymbol{\sigma_y}$ are activation functions (in the image 2.35 is σ_h = tanh) , $\boldsymbol{W_h}$ $\boldsymbol{W_y}$ $\boldsymbol{U_h}$ $\boldsymbol{b_h}$ $\boldsymbol{b_y}$ are parameter-weights vectors and $\boldsymbol{h_{t-1}}$ is the hidden state of the previous time.

Note that the above neuron may be a single layer neuron of the repetitive neural network and the network may consist of several layers. In this case, the input vector of the neuron comes from the output of the previous layer, while the hidden state vector from the same layer from a previous time.

Theoretically, RNNs are perfectly capable of handling "long-term dependencies", that is, dependencies on inputs that are too far in time or otherwise very distant ancestors. In practice, RNNs become very ineffective when the gap between the relevant information and where they are needed is too large. This is because information is transmitted at every step and the longer the chain, the more likely the information is to be lost along the chain. The problem has been investigated in depth by Hochreiter (1991) [German] and Bengio, et al. (1994), who found some very fundamental reasons why this might be the case. LSTMs, a special type of neural network RNN, try to solve this problem of long-term dependency.

2.5.2 Long Short Memory Networks

Long Short Term Memory (LSTM) networks are a special type of RNN capable of learning long-term dependencies. Created by Hochreiter & Schmidhuber (1997) [36] and improved and disseminated by many people. They work extremely well in a wide variety of problems and are now widely used.

LSTMs are explicitly designed to avoid the problem of long-term dependency. Memorizing information for long periods of time is essentially their default behavior, not something they find difficult to learn.

In addition to the cache state (h_t), which can be thought of as short-term memory, they use another state called cell state (c_t), which can be thought of as long-term memory. The

cell state runs along the entire chain, with only a few small linear interactions. Thus, it is very easy for information to flow unchanged.

In their internal structure, LSTMs consist of three portals:

- The forget gate, which takes as input the cell state of the previous time c_{t-1} and forgets anything that is not useful.
- The input gate, which takes as input the hidden state of the previous time h_{t-1} but also the input of the neuron x_t which is the new information and puts them together.
- The output gate, which takes as input the information we have just learned along with the information of the long-term memory (cell state) that have not yet been forgotten and uses it to make a prediction, i.e. to produce the output and update the short-term memory (hidden situation).

The following figure shows in detail the internal structure of the LSTM as well as the portals of which it consists [37]:

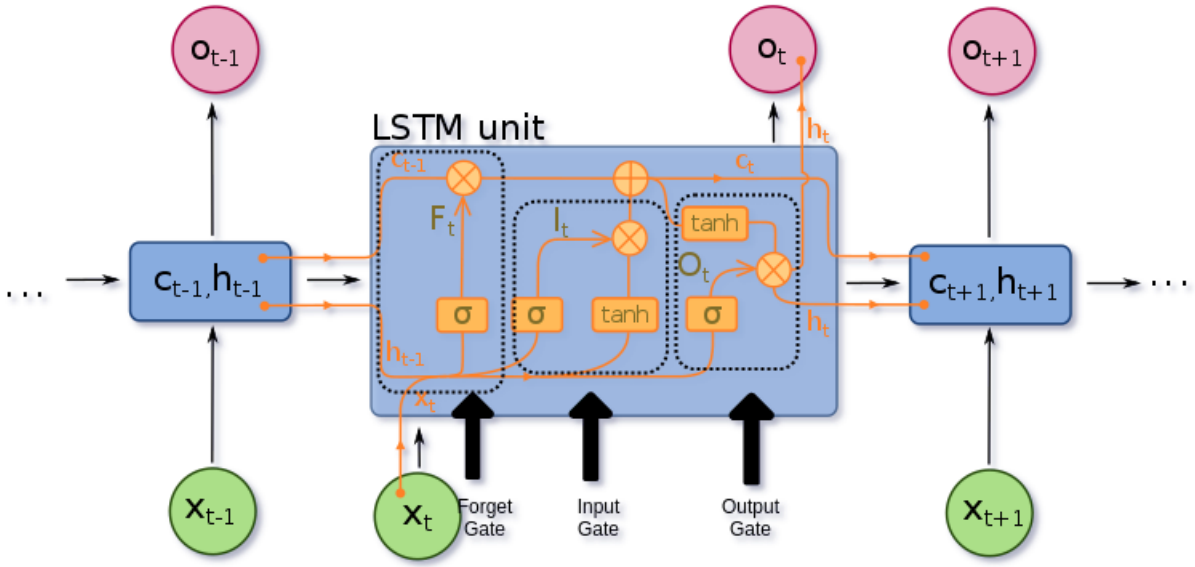


Figure 2.37: LSTM Neuron Structure

In detail, the mathematical operations that accompany the above architecture are the following:

$$F_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{2.71}$$

$$I_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{2.72}$$

$$I'_t = \sigma'_i(W'_i x_t + U'_i h_{t-1} + b'_i) \tag{2.73}$$

$$O_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{2.74}$$

The new cell state c_t, which will spread to the same neuron at the next time in the sequence, is essentially what we do not forget from the previous state and what we have learned from the new information. Defined by the following relation:

$$c_t = F_t \otimes c_{t-1} + I_t \otimes I'_t \tag{2.75}$$

The output of the neuron o_t but also the new hidden state h_t that will spread to the same neuron, at the next time point of the sequence is defined as follows:

$$o_t = h_t = O_t \otimes \sigma_h(c_t) \tag{2.76}$$

In the above relations, all W, U and b are parameters-weights that the model learns during training. The σ are activation functions which in this case in the figure 2.36, σ_g and σ_h are the sigmoid function, while σ'_i is the tanh function. These activation functions can of course have several variations.

In conclusion, the same problem that generally occurs with RNNs, occurs also with LSTMs, i.e. when the suggestions are too large, the LSTMs are still not efective. The reason for this is that the probability of preserving the environment from a word that is far from the current word being processed decreases exponentially with the distance from it. This means that when for example the data is text and the sentences are large, the model often forgets the content of the remote locations in the sequence. Another problem with RNNs and LSTMs is that it is difficult to parallel the task of editing sentences, as they have to be processed word for word

2.6 Transformers

Transformers is an artificial neural network architecture created in 2017. They are mainly used in the development of systems in the field of natural language processing, but recent research has also developed their application in other tasks such as video comprehension. Transformers were developed to address issues related to the problem of sequence transduction or neural machine translation. This means any task that converts an input sequence to an output sequence. This includes speech recognition, text-to-speech, translation, etc.

Like repetitive neural networks (RNNs), transformers are designed to handle sequential input data, such as natural language, for tasks such as translation and text summarization. However, unlike RNNs, transformers do not require sequential data processing in sequence. Instead, the attention mechanism they use identifies the box for any position in the input sequence. For example, if the input data is a natural language sentence, the transformer does not need to process its beginning before the end. Instead, it identifies the context that gives meaning to a word in the sentence. Because of this feature, the transformer allows much greater parallelism than RNNs and therefore reduces training times.

Transformers have quickly become the model of choice for NLP problems, replacing older iterative neural network models such as LSTMs. As the Transformer model facilitates more parallelism during training, it allowed training in larger datasets than was possible before. This has led to the development of pre-trained systems such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trainer Transformer), which have been trained with huge language datasets, such as Wikipedia Corpus and Common Crawl, and can be customized through the fine-tuning method in specific language processes.

2.6.1 Mode operation

Transformers are based on **encoder-decoder** architectures and **attention mechanisms**. More specifically, they consist of 6 encoders and 6 decoders as shown in the following figure

[38]:

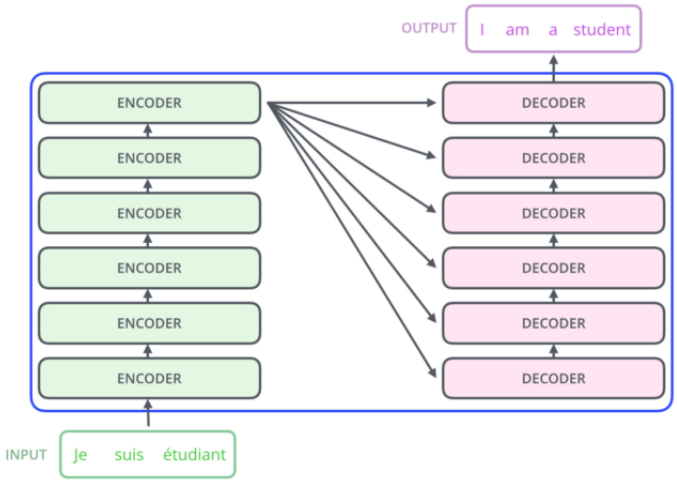


Figure 2.38: Transformers architecture

In general, the encoder is a neural network that receives sequential input. It could be RNN, but also CNN or some other architecture. The role of the encoder is to read the input and create some kind of state (similar to the state in RNN) that can be thought of as a numerical representation of the concept of the input with which the machine can work. The notion of a certain entity, whether it is an image, text, or video, is usually a vector or matrix containing real numbers. This vector (or table) is called in the machine language **embedding** the input. The decoder is another neural network that takes an integration as input and is capable of generating an output sequence. As one might have guessed, the integration comes from the encoder.

What practically differentiates transformers from other sequential models is the attention mechanism and the positional embeddings they use, two concepts that we will analyze below.

In the Transformers architecture, all encoders have the same internal architecture. They consist of two levels: the first level is an attention mechanism called **self-attention** and the second level is a **fully connected feed-forward network**. This internal structure is shown in the following figure [38]:

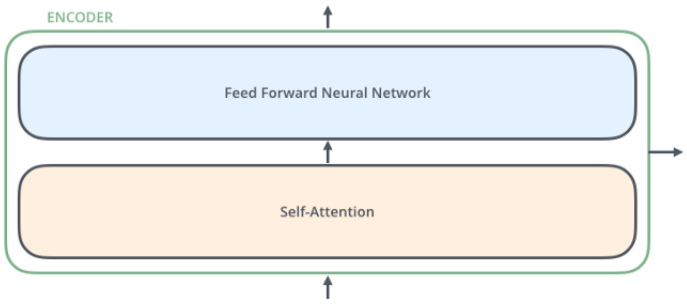


Figure 2.39: Encoders Architecture

All decoders, respectively, present the same internal architecture and consist of the two levels we saw in both encoders, with an additional layer interposed between the other two. The extra level is again a watch mechanism that monitors the output of the coding stage. Schematically the internal structure of the decoder is shown below [38]:

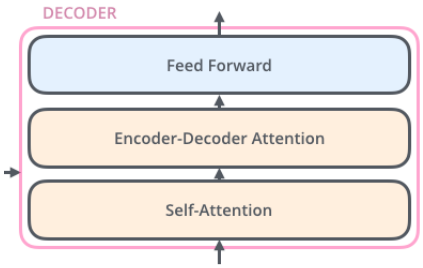


Figure 2.40: Architecture of Decoders

Each encoder as well as each decoder is connected to its predecessor and even the output of the last encoder is transmitted to all six encoders as shown in the figure 2.37.

Self-Attention

At this point, we will examine how the attention mechanism presented above works. Attention mechanism is a technique that is applied to artificial neural networks and essentially implements what its name implies: it focuses its attention on a part of the whole that it receives as input each time.

At first, we convert each input word into a vector through embedding algorithms. These verbal embeddings of the input sequence flow through each of the two layers of the encoder as shown below [38]:

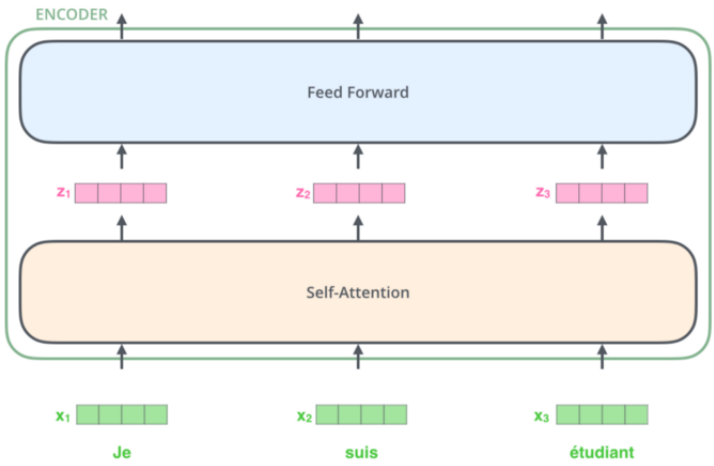


Figure 2.41: Enter the first encoder

Here we begin to notice a key property of Transformer, i.e. that the word in each position flows through its own path in the encoder. There are dependencies between these paths at the self-attention level. The feed-forward layer does not have these dependencies, however the various paths can be run in parallel while flowing through the feed-forward layer.

The reasonable question that arises here is what functions are performed in the self-attention layer to produce the new z vectors shown in the image. Below are all the steps to follow in the self-attention layer:

1. Three vectors are initially created for each of the encoder input vectors. In the case of the first encoder, the input vectors are the verbal embeddings, while for the subsequent encoders the input vectors are those that have emerged from the output of the immediately preceding encoder. The three new vectors that are created are called query, key and value. These vectors are produced by multiplying the input vector by

three arrays that we trained during the training process, which are the well-known weight arrays we came across in neural networks. An example of this process of creating the three vectors is shown in the following figure [38]:

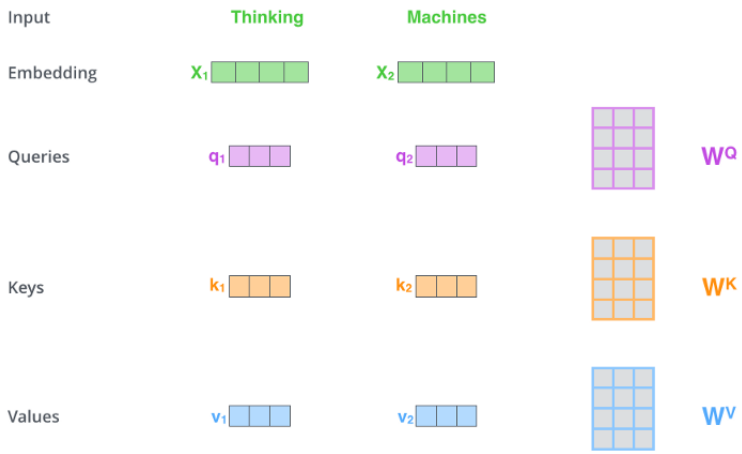


Figure 2.42: Query, Key and Value Vectors

In the example above, two words of the input sequence are presented: the words thinking and machines. X_1 and x_2, respectively, are the vector embeddings of these two words. The vector query of the word thinking symbolized as q_1 is obtained by multiplying x_1 by the weight table W_Q. The key vectors (k_1) and value (v_1) are obtained respectively, multiplying the vector x_1 by the tables W_K and W_V respectively. The same procedure is followed for the word machines which has a vector representation of x_2 as for any other word in the input sequence.

2. The next step in calculating self-attention is to calculate a score. For example, for the first word "thinking" we have to rate each word of the introductory sentence against it. This score determines how much we should focus on other parts of the introductory sentence when coding a word in a particular position. The score is calculated by taking the inner product of the query vector of the word we are encoding with the key vector of the corresponding word we are scoring. Therefore, if we process the self-attention for the word in position 1, the first score would be the internal product of q1 and k1. The second score would be the internal product of q1 and k2 and so on. This procedure is shown in the following figure [38]:

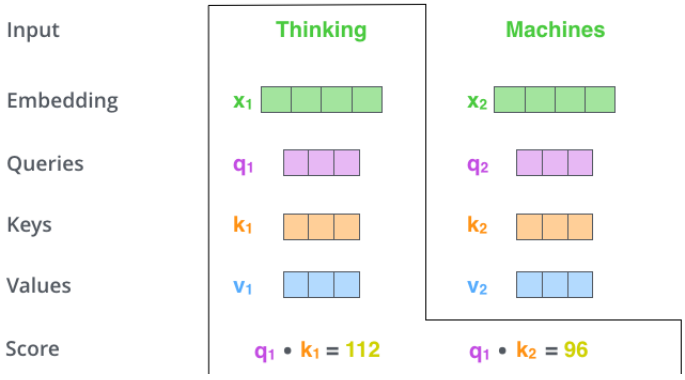


Figure 2.43: Score calculation

3. Next, we divide this score by the square root of the key vector dimension. In the original paper this dimension is 64 and therefore its square root is 8.

4. The result of the scores obtained from the previous step, we pass it through a softmax activation function which normalizes the scores and gives all positive values which all add up to the value 1. The result of the softmax function, essentially expresses how much each word will affect the coding of the word in question. It is clear that the word under consideration in this post will have the highest softmax rating, but sometimes it is helpful to watch another word related to the current word. Schematically the procedure of the last two steps is shown below [38]:

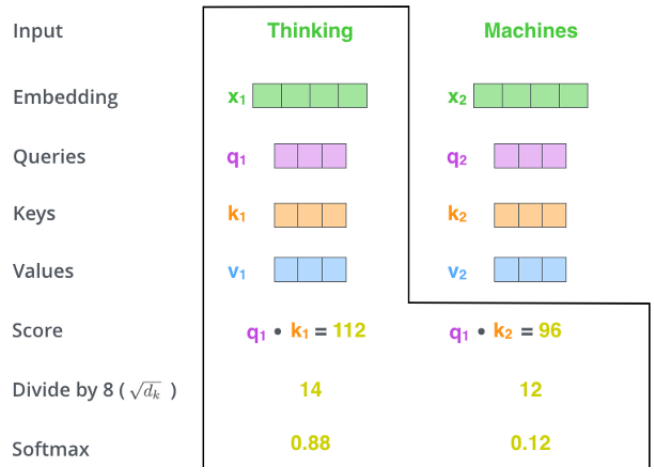


Figure 2.44: Division and application of softmax function

5. The final step is to multiply each value vector by the softmax score and then add these results. This is done to keep intact the values of the words (values) on which we want to focus and to weaken the irrelevant words (multiplying them by tiny numbers such as, for example, 0.001).

   The final vector resulting from the above sum is the z vector of the word in question, which we can now send to the layer of the feed-forward neural network.

   The procedure for this step is shown in the following figure [38]:

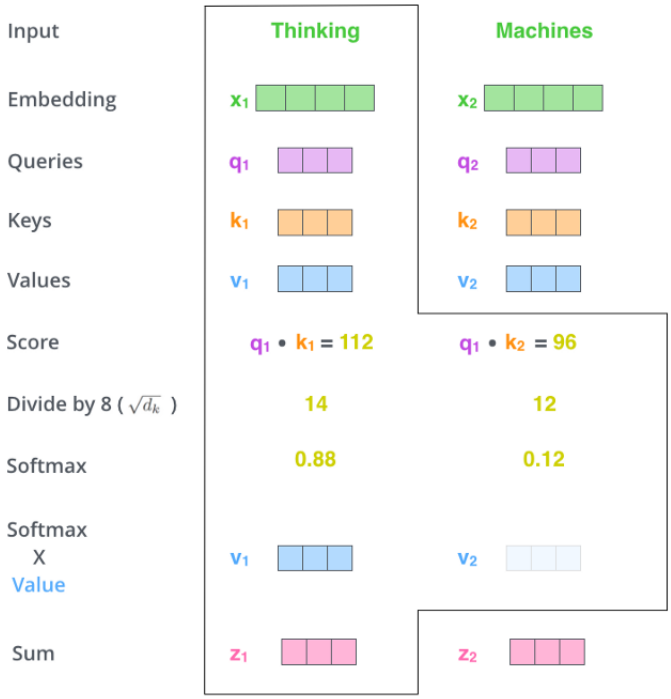


Figure 2.45: Multiply softmax scores by the vectors of values and sum

In practice, the attention mechanism is calculated in a set of queries at the same time, packed together in a Q table. The keys and values are also packed in tables K and V. Thus, the output matrix is calculated as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.77}$$

Multi-Head Attention

In practice, Transformers use multiple self-attention mechanisms in parallel with a technique called multihead attention.

The idea behind this is that if for example we translate a word, we can pay different attention to each word based on the type of question we ask. So, every time we translate the word "kicked" into the sentence "I kicked the ball", we can ask "Who?" . Depending on the answer, the translation of the word into another language may change. Similarly, we can ask other questions, such as "Did what?", "To whom?" etc.

Multi-head attention allows the model to jointly monitor information from different representation subfields in different positions.

The mathematical expression of this mechanism of attention is given as follows:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O, \quad where \quad head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.78}$$

The comparison between a single attention layer with multi-head attention is shown in the following figure [39]:

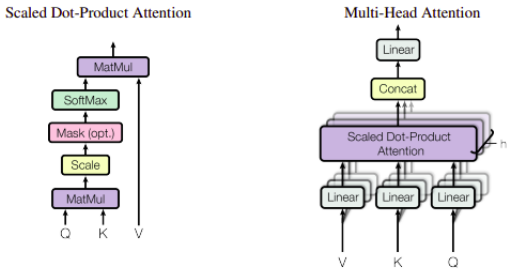


Figure 2.46: (left)Scaled Dot-Product Attention. (right) Multi-Head Attention consisting of several attention layers running in parallel

Position Encoding

Another important step of transformers is to encode the position, i.e. to enter information about the position of each word in the input sequence. To achieve this purpose, additional information about the position of each word that the model learns to recognize is inserted into the word vectors. The intuition is that a position expression vector is added to each word in the first level of coding. Position coding introduces important information to the model, which it will then use to give meaning to the words.

Finally, the fully connected feed-forward network, which exists in both encoders and decoders as we have already seen, consists of two linear transformations with ReLU activation between them, which are given by the following relation:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.79}$$

CHAPTER 3

System Architecture

The previous chapter provided an extensive analysis of the whole theory that will be used for the purposes of this dissertation. This analysis will be a valuable tool for understanding the chapters that follow.

For the purpose of speech evaluation, we use information that we extract from both audio and text. By the term sound we mean the speech of a person in audio form (sound signal), while by the term text we mean the same speech in the form of text. Each of these two pillars (audio and text) compiles a separate system of processing and analysis, which has as its ultimate goal to help in the final evaluation of speech. Below are these two systems and their combination

3.1 Audio Analysis System

As for the audio part, we take the following steps:

1. The system receives speech in audio format (audio file).
2. This sound is then split into sections. This split is performed through predefined time windows.
3. Each part of them is used as input to trained class models, which are segment classifiers. Specifically, the system contains three such models: one for predicting speech emotion (recognition emotion recognition), one for predicting arousal / excitement (arousal) and one for predicting valence / activity (valence).
4. Classifier outputs generate a prediction for each audio segment separately (class posters). But the purpose of the system is to characterize the overall speech. In this way the labels provided for each section are gathered, in order to produce information for the entire audio file. This concentration is created by calculating the average of the samples per label. For example, if an audio file contains 2 sections labeled "joy" and 3 sections labeled "stress" then the total audio-speech will be 0.4 labeled "joy" and 0.6 labeled "stress".
5. It is obvious that with the above steps we have already acquired information or otherwise features for speech, which could be used by a model that will evaluate its quality. These features, however, may not be sufficient. For this reason, some

additional features related to the total sound are added to this step, such as the average duration of silence, the number of pauses, the total duration of speech, the rhythm of words etc. These features are called high-level features or recording-level features, to distinguish them from those of the previous step that resulted from the segment-level features.

6. Finally, combining the features of the previous two steps, we end up with an overall representation of the sound.

The system described in the previous steps is shown in the following figure:

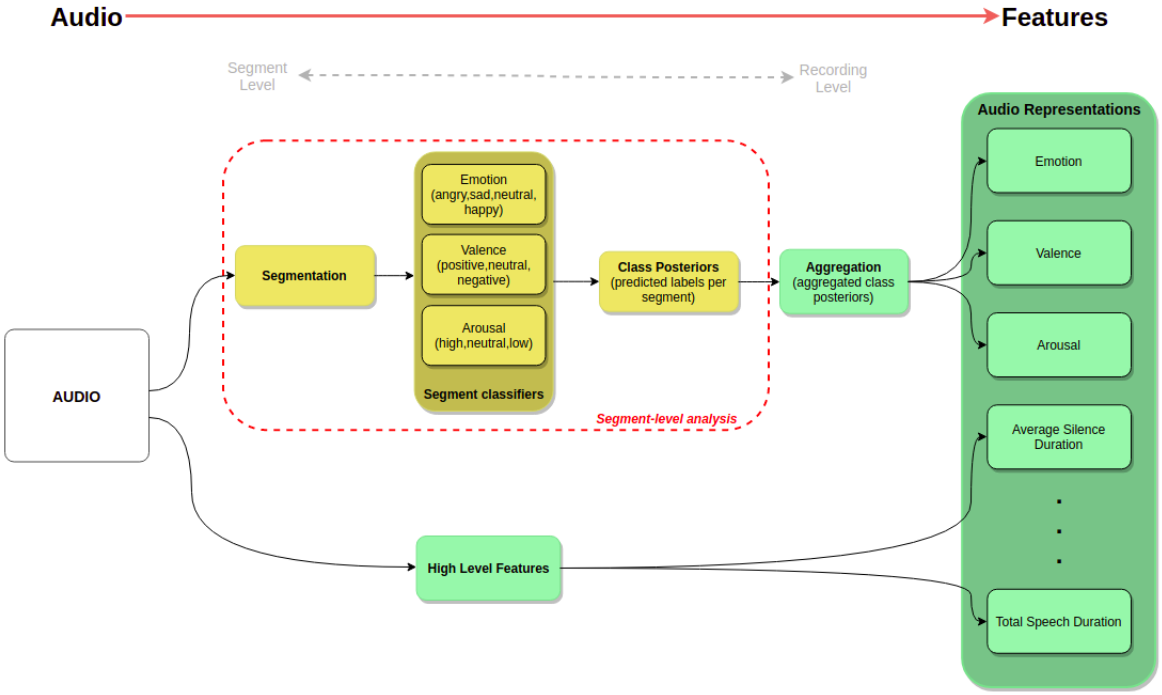


Figure 3.1: Audio Information Analysis Process

We observe in the figure 3.1, that the part of the system that concerns the division of sound into parts and the use of segment classifiers is outlined with dashed lines and is called segment-level analysis. This section will be discussed in detail in the next chapter (4). In contrast, the remaining pieces on the characteristics of the overall sound compose the recording-level analysis, which will be developed in detail in the chapter 5.

3.2 Text Analysis System

The text of the speech is derived from the audio, using artificial intelligence technologies from Google Cloud. Specifically, Cloud Speech-to-Text was used, which applies Google's most advanced deep learning neural network algorithms for Automatic Speech Recognition (ASR) [40].

As for the piece of text, the exact same steps were followed with the audio system, again splitting the text into sections and using section sorters. What changes here is step 5, where the high-level features for the text are now word rhythm, number of unique words, word frequency histograms, etc.

The text analysis system is shown in the following figure:

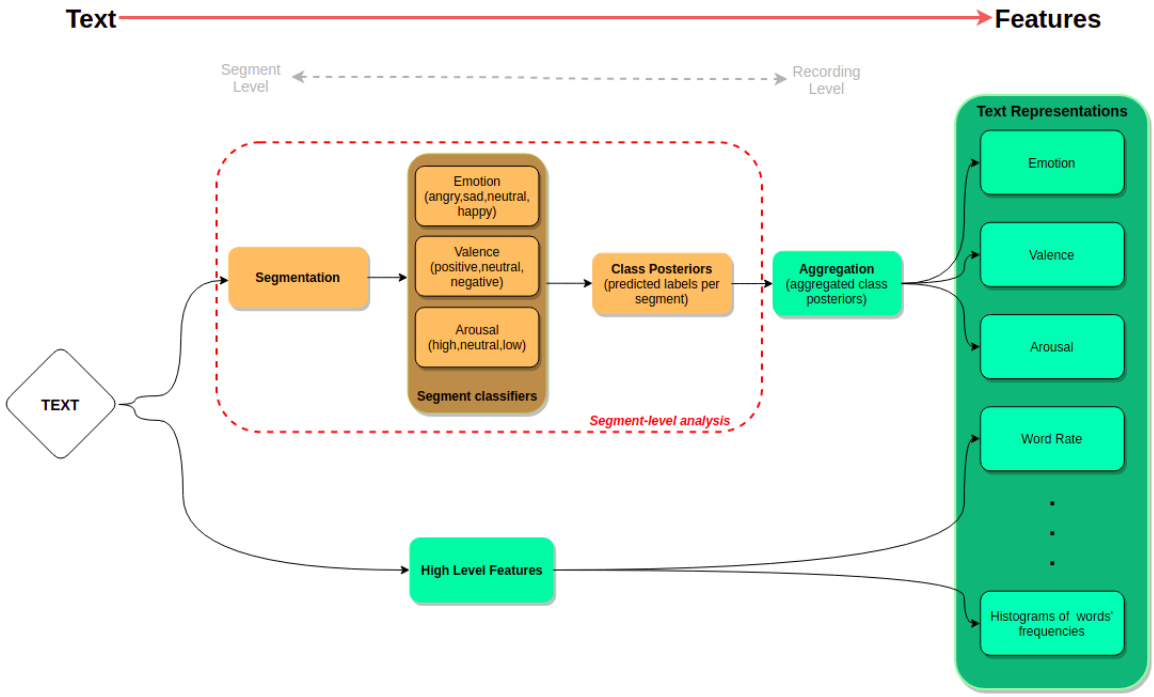


Figure 3.2: Text Information Analysis Process

In the figure 3.2 we distinguish again the areas of segment in level analysis in log-level analysis, which will be developed in detail in the following chapters.

3.3 System Completion

Having now extracted features from both the overall sound and the overall text, the ultimate goal is to use them in a model that will be trained to predict the quality of speech. In this way, the system outputs presented in the previous subchapters will be used as inputs to the final classifier model. The following figure shows the overall architecture of the system:

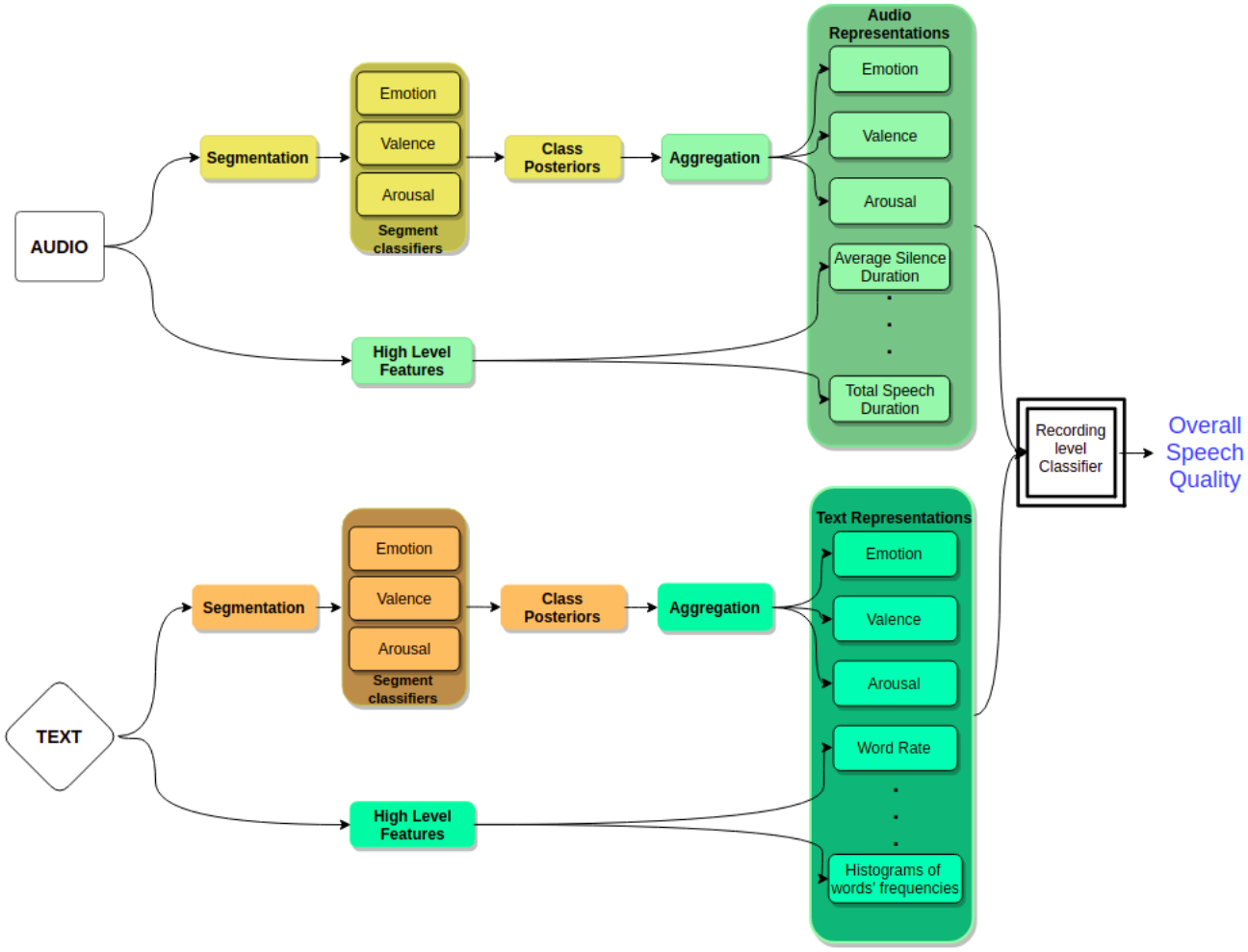


Figure 3.3: Final Classifiers

As shown in the figure 3.3, the final model is called the recording-level classifier because it concerns the total speech, i.e. the total recording / recording as one might say. At this point it is important to note that it is not necessary to use both audio and text systems. Conversely, the final classifier can use only audio features or text-only attributes to evaluate speech quality. It is thus set to investigate which of the three systems (sound only, text only, combination), will give the most valid results, i.e. will help the final model to learn better.

CHAPTER 4

Segment-Level Analysis

This chapter will provide a detailed analysis of the part of the architecture that uses audio and text segments, respectively (segment-level analysis).

4.1 Audio Analysis

This subchapter will present the techniques for splitting the audio signal into windows, the set of [41] audio features used for segment classifiers, the data sets used for classifier training, and the experiments performed.

Specifically, the following part of the system architecture will be analyzed:

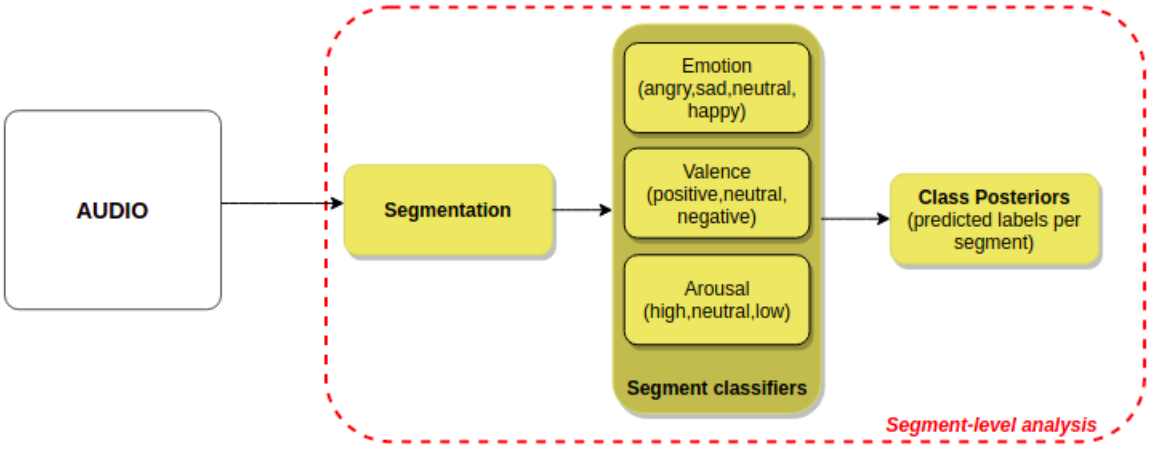


Figure 4.1: Segment Level Analysis

As an introduction it would be important to mention that a signal that we denote as x(t), is a continuous time signal as shown in the following figure [43]:

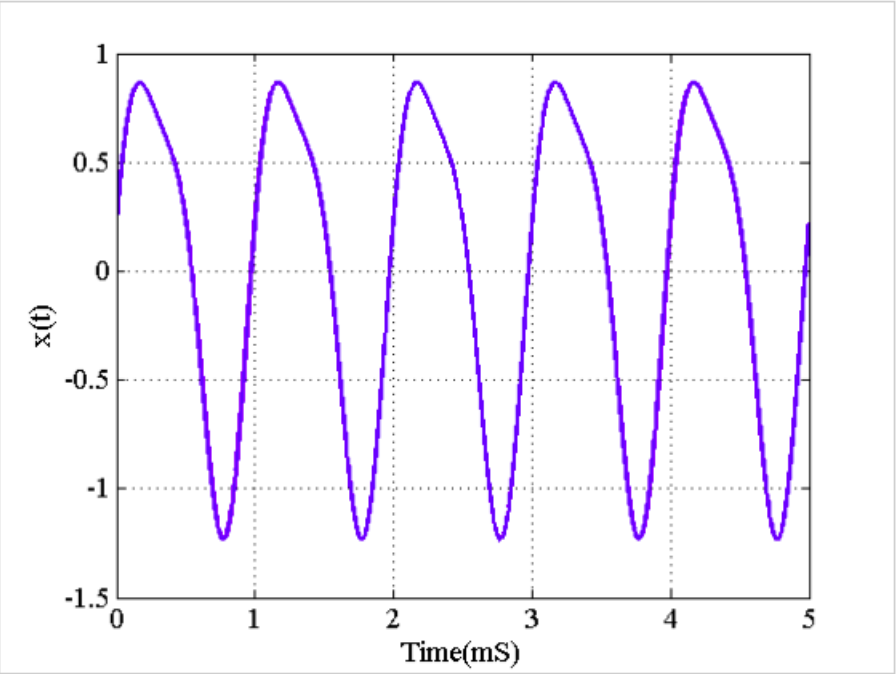


Figure 4.2: Continuous Time Signal

A digital computer, however, cannot operate with a continuous time signal for this reason it is necessary to take some samples of x(t) and work with these samples instead of the original signal. Additionally, the original signal extends indefinitely, but the digital computer can only process a finite number of samples. Therefore, in general, a finite length sequence is used to represent this analog continuous time signal that can extend to a positive infinity on the time axis.

For example, we can sample the x(t) signal of the figure 4.2 with a sampling frequency of f_s = 8000 samples per second . So for a period of the original signal x(t) which is equal to 1msec we will get 8000 samples/sec x 0.001sec = 8 samples. The result is shown in the following figure in red [43]:

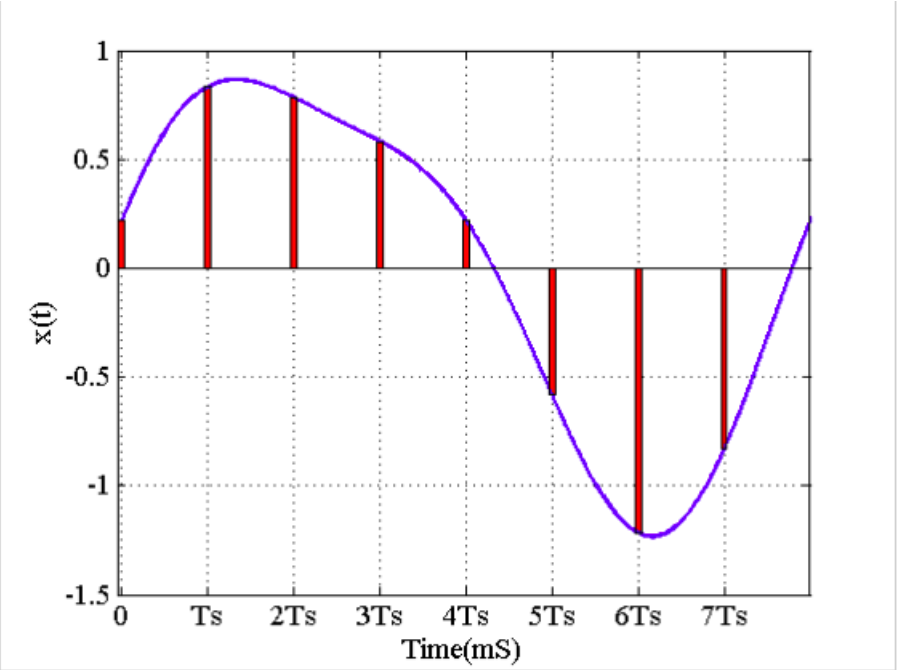


Figure 4.3: Continuous Time Signal Sampling

Continuous Time Signal Sampling If we normalize the time axis in the sampling period $T_s = \frac{1}{f_s}$, we will get the distinct sequence x(n) for the 8 samples (x(n=0) = x(t=0), x(n=1) = x(t=T_S), ..., x(n=7) = x(t=$7T_s$)).

4.1.1 Short-Term Audio Processing

Short-term audio processing, is a technique in which the audio signal is split into **short-term windows**.

An oral sentence is a sequence of phonemes. Speech signals, therefore, change over time. To extract information from a signal, we need to break the signal into several short sections. In other words, we want to export parts that are small enough that the properties of the speech signal do not change within them.

An example of a speech signal with the corresponding phonemes is shown below [42]:

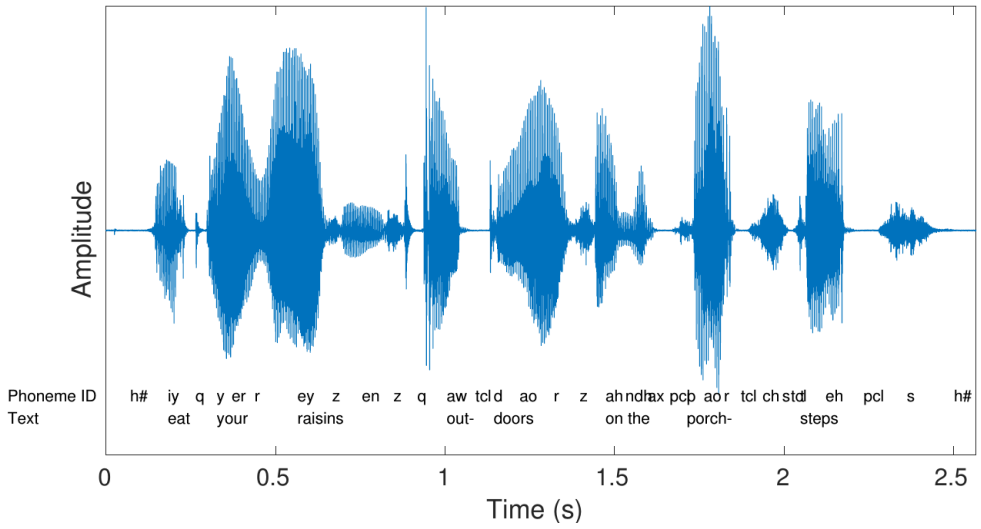


Figure 4.4: Audio Speech Signal

For this purpose, in most applications the audio signal is split into possible overlapping windows and the sound analysis is performed on a segment basis. The reason why this split occurs is because, as we have mentioned, sound signals are not static over time, that is, their properties change. For example, a speaker's emotion may change over time as he or she speaks. The same goes for the volume, the tone of voice, the rhythm of speech, etc. So it would be incomplete and discriminatory to characterize the whole speech with a single label of emotion, intensity and so on. More specifically, if we had a signal, which contains parts of both normal and high intensity, and calculated the average intensity of the total signal, then the result would be dominated by samples from the high intensity part, thus ignoring the part of the total signal in which the volume was normal. Conversely, if we calculated the intensity separately for each segment, then we would have a more accurate characterization of the overall signal.

From a mathematical point of view, the above process is as follows: given an sound signal x(n) with n = 0, ..., N-1, which has N samples, at each step of the short-term processing, we multiply the signal by one predefined window w(n). The signal portion of the signal in step i is as follows:

$$x_i(n) = x(n)w(n - m_i), \quad i = 1, ..., K - 1 \tag{4.1}$$

where $\mathbf{i}$ is the number of steps, i.e. the number of partition-windows in which the signal will be split $\boldsymbol{m_i}$ is the index of the sample in which the i-th window starts. It basically shows the number of samples that window w must move in order to produce the i-th segment.Two more metrics are the **window length**, which is fixed and denoted as $\boldsymbol{W_L}$ and the **hop size / step** also fixed with which the window moves, denoted as $\boldsymbol{W_S}$. Also, the total number of segments into which the audio signal is finally divided is calculated as $\frac{N-W_l}{W_S} + 1$.

The following is an example of signal splitting in windows[42]:

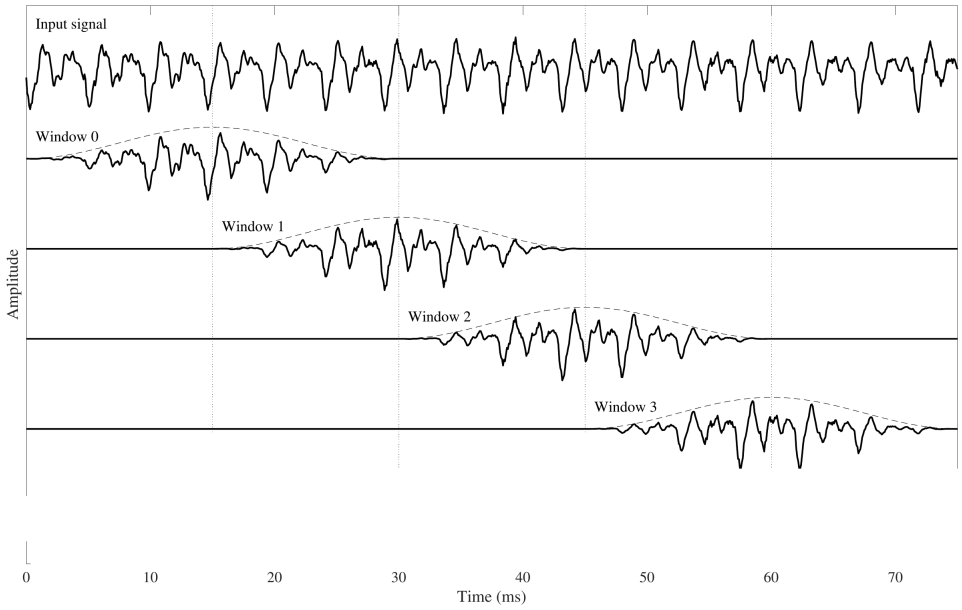


Figure 4.5: Signal Splitting in Windows

The following is a numerical example: if we have a signal with a sampling frequency of F_s = 8000 Hz, which we want to break with step windows W_S = 10 and length W_L = 20 ms or otherwise $W_L = 0.02 \cdot 8000 = 160$ samples, then the variable $m_i = i \cdot W_S \cdot F_s = i \cdot 0.01 \cdot 8000 = i \cdot 80$ samples. This means that each step that is 0.01 sec is translated to 80 samples and the i-th window starts at $i \cdot 80$ and ends at $i \cdot 80 + W_L - 1$. So the 3rd window in this case would start in

the sample$3 \cdot 80 = 240$ and would end in the sample $240 + 0.02 \cdot 8000 - 1 = 240 + 160 - 1 = 399$, while the 4th window would start in the sample $4 \cdot 80 = 320$ and would end in the sample $320 + 0.02 \cdot 8000 - 1 = 320 + 160 - 1 = 479$.

As one can assume, these two consecutive windows overlap, since the 4th starts before the 3rd ends. In fact there is a 50 % overlap. This is because the window pitch is shorter than its length ($W_S > W_L$). For steps greater than or equal to the length, no overlap occurs. Accordingly, the smaller the pitch relative to the length of the window, the higher the overlap rate.

There are several types of windows that can be used. The most common and used in this dissertation is the **rectangular window**, which has a value of 1 within its length and 0 anywhere else:

$$\begin{cases} 1, & 0 \leq n \leq W_L - 1 \\ 0, & \text{elsewhere} \end{cases}$$

Having thus broken down the audio signal into short-term audio segments, we can now export features for each section separately, thus creating a sequence of feature vectors. The dimension of these vectors depends on the number of features we will use which are analyzed in the next subchapter

4.1.2 Mid-Term Audio Processing

Mid-term audio processing, is the process by which the audio signal is split into larger **mid-term / texture windows** than in the case of short-term processing and then for each of the exported parts, the short-term processing is followed.

Essentially, the signal is divided into larger windows, each of which contains a number of smaller windows. In this way, a feature vector is extracted for each mid-term segment of the sound, which represents the feature statistics, i.e. it can be derived from the average value of the features in the short-term windows, from their standard deviation etc.

If, for example, we have a 10-second signal and use 2-second mid-term windows and 1-second short-term windows, then the signal will be divided into 10/2=5 mid-term windows, each of which contains 2/1=2 short-term windows. So if we want to export for example the zero pass rate, which is a sound feature that will be presented next, then we will calculate it for each of the two short-term windows and we will take the average of these values and / or the their standard deviation, so that a mid-term representation will emerge.

In many applications, mid-term feature extraction is used for a longer-term representation. In these cases, it is best to export a single feature vector representing the overall sound, as opposed to multiple vectors from each mid-term segment. Thus, the **long-term averaging** of mid-term feature statistics is calculated, extracting , in this way, a unique vector of features. In this final vector, the stimulation of the important features and the rejection of the features that are presented only on a temporal basis will have been achieved. The above process of medium-term audio processing is presented in the following figure:

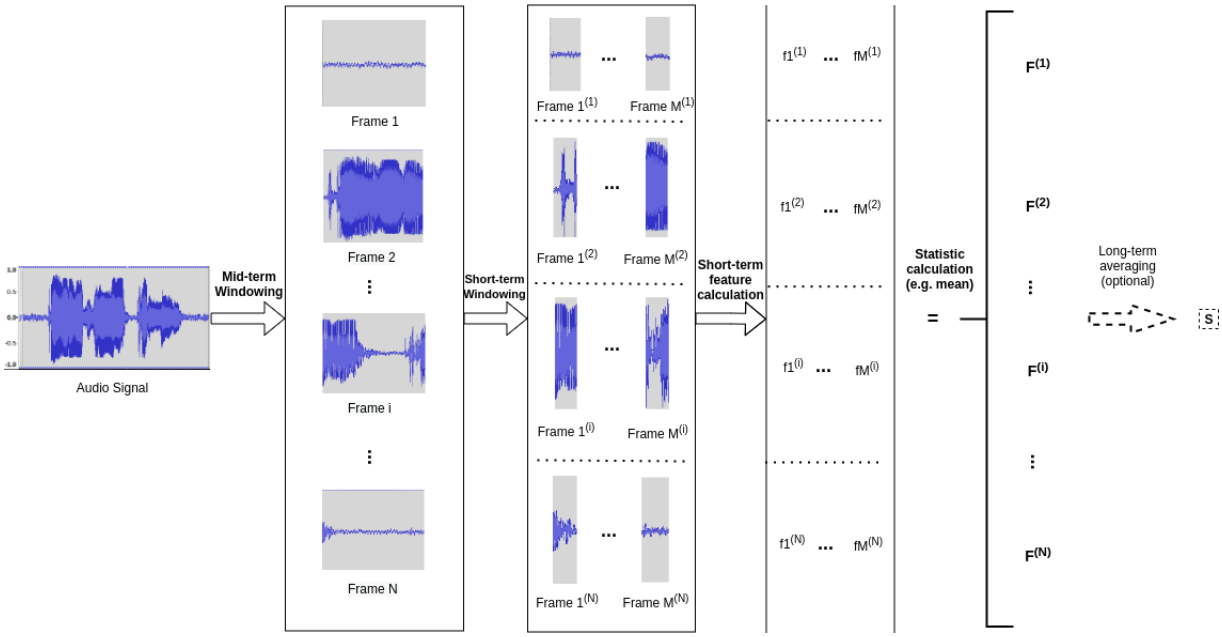


Figure 4.6: Mid-term Audio Processing

The above figure (4.6) represents an audible signal, which splits into N mid-term windows (mid-term windowing) and then each of these windows splits into M short-term windows (short-term windowing). A vector of f is extracted from each short-term window (short-term feature calculation). Then a statistical calculation is been performed (for example, average value calculation), for each mid-term window (statistic calculation). Thus, we end up with N attribute vectors in total ($F^{(1)}...F^{(N)}$). In the last step, the average of these vectors can be optionally calculated and we end up with a unique vector (S) that characterizes the total sound signal (long-term averaging).

4.1.3 Audio Feature Extraction

An audible signal is a signal that contains information in the audio frequency range. Audio representation refers to the extraction of audio signal properties, or characteristics, that are representative of the acoustic signal composition (both in time and spectral field) and the behavior of the audio signal over time. Feature output is usually combined with feature selection, which determines the best feature set for the intended operation of the audio signal. The aim is to extract features from the audio data (speeches), which will have information power for the models to be trained.

Having split the audio signal into short-term windows, for each of these windows some characteristics must be calculated, from which statistical values per medium-term window will then be extracted.

There are many metrics that can be used as features in audio analysis. This section provides a brief description of the various features used in the system design.

4.1.3.1 Time-Domain Audio Features

The time characteristics are output directly from the audio signal. They are often combined with some more sophisticated spectral features, which will be presented in the next chapter. This chapter analyzes the time characteristics of the audio signal used in the system for training segment classifiers.

Energy

Given a sound signal of discrete time x(n), the mathematical expression of its energy **(energy)** is defined as follows:

$$E_s = \sum_{n=-\infty}^{\infty} |x(n)|^2 \tag{4.2}$$

However, because we work at the level of short-term parts of the signal, the short-term energy is calculated as follows:

$$E(i) = \sum_{n=1}^{W_L} |x_i(n)|^2$$

where $\boldsymbol{x_i(n)}$ για n=1...W_L, is the sequence of audio samples of the ith window, which has length $\boldsymbol{W_L}$.

To make the energy independent of the length of each window, it is usually normalized by dividing it by this length. So, finally, the energy is given as follows:

$$E(i) = \frac{1}{W_L} \sum_{n=1}^{W_L} |x_i(n)|^2 \tag{4.3}$$

The energy can show many alternations and large deviations between the speech segments, since the speech signals contain phoneme which can be weaker or even periods of silence / pauses between words and sentences. For this reason, a mid-term statistic that can be derived from short-term energy, is the standard deviation of σ^2. If, for example, we compare the standard energy deviation of a speech segment with a music segment, we will notice that in the case of speech the standard deviation takes on a higher value than in the case of music, since music does not show such strong alternations.

Energy Entropy

Entropy of energy, is a metric that shows the abrupt changes in the energy of an acoustic signal. To calculate the energy entropy of a short-term window, we break the window into K sub-windows of a specific duration. Then for each of these sub-windows we calculate the energy according to the equation (4.2) and divide this value by the total energy of the short-term window. So we finally have the value sequence for each sub-window j as follows:

$$e_j = \frac{E_{subFrame_j}}{E_{shortFrame_i}} \tag{4.4}$$

where j is one of the K sub-windows (j = 1, ..., K) and i is one of the short-term windows of the total signal. This division converts the sub-window energy into a probability. The denominator of the above equation is given by the expression:

$$E_{shortFrame_i} = \sum_{j=1}^{K} E_{subFrame_j} \tag{4.5}$$

where the energy of the total short-term window I, is calculated from the sum of the actions of all its sub-windows.

The final step calculates the entropy of the e_j sequence as follows:

$$H(i) = -\sum_{j=1}^{K} e_j \log_2(e_j) \tag{4.6}$$

What one can observe is that if a sub-window has a high energy, then one of the probabilities of the e_j sequence will be high, resulting in a decrease of the entropy of the sequence (4.6). Therefore, this feature has a lower value if there are abrupt changes in the window energy, which is why it is used to detect significant energy changes. Thus, it is a feature that can help a classifier learn class separation. If, for example, we wanted to separate the types of music between electronic and classical, the entropy of energy would probably give a lower value to samples of electronic music, which show abrupt changes, than to samples of classical music.

Zero-crossing rate

Zero-crossing rate represents the rate at which the amplitude of an audio signal changes sign within a frame, that is, from positive, zero and negative, or vice versa. Mathematically defined by the following expression:

$$Z(i) = \frac{1}{2W_L} \sum_{n=1}^{W_L} |sgn[x_i(n)] - sgn[x_i(n-1)]| \tag{4.7}$$

where **sgn** is the sign function that gives 1 when its argument is non-negative and -1 when its argument is negative:

$$sgn[x_i(n)] = \begin{cases} 1, x_i(n) \geq 0 \\ -1, x_i(n) < 0 \end{cases}$$

also the expression is again been divided by the length of the window W_L.

Zero pass rate is a characteristic that can indicate how noisy a signal is. In noisy parts of a signal, it has a higher value than in non-noisy parts. It is also often used as a feature in speech recognition for voice activity detection. On the other hand, if we take the standard deviation of this feature, as we did in the previous one, we will notice in the case of speech, a greater value than in the case of music. This is because speech has a higher signal turnover rate. Therefore, this feature can be quite informative and helpful for classifiers, as in the case of binary classification between speech and music.

4.1.3.2 Frequency-Domain Audio Features

To export frequency-domain features we must first calculate the textbfdiscrete Fourier transform (DFT) of the sound windows [43] [44].

Discrete Fourier Transform

This transformation produces a frequency domain representation of the signal. That is, it converts a finite sequence of samples, equivalent distance, of a function, into a sequence of the same length, consisting of equivalent distance samples of a complex frequency function. This frequency function is the **discrete-time Fourier transformation (DTFT)**. The interval during which the DTFT is sampled is the inverse of the duration of the input sequence.

Therefore, DFT is the transformation, i.e. the way in which the conversion is done in frequency domain representation (finite sample sequence), while DTFT is the frequency

function from which the result sequence is derived. If the original sequence extends to all non-zero values of a function, its DTFT is continuous (and periodic) and DFT provides discrete samples of a cycle. If the initial sequence is a cycle of a periodic function, DFT provides all non-zero values of a cycle of DTFT.

Note that an effective algorithm for calculating the coefficients of this transformation is **fast Fourier transform (FFT)**.

The DTFT of an input sequence x(n), n = 0, ..., N-1, is given as follows:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} x(n)e^{(-jn\omega)} \tag{4.8}$$

$\boldsymbol{X(e^{j\omega})}$ which is given by the above relation is a continuous function of *omega*. Therefore, a digital computer cannot directly use this relation to parse $x(n)$. However, we can use samples of $X(e^{j\omega})$, to find an approximation of the$x(n)$. The idea of sampling $X(e^{j\omega})$ at points equal distances is actually the basis of DFT. As discussed above, DFT is based on DTFT sampling, given by the equation (4.8), at points of equal distances. $X(e^{j\omega})$ is a periodic function in ω with period 2π. If we take N samples in each period of $X(e^{j\omega})$, the distance between the frequency points will be $\frac{2\pi}{N}$. Therefore, the frequency of all the sinusoids we are looking for, to convert x(n) to frequency domain l form, will be of the form $\frac{2\pi}{N} \times$ k, where we can choose k = 0, 1, ..., N-1, with k the respective point / frequency sample of the sequence. Using compound exponentials similar to the equation (4.8), base functions will be $e^{(-j\frac{2\pi}{N}kn)}$.

Therefore, in a nutshell, DFT converts a time sequence / signal from N samples x(n), n = 0, ..., N-1, to another frequency domain sequence of complex numbers X(k), k = 0, ..., N-1 as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{(-j\frac{2\pi}{N}kn)}, k = 0, ..., N-1 \tag{4.9}$$

Finally **X(k)** is a sequence of N coefficients. K is one of the N samples in the frequency domain.

Inverse DFT - IDFT, does the reverse process, providing the exact reconstruction of the original x(n) signal. Essentially, it is a Fourier series, which uses DTFT samples as complex sinusoidal coefficients at the corresponding DTFT frequencies.

The following is the inverse of DTFT (DTFT):

$$x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{jn\omega}d\omega, n = 0, ..., N-1 \tag{4.10}$$

The following is the reverse DFT (inverse DFT - IDFT):

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{(j\frac{2\pi}{N}kn)}, n = 0, ..., N-1 \tag{4.11}$$

where X(k) indicates the coefficient / weight used for the complex exponential $e^{(j\frac{2\pi}{N}kn)}$. n is one of the N samples in the time-domain. With another approach, if we wanted to find the correlation (4.11) with which we obtain a discrete time signal from its corresponding frequency domain representation, we could think as follows:

1. All we need is to create a periodic signal from the N samples of the -finite term-sequence x(n). Thus, to calculate the DFT of N-points, we compose a periodic signal p(n) which is equal to x(n) for n = 0,1, .., N-1.

2. Then applying the discrete-time Fourier series extension, we can arrive at a –frequency domain- representation of the periodic signal, which is defined as follows:

$$\alpha_k = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{(-j\frac{2\pi}{N}kn)} \tag{4.12}$$

where N is the period of the signal.

3. From the above relation, the time-domain signal is defined as follows:

$$x(n) = \sum_{k=0}^{N-1} \alpha_k e^{(j\frac{2\pi}{N}kn)} \tag{4.13}$$

4. α_k differs from the DFT coefficients defined in the relation (4.9), only by the factor $\frac{1}{N}$. This is because the Fourier series of discrete time is periodic, while the coefficients DFT, X(k), are a finite duration sequence. Therefore, multiplying the relation (4.12) by N, we end up with X(k):

$$X(k) = N\alpha_k = \sum_{n=0}^{N-1} x(n)e^{(-j\frac{2\pi}{N}kn)} \tag{4.14}$$

5. Finally, the inverse DFT is calculated according to the above two relations (4.13) (4.14) as follows:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{(j\frac{2\pi}{N}kn)}$$

The following figure represents the processes followed in step 1, i.e. the initial discrete time sequence x(n) from which we create the periodic signal p(n) [43]:

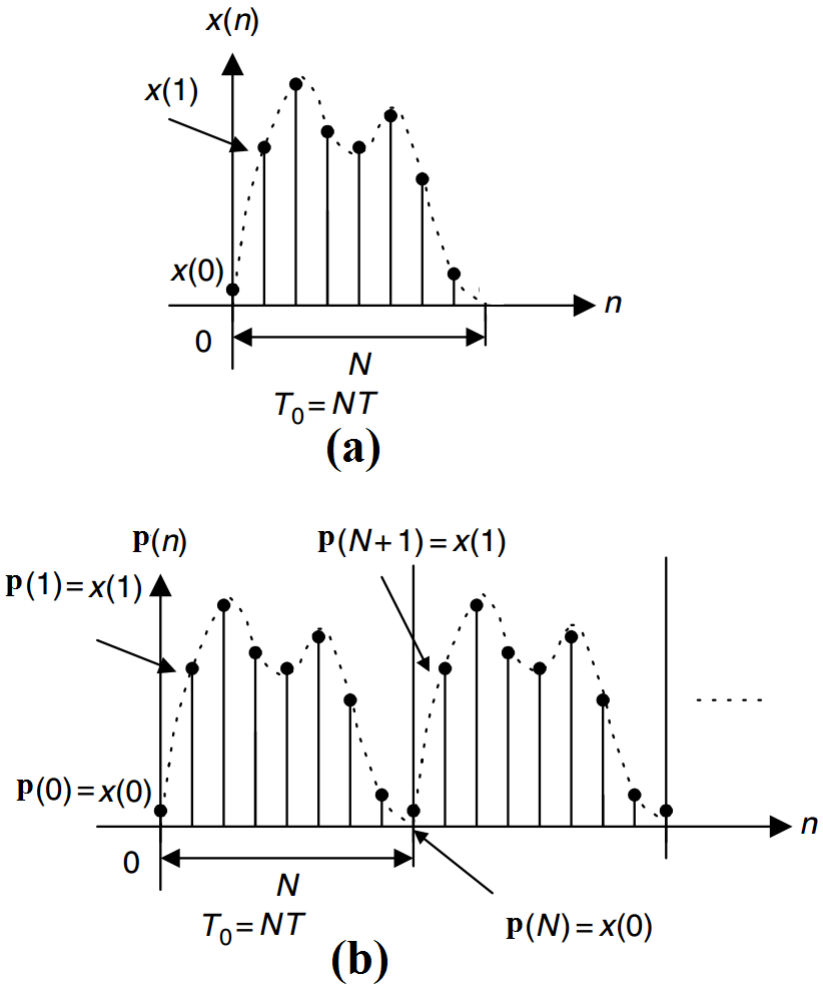


Figure 4.7: (a) The finite sequence x(n). (b) The periodic signal p(n), obtained from x(n).

Finally, an important observation can be made is : sampling a signal in the time domain leads to copies of the source signal in the frequency domain. Respectively, sampling $X(e^{j\omega})$ (4.8) in the frequency domain leads to copies of x(n) in the time domain.

Spectral Centroid

Having now analyzed the DFT coefficients, we are ready to present all the frequency domain characteristics that use them

One of them is the **spectral centroid**. The spectral centroid is the center of "gravity" of the spectrum. It shows, that is, where the center of mass of the spectrum is. Respectively, it has a strong connection with the impression of the brightness of a sound . Brightness reflects the amount of high-frequency information and is measured by correlating energy over a predetermined cutoff frequency with total energy. Thus, higher spectral centrifugal values correspond to brighter sounds. For example, a background noise or silence will have lower values of this statistic than some abrupt sounds.

The spectral center C_i of the ith sound window (short-term window) is defined as the mean frequency weighted by the backs, divided by the sum of the backs:

$$C_i = \frac{\sum_{k=1}^{Wf_L} kX_i(k)}{\sum_{k=1}^{Wf_L} X_i(k)} \tag{4.15}$$

where $X_i(k)$ with k = 1, ..., Wf_L is the magnitude of the DFT coefficients of the ith

window, or the width corresponding to bin k in the DFT spectrum, and k is one of the Wf_L samples in the frequency domain.

Note that normally the length of the short-term window is W_L, as is the number of DFTs of the window. However, the spectral characteristics are sufficient to use only the first half of these coefficients, since the second half serves to reconstruct the original signal. So Wf_L is finally, the number of coefficients used.

Spectral Spread

Spectral spread, is a characteristic that shows the mean deviation of the spectrum from the spectral centroid, which is usually related to the bandwidth of the signal. Defined as follows:

$$S_i = \sqrt{\frac{\sum_{k=1}^{Wf_L}(k - C_i)^2 X_i(k)}{\sum_{k=1}^{Wf_L} X_i(k)}} \tag{4.16}$$

Spectral spread measures how the spectrum spreads around the centroid. Thus, low values of spectral propagation correspond to signals whose spectrum is tightly concentrated around the spectral centroid. Noise-like signals usually have a large spectral propagation, while individual tonal sounds with isolated peaks will lead to a low spectral propagation. For example, if we compared the price of the spectral centroid for an electronic music window, we would probably get a higher value compared to similar classical or jazz music windows. This is because electronic music usually has a wider distribution around the spectral centroid.

Spectral Entropy

Spectral entropy, is calculated in a similar way as the energy entropy we came across in the temporal characteristics, with the difference that it takes place in the frequency domain.

Initially the spectrum of each short-term window is divided into L sub-windows (sub-bands /bins). The energy of the f sub-window is then calculated for f = 0, ..., L-1 which is normalized by the total spectral energy. This relationship is defined as follows:

$$n_f = \frac{E_f}{\sum_{f=0}^{L-1} E_f} \tag{4.17}$$

where n_f is the normalized spectral energy of the f sub-window.

The entropy of this action for all sub-windows is then calculated as follows:

$$H = -\sum_{f=0}^{L-1} n_f \log_2(n_f) \tag{4.18}$$

Entropy can be used to represent how many peaks spectral representation (peakiness) has. The resulting characteristic is low for a spectrum with many different spectral peaks and high for a flat spectrum. This can be justified by the fact that if we have high spectral energy in a sub-window, the probability of n_f in that sub-window increases and therefore the entropy decreases for the overall short-term window. Therefore, the more high-energy sub-windows we have (peaks), the lower the entropy value and vice versa.

As an example, if we calculated the standard deviation of the spectral entropy sequences of some parts of speech-windows or ambient sounds, then it is very likely that this value was lower for ambient sounds and higher for speech. This is because speech presents many different spectral peaks and changes. Therefore, the spectral entropy changes from window

to window (windows with more vertices have lower spectral entropy, while windows without many vertices have higher spectral entropy), resulting in a larger standard deviation. In contrast, in the case of ambient sounds, a more normalized spectrum (not steep peaks) is presented, which produces a small standard deviation of spectral entropy.

Spectral Flux

Spectral flux is a measure of the change in power spectrum between two consecutive frames / windows, which is calculated as the square difference between two normalized values of the spectrum. Its mathematical expression is defined as follows:

$$Fl_{(i,i-1)} = \sum_{k=1}^{Wf_L} (EN_i(k) - EN_{i-1}(k))^2 \tag{4.19}$$

where $EN_i(k)$ is the k normalized DFT factor of the i-th box / window and is defined as follows:

$$EN_i(k) = \frac{X_i(k)}{\sum_{l=1}^{Wf_L} X_i(l)} \tag{4.20}$$

As an example, if we compared the mean value of spectral flow sequences calculated from segments of two audio classes (music and speech), then it is more likely that the spectral flow values would be higher for the speech class. . This is because speech contains many local spectral changes due to the rapid alternation of phonemes.

Spectral Roll-off

Spectral roll-off is the frequency defined for each frame / window as the central frequency for a bin of the spectrogram, so that a certain percentage of roll energy (roll percentage) in that frame , is contained in this bin and in the smaller bins. The usual values used as a percentage are between 0.85 and 0.95.

Giving a mathematical definition of spectral attenuation, if the m-bone DFT coefficient corresponds to the spectral attenuation of the i-th frame, then it meets the following equation:

$$\sum_{k=1}^{m} X_i(k) = C \sum_{k=1}^{Wf_L} X_i(k) \tag{4.21}$$

where C is the above percentage. Essentially, if we had set C=90%, then the above equation shows us that in order for mth bin to be the frequency we are looking for (i.e. spectral attenuation), we need the sum of all DFT coefficients from bin 1 to bin m to be equal to 90 % of the sum of all DFT coefficients within each box (i).

The spectral attenuation frequency is usually normalized by dividing it by the length of the window Wf_L, thus taking values from 0 to 1. The value 1 corresponds to the maximum frequency of the signal.

Spectral attenuation describes the spectral shape of an audible signal and can also be used to differentiate between phonetic and non-phonetic sounds.

As an example, if we calculated the spectral attenuation of sound sequences from different classes such as classical music, electronic music, and jazz music, we would probably observe higher values in electronic music tracks than in the rest. This is because most of the energy of electronic music's spectrum occurs at high frequencies and consequently the values of its spectral attenuation are higher. Classical and jazz music, on the other hand, mostly concentrate their spectral energy on lower frequencies. Also, the deviation of spectral

attenuation would be more pronounced in the case of electronic music, since it has a wider range.

MFCCs

Mel-Frequency Cepstrum Coefficients (MFCCs) are very popular in speech analysis and are a type of **cepstral** signal representation.

Before we get into the details of MFCCs, it would be helpful at this point to explain what exactly **cepstrum** is. Cepstrum is the information of the rate of change of the spectral bands. Cepstrum comes from the word spectrum, inverting the first four letters (spec -> ceps). The periodic signals in the time domain are represented as sharp peaks in the frequency domain (frequency spectrum), following the application of the Fourier transform as we have noticed. Using the logarithm of the widths of the spectrum and then recalculating the spectrum of this logarithm through a cosine transform, the new spectrum displays a peak whenever there is a periodic element in the original time signal. This new spectrum is neither in the time domain nor in the frequency domain. So Bogert et al [45]. called it "quefrency domain". Cepstrum, then, is the spectrum of the logarithm of the spectrum of the time signal. The following figure shows the steps taken to produce cepstrum [46]:

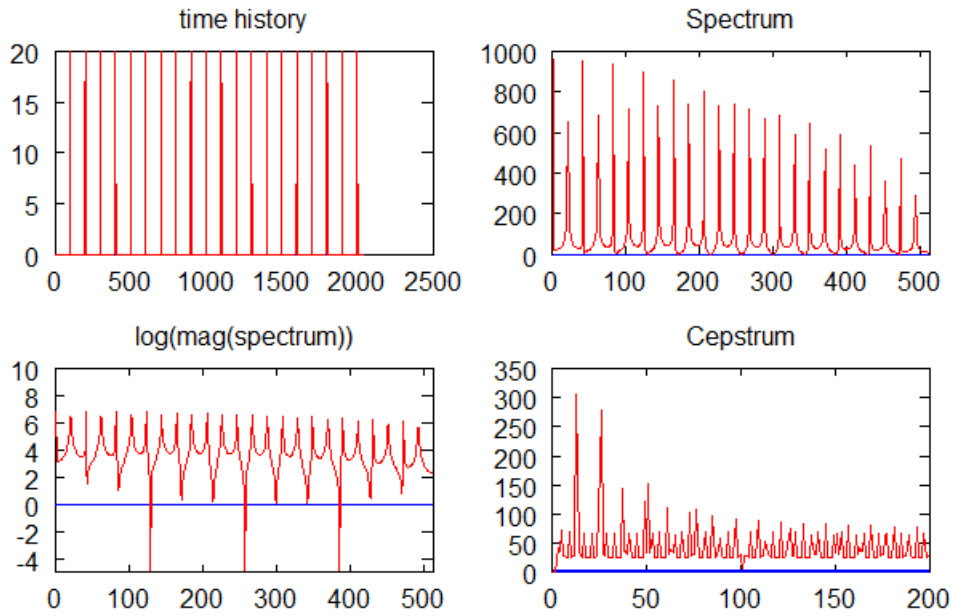


Figure 4.8: Steps to Form Cepstrum from Time History

The next parameter to be analyzed is Mel-frequency. This term refers to frequencies that have been scaled according to the **Mel scale**. This scale correlates the frequency of the tone perceived by the human ear with the actual frequency. In essence, it scales the frequency so that it is closer to what the human ear can hear. For example, it is known that humans can distinguish frequencies more easily in the low frequency range. Thus, if one hears two sounds at 300 Hz and 400Hz respectively and two sounds at 900Hz and 1kHz respectively, then the perceived distance between the last two sounds is probably much larger than that between the first two sounds, even though it is the same (100Hz).

In other words, the Mel scale is a perceptual scale of frequency intervals, which if judged by human hearing, are perceived as equal distance intervals.

There are several **formula** for Mel scale conversion or otherwise **frequency warping functions** that have been proposed over the years. A known such formula is the following [47]:

$$m = 2595 \log_{10}(1 + \frac{f}{700}) = 1127 \ln(1 + \frac{f}{700}) \tag{4.22}$$

where **m** is the frequency in Mels and **f** is the frequency in Hz.

The above formula can also be defined for the reverse process (from Mels to Hz) as follows:

$$f = 700(10^{\frac{m}{2595}} - 1) = 700(e^{\frac{m}{1127}} - 1) \tag{4.23}$$

Having explained the key concepts, the steps to be taken to generate MFCCs are as follows:

1. DFT of the time signal for each window is calculated (short-term window), since as we have mentioned the export of the characteristics is done at the level of short-term windows. The following steps show the procedure for a window.

2. the spectrum output after DFT is given as input to a **filter bank** Mel-scale, which consists of M filters, which are usually overlapping triangular frequency responses. To create these filters, we first set the lowest (f_1) and highest (f_h) frequency of the filter bank in Hz. Then we convert these frequencies to Mels according to the relation (4.22) and depending on the number of filters we will use, we divide the range ($f_h - f_1$) into points of linear distance (equal distance) . Each filter in the filter bank, as we have mentioned, is triangular with a response of 1 at the center frequency which decreases linearly to 0 until it reaches the center frequencies of the two adjacent filters where the response becomes 0. Thus, the number of points at which divide the range is M + 2, where M is the number of filters. If for example we have 10 filters, then we divide into 12 points. Usually, 26-40 filters are actually used. Then we convert the Mels to Hz according to the relation (4.23) and finally we create the filters as follows:

$$H_m(k) = \begin{cases} 0 & , k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & , f(m-1) \le k \le f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & , f(m) \le k \le f(m+1) \\ 0 & , k > f(m+1) \end{cases} \tag{4.24}$$

where **m** is the filter number, **f (.)** is the list of M+2 mel-spaced frequencies we created above and **k** are the points of the original spectrum (frequencies).

An example of filters is shown below:

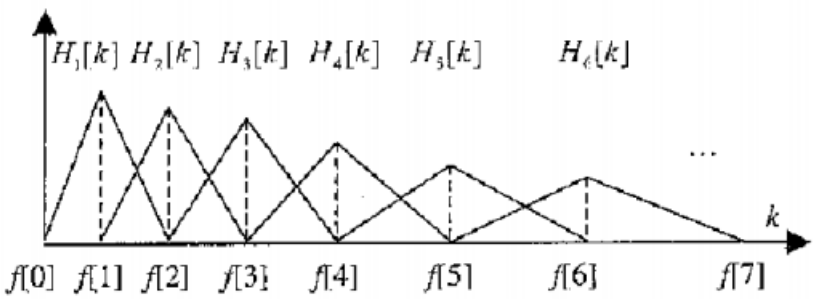


Figure 4.9: Triangular Filters for Mel-Cepstrum Computing

The mel spectrum created after the application of the filters is given by the following equation:

$$s(m) = \sum_{k=0}^{N-1} [|X(k)|^2 H_m(k)] \quad , 0 \leq m \leq M-1 \tag{4.25}$$

where **X (k)** are the widths of the spectrum (coefficients), $\boldsymbol{H_m(k)}$ are the mathematical expressions of the filters and **m** each of the M total filters. An example of applying filters to the spectrum is shown below [48]:

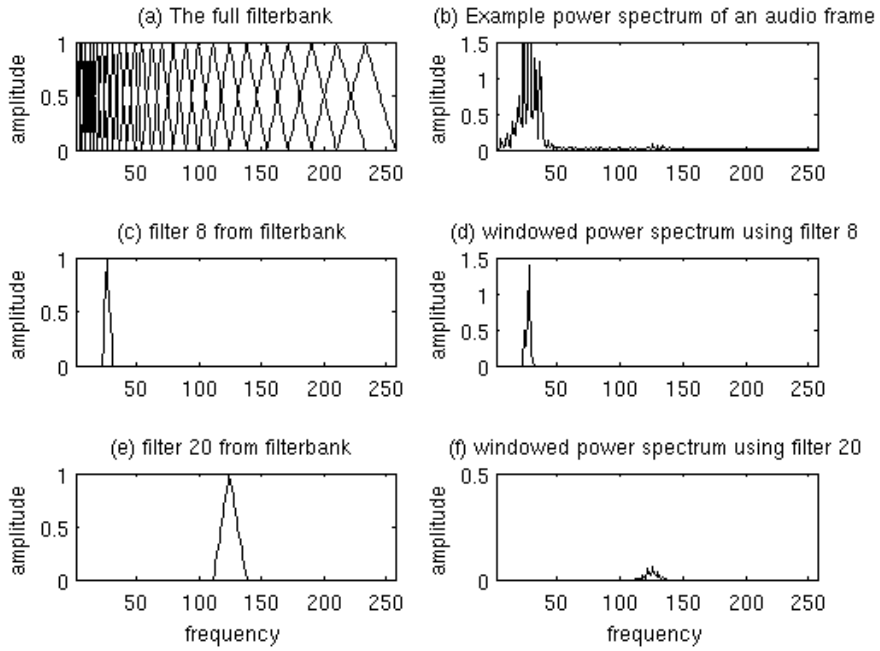


Figure 4.10: Mel Filters on power spectrum windows

3. In the next step we calculate the logarithm for each of the actions exported by the filters.

4. Finally, we calculate the **discrete cosine transform (DCT)** or otherwise the **Fast Fourier Transform (FFT)** of the above logarithms to arrive at the cepstral coefficients.

   The mathematical expression of the coefficients is defined as follows:

$$c(n) = \sum_{m=0}^{M-1} \log_{10}(s(m)) cos(\frac{\pi n(m-0.5)}{M}) \quad , n = 0, ..., C-1 \tag{4.26}$$

   where **C** is the number of MFCCs factors used.

   In most applications, it has been established that the first 12 to 13 coefficients are chosen because they are considered to give enough information to the classifiers to help them separate the classes.

   The final characteristics (13 numbers per short-term window) are called Mel Frequency Coefficients - MFCCs.

All of the above steps for extracting MFCCs are summarized in the following figure [49]:

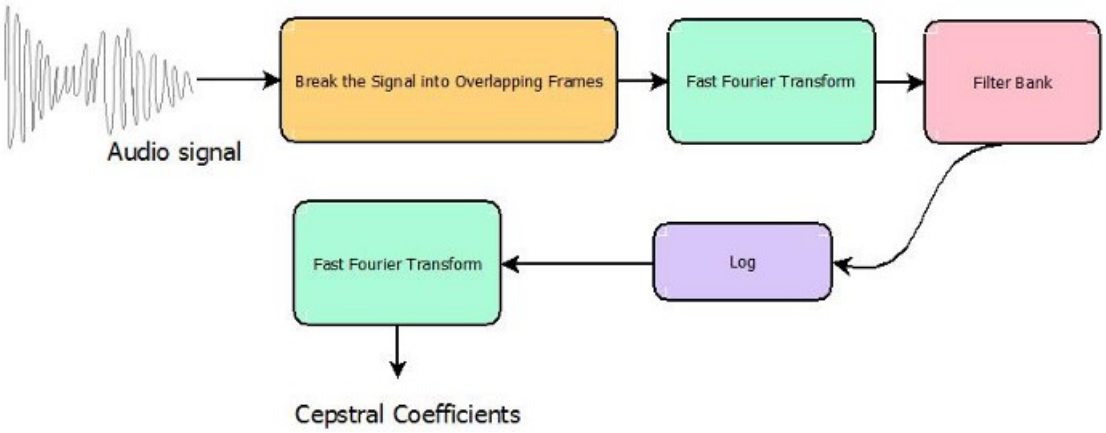


Figure 4.11: Steps to Export MFCCs

Chroma Vector

The **chroma vector** or **chromogram** or **pich class profile (PCP)** is a representation of 12 elements of the spectral energy of an acoustic signal.

For a better and easier understanding of this feature, it is important to first analyze some concepts.

Tone (pitch) is a perceptual property of sounds that allows them to be classified on a frequency-related scale. In other words, the tone is the quality that makes it possible to rate the sounds as "higher" and "lower" in the sense associated with musical melodies. The tone can only be determined to sounds that have a frequency that is clear and stable enough to stand out from the noise [50].

In music, a **octave** or a perfect octave is the space between one tone of music and another at twice the frequency [51].

Pitch class, is a set of all tones that differ by one or more octaves. For example, the tonal class C, consists of Cs in all octaves and is defined as follows: $C_n : nisaninteger = ..., C_{-2}, C_{-1}, C_0, C_1, C_2, C_3, ...$ [52].

An **equal temperament** is a musical **tuning system** which approaches intervals by dividing an octave (or a space) into equal steps. This means that the ratio of any two adjacent note frequencies (notes) is the same, which gives the perception of an equal step size, since the tone - which is the perceptual property - is considered approximately equal to the logarithm of the frequency.

In classical music and (Western – type music) in general, the most common tuning system is the **twelve -tone equal temperament (12-TET)**, which divides the octave into 12 parts that are equivalent on a logarithmic scale. Each of these intervals is called **semitone** [53]. The human perception of tone is periodic in the sense that 2 tones are perceived as similar in "color" (play a similar harmonic role), if they differ by one or more octaves, where in our case, an octave is defined up to a distance of 12 tons.

A tone can be divided into two components: **tone height** and **color (chroma)**. Tone height refers to the octave number, while color refers to the corresponding orthographic feture of tone. In Western musical notation, the 12 tone characteristics are given by the set: : {C, C#, D, D#, E , F, F#, G, G#, A, A#, B}. By enumerating the color values, we define this set with indices [0: 11] where c = 0 refers to the color C, c = 1 in C# and so on. Note that in **equal-tempered scale** different tone spellings, such as C# and Db, refer to the same color.

The following figure shows the chroma circle, i.e. the periodicity of tones that is perceived by humans and which cannot be represented by a linear tonal scale, except by an

equal-tempered scale we saw above [54]:

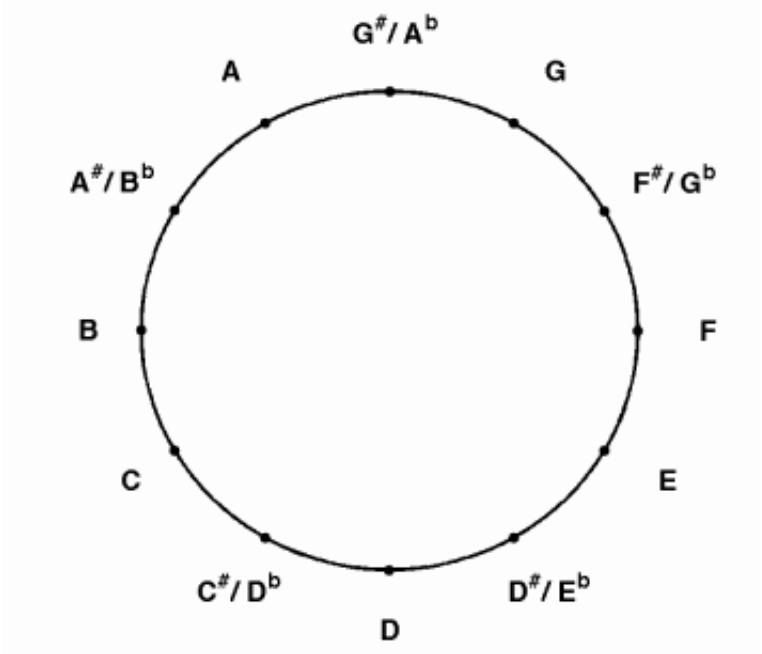


Figure 4.12: Color Circle

Finally, the main idea of color characteristics is to gather all the spectral information related to a given tonal order, in a single factor. Thus, to calculate the color vector, the DFT factors of a short-term window are grouped into 12 points (bins), where each point represents one of the 12 equal-scale tonal order of the western music we mentioned above. Then, the mean value of the logarithm of the width of the corresponding DFT coefficients ($\log_{10} X_i(k)$) is then calculated for each point (bin).

The mathematical expression for each bin / tonal class is given as follows:

$$v_z = \sum_{n \in S_z} \frac{\log_{10} X_i(n)}{N_z} \quad , z \in 0, ..., 11 \tag{4.27}$$

where $\boldsymbol{v_z}$ is the sum of the **pitch coefficients** belonging to the color (chroma) z, $\boldsymbol{X_i(n)}$ is the DFT frequency factor **n** for the short-term window **i**, **z** is the octave index (from 0 to 11), $\boldsymbol{S_z}$ is a subset of the frequencies corresponding to the DFT coefficients of bin z and $\boldsymbol{N_k}$ is the number of components of the subset S_z (i.e. the number of frequencies belonging to this subset).

The color vector or chromogram consists of the above v_z for each z, i.e. for each of the 12 colors (chroma).

A chromogram representation is shown in the following figure [55]:

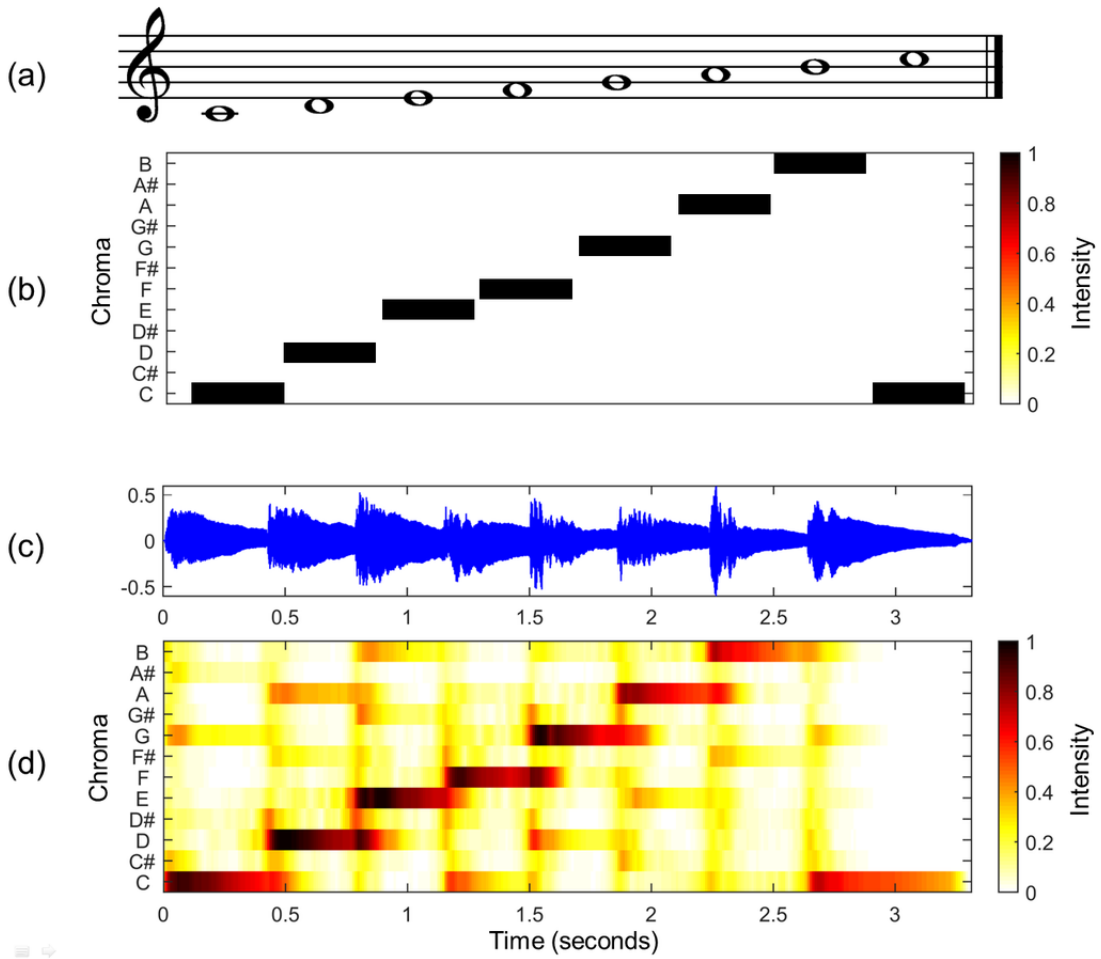


Figure 4.13: (a) Musical score of a C-major scale. (b) Chromogram extracted from the score. (c) Audio recording of a C-major scale from a piano. (d) Chromogram obtained from the audio recording

.

The image 4.13 chromograms for a C-major scale, which have been extracted either from the musical score (a -> b) or from the sound recording (c -> d).

As an example of the application of this feature, if we had a chromogram of a musical signal versus a chromogram of a speech, we would most likely observe in the first case certain color coefficients predominating, which would be constant for short periods of time, while in the second case, noisy color ratios. This can be explained by the fact that in music, there are intervals where a certain note or notes from certain tonal classes dominate, while in speech the phonemes differ from each other in tonality and the alternations are very fast. After all the audio features have been described, we can summarize how many and which features are extracted from the audio signals and used to train system classifiers:

1. Zero-crossing rate
2. Energy
3. Energy entropy
4. Spectral centroid
5. Spectral spread
6. Spectral entropy

7. Spectral flux
8. Spectral rolloff (spectral attenuation)
9. MFCCs (13)
10. Chroma (color vector) (12)
11. Chroma standard deviation-std (standard color vector deviation)

Of the Mel-Frequency Cepstrum (MFCCs), the first 13 are used as characteristics. The color vector (chroma) also contains 12 values as we have mentioned, while finally the standard deviation of the values of this vector is calculated, which is counted in the characteristics.

In conclusion, we come up with 34 features in total, which as we have already mentioned, are exported for each short-term window separately.

For each short-term window, the difference between these 34 values, characteristic of each window and the characteristic values of the immediately preceding window is also calculated. Therefore, in each short-term feature sequence, we add its derivative, i.e. its difference from the previous frame. Thus, another 34 (the 34 derivatives / deltas) are added to the 34 features and we end up with 68 features per short-term window. Then, for each medium-term window, 2 statistics are calculated: the average and the standard deviation of the values that each feature gets in the short-term windows. For example, if each medium-term window contains 2 short-term windows, then the mean and standard deviation of the zero-passage rate in these two windows will be calculated to extract 2 statistics for the medium-term window. The same goes for the other features. Therefore, each attribute gives 2 statistics at mid-term level. So we end up with 68 x 2 = 136 features per medium-term window.

The reason why both statistics are used (mean and standard deviation), is because as we have seen, each of them can give different information about the sound signals and consequently contribute to the separation of classes.

The python open source library called pyaudioanalysis was used to extract these features [56] .

4.1.4 Segment Classifiers - Emotion Recognition

The classifiers used in the system refer to emotional characteristics, either categorical or dimensional. For a better understanding of these concepts we will explain the psychology of emotion [63].

The psychology of emotion

Due to the subjectivity of emotion, different scholars tend to have different perceptions and thus rarely reach a consensus in the bibliography on the definition of emotion. According to the modern study [64], emotion is defined as "the complex pattern of physical and mental changes that includes physiological stimulation, emotions, cognitive processes, visible expressions and specific behavioral reactions in response to a situation that is considered as personally important ".

There are two categories of characteristic emotions: **categorical attributes** and **dimensional attributes**.

Categorical Attributes consist of some basic classes of emotions. A different set of emotions may be required for different fields and different systems / models. According to Ortony and Turner [65], basic emotions are often the primitive building blocks of other non-essential emotions, which are considered variations, or mixtures of basic emotions. Ekman [66] in his study of the analysis of facial expressions, suggested six basic emotions: anger, disgust, fear, joy, sadness and surprise. Although Ekman's core emotion set is often

used for emotion mining and modeling, there are also a wide variety of alternative emotion classification systems that have been accumulated by various other scholars.

The main disadvantage of the categorical model is that it has a lower resolution than the dimensional model because it uses categories. The number of emotions and their tones encountered in different types of communication are much richer than the limited number of emotion categories in the model. The smaller the number of groups in the categorical model, the greater the simplification of the description of emotions [67]

In recent decades, to better understand human emotions, researchers have tried to create dimensional spaces that can mainly capture the similarities and differences between different emotional experiences. Wundt [68], proposed the first **dimensional model** by disassembling the space of emotions along three axes, namely: **valence (positive-negative)**, **arousal (calm-excitment)** and **intensity (intensity-relaxation)**. Although proposed more than a century ago, Wundt's model successfully laid the groundwork for later work and has become the basis for many later theoretical developments. Inspired by Wundt's work, several dimensional models of emotions were developed, although many of which incorporated only two of the three dimensions proposed in Wundt's work (dimensions of valence and arousal, but not the dimension of intensity). The predominant two-dimensional emotion models include: the circular model [69] and variants of [70], the vector model [71] and the positive activation-negative activation model (Positive Activation Negative Activation - PANA) [72]. Of all the dimensional models, **Circumplex model** is the most widely used in emotion research. The **circumplex model** [69] suggested that all emotional experiences be distributed in a circular space consisting of vertical axes, **valence and arousal**. Although there are other three-dimensional models of emotions, it is argued that the majority of emotions can be better explained in terms of arousal and valence dimensions using the two-dimensional model.

In the literature, the field of Speech Emotion Recognition (SER) has been very busy, with many researches being conducted and published on this subject. Some of the state-of-the-art implementations use neural networks, by combining convolutional networks (CNN) with long short-term memory networks (LSTM) [73], by looking at deep multi-layer neural networks [74], or by using recurrent encoders and combining information from both the audio and the text of the speech (Multimodal Dual Recurrent Encoder) [75].

Moreover, in addition to the information of the audio or the text of the speech, corresponding researches have been also carried out in the part of the analysis of the emotional behavior (affective behavior analysis) from audiovisual information, ie the automatic analysis of the three main behavioral tasks: valence-arousal estimation, recognition of basic emotional expressions (categorical characteristics) and action unit detection(facial expressions and movements). An audiovisual database containing labels for all three different behavioral tasks listed above is Aff-Wild2 [76]. On this dataset, experiments have been performed with different types of deep neural network architectures, using mainly cropped images from the original data and investigating either each task separately [77] [78] or utilizing the use of multi-task learning [79]. In some other proposed methods, the visual information is considered separately from the audio one, using a convolutional network and a recurrent network (CNN-RNN) or even a combination of these two information [80].

In this dissertation the audio and textual information of speech will be used, in order to investigate the problem of speech emotion recognition both in terms of its categorical characteristics and in terms of its dimensional characteristics. Having understood the meanings of the categorical and dimensional characteristics in terms of emotions, but also the meaning of the two-dimensional circumplex model, we can now analyze the classifiers used by our system.

As shown in the figure 4.1, the system uses three partition classifiers for audio. A

classifier that sorts sounds into four **emotions** or otherwise categorical characteristics: anger, sadness, neutrality, joy. A classifier that sorts sounds according to **valence**: positive, moderate, negative and one that sorts them based on **arousal**: high, medium, low. Valence and arousal, as we saw above, are a two-dimensional space of emotional characteristics.

4.1.5 Datasets

The following data sets were used to train these three classifier models:

- **Emovo**
  Emovo is an Italian emotional speech database. It was created by the voices of 6 actors (3 women and 3 men), who said 14 sentences that simulate 6 emotional states (disgust, fear, anger, joy, surprise, sadness) plus the neutral state. These emotions are the well-known Big Six found in most emotional speech bibliography [57].

- **Emo-DB**
  Emo-db is an emotional speech database created by the Technical University of Berlin. Contains recordings of 10 speakers (5 women and 5 men) aged 21 to 35, who said 10 sentences that simulate 6 emotional states (anger, boredom, disgust, anxiety / fear, joy, sadness) plus the neutral state [58].

- **SAVEE**
  The SAVEE database was recorded by four Native English male speakers, graduate students and researchers at the University of Surrey aged 27 to 31 years. Emotion has been psychologically described in 6 distinct categories: anger, disgust, fear, happiness, sadness and surprise plus the neutral state. This database includes 15 TIMIT sentences per emotion: 3 common, 2 based on each emotion and 10 general that are different for each emotion and vocally balanced. The 3 common and the $2 \times 6 = 12$ specific sentences of emotions were also recorded as neutral to give 30 neutral sentences [59].

- **RAVDESS**
  RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) provides speeches and songs in audio and video format. For this speech quality assessment system, only audio speech samples from this database were used. RAVDESS database contains 1440 files: 60 tests per actor x 24 actors = 1440 samples. Speakers are 24 professional actors (12 women, 12 men). Speech emotions include 6 expressions: calm, joy, sadness, anger, fear, surprise and disgust. Each expression is produced on two levels of emotional intensity (normal, strong), with an additional neutral expression. Therefore, 7 emotional expressions in total [60].

- **IEMOCAP**
  IEMOCAP (Interactive Emotional Dyadic Motion Capture) database is an active, multimodal, and polyphonic database compiled at the University of Southern California. Contains approximately 12 hours of audiovisual data, including video, speech, motion capture, text transcripts. It consists of binary sessions where the actors perform improvisations or scripts, specially selected to evoke emotional expressions. Contains recordings by 10 actors (5 women and 5 men), which are highlighted by many commentators in categorical attributes: anger, joy, excitement, sadness, frustration, fear, surprise, but also a neutral state, as well as dimensional attributes such as valence, activation and dominance [61].

- **Emotion Speech Movies**
  This database is not open source like the previous ones, i.e. it is not available for free

for research use. It is a base that has been created in the framework of a doctorate by the researcher Theodoros Giannakopoulos [62]. Contains audio files, scenes from movies that are divided into the following classes of emotions: anger, fear, joy, sadness and neutrality.

In the system, the emotion classifier is trained in samples of four classes as mentioned above: anger, sadness, neutrality and joy. Nevertheless, one can observe that the data sets used contain more classes of emotions. Thus, from the datasets, only the samples of the classes corresponding to the classes of interest were used, while where there was the class "excitement" it was combined with happiness, since they are quite related expressions.

The following table shows the emotion classes we end up with, as well as how they result from the emotion classes contained in the data sets:

	Final Classes	Emotions (all datasets)
Emotion	**Sad**	Sad
	Angry	Angry
	Neutral	Neutral
	Happy	Happy, Excitement

Table 4.1: Final classes of emotion

For the valence classifier, one can observe that the only data that exists is that from the iemocap database, which in addition to emotions, also contains dimensional tags such as valence. This label is a continuous real number with a maximum value of 5.5 and a minimum value of 1, which indicates the degree of vigor. The higher the value of this label, the stronger the vigor and vice versa.

The model used in the system to predict the strength is a classification and not a regression model. This means that it does not anticipate continuous values but discrete classes. For this reason, the above tag, which is a continuous value, was converted into three distinct classes. This conversion was done by dividing the value range (1 to 5.5) into three equal intervals. Thus, samples with a valence label in the range [1,2.5) are considered as "negative valence", samples with a label value in the range [2.5,4) are considered as "neutral valence" and samples labeled in the interval [4,5.5] are considered as "positive valence". A similar procedure was followed for data collection for the arousal classifier. In this case, the iemocap database again contains samples labeled "activation" corresponding to the arousal. This label is continuous, as in the case of valence, with a maximum value of 5 and a minimum value of 1. Thus, the conversion to distinct classes was performed as follows: samples with an activation label in space were considered "low arousal" samples [1,2.3). "Neutral/moderate arousal" samples were considered to be samples labeled in the interval [2.3,3.6). Finally, samples with an activation tag in space [3.6.5] were considered "high arousal" samples.

The other bases (except iemocap) do not have samples sorted by arousal, but only by emotion. Thus, in the same way as before, emotion categories were collapsed to produce the desired classes. More specifically, the categories of low arousal, moderate / neutral arousal and high arousal were formed as follows: in the class "low arousal" were included the samples of the following emotional expressions: sadness, boredom, calm. The "moderate / neutral arousal" class included samples with emotional expressions: neutrality. Finally, in the class "high arousal" were included the samples that are marked as: joy, anger, surprise, fear, disgust.

The following table shows the final classes for valence and stimulation, as well as how they are derived from the data contained in the databases:

	Final Classes	Values of **Valence (V)** and **Arousal (A)** (iemocap)	Emotions (rest datasets)
Valence	**Negative Valence**	$1 \leq V < 2.5$	Sad, Angry, Fear, Disgust, Boredom
	Neutral Valence	$2.5 \leq V < 4$	Neutral, Surprise
	Positive Valence	$4 \leq V \leq 5.5$	Happy, Calm
Arousal	**Low Arousal**	$1 \leq A < 2.3$	Sad, Boredom, Calm
	Neutral Arousal	$2.3 \leq A < 3.6$	Neutral
	High Arousal	$3.6 \leq A \leq 5$	Happy, Angry, Surprise, Fear, Disgust

Table 4.2: Final classes of arousal and valence

The above distribution of emotion data in the valence and stimulation tag classes follows the two-dimensional circumplex model described in the previous subchapter, which in this case concerns all the set of emotions presented in the databases used. Thus, in this case it is formed as follows:

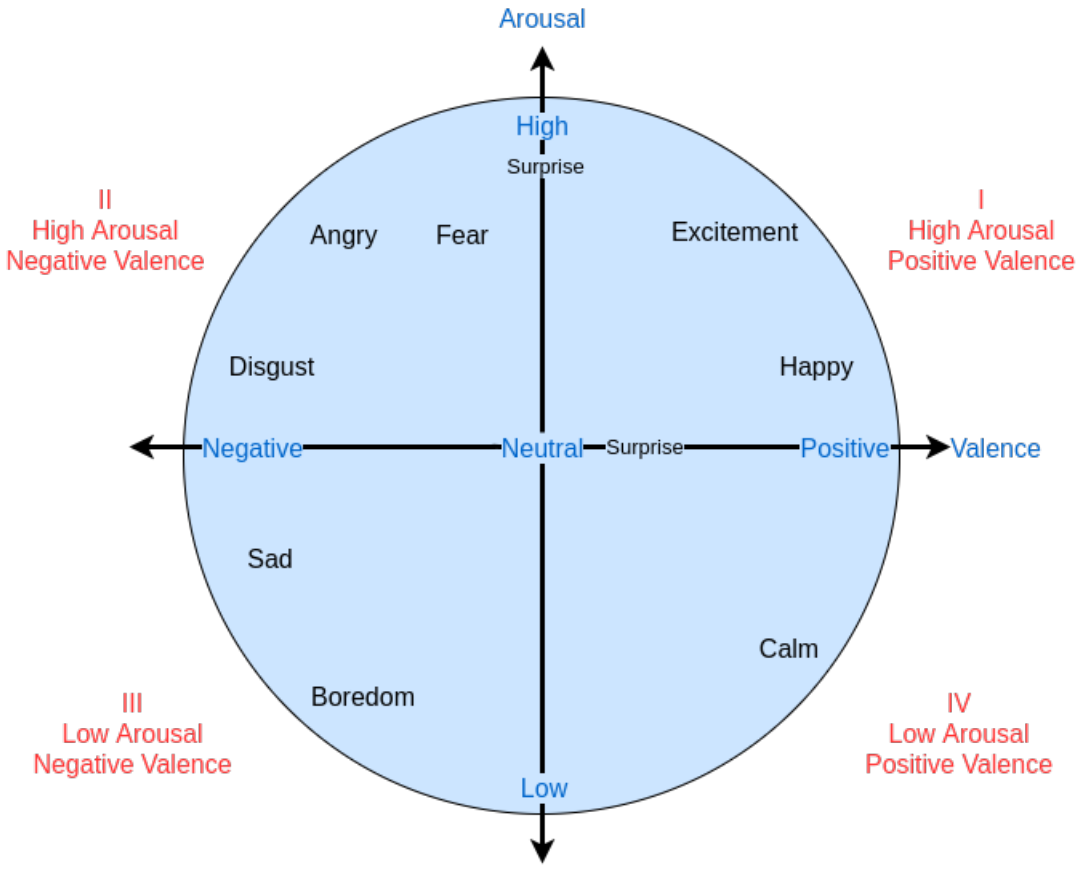


Figure 4.14: Database emotions introduced in the circular model (circumplex space)

Finally, after the new configuration of the data, each sample follows the procedure described in the previous subchapters: splitting into short-term windows, extracting characteristics for each of these windows, and finally producing statistics for each medium-term window. For the department-classifier models trained in this dissertation, mid-term windows of 3 seconds and short-term windows (0.05 seconds) were used. These window lengths

are ideal for speech, as they are neither too small to make it difficult to find information, nor too long to shrink too much information in one window. There is another step that is been followed for education, that of the long-term averaging we mentioned at the end of chapter 4.1.2. Since, in datasets, each audio file is classified into a class, the data is already considered presegmented and for this reason we need a long-term representation, i.e. a single vector of attributes for each file. To achieve this, we take the average of the characteristics-statistics of the medium-term windows. Having 66 features per audio file, we are ready to train the models.

4.1.6 Experiments and Results

For the training process and the evaluation of the models, experiments of three different types were performed: experiments for each data set separately (inner-dataset evaluation), training experiments in all data sets except one that is used for evaluation (testing) (cross-dataset evaluation) and finally merged data set evaluation experiments.

Inner-Dataset Evaluation

The following table shows the performance of the models for emotion, valence and stimulation, in each dataset separately (inner dataset evaluation). The metrics presented are f1 macro, i.e. f1 is calculated for each class separately and at the end the average of these f1. The machine learning algorithm used is svm with kernel rbf. Hyperparameter tuning was performed via grid search for different values of parameter C ([0.001, 0.01, 0.5, 1.0, 5.0, 10.0, 20.0]). Cross validation was used with 2 repetitions for cases where the samples exceed 10000 in number, 5 repetitions for cases where the samples exceed 2000 in number, 10 repetitions for cases where the samples exceed 1000 in number and 100 iterations if the samples are less than or equal to 1000. This means that for each parameter value, the data were randomly divided k times (as many as the iterations). In each separation, two sets emerged from the random separation of data into 80 % which was for training and 20 % which was for evaluation. Then, the model for the respective hyperparameter value and the given split was trained and evaluated. The final performance of the model for each hyperparameter value was given by the average of the returns in all iterations / splits. Finally, the hyperparameter with the maximum overall performance was selected. The results of the best models are presented below:

Classification Task	Dataset						Average
	Iemocap	Savee	Emovo	Emo-db	Ravdess	EmotionSpeechMovies	
Emotion	79.7	71.2	75.5	80.6	67	50.4	**70.7**
Valence	52.9	62.3	59.5	76	63.9	53.2	**61.3**
Arousal	60.3	75.3	79.9	81.9	68.3	64.1	**70**

Table 4.3: Inner-dataset Evaluation

Cross-Dataset Evaluation

The following table presents the results for the second category of experiments (cross-dataset evaluation). In this category as we have mentioned, in the training all datasets are taken into account except one which is used during the testing of the model. The metrics presented concern the f1 macro. The algorithm used here is svm rbf and again followed the same grid search procedure with cross-validation as above, during the training of the model. The results no longer relate to what emerged from cross validation during training, but what emerged from testing in the test dataset, after training the model:

Classification Task	Test Dataset						Average
	Iemocap	Savee	Emovo	Emo-db	Ravdess	EmotionSpeechMovies	
Emotion	39	36	45.5	57.6	29.8	36.6	**40.8**
Valence	39.7	37.9	32.7	37	26.3	42	**35.9**
Arousal	40.1	41.8	40.3	51.1	38.4	38.6	**41.7**

Table 4.4: Cross-dataset Evaluation

Comparing the results of the above two tables, we can observe that in the cross-dataset evaluation (table 4.4), the results are very bad. It seems to be a little better than a random classifier. On the contrary, in the case of inner-dataset evaluation the percentages are quite satisfactory. This means that the problem that these models are called upon to solve is directly dependent on the specific subsector. As a specific sub-sector, we mean the type of speech which differs per dataset, the conditions under which the samples were created, the subject of the speeches, the speakers and their gender, the way the samples are marked etc. This explains the fact that the model gives much higher performance when evaluated in the same dataset in which it was trained, while when evaluated in a different one, its performance drops dramatically. This is an expected phenomenon as it is known that the problem of Speech Emotion Recognition (SER) is difficult precisely because it depends on and is influenced by many factors.

Merged-Dataset Evaluation

For the above reasons, the third type of experiment was created, which contains all the datasets together. The reason why datasets were merged is to include as many domains as possible, from those representing each dataset separately. The data were pooled and then separated into train and test set. Here the same grid search procedure was performed again with cross-validation that mentioned before, during the train, while the results presented concern the f1 macro metric which was calculated in the test set. In these experiments which they give and the final models that will be used by our system, were included tests with the xgboost algorithm as well as with the neural cnn, in addition to svm rbf.

The python library called deep_audio_features[81] was used for the cnn training, which uses mel spectograms as features, makes a resizing so that the height of each sample is equal to the average of lengths of all samples and uses 4 convolutional layers with kernels 5×5, stride=1, padding=2 and maxpooling=2. The output channels (i.e. the third dimension), for the first layer are 32 and for each of the next layers multiplied by an increasing force of 2 (e.g. for the second layer it will be 32×2^1, for the third layer 32×2^2 etc). Finally, there are 3 linear layers, with the first having an output dimension of 1024, the second 256 and the third equal to the number of classes.

The following figure shows in detail the architecture of this cnn up to the second linear layer. The third which gives the output a vector of dimensions as many as the classes, is not displayed as it is different for each classification task (specifically it has dimension 4 for emotions and 3 for arousal and valence):

Classification Task	Merged Dataset			
	Train/Val Xgboost	Train/Val CNN	Train/Val SVM	Test SVM
Emotion	60.4	60.5	64.9	**62.8**
Valence	51.7	52.6	56	**49.6**
Arousal	64.2	69.3	68	**56.7**

Table 4.5: Merged-dataset Evaluation

From the above results it seems that svm is doing better than xgboost during validation.

CNN is only doing a little better on arousal. But this difference is negligible as we are talking about 1%, for this reason we will keep the svm model because it is less costly in time but also in memory and because this dataset may have contained enough data for cnn training. but in future uses there may not be as much data and therefore svm is a more general and secure option. Compared to the table experiments 4.3, which, as we mentioned, involve training in individual datasets and are very specialized, the performance here (from the train/val svm column) seems to be very close to the average (column average). This is very encouraging, as even though we involve many different domains through different datasets, they manage to do almost as well as the single domain models. Finally, it is important to analyze the results of the "test svm" column. The test set was created by speakers that are not included in the train set. Specifically, 1 to 2 speakers were selected from each dataset and the samples related to these speakers were all collected in the test set. In fact, the choice of speakers was made with a gender balance (4 men and 4 women). In this way, the performance that the model gives to the test set is more realistic as it concerns speakers that the model has never come across during the training. This is why the performance appears lower than the Train/Val SVM column (2-10% deviation), as the problem of speech emotion recognition is also dependent on the speaker.

For all the above reasons, the models that are finally selected to use in our system are the svm rbf from the merged dataset we mentioned in the last table.

Below are the results of the svm test in graphs.

Emotions

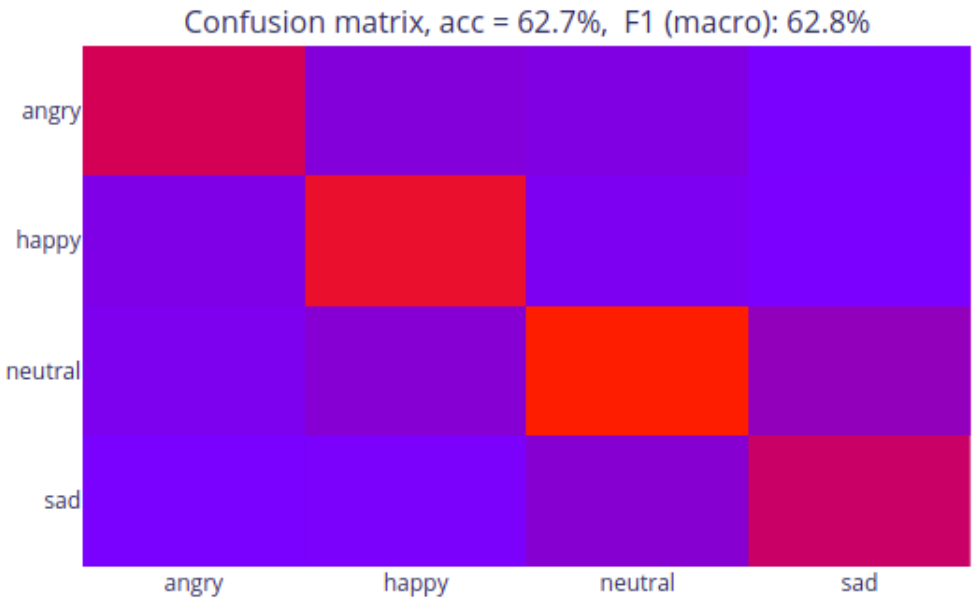


Figure 4.15: Confusion Matrix of Emotions

As we see from the confusion matrix, most of the samples have been sorted in the diagonal since it has more intense and bright colors. The diagonal is made up of samples that have been sorted correctly so this result is quite encouraging.

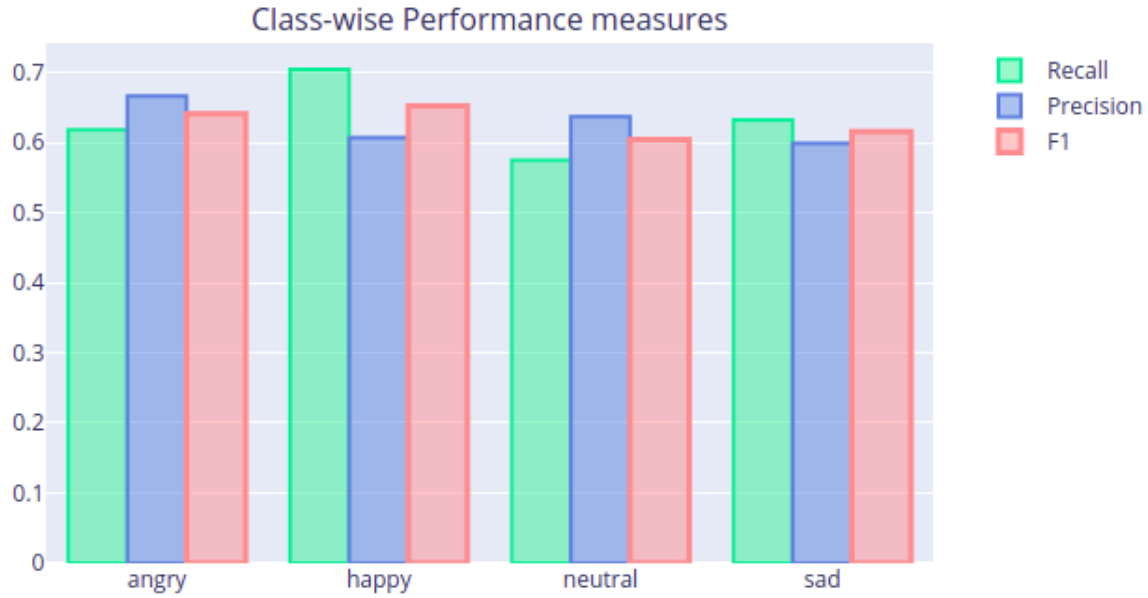


Figure 4.16: Performance Metrics per Emotion Class

From the above figure we can now clearly extract the performance measures per class and we can observe that the performance is almost the same for all classes, with small differences. The most striking differences we can observe is that the happy class surpasses all other classes in terms of metric recall, which means that it has the more correctly sorted samples, while the neutral class is the lowest in terms of recall and f1, without of course these differences being large. The last observation was expected for the neutral class, as it is the most neutral class, so some of the samples of the other classes may be incorrectly classified in it. In the above graphs we can see the performance of the model for each class separately, if different probability thresholds were applied. This threshold practically determines the value of the probability above which the sample will be classified as positive, i.e. in the respective class.

As for the graphic precision vs recall, if we wanted to have both good precision and good recall, then we would choose the threshold in which the two curves intersect.

As for the graphical ROC, we can see for sure that all the classes are doing much better than they would in a random classifier, as it is above the line x = y, which represents this classifier. We can also compare the classes with each other, as when the curve is higher than another, i.e. the area below this curve is larger, it means that it has higher true positive rate and therefore has better results and will always do better whatever threshold we apply. In this case, it seems that all the classes are almost equivalent, with the neutral class perhaps having a slightly worse performance, since its curve is presented a little lower than the others.

Valence

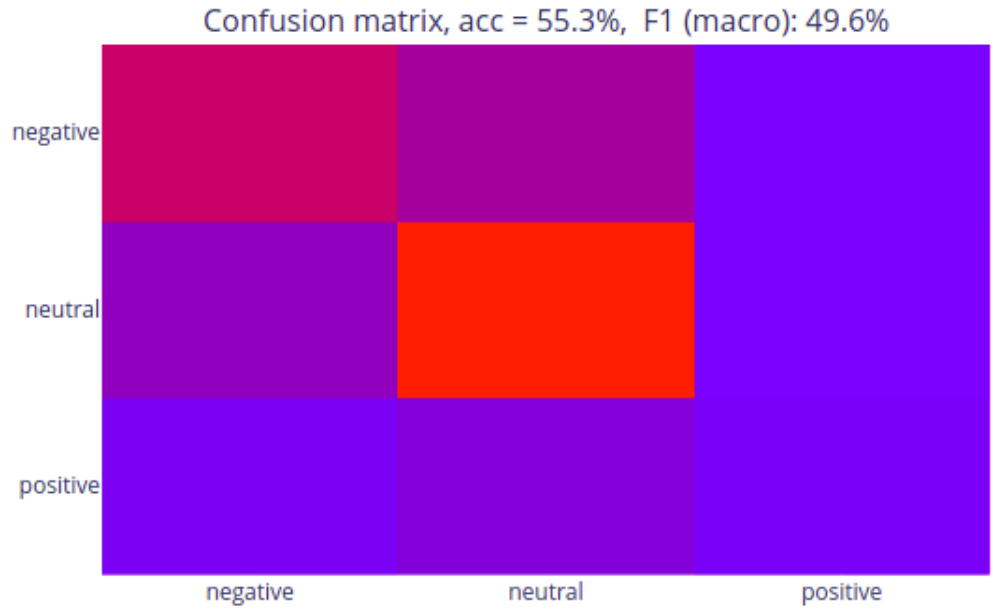


Figure 4.17: Confusion Matrix of Valence

From this confuson matrix we can assume that the positive class is not doing so well, as there are not many samples that have been classified as true positive in this class. On the contrary, most of the samples in this class seem to be classified as neutral.

Figure 4.18: Performance Metrics per Strength Class

What we observed above with the confusion matrix is confirmed by the performance metrics shown here per class. We see that the positive class has a fairly low recall, while showing greater precision. This means that the model classified several samples of the positive class into other classes, but did better in the samples it classified in this class. Therefore, it is more difficult to classify samples of another class in the positive class than vice versa. Finally, we observe that the neutral class has done better than the other two, with perhaps a slightly lower precision than the negative.

One reason why the positive class does not seem to be doing well is the fact that the dataset is imbalanced, since the samples belonging to the negative class were 4746 in number, the samples belonging to the neutral class were 4695 and the samples belonging to the positive class were 2189. This means that approximately 41% of the train set were samples of the class negative, 40% of the neutral class and 19% of the positive class. Therefore, the positive class was at a disadvantage, as it had much fewer samples than the other two and this, in turn, made it difficult for the model to better understand its characteristics and consequently to classify it correctly.

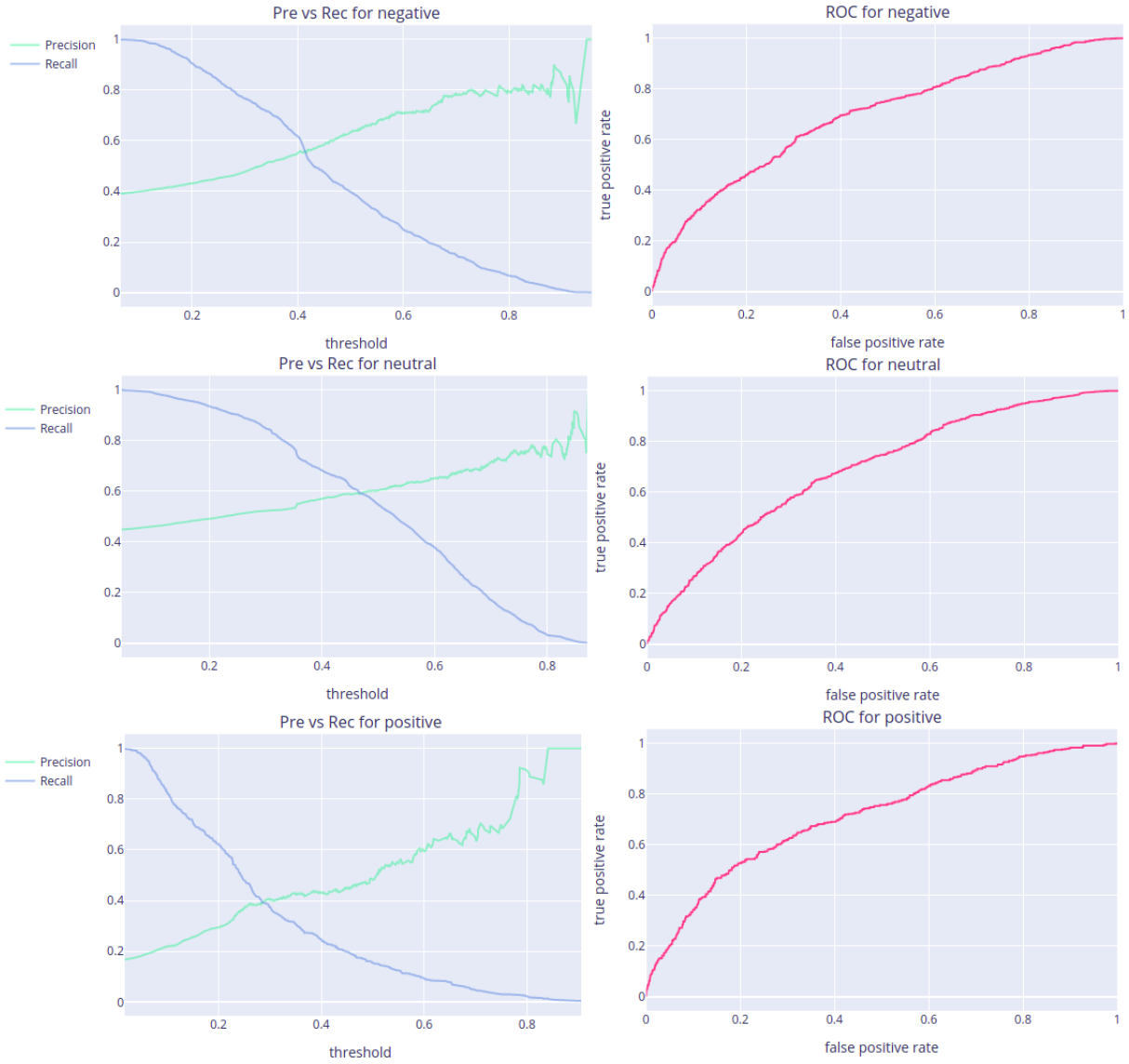


Figure 4.19: Precision vs Recall and ROC curves

From the ROC curves, we conclude that the three classes have almost equivalent performance on all thresholds.

Arousal

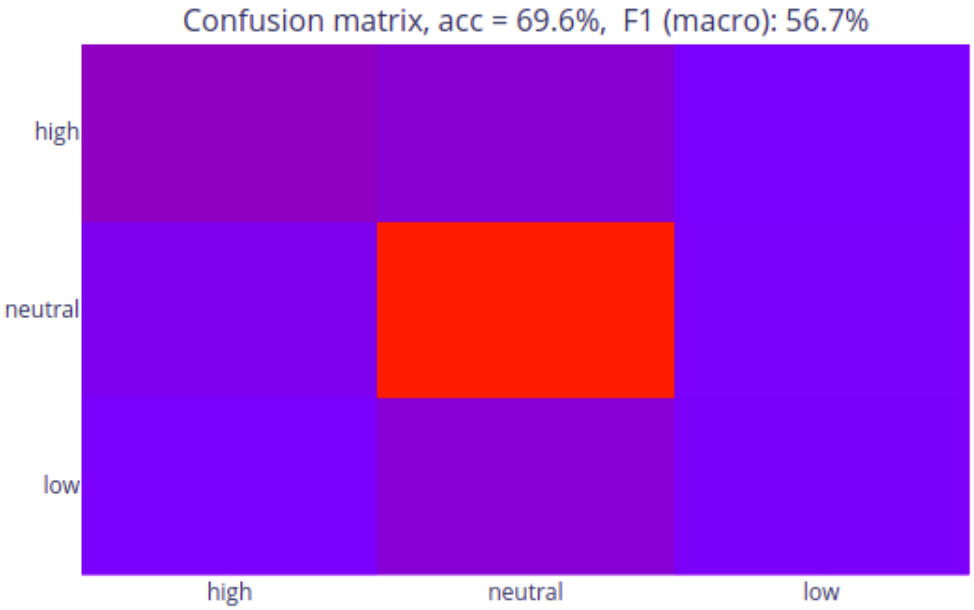


Figure 4.20: Confusion Matrix of Arousal

Here we notice that the class that is doing best is again neutral, the next one is high, while the class low is doing the worst of all. For the high and low classes it seems that many samples are wrongly classified as neutral, which is to be expected, as we have mentioned before, the neutral class is neutral and can easily "mislead".

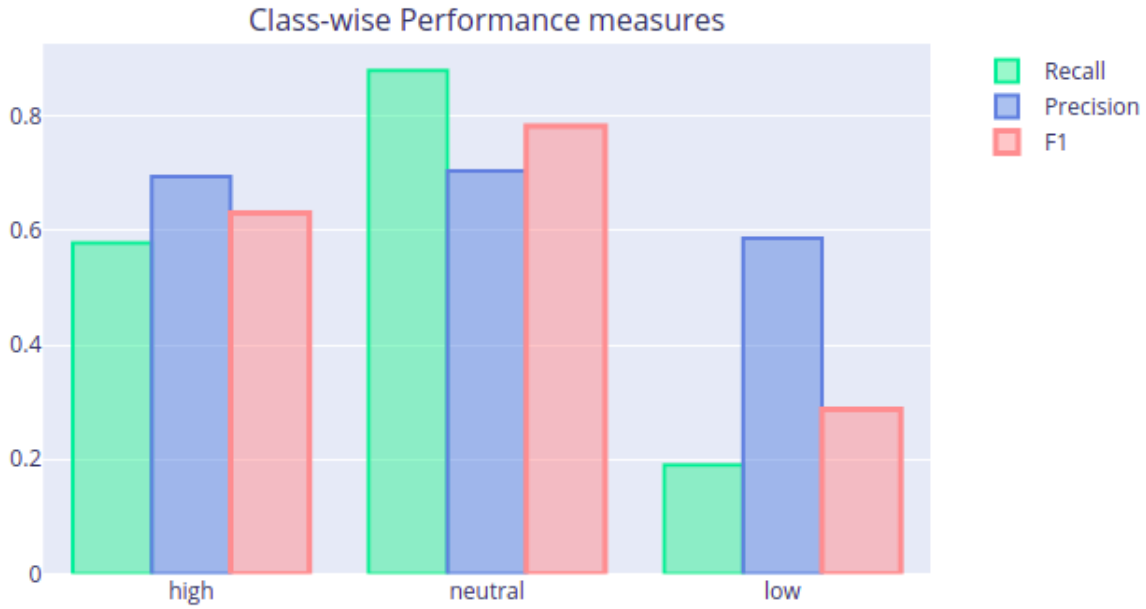


Figure 4.21: Performance Metrics per Arasousal Class

From the performance metrics per class, we observe the low class has a very bad recall, which means that very few of the samples in this class were predicted correctly, while the precision is much higher indicating that most of the samples predicted in this class, are correctly classified.

In this case, as in valence, the dataset is imbalanced and for this reason the low class

seems to be doing much worse than the other two. More specifically, the high class contained 3690 samples, the neutral class 6413 and the low class 1527. That is, approximately 32% of the train set were samples of the high class, 55% of the neutral class and 13% of the low class .

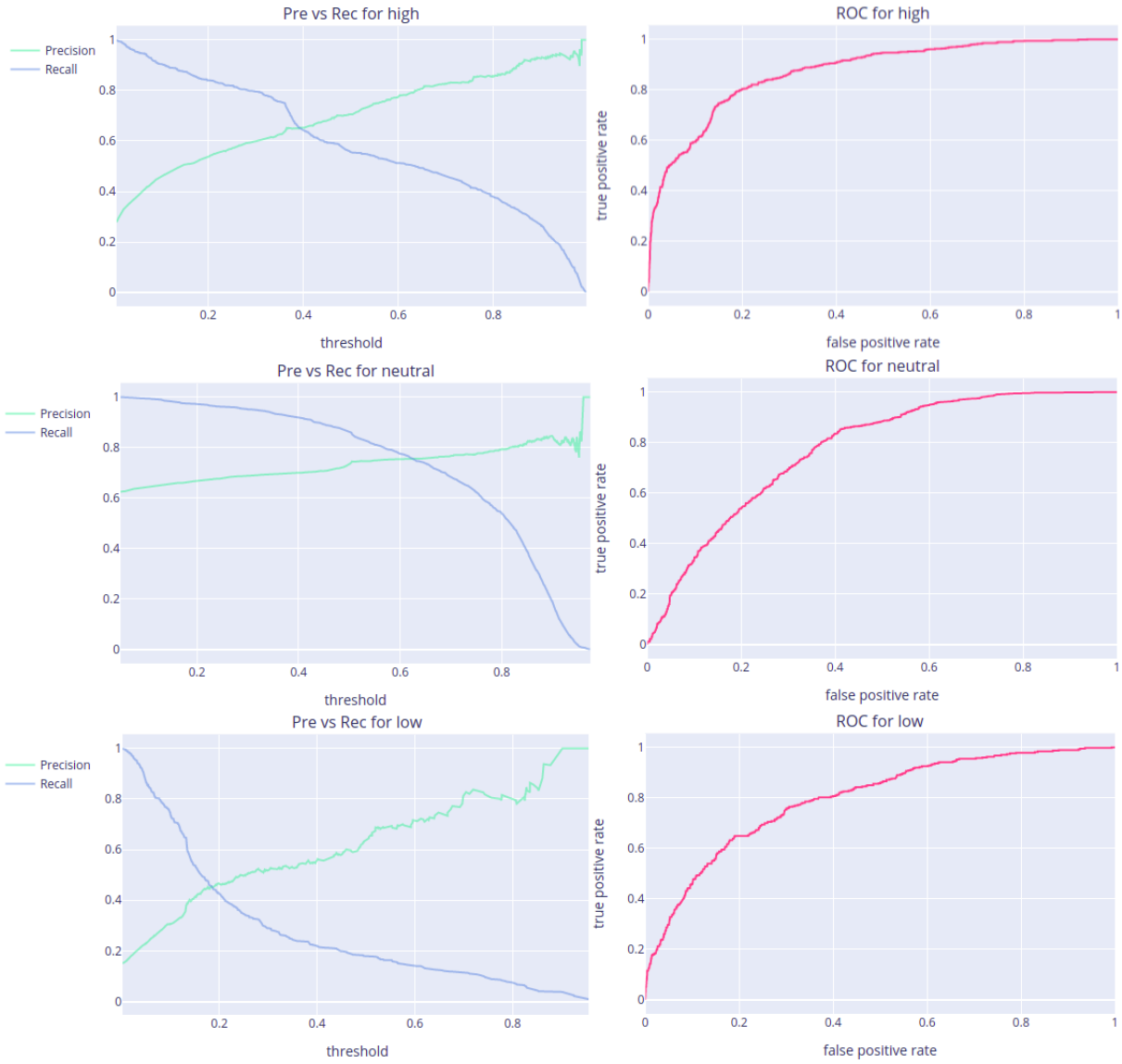


Figure 4.22: Precision vs Recall and ROC curves

From the above ROC curves we can conclude that all three classes are almost equivalent, with the neutral class going a little worse in lower thresholds, compared to the other classes, since it has a lower true positive rate.

In order to increase the performance of the above three models that will eventually be used by the system, but also to manage the imbalance of datasets, some additional methods were applied. Specifically, svm models were re-trained with kernel rbf again in the merged dataset.

This time a pipeline was applied, which uses the following steps:

- The sampler SMOTETomek [82], which is applied only in case the dataset is imbalanced, doing upsampling or downsampling as appropriate, thus eliminating the imbalance of

the dataset. The imbalance of the dataset is checked as follows: an equal proportion of samples per class is initially calculated. That is, if we have 2 classes the ideal ratio would be 50% of the samples belonged to the first class and the remaining 50% to the second. If we have 3 classes the ideal ratio would be a $\tilde{3}3\%$ of the samples belonged to the three classes respectively. Consequently, the actual ratio is then calculated by dividing the number of samples in each class by the total number of samples and if this ratio is less than 80% of the ideal for any class, then the dataset is considered imbalanced.

- StandardScaler [83] for attribute normalization, which normalizes attributes by subtracting the mean value and dividing by the standard deviation.
- VarianceThreshold [84] with value threshold = 0, which practically removes all features that have zero variance, i.e. those that have the same value in all samples and therefore do not offer any information.
- The PCA [85], which is responsible for reducing the linear dimension using Singular Value Decomposition of the data to display it in a lower dimension space.

During the training, a gridsearch was applied which uses RepeatedStratifiedKFold cross validation [86] and more specifically, it uses 5 folds, i.e. each time it splits the dataset into 5 parts and repeats the cross validation 3 times, with different randomization of folds , in each iteration. In this gridsearch hyperparameter tuning is performed in three hyperparameters: the number of components of the pca that are maintained, the hyperparameter γ and the hyperparameter C of the SVM. The following values are considered for the last two parameters: 'gamma': ['auto', 'scale'], 'C': [0.001, 0.01, 0.5, 1.0, 5.0, 10.0, 20.0]. Finally, the parameters that give the best average performance are selected, from the average performance of the repetitions. The performance metric that has been selected is again the f1 macro.

The values of the hyperparameters given by the best models for each classification task are as follows:

- **Emotion**:
  Parameters of best svm model: 'SVM C': 5.0, 'SVM gamma': 'auto', 'pca n_components': 'mle'
- **Valence**:
  Parameters of best svm model: 'SVM C': 1.0, 'SVM gamma': 'scale', 'pca n_components': 'mle'
- **Arousal**:
  Parameters of best svm model: 'SVM C': 1.0, 'SVM gamma': 'scale', 'pca n_components': 'mle'

The following table presents the new results, for the best models that resulted from the procedure described above, in the merged dataset, both in the validation during training (training set with cross validation), and in the evaluation during testing (test set) :

Classification Task	Merged Dataset	
	Train/Val SVM	Test SVM
Emotion	64.4 (+/-0.9)	**62.9**
Valence	55.2 (+/-1.2)	**51.8**
Arousal	66.8 (+/-1.1)	**60**

Table 4.6: Merged dataset - Performance Improvement

If we compare the results in the test set of the above table, with those of the table 4.5, we will see that all classification tasks have been improved, with emotion showing the slightest improvement, as expected, as this dataset did not show previously the problem of imbalance, like the other two.

The following shows in more detail the confusion matrices as well as the performance metrics per class, for each classification task:

Emotions

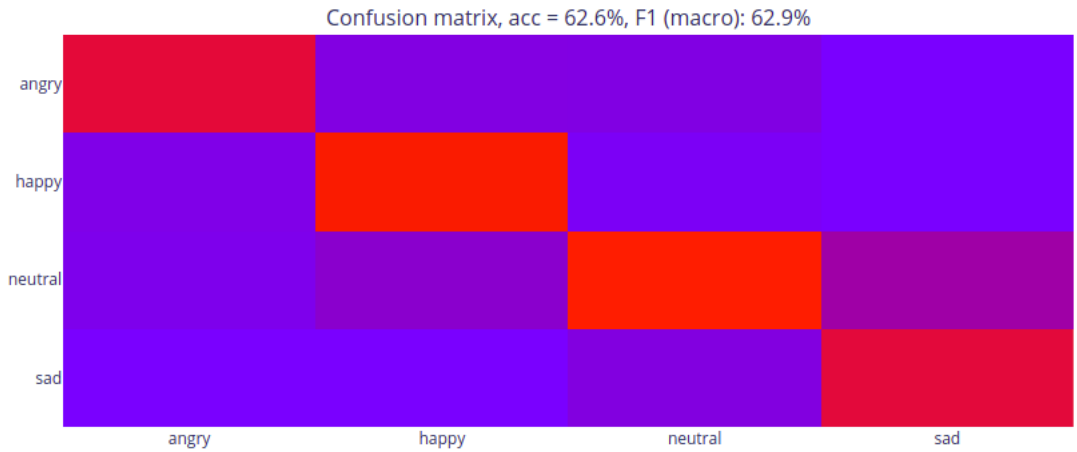


Figure 4.23: Confusion Matrix of Emotions

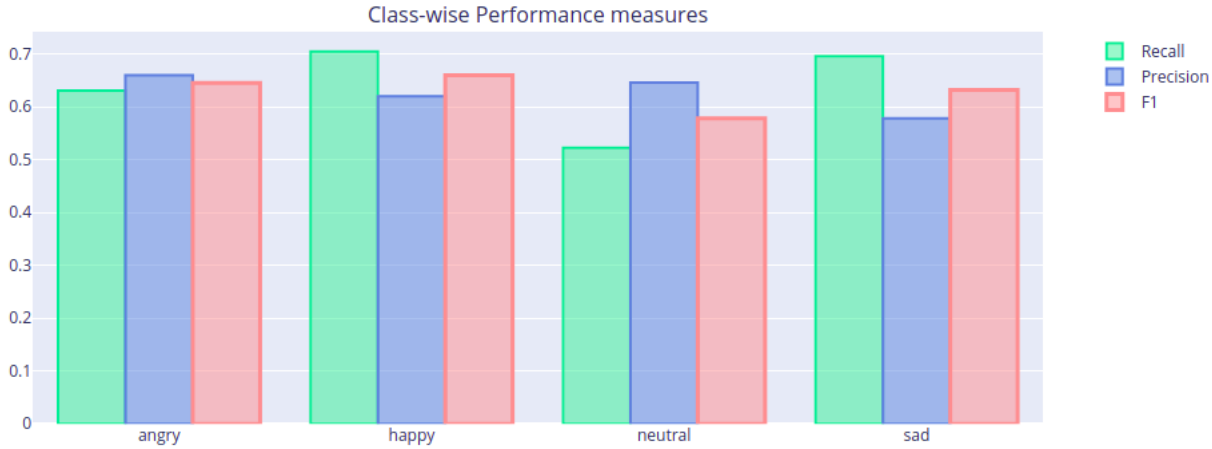


Figure 4.24: Performance Metrics per Emotion Class

If we compare the above graphics, with the graphics 4.15 and 4.16, we observe an

improvement across the diagonal of the confusion matrix, as it is presented in a lighter red color compared to before, which means that more samples have been classified as true positives in each class than before. This improvement seems more pronounced in the sad class. Also, observing the performance metrics per class, we observe a slight improvement in all recalls except the neutral class where the recall suffered some reduction and a slight improvement in all precision, except in the sad class where the precision suffered a slight decrease in relation to before. F1 has more or less remained the same, except for the sad class where f1 was increased.

Valence

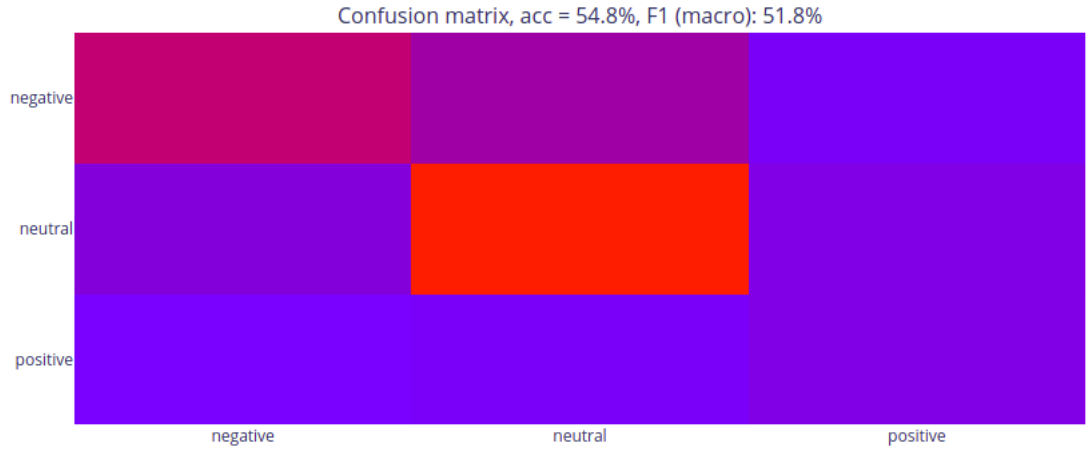


Figure 4.25: Confusion Matrix of Valence

Figure 4.26: Performance Metrics per Strength Class

If we compare the above graphics, with the graphics 4.17 and 4.18, we see that the recalls of the negative and neutral classes have decreased compared to before, but the precision has increased while for the positive class which was also the one with the fewest samples we see a noticeable improvement where the recall has risen by 20% with the precision having fallen slightly. The f1s have again remained more or less the same, except for the f1 of the positive class which has been improved.

Arousal

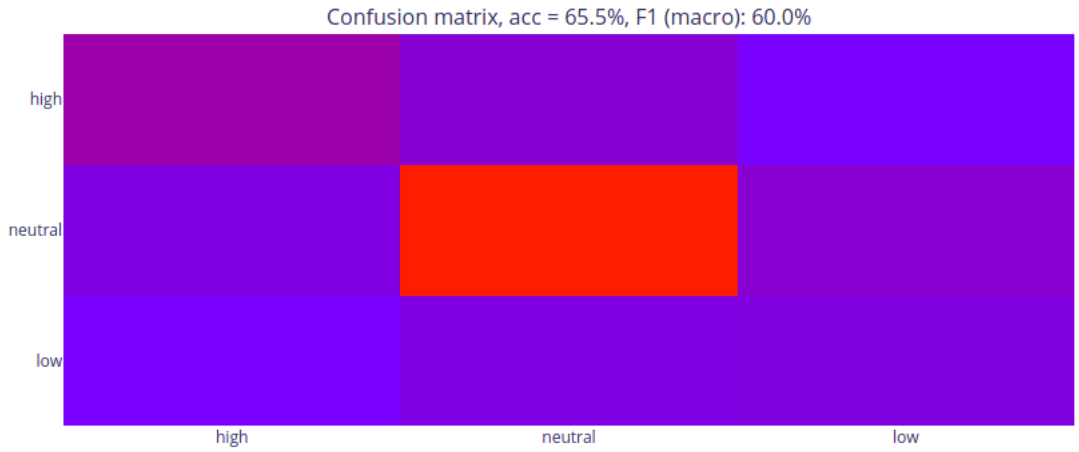


Figure 4.27: Confusion Matrix of Arousal

Figure 4.28: Performance Metrics per Stimulus Class

If we compare the above graphics, with the graphics 4.20 and 4.21, we observe that the recall has increased for both the high class and the low class, in which in fact we notice a huge difference since from 19% it went up to 46%. This shows that the problem of dataset imbalance, where the low class was the one with the fewest samples, was solved to some extent. In the neutral class, the recall fell again but is maintained at a fairly satisfactory price. As for the precision, we observe that it has risen in high class and neutral class and has fallen to the low class. The f1 has been improved for the high and low classes, with the low classes showing a large improvement, while it has decreased slightly for the neutral class.

As a whole, the models seem to have improved compared to before and mainly addressed the problem of datasets imbalance. Therefore, these are the models that will ultimately frame our system, in terms of audio segment classifiers.

4.2 Text Analysis

As we mentioned in the chapter 3.2, the text of speech is derived from sound by making a speech to text conversion.

This subchapter will present the text splitting techniques in sections, the set of textual features used for section classifiers, the set of data used for classifier training, and the experiments performed. Specifically, the following part of the system architecture will be analyzed:

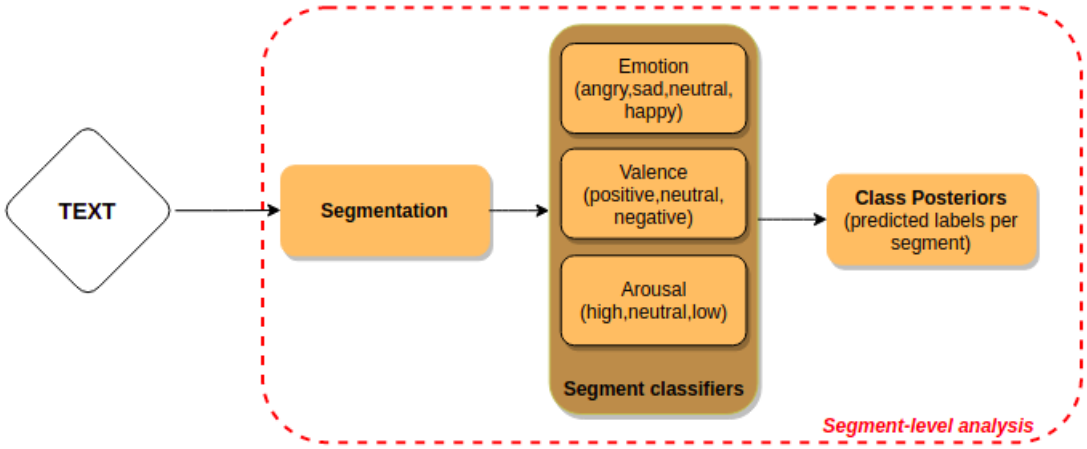


Figure 4.29: Segment Level Text Analysis

4.2.1 Text Segmentation

As in audio, so in text it is important to break down information into segments. The reason for this split is because the textual information, which has come from the audio signal, is not static over time, i.e. its properties change. For example, a speaker's emotion may change from sentence to sentence and from minute to minute as he speaks. The same goes for the volume, the tone of voice, the rhythm of speech, etc. So it would be incomplete and discriminatory to characterize the whole text with a single label of emotion, volume and so on.

The splitting of the text into sections can be achieved through the system in three ways:

1. **Split by sentence**
   In this case each sentence is a separate sample for the classifier and is characterized separately.

2. **Split into fixed size sections**
   In this case a threshold for the number of words per section is set. For example, text can be split into 3-word sections.

3. **Split in fixed time windows**
   In this case a threshold of seconds is set which is the duration of each section / window. For example, text can be split into 3-second segments. So in each section the words will be contained that were said within the 3 seconds each time.

There is a choice of one of three splitting modes. For the department classifier models trained in the present dissertation, the first split method (per sentence) was used. This method of splitting is safer, as it is more likely that any change in speech characteristics will

occur at the sentence level. A smaller cleavage may be incomplete to identify specific patterns / features, while a larger cleavage could contain more than one attribute with the risk of missing some of them. Of course, this does not mean that the other two ways of splitting are not beneficial. In any case, the cleavage threshold in these modes can be investigated, depends on the process and must be chosen wisely, after testing.

4.2.2 Text Feature Extraction

In word processing models, the words that make up the text or the characters that make up the words do not show any direct numerical correspondence. In **Natural Language Processing (NLP)**, **Word Embedding** is a term used to represent words in text analysis, usually in the form of a real value vector that encodes the meaning of the word so that the words closest to the vector space are expected to have a similar meaning. Word embeddings can be obtained using a set of language modeling and feature learning, where words or phrases from the vocabulary are mapped to real number vectors. Conceptually it includes a mathematical integration from a space with many dimensions per word to a continuous vector space with a much lower dimension [87].

The incorporation of words and phrases, when used as input representations, has been shown to enhance performance in natural language processing tasks, such as syntactic analysis and emotion analysis.

In the present dissertation, the classifiers used in the text part are emotion analyzers, as in the case of the audio information we saw above. Thus, word embedding is used as input text representation (as text features) of the models.

There are several methodologies for calculating word vectors (text embedding). Below are the most well-known of these, as well as the one that is ultimately used by the system.

4.2.2.1 Bag of Words

The method **Bag of Words (BoW)**, is the simplest technique of converting text to vector. With this method words, phrases, sentences and texts are represented with sparse long vectors in a high dimensional vector space. As the name implies, the piece of text is treated like a bag of words. This practically means that we ignore the order of the words and are only interested in the presence or frequency of words in the text.

Given a dictionary of V=$[v_1, v_2, ..., v_n]$ consisting of n words, BoW breaks down the words of a text K consisting of k<n words into individual word count statistics. That is, it constructs a vector w=$[w_1, w_2, ..., w_n]$ of length n, where each element w_i of the vector expresses how many times the word v_i is in the text. For example, if we have the following dictionary: V=['you', 'my', 'great', 'very'] and the text K="you are very very beatiful", then the generated vector will be: w = [1,0,0,2]. This representation is called **Term Frequency**.

Another example of a BoW representation with term frequency is shown schematically below [88]:

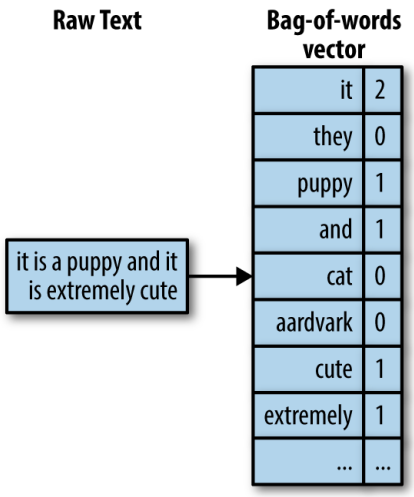


Figure 4.30: Convert raw text to word bag representation

Another type of BoW representation is the representation of words with one-hot vectors textbf one-hot vectors which is essentially a vector with only one element being 1 and all others being 0. The length of the vector is equal to the size of the total unique dictionary. Conventionally, these unique words are coded alphabetically. That is, we should expect the one-hot vectors for words starting with "a" to have a value of "1" at a lower index, while for words starting with "z" to have a value of "1" at a higher index.

An example of a one-hot word representation is shown in the following figure [89]:

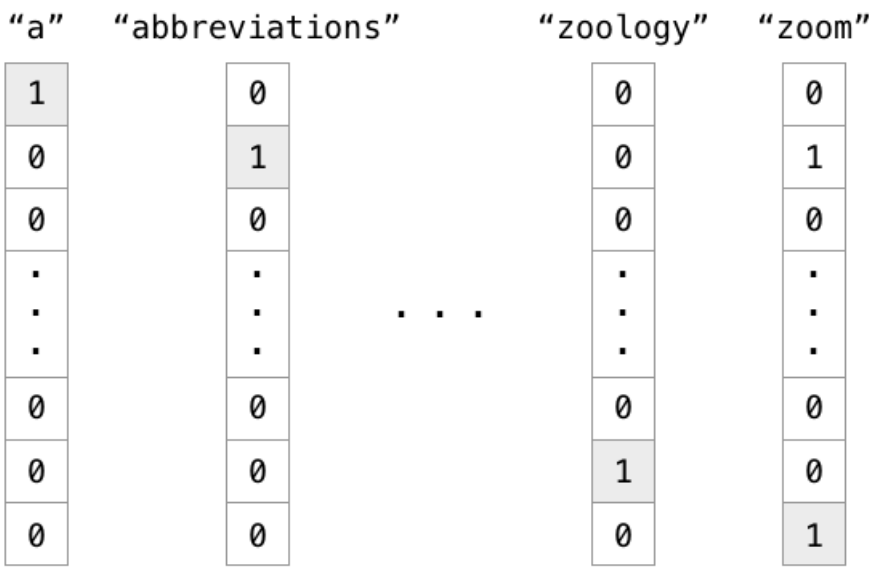


Figure 4.31: One-hot word vectors

In the same way, if we wanted to represent with a vector, the whole document and not just individual words, then the vector would have a value of 1 in the positions corresponding to the dictionary words, which appear inside the document. This representation is called **Term Occurrence**.

BoW representations are often used in document classification methods where the frequency of each word is a useful feature for classifier training. One challenge with BoW word representation is that they do not encode information about the meaning of a given

word, as well as information about the syntax / sequence of words in a sentence or text. It is well known, however, that two sentences that can have exactly the same words in different order can give a completely different meaning. This representation, however, will give two identical vectors for these two sentences.

At BoW, word impressions are evenly weighted. However, in most NLP tasks some words are more relevant than others.

4.2.2.2 Bag of n-grams

The Bag of n-grams method is an extension of the BoW method, which addresses in part the problem of structural structure. N-gram is just any sequence of n words. That is, instead of splitting the text into a bag of words, we split it into a bag of word sequences. For example, the sentence "You are very very beatiful" that we mentioned before could be broken down into:

- **1-grams**: 'You', 'are', 'very','very','beautiful'
- **2-grams**: 'You are', 'are very', 'very very', 'very beautiful'
- **3-grams**: 'You are very', 'are very very', 'very very beautiful'
- **etcetera**

The n-grams bag may be more informative than the bag of words we saw before, because it captures more content around each word (i.e. "I love this dress" is more informative than "dress"), that is, it also utilizes the context, or otherwise gives importance to the syntax. However, this also comes at a cost, as the n-grams bag can produce a much larger and thinner set of features than the word bag. Usually, 3-grams is about as high as we want, as using higher n-grams rarely increases efficiency due to thinness.

4.2.2.3 TF-IDF

The **TF-IDF (Term Frequency - Inverse Document Frequency)** methodology is a statistical process that calculates the importance of a word or an n-gram in a text from a set of texts. This representation consists of two quantities: the **Term Frequency** and the **Inverse Document Frequency**. The first value is the number of times the word appears in the document, while the second value is the frequency of the word in all documents and is calculated as the logarithm of the ratio of the total number of documents to the number of documents in which the word.

Its mathematical representation is shown below:

$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_i} \tag{4.28}$$

where $\boldsymbol{tf_{i,j}}$, is the number of times the word i appears in the document j, $\boldsymbol{df_i}$ is the number of documents in which the word i appears and **N** is the total number of documents.

The above mathematical expression when it gives values close to 0 means that the importance of the word in the text is minor, while when it gives values close to 1 it indicates great importance

The main advantage of the tf-idf method is that it examines the importance of words vs a large set of documents. For example, words that often appear in a text such as 'is', 'the' are of little importance to the semantics of the document. This algorithm manages to indicate this, since these words are common to all documents and therefore the ratio $\frac{N}{df_i}$

representing the reverse frequency of documents (idf) will be low, thus reducing the final value of w (tf-idf). Instead, the appearance of a word that is directly related to the subject of a particular document will be common in this document but not in the rest, thus giving a high value to idf and a corresponding high value to the final metric tf-idf.

However, although tf-idf and BoW representations provide weights in different words, they are not able to represent the meaning of the word and take the context into account. Another problem with tf-idf is that this methodology requires a large number of documents to be effective.

As the famous linguist J. R. Firth said in 1935, "The full meaning of a word is always contextual and the study of meaning outside of context / context cannot be taken seriously."

4.2.2.4 Lexical Co-occurrence Matrix

The **co-occurrence matrix** for a given body of text (corpus), is a table that shows, in pairs of words, the number of times these words coexist / appear together in the text. Mathematically expressed by the relation C = $\{c_{ij}\}$ with $c_{ij} = f(w_i, w_j, S)$, where $f(w_i, w_j, S)$, where $f(w_i, w_j, S)$ is the frequency of occurrence of the $w_i w_j$ ή $w_j w_i$ sequence throughout the text body.

For example if we have the body /corpus = **He is not very pretty. He is happy. He is intelligent**, then the coexistence table, ignoring the dots would be as follows:

	He	is	not	very	pretty	happy	intelligent
He	0	3	0	0	1	1	0
is	3	0	1	0	0	1	1
not	0	1	0	1	0	0	0
very	0	0	1	0	1	0	0
pretty	1	0	0	1	0	0	0
happy	1	1	0	0	0	0	0
intelligent	0	1	0	0	0	0	0

Table 4.7: Coexistence Table Example

But this coexistence table is not the vector representation of the word generally used. Instead, this coexistence table is decomposed using techniques such as PCA, SVD, etc. in factors and the combination of these factors forms the word representation of the word (word vector representation).

With the SVD (Singular Value Decomposition) method the above sparse table can be factorized as follows [92]:

$$\underset{|V|}{\overset{|V|}{\begin{bmatrix} \\ X \\ \\ \end{bmatrix}}} = \underset{|V|}{\overset{|V|}{\begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix}}} \underset{|V|}{\overset{|V|}{\begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}}} \underset{|V|}{\overset{|V|}{\begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix}}}$$

Figure 4.32: SVD application on a square coexistence table X

where $\boldsymbol{\Sigma}$ diagonal array, $\boldsymbol{\sigma_i}$ are called singular values, $\boldsymbol{u_i}$ left-singular vectors and $\boldsymbol{v_i}$ right-singular vectors. By selecting the k largest singular values and the singular vectors left and right corresponding to them, we obtain the degree k approximation of table X with the smallest error according to Frobenius norm. This practically means that if the original array is a square V × V, that is, it contains V columns of V dimensions or V rows of V dimensions, selecting the k largest singular values and the corresponding k vectors u and

k vectors v, the k vectors u give k -dimensional approach of V rows and k vectors v the k-dimensional approach of V columns. SVD factorization is therefore a method of reducing dimensionality.

The effect of reducing the dimensionality by selecting the first k special values is shown below [92]:

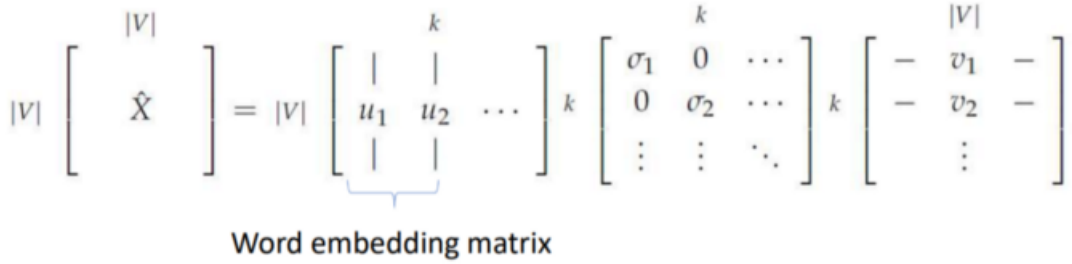


Figure 4.33: Dimension Reduction with SVD

Where now the dimension of X is |V|*|V|, the dimension of U is |V|*k, the dimension of Σ is k*k and the dimension of V^T is k*|V|.

Therefore, in our example where the table C is 7x7, we can take k-dimensional (k <7) approaches to the columns or rows, i.e. the words, and end up with k-dimensional word vectors (e.g. for k=2 or k=3). Thus, a single word, instead of being represented in dimensions V, will be represented in dimensions k, while capturing almost the same semantic meaning. K is generally in the order of hundreds. What also matters is that the inner product of U and Σ gives the vector representation of the word **word vector representation** and V gives the environment representation of the word **word context representation**. Reducing the dimentionality of the co-occurrence table is a simple method for extracting vector word representations. However, it can also be extended to document representations. In this case the co-occurrence table is non-square and each line refers to a word or generally a term (n-gram) while each column is a document. Each element of the table refers to the coexistence of terms with documents, i.e. each column is a term occurrence, term frequency or tf-idf representation. Therefore with SVD factorization we get k-dimensional approaches to terms and documents. This method of global matrix factorizations, when applied to term frequency matrices, is more commonly known as **Latent Semantic Analysis (LSA)**.

4.2.2.5 Word2Vec

The methodologies presented in this subchapter and in the next two subchapters for the calculation of word vectors are **neural embeddings** or otherwise **neural language models**,i.e. they are based on neural networks /models. Neural prediction models learn to predict their vectors by improving the prediction loss, such as the loss of predicting a vector of a target word from the vectors of the surrounding words.

Word2Vec is one such predictive embedding model. Developed by Tomas Mikolov, et al. at Google in 2013.

The algorithm works with **context windows**. Basically, a central word is selected and its neighboring context words, left and right. The number of adjacent words is determined by the length of the window. Below is an example for context window = 2:

Quick	rabit	jump	over	the	turtle

Table 4.8: Example Context Window

Quick	rabit	jump	over	the	turtle

Table 4.9: Example Context Window

where in the above examples, the green words form the context window (2 around) around the central words that are colored orange.

There are two main Word2Vec architectures used to generate a distributed representation of words: CBOW and Skip-gram.

CBOW

CBOW (Continuous bag-of-words). Is based on the architecture of a simple neural network. The idea is that given a context, we want to know which word is most likely to appear in it. The neuron receives the words around the target word as input and predicts the target word at the output. For example, in the sentence "I have a cute dog", the input could be "I", "have", "cute", "dog" and the output "a". The input and output data are of the same dimension and encoded with one-hot as we mentioned in chapter 4.2.2.1. The grid contains 1 hidden layer whose dimension is equal to the embedding size, which is smaller than the size of the input / output vector.

The following figure shows the architecture of this neuron for a context word and a target word [90]:

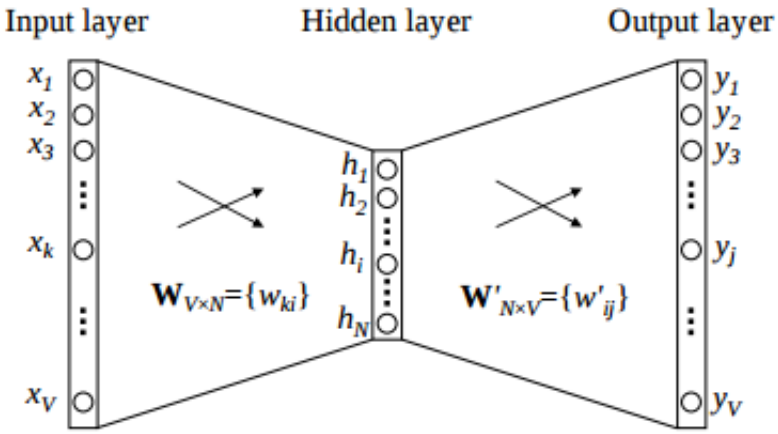


Figure 4.34: CBOW Architecture for a Context Word

From the above form of the architecture we can point out the following:

1. It is In the input layer the context word , while in the output layer it is the target word that the neuron seeks to predict, given the context word. Both words are coded with one-hot encoding, i.e. they have 1 in one position and 0 in all the others. The dimension of the input and output vector is V [1 X V] as much as the words in the dictionary, which is constructed from the body of the text (context). For example, for the sentence "I have a cute dog", V is equal to 5.

2. There are two groups of weights. One is between the input layer and the hidden layer and the second between the hidden layer and the output layer. Input-Hidden layer matrix size = [V X N], hidden-Output layer matrix size = [N X V]: where N is the dimension we choose to represent the word. It is a hyperparameter of the nervous system, since it is also the number of neurons in the hidden layer.

3. The input is multiplied by the weights between the input-hidden layers. In essence this product gives the k series of weights $W_{V\times N}$, where k is the position at which the input vector x has 1, i.e. $x_k = 1$. From this multiplication results the hidden vector h of dimension N.

4. Then the hidden vector is then multiplied by the second set of weights between the hidden-output layers ($W'_{N\times V}$). From this product results a length V vector to which a softmax activation function is applied so that all values add up to the value 1. After applying the activation function, the output vector y is obtained. In this way, each value y_j of the output vector y, is a probability that describes how likely the vocabulary word j is to be the target word we are looking for.

5. The optimization criterion used in this neural is to maximize the probability of observing the actual target word which, as we have mentioned in theory, is equivalent to maximizing the probability logarithm. But because the cost function must be minimized, the same problem is defined as minimizing the negative probability logarithm or minimizing cross entropy. Therefore the cost function or objective function is defined as follows:

$$-\log(p(w_0 = w_{j^*}|w_I)) \tag{4.29}$$

   where the $p(w_0 = w_{j^*}|w_I)$ is defined as follows:

$$p(w_0 = w_{j^*}|w_I) = \frac{\exp(v'^{\top}_{w_{j^*}} \cdot v_{w_I})}{\sum_{j'=1}^{V} \exp(v'^{\top}_{w_{j'}} \cdot v_{w_I})} \tag{4.30}$$

   where w_0 is the output word, w_I is the input word (context word), j^* indicates the index of the word w_{j^*}, which is actually the target word in the dictionary, v is a part of the weights W and v' of the weights W', while v_w and v'_w, are the input and output vectors (input and output weights) respectively, which characterize the word w.

   The relation 4.30, is essentially the probability that the output word w_0 is equal to the target word w_{j^*}, given the context input word w_I. This possibility, as we mentioned in the previous step, has arisen after the application of the softmax activation function, which we mentioned in the chapter 2.3, hence the expression 4.30, has this form .

6. The error we came across in the previous step (cross entropy loss) is propagated backwards using stochastic gradient descent and backpropagation methods to update / reset all weights.

7. After the end of the neuronal training, the vector that finally gives us the representation of the target word, is the weight between the hidden layer and the output layer and has dimension N. That is it is $v'_{w_{j^*}}$ where j^* where j^* is the location of the target word.

In fact, the above example was a simplification of the architecture with a single input word. Given a body of text, the algorithm works in windows where each window contains a central target word and C words in total to the right and left of the central (context words). In the general case where we have more than one context word as input, the architecture is shown below [90]:

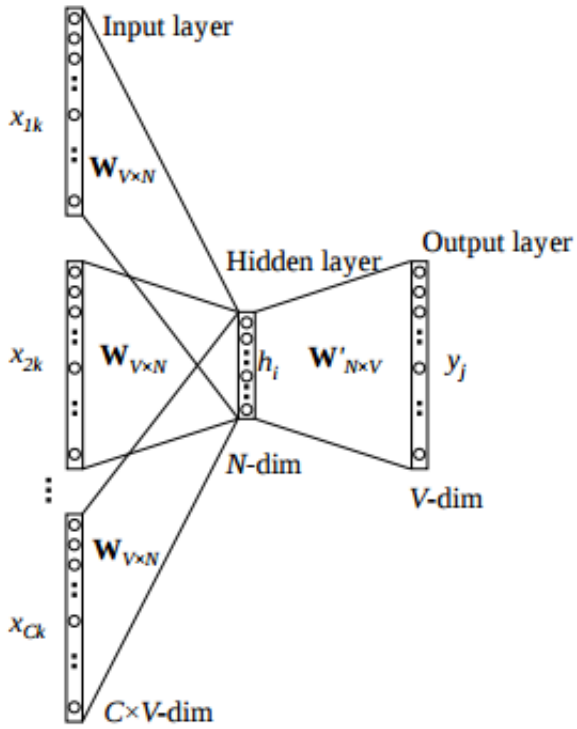


Figure 4.35: CBOW Architecture for Many Context Words

In the figure above we see that we now have C context input words, each one of which is dimension V. The steps remain the same as before, only step 3 changes which is the calculation of the hidden vector h. Instead of simply copying the corresponding rows of the input-hidden weight matrix ($W_{V\times N}$) to the hidden layer, an average of all the corresponding rows in the table is obtained. In essence, before we only had one input vector x which had a value of 1 in position k and so multiplication $x \times W_{V\times N}$ gave the array k of the table $W_{V\times N}$. Now, C such multiplications (as many as the vectors x) are performed, each of which gives a different order of the array. In the end, the average of these series is calculated based on the elements (element-wise) and a vector of h, N dimensions emerges again.

Skip gram

The continuous skip-gram is very similar to CBOW, except that it exchanges input and output. The skip-gram architecture is a neuron that accepts a target word as input and predicts the context words around the target word at the output. For example, in "I have a cute dog", the input could be "a" and the output "I", "have", "cute", "dog".

The following figure shows the architecture of this neural [90]:

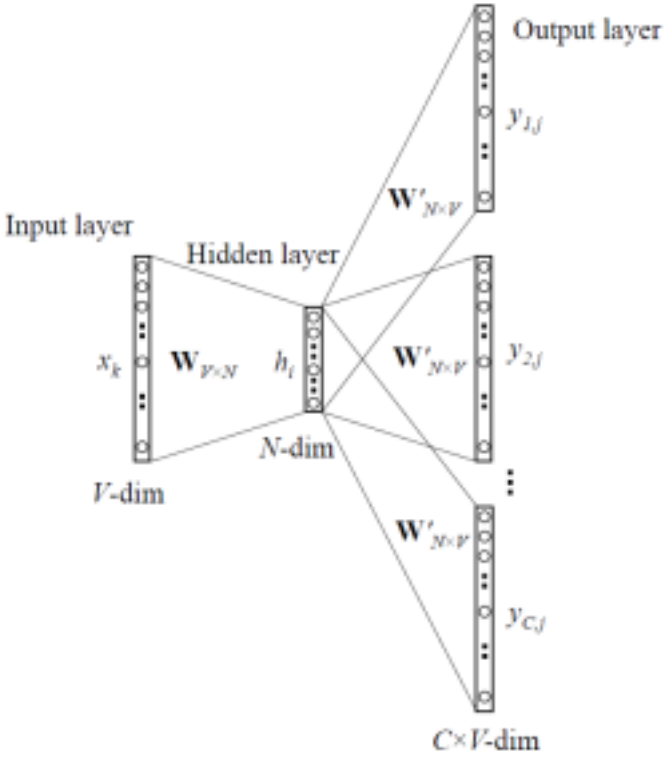


Figure 4.36: Skip-gram architecture

Here again the same steps as before are followed, from 1 to 3. After calculating the vector h, then in step 4, this vector is multiplied by all the hidden output weights $W'_{N\times V}$. From this multiplication C dimension vectors result in V. In each of these vectors the softmax function is applied again and from there C-in a number of output vectors y emerge. Thus $y_{c,j}$ is characterized by the probability that the context word c is the word j in the dictionary with c = 1,2, ..., C and j = 1,2, ..., V. The goal of skip-gram model training is now to maximize the likelihood that context words are the words actually are observed in the sample. Therefore, the cost function is defined here as follows:

$$-\log(P(w_{0,1}, w_{0,2}, ..., w_{0,C}|w_I)) = -\log(\prod_{c=1}^{C} P(w_{0,c}|w_I)) = -\sum_{c=1}^{C} \log(P(w_{0,c}|w_I)) \tag{4.31}$$

where in essence the total error is calculated by summing the individual errors for each context word. That is, calculate the relation 4.30, for each output word $w_{0,c}$ with c=1, ..., C, which is the probability of observing the actual word-context, and then add the negative logarithms of these probabilities. The neuronal weights are then redefined by stochastic gradient descent and backpropagation methods, as we mentioned before.

After training, the vector that finally gives us the representation of the target word, is the weight between the input layer and the hidden layer, which has dimension N. That $v_{w_{j^*}}$ όπου j^* is the location of the target word.

Both models focus on learning words given their local usage environment, where the environment is defined by a neighboring word window. This window is a configurable model parameter.

With word2vec, the representation dimension of a word is reduced by the size of the vocabulary (V) to the length of the hidden level (N). In addition, vectors have more "meaning"

in describing the relationship between words. Vectors obtained by subtracting two related words sometimes express an essential concept such as gender or verb, as shown in the following figure [91]:

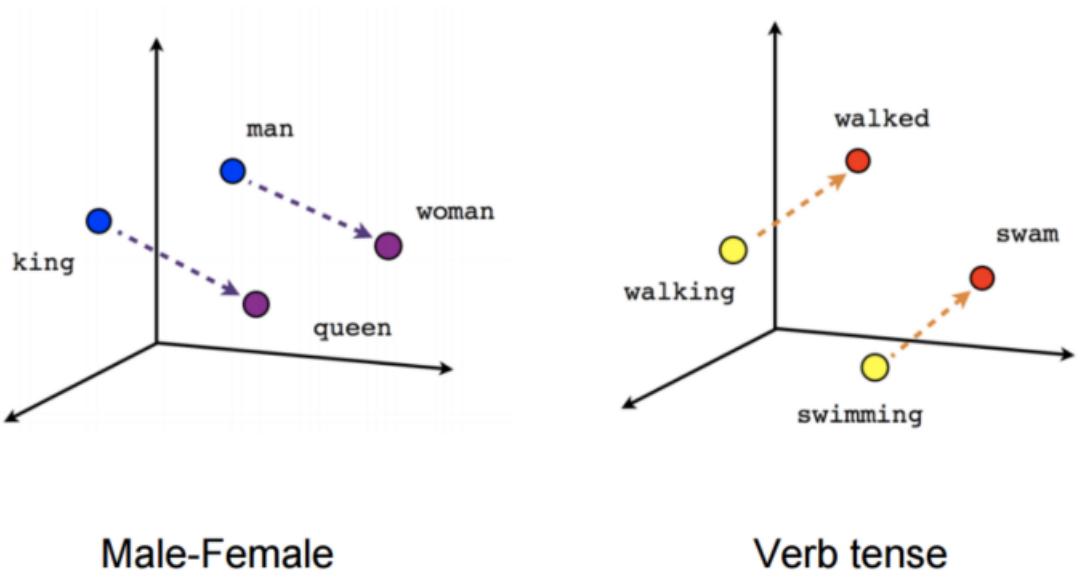


Figure 4.37: Word Vector Visualization

Thus, in this method we see that the contexts are taken into account, or in other words the syntax and the order of the words, something that did not happen in the previous methods we studied. Also, this method can give more than one representation / semantics in one word, since for each context a different representation vector is extracted. Therefore, the same word in different contexts can be represented differently. Finally, it is argued that CBOW is faster than skip-gram, but skip-gram tends to do better for rare words. Skip-gram with negative sampling generally outperforms any other method.

4.2.2.6 Glove

The **Global Vectors or GloVe** algorithm is an extension of the word2vec method for efficient learning of word vectors, developed by Pennington, et al. in Stanford. This model produces vector representations incorporating global information.

A disadvantage of the word2vec algorithm is the non-integration of information, in vector representations, for the total statistics of the text body. In the above chapters, among others, we studied two general ways of exporting vector representations, where the former exploits the overall statistics of the text body and usually involves methods of derivation of the co-occurrence table 4.2.2.4, while the latter exploits local information using text boxes 4.2.2.5.

Classic vector representations of vector space models were developed using vector factorization techniques such as Latent Semantic Analysis (LSA) seen in 4.2.2.4, which do an effective job of using global text statistics (such as co- occurrence table), but are not as good as trained methods, such as word2vec, in capturing semantics and demonstrating it in tasks such as calculating word ratios. Such works are for example the removal of the "male" element from the "king" and the addition of the "female" which leads to the word "queen", recording the ratio "the king is to the queen as the man is to the woman". Another such analogy may be between the tenses of a verb: for example "dance is to dancing as fly is to flying".

GloVe is an approach to combine both of the above: **global statistics** elements of table factorization techniques such as LSA, with local context-based learning **(context-based)** of word2vec .

The Glove algorithm works like this: instead of extracting the integrations from a neural network designed to perform a different task such as prediction of neighboring words (CBOW) or prediction of the target word (Skip-Gram) , the integrations are optimized directly , so that the interior product of two word vectors is equal to the logarithm of the number of times the two words appear close to each other. This forces the model to encode the frequency distribution of words that appear close together in a more global context.

Below are presented some logical steps to understand the idea of this algorithm and the way it was created:

- At first, we construct the co-occurrence matrix X of a text box. This table is constructed as we mentioned in 4.3, with the distinction that here we can define context-windows based on which the coexistences of the table will be calculated. That is, the occurrence frequency of the two-word sequence does not need to be calculated just for the case the two words are adjacent by one position, but can be calculated for larger neighborhoods / environment windows, such as those came across in 4.4 and 4.5. Of course, the choice of how big the environment window will be but also whether it will be symmetrical or not (whether the left environment should be separated from the right), are hyperparameters of the model and are being investigated.

- Next, from the coexistence table, a metric can be calculated which shows the semantic similarity between the words. This metric is $P_{ik} = X_{ik}/X_i$, where P_{ik} is the probability that the word i is close to the word k and is calculated by dividing the number of times the word i and the word k appeared together X_{ik}, with the total number of times the word i appeared in the X_i box.

- What we are interested in is given three words i, j and k, to calculate the expression P_{ik}/P_{jk}, which gives a ratio of the probability that the word i and k appear together ,in relation to the probability that the word j and k appear together. This practically means that if the word k is more similar to i than to j, then the ratio will give a value> 1, if it is more similar to j than to i, then the ratio will give a value <1 and if it is equally related to both then it will give a value close to 1.

- The above expression to define in the form of equality is given as follows: $F(w_i, w_j, u_k) = P_{ik}/P_{jk}$, where w and u are two separate levels of integration. Otherwise, the equation can be considered as follows: $F(w_i - w_j, u_k) = P_{ik}/P_{jk}$. This form, which contains the difference $w_i - w_j$, makes sense because word vectors are linear systems and so can be performed arithmetic in the integration space, but also because the distance between word vectors is important information. On the other hand, this distance can be related to the probabilities P_{ik} και P_{jk} as shown in the following figure [93]:

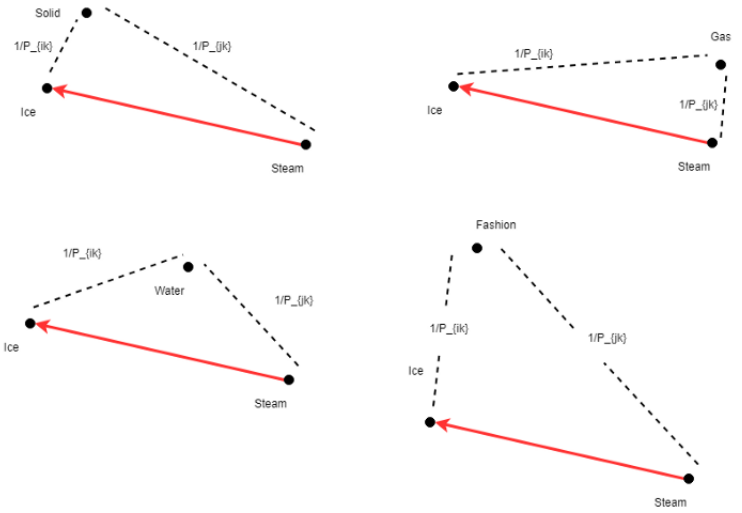


Figure 4.38: Behavior of vector distances in relation to a word

where we observe that as the word k changes, so do its distances from the other two fixed words j and i. These distances are defined as the inverse of the probability $1/P_{ik}$ και $1/P_{jk}$, as the greater the probability that the words are together, the more similar they are and therefore the shorter their distance in space. But the distance between the fixed words i, j $(w_i - w_j)$ remains the same.

- Next, a major concern is to convert the vector to the left of equation, to vector scalar as is the right of equation. This conversion is easily done using transpose and internal product between two entities: $F((w_i - w_j)^*.u_k) = P_{ik}/P_{jk}$.
- Next, we assume that F is a uniform function and we conclude with the following implications: $F(w_i^* u_k - w_j^* u_k) = F(w_i * u_k)/F(w_j^* u_k) = P_{ik}/P_{jk}$.
- With a more abstract view it could be assumed that $F(w_i^* u_k) = P_{ik}$ and if F=exp then we end up with the following: $\exp(w_i^* u_k) = P_{ik} = X_{ik}/X_i \implies w_i^* u_k = \log(X_{ik}) - \log(X_i) \implies w_i^* u_k + \log(X_i) = \log(X_{ik})$.

  Now since we do not yet have bias terms in the equation, the equation could be reconfigured as follows:

$$w_i^* u_k + bw_i + bu_k = log(X_{ik}) \tag{4.32}$$

with bw and bu being biases of the neuronal.

From the relation (4.32), it finally appears that the interior product of two word vectors is required to be equal to the logarithm of the number of times the two words appear close to each other, as we mentioned above.

Therefore, the cost function of the neuron in this case is defined as follows:

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(w_i^T u_j + bw_i + bu_j - log(X_{ij}))^2 \tag{4.33}$$

where the error we came across above (4.32) has been squared (MSE-mean squared error), and has been multiplied by $\boldsymbol{f(X_{ij})}$ which is a weighting function. In the end, the errors of

all pairs of words for neuronal training are added up. The weighting function f is used to prevent the problem that would occur if $X_{ij} = 0$ and is defined as follows:

$$\begin{cases} f(x) = (\frac{x}{x_{max}})^a & , if \quad x < x_{max} \\ 1 & , otherwise \end{cases}$$

A more detailed analysis of the Glove algorithm as well as its parallelism with other models such as skip-gram can be found here [94].

4.2.2.7 FastText

FastText is an open source library, created by Facebook's artificial intelligence research team (Bojanowski et al) in 2016, for effective learning of word representation and sentence classification [95]. It is an extension of word2vec, which uses the n-grams of words (4.2.2.2).

Instead of supplying individual words to the Neural Network, FastText breaks words down into several n-grams (sub-words). For example, the 3-grams for the word "artificial" are <ar, art, rti, tif, ifi, fic, ici, ial, al> (where the square brackets indicate the beginning and the end of the word). While this adds a lot of extra computation to the training, it allows word incorporations to encode sub-word information. This helps to grasp the meaning of the smaller words and allows the integrations to understand the suffixes and prefixes Once the word is represented using n-grams characters, a skip-gram model is trained to learn the embodiments. This model is considered a word bag model with a sliding window over a word because it does not take into account the internal structure of the word. As long as the characters are in this window, the order of the n-grams does not matter.

The structure and training of this neuron is presented in more detail in the following steps:

1. Initially the target word to be fed to the neural skip-gram as input is broken down into n-grams. Suppose we have the word eating and break it down into 3-grams. So we get the following 3-gram character list of the word [96]:

Figure 4.39: 3-grams of the word "eating"

2. Then a hashing is applied. Instead of learning one embodiment for every single n-gram, we learn total embodiments K where K denotes the bucket size. More specifically, to limit the memory requirements of the model, a hash function that corresponds to n-grams in integers 1 to K is used. [99]. Finally, a word is represented by its index in the word dictionary and the set of fragmented n-grams it contains.

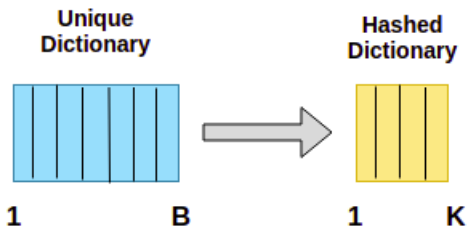


Figure 4.40: Hashing

3. At this point, the integration for the target word, i.e. its vector, is calculated by summing the vectors of all its n-grams as well as the same [96]:

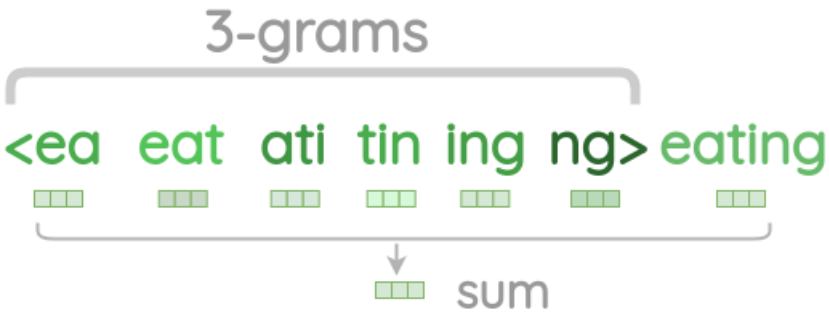


Figure 4.41: Embedding of target word

4. For context words that are defined as we have seen from a context window, are taken directly from their vectors themselves without adding n-grams [96]:

Figure 4.42: Context λέξεις

5. The skip-gram algorithm presented in chapter 4.2.2.5 is called **vanilla Skip-Gram** and is interpreted as a multiclassification task and the output vectors are dimension V, as well as the total number of words in the dictionary. These vectors represent the probability that arises for each word in the dictionary after applying the softmax function to the output of the neuron, which in the case of words that really belong to the context, is sought to be equal to 1. Essentially, it is a classification between V classes and the resulting vectors contain the probabilities that the prediction word belongs to each class. The softmax activation function is computationally very expensive, as it requires scanning the entire output integration table (Woutput) to calculate the probability distribution of all V words, where V can be millions or more. The total calculations made to derive the output vector are O(V+V), as V calculations are needed, one for each dictionary word, plus V for calculating the normalization factor of the denominator, which is calculated once and is the same for all words [97]:

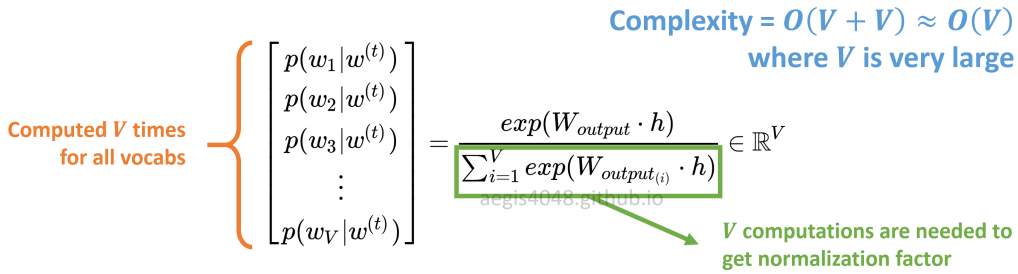


Figure 4.43: Complexity of the Vanilla Skip-Gram algorithm

Due to this computational inefficiency, softmax is not used in most Skip-Gram applications, nor in the FastText algorithm. Instead, we use an alternative called negative sampling with the sigmoid function, which restates the problem into a set of independent binary sorting tasks.

More specifically, in recasting the problem as a set of independent binary classification tasks, the goal is to independently predict the presence (or absence) of context words. Therefore, for a target word, we consider all context words as positive examples and all other words as negative. Instead of trying to predict the probability that a word is close to all the words in the vocabulary, we try to predict the probability of whether the training word samples are neighbors or not. For example, having the word "orange" as the target word and the word "juice" as the context word, we are not trying to predict the probability that the juice is a close word, i.e. P (juice | orange), we are trying to predict whether the words (orange, juice) are close words to or not by calculating P (1 | <orange, juice>). So instead of having a giant softmax - classification between V classes, we have turned it into V binary classification problems.

The new goal is to predict, for each given environment-word pair (w, c), whether the word (c) is in the context window of the central word (w) or not. Since the goal is to define a given word as True (positive, D = 1) or False (negative, D = 0), we use the sigmoid function instead of the softmax function. The probability of a word (c) appearing in the context of the central word (w) can be defined as follows:

$$p(D = 1|w, c; \theta) = \frac{1}{1 + exp(-\bar{c}_{output_{(j)}} \cdot w)} \in \mathbb{R}^1 \tag{4.34}$$

where c is the word we want to know if it belongs in the context window or not, w is the input word (center) and ϑ is the weight table that passes to the model. Note that w is equivalent to the hidden level (h) and $\bar{c}_{output_{(j)}}$ is the word vector from the output weight table (Woutput). At this point another method is applied that has not been previously mentioned in skipgram. This method is called negative sampling. Negative sampling allows us to modify only a small percentage of the weights, and not all weights for each training sample. Thus, we further simplify the problem by randomly selecting a small number of "negative" words k (a hyper-parameter, let's assume 5)for which to inform the weights (a "negative" word is the one for which we want the network to export 0 , i.e. not belong to the context window). For our educational sample (orange, juice), we will take five words, let's say apple, dinner, dog, chair, house and use them as negative samples. For this particular iteration we will only calculate the odds for juice, apple, dinner, dog, chair, and house. Therefore, the loss will only be transmitted for them and so, only the weights corresponding to them will be updated.

The total calculations that are made, now, to obtain the output vector are O(K + 1), since the probabilities are calculated only for the respective context word for the corresponding K negatives that have been selected:

$$\begin{bmatrix} p(D=1|w,c_{pos}) \\ p(D=1|w,c_{neg,1}) \\ p(D=1|w,c_{neg,2}) \\ p(D=1|w,c_{neg,3}) \\ \vdots \\ p(D=1|w,c_{neg,K}) \end{bmatrix} = \frac{1}{1+exp(-(\{\bar{c}_{pos}\} \cup \bar{W}_{neg}) \cdot h)} \in \mathbb{R}^{K+1} \tag{4.35}$$

The new objective function for skip-gram with negative sampling is given by the following expression:

$$J(\theta; w, c_{pos}) = -log\,\sigma(\bar{c}_{pos} \cdot h) - \sum_{c_{neg} \in W_{neg}} log\,\sigma(-\bar{c}_{neg} \cdot h) \tag{4.36}$$

where ϑ is the table of weights passed in the model, w is the central target word, c_{pos} is the positive sample word, i.e. the one belonging to the environment of w, σ is the sigmoid function, $\bar{c}_{pos}$ is the vector of the positive word c_{pos} from the output weight table (Woutput), h is the vector of the hidden level, and $\bar{c}_{neg}$ is the vector from the output weight table (Woutput) of the negative word c_{neg}, which belongs to the set of negative words W_{neg} that we have sampled.

By minimizing the above cost function, the first term tries to maximize the probability of appearing for words that are actually in the context window, i.e. coexist, while the second term tries to pick up some random words j that are not in the window and minimize the possibility of their coexistence with the central word.

For the example we have seen so far, with the central word "eating" and context words "am" and "food", let us sample two negative words [96]:

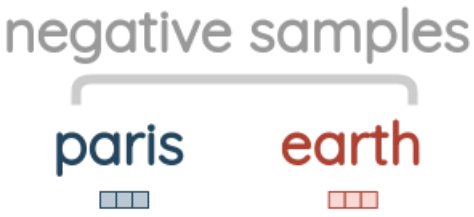


Figure 4.44: Negative samples

6. Then, after the ngrams word vectors of the central word "eating" have been entered into the neuronal, together with the word itself and their average has been calculated, then this average that constitutes the vector h of the hidden layer , is multiplied by internal product with Woutput weights.

7. From the resulting vector we are only interested in the rows that correspond to the K + 1 elements, i.e. the context word and the K negative samples. For these series of the resulting vector, the sigmoid function is computed, so that a probability arises for each of them. This probability reflects how likely these words are to appear in the context.

In our case, where the context words are "am" and "food" and the negative samples are "paris", "earth" this process is shown below [96]:

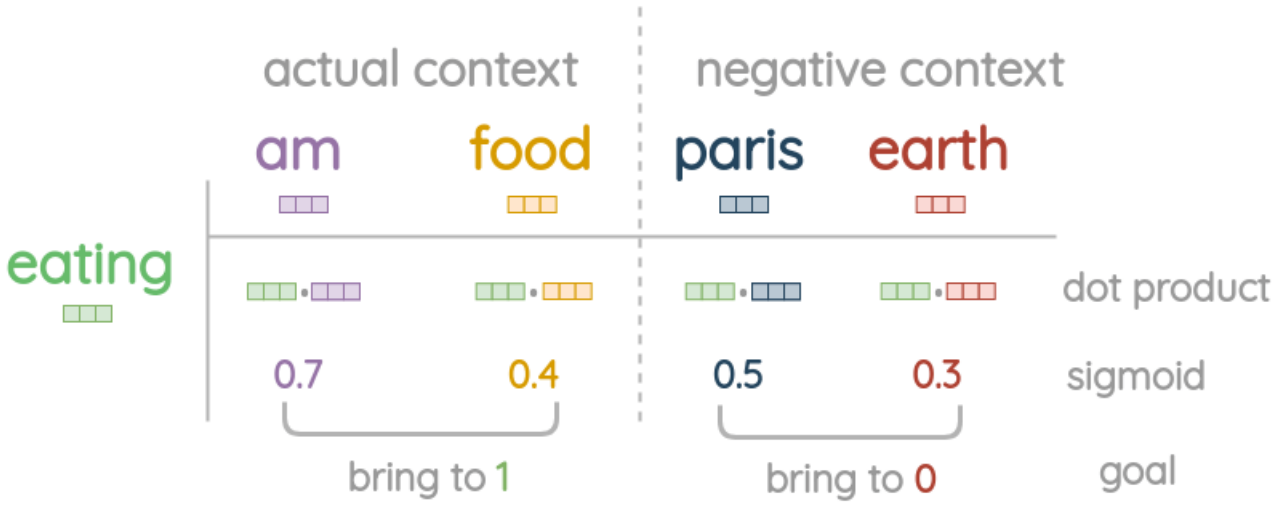


Figure 4.45: Fast Goal of FastText

As shown in the figure, after applying the sigmoid function, the goal is for the chances of the actual context words to come as close as possible to 1, that is, to be classified as likely as possible , as positive samples, while the chances of the negative ones samples to come as close as possible to 0, i.e. to be classified with the least possible probability as positive samples.

A huge advantage of FastText is that it can export vector representations for words that do not exist in the dictionary, that is, that the algorithm has not come across in training. This is achieved through the n-grams used, since the representation of a word is the sum of the representations of its n-grams and even if the word does not exist as it is, one of its n-grams can be found in the dictionary. In short, this simple model allows the sharing of representations between words, thus helping to learn reliable representation for rare words. This phenomenon does not occur in the word2vec and glove models, which when a word does not exist in the dictionary give output out-of-vocabulary (OOV).

FastText also does better in morphologically rich languages, such as German where many words are complex. For example, the noun phrase "table tennis" is spelled with one word as "Tischtennis". Taking advantage of the character similarities between "Tischtennis" and "Tennis", the model does not treat the two words as completely different words, but can assume that there is a degree of affinity since they have some n-grams in common A word that has a spelling mistake and consequently the speaker has not pronounced it correctly, is not a problem for this model, as through n-grams he can connect it with the correct -correctly spelled- version. This is a great advantage in speech assignments, as is the case in the present dissertation, as it is very likely that there are incorrect pronunciation of words that should not lead to wrong semantics.

FastText combines information from the internal structure of a word (n-grams) but also from the context words that enclose it (context words). This allows FastText to compare words based not only on lexical variants, as traditional word distance calculation methods do, but also on semantically relevant words.

In the present dissertation, the FastText model was tested, among other things, to extract word incorporation or otherwise its characteristics. More specifically, we used a pre-trained FastText model that has been trained in Wikipedia 2017 data [95] [98].

This pre-trained model has been trained with the following features:

- The skip-gram architecture (not CBOW) has been used.
- Word vectors have dimension = 300. This means that the hidden layer consists of the vector h which has dimension 300.
- For each positive sample, i.e. a sample belonging to the context window, 5 negative samples have been selected. The way these samples were sampled is with a probability proportional to the square root of the uni-gram frequency and this distribution is called **noise distribution**. The mathematical expression of the distribution is given as follows:

$$P_n(w) = \left(\frac{U(w)}{Z}\right)^{\alpha} \tag{4.37}$$

  is the number of times the word (w) appears inside the body (corpus), divided by a normalization coefficient Z so that the distribution becomes a probability distribution of amplitude

$$0,1$$

  and adds up to 1. In the case of the pre-trained FastText model α equals to $\frac{1}{2}$.
- The context window size is size c, where c is evenly sampled between 1 and 5.
- The most common words are been subsampled with a rejection limit of 10^{-4}. Subsampling is a technique in which the number of samples for a word is limited , limiting the frequency of its occurrence.
- A Fowler-Noll-Vo hash function is used that assigns n-grams to integers from 1 to K, with K = 2.10^6.
- While creating the word dictionary, the words that appear at least 5 times in the training set are preserved.
- Step size γ_0 is set to 0.05 for the model.

To export text attributes to our system, it was tested the use of all embeddings of this particular pre-trained dictionary, but also the use of only the 5.10^5 word vectors. This was done for memory reasons, as loading the entire pre-trained Fasttext model required very large resources in terms of ram and conventional machines were unable to load it. For this reason, we tried this small cutback that did not seem to cost enough in performance, since the words in the dictionary are sorted by frequency of appearance and therefore the first 5.10^5 are the most common. Of course, at this point it should be noted that when a part of the pre-trained Fasttext is loaded, the ability to export word vectors based on n-grams is lost and therefore there is a greater chance of words appearing outside the dictionary (out-of-vocabulary words).).

In conclusion, as we have already mentioned, the features at the segment level are extracted after segmenting the information. In this case, after the text has been split into sections in one of the ways presented in 4.2.1, the word representations are extracted for each section separately. This practically means that for each word of a section, 300 attributes are extracted from the above dictionary, or otherwise a dimension vector of 300. For the representation of the whole section, an average of these vectors is calculated from all the words contained in the section. Finally, we end up with 300 features per section / sentence.

An additional step that has not been mentioned so far is the **preprocessing** of the text. After we have divided the text into parts, we apply a process to each part, with which we

remove the special characters and the punctuation marks, we convert the numbers into words / holograms and we make all the letters from capital to small. This preprocessing is done to help detect verbal representations. For example, the word "apple" and the word "Apple", we want to give the same vector representation, which should not be altered because of the capital letter. The same goes for punctuation marks such as "apple" and "apple.". Of course, there are many other methods of word preprocessing in the field of natural language processing, such as stemming, lemmatization, removal of stopwords, etc. These methods, change the form of the word, for example isolate its root (e.g. from troubled to troubl with stemming or trouble with lemmatization) or in the case of stopwords removal, remove frequently used words such as "is" "the "etc. In emotion recognition tasks, however, such distortions can be fatal as, for example," do "from" donnot "may indicate a different emotion or expression, which will be recognized as the same if" donnot " will be converted to "do" or the separator information will be completely lost if the words are removed from the text. For this reason, such pretreatment methods were avoided for the present work.

Despite the great advantages of Fasttext models, their use continued to give us words outside the dictionary, thus forcing us to completely reject these words, since it was not possible to represent them. This fact, combined with the fact that we wanted to try other methods of extracting word representations, with the ultimate goal of improving the performance of segment classifiers, led us to try and use bert embeddings.

Bert embeddings are essentially extracted from transformer architecture (2.6), a type of artificial neural network that appears very strongly in the field of natural language processing and could not be overlooked in this dissertation. Their exact operation is given in the following chapter.

4.2.2.8 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a recent work published by Google AI researchers. It has caused a turmoil in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, such as Question Answering, Natural Language Inference and more. The key technical innovation of BERT is the application of Transformer's two-way training (2.6), a popular attention model, to language modeling. This contrasts with previous attempts that examined a sequence of text either left-to-right or with combined left-to-right and right-to-left training. The results of the publication [100] show that a two-way language model trained can have a deeper sense of the language environment and flow than one-sided language models. In the essay, the researchers describe in detail a new technique called Masked LM (MLM) that allows two-way training in models in which it was previously impossible.

BERT uses Transformer, an attention mechanism that learns the relationships between words (or sub-words) in a text. As we have seen in chapter 2.6 of the theory, transformers consist of two building blocks: the encoders and the decoders. The former read the text input and the latter produce a prediction for a specific task. Since the goal of BERT is to create a language model, only the coding mechanism is required.

Unlike directional models, which read text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire word sequence at the same time. It is therefore considered a bidirectional model, although it would be more accurate to say that it is non-directional. This feature allows the model to learn the content of a word based on all its context (left and right of the word).

Training Strategies

The question here is what will be the prediction goal of the BERT model. Most language models predict the next word in the sequence or some surrounding words such as Word2Vec

and Fasttext. Bert uses two training strategies:

1. **Masked LM (MLM)**
   Before the word sequences in the model are fed, 15 % of the words in each sequence are replaced by a token [MASK]. Thus, the model tries to predict the initial values of these words, which were hidden through the mask, based on the environment formed by the other words in the sequence. To make this prediction, a sort layer is added to the encoder output, which is essentially a fully connected layer with a gelu activation function and regularization. Then the output vectors are multiplied by the embedding table to get the vocabulary dimension and finally, they go through the activation function sodtmax to give a possibility for each word in the dictionary.

   The cost function only takes into account the predictions of masked words and ignores the predictions of not-masked words.

   The following figure shows this architecture [101]:

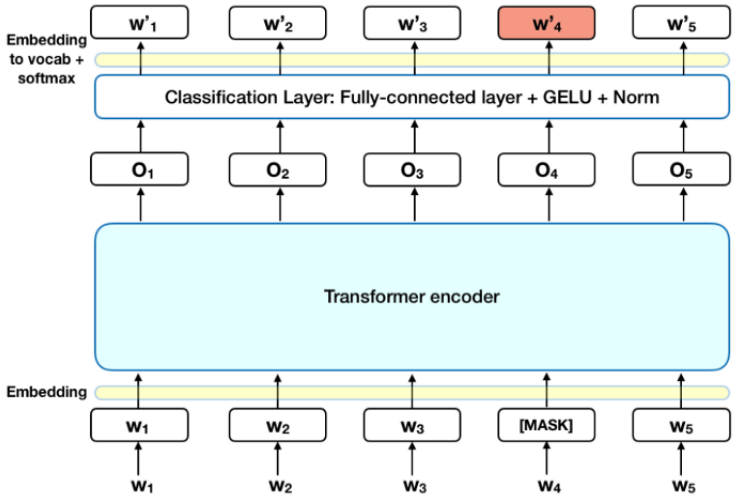


Figure 4.46: Bert Architecture with Masked LM Strategy

   Although the [MASK] indications allow us to obtain a two-way pre-trained model, the creators of the model point out that the disadvantage is that there is a mismatch between pre-training and fine-tuning, which is the use of pre-training model for other classification tasks, as the token [MASK] does not appear during fine-tuning. To mitigate this, we do not always replace the "covered" words with the token [MASK]. The training data generator randomly selects 15% of the tokens for prediction. If the i-th token is selected, we replace the i-th token with (1) the [MASK] token in 80% of cases (2) a random token in 10% of cases and (3) the unchanged i-th token in 10% of cases.

2. **Next Sentence Prediction (NSP)**
   The next strategy for model training is to predict the next sequence. Essentially, the model is fed with sequence pairs and tries to predict if the second sequence is immediately after the first in the original text. During the training process 50% of the pairs consist of consecutive sequences in the text, while the remaining 50% consist of pairs of randomly selected sequences.

   And in this case, to make the prediction, a sort layer is added to the coder output, trying to sort the outputs into the IsNext, NotNext classes, that is returning a dimension 2 vector to which the softmax function is then applied again to give a probability .

Fine-Tuning

The above two strategies are used in the pre-training process of the BERT model. However, when this model needs to be used in a specific application, for example in a specific classification task such as emotion analysis or in a Question Answering task, then a decoder or a classifier is added to the end of the model (shallow fully connected network). This process is called **Fine-Tuning** and during its implementation, only the additional layers update their parameters and perhaps some of the last network layers, while most of the system parameters are not affected.

Input Representation

BERT expects a specific representation of the entry. The steps to pre-process the input are as follows:

- Initially, a sample sentence is converted to a sequence of units (tokenization). BERT has its own tokenizer, which, given a sentence in its input, will construct at the output a sequence of words (tokens) that it will find in the vocabulary. Specifically, it uses the WordPiece technique, which essentially splits a token such as "playing" into "play" and "##ing". This is mainly done to cover a wider range of words outside the vocabulary (Out-Of-Vocabulary/OOV). For input words that are not recognized in the vocabulary the tokenizer will try to break them down in the vocabulary terms with a maximum number of characters. In the worst case, it will split a word into the individual characters that make it up. If a word is split, the first term will appear in the sequence as it is, while in the other terms, the double symbol # will be added at the beginning fro the model to identify that these are terms that have been preceded by a split

- Next, the token [CLS] is added to the beginning of each sequence, i.e. each sample. It is a classification token commonly used in conjunction with a softmax level for classification tasks. For anything else, it can be safely ignored.

SEP token is also added, which is inserted at the end of each sequence as a token separator. It is used in pre-training for sequence pair tasks, i.e. predicting a next sentence, such as the Next Sentence Prediction we mentioned. When only one sequence is used, it is attached to the end.

After the above pre-processing procedure, BERT represents each sequence of three vectors as shown in the following figure [101]:

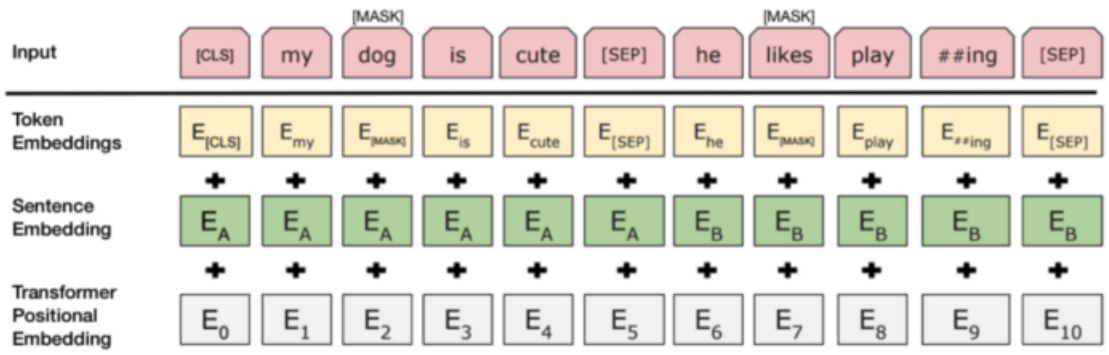


Figure 4.47: BERT input representation

First, we have the input from the tokens we created in the way discussed above. It is noted that these tokens may include the token [MASK], if we are referring to pre-training

with Masked LM strategy. Then, the three vectors are created: Token embeddings, Sentence Embeddings, Transformer Positional Embedding. Token embeddings are the IDs of each token in the vocabulary. Sentence embeddings are a binary value for sentence separation. Finally, tranformer positional embeddings indicate the position of each word in the sequence. Thus, for a given token, its input representation is constructed by summing the corresponding 3 integrations: token, sentence and positional.

Export Bert Embeddings

In this dissertation, what we are interested in is to use a pre-trained BERT model, in order to export word representations and consequently whole samples, i.e. embeddings, which will then be used as features by segment classifiers, which are asked to sort the text samples.

Google provides two types of models: base and large. Their difference lies in size. The base has the following characteristics: L = 12, H = 768, A = 12, Total Parameters = 110M, i.e. 12 levels with hidden layer size equal to 768 and 110 million total parameters. The large one has the following characteristics: L = 24, H = 1024, A = 16, Total Parameters = 340M, i.e. 24 levels, hidden level size equal to 1024 and 340 million total parameters. It also provides the following two versions: case and uncased. In case uncased, the input words are expected to be in English without uppercase characters, while in case case, the words can also contain uppercase characters. The pre-training data they used is from BooksCorpus (800 million words) (Zhu et al., 2015) and English Wikipedia (2,500 million words).

In this work we used the bert base cased version. The architecture of this model internally consists of 12 identical 768 size levels, as mentioned, where each level simulates an encoder. Each encoder consists of various processing parts of the vector integrations as we have mentioned in Transformers theory (2.6). Specifically, the first part is a multihead-attention mechanism with 12 focus points. The next section is a normalization layer followed by a feed-forward network, structured by two fully connected levels of neurons with a relux function. The last part of the encoder is the position coding mechanism by inserting position integration vectors in parallel with the word integration vectors.

The internal structure of the BERT model is shown in the following figure [101]:

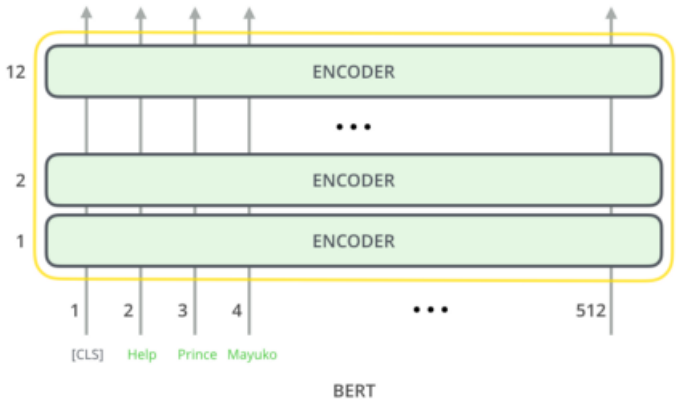


Figure 4.48: Internal structure of the BERT model

Therefore, the steps followed to obtain the vector embodiments of our samples are as follows:

- First each sample is divided into tokens which it consists of through the bert special tokenizer.
- Next, we add at the beginning of each sequence of tokens, the token [CLS] and at the end the token [SEP], as even if a sample consists of more than one sentence, we

consider it as a total sample, i.e. one sentence , since the [SEP] sentence separation is only useful during pre-training.

- The model, as we have already seen, expects three parallel vectors of constant length to represent each sequence at the input. Since our samples have different lengths, since they each have a different number of words, we choose as a fixed length the maximum length of all samples defined by the maximum number of words in the sample. Thus, samples consisting of a number of tokens less than the maximum length are completed with the token [PAD] and for those consisting of a number of tokens greater than the maximum length, we cut off the excess terms.

- The vectors we finally construct are the vector tokens_ids from the IDs for each term in the sequence, as provided by the model vocabulary, and the attention_mask vector, which consists of aces in the first n positions, where n is the length of the input sequence, and from zero to the remaining max_lengthn positions. Since we do not use more than one sentence in the sample, the vector sentence_ids is omitted and we will not use it.

Finally we pass the above two vectors to the pre-trained model per batch and take the embeddings of each token, summing the output vectors from the last four levels. What we are looking for is to find the representation of the whole sample. This representation is calculated from the average of the embeddings of the tokens that make up the sample.

The reason we choose to add the integrations from the last 4 levels is because empirically BERT inspirers suggest that this is a good tactic for high performance.

The following figure shows indicative different combinations of these vectors, from different levels and their f1 scores, as they were examined in the publication [100]. Of course, the choice of the appropriate combination depends on the problem [103] each time:

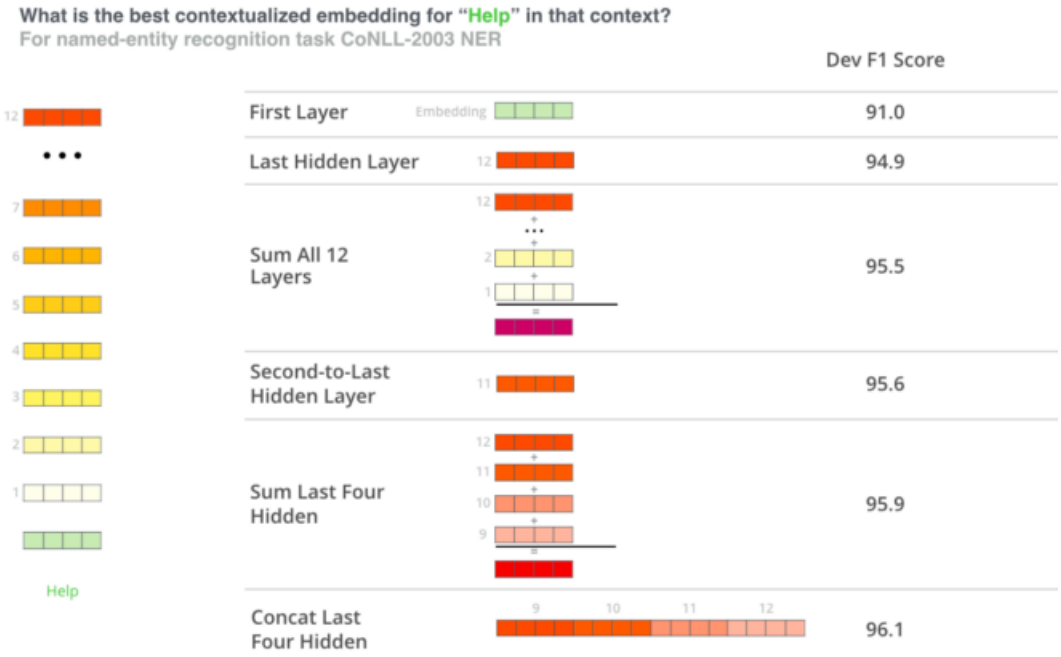


Figure 4.49: Vector combinations of different levels of bert

In conclusion, after segmenting the speech, we will finally end up with a dimension 768 vector which is the representation of the part-sentence and which will be used as features

for the text segment classifiers that we will analyse in the next chapter.

Vectors produced by the BERT methodology are superior to previous methods because the representation of each word is not static, such as Word2Vec and Fasttext, where the representation is context-independent, but each representation is dynamically updated by the surrounding words.

4.2.3 Segment Classifiers - Emotion Recognition

The text segment classifiers are the same as the corresponding audio segment classifiers we analyzed in chapter 4.1.4. As shown in the figure 4.29, three segment classifiers are used. A classifier that classifies texts into four **emotion** or otherwise categorical characteristics: anger, sadness, neutrality, joy. A classifier that classifies texts according to **valence** : positive, moderate, negative and one who classifies them based on **arousal**: high, moderate, low. Vigor and arousal, as we have analyzed, are a two-dimensional space of emotional characteristics.

4.2.4 Datasets

Only the iemocap dataset presented in chapter 4.1.5 was used to train these three classifier models, as it is the only one containing transcriptions.

For the emotions, only the samples belonging to the 4 basic emotions examined in the present work (sadness, anger, neutrality, joy) were retained and the samples belonging to the "enthusiasm" class were considered "joy" samples (table4.1).

For valence and arousal, as we have mentioned the iemocap database contains continuous values, 6 classes of emotions were created (3 for vigor and 3 for stimulation), in the same way as in the case of sound. The classes were divided into negative, neutral, positive for the valence and low, neutral, high for the arousal based on the value thresholds shown in the table 4.2.

Finally, after re-formatting the data, each sample text follows the procedure described in the previous subchapters. The first step is to split into sections. Specifically, for our experiments, the split per sentence was used, because a sentence is considered a sufficiently large context in which it manages to develop an emotion, but also a sufficiently small one in order not to involve more than one emotions/characteristics. The next step is to extract attributes for each section and the last step is to feed them to the section classifiers so that each section is labeled with an emotion, vigor and arousal tag. It is noted that for the training process, the text data is considered presegmented, as in the iemocap data set each sample consists of a single tag, even if it contains more than one sentence

4.2.5 Experiments and Results

Experiments with both fasttext embeddings and bert embeddings were performed on all three classification tasks. During the training and evaluation, the same pipeline we analyzed in chapter 4.1.6 was applied. Specifically, SMOTETomek was used to manage potential imbalances, StandardScaler, VarianceThreshold and PCA. In addition, gridsearch was applied with Repeated-StratifiedKFold cross validation which uses 5 folds and 3 repetitions. In this gridsearch hyperparameter tuning is been performed in the number of components of the pca and for the SVM models, in the parameters C and gamma. The results of the experiments during validation are shown in the table below. It is noted that in this case there is no need for evaluation in a separate test set, as these are texts that are not dependened on any other parameter such as the speaker. On the contrary, the evaluations that emerged from the cross

validation are much more realistic as they are based on many different splits and not on a single test set.

Classification Task	Iemocap			
	Train/Val SVM with Full FastText Embeddings	Train/Val SVM with 500k FastText Embeddings	Train/Val Xgboost with Bert Embeddings	Train/Val SVM with Bert Embeddings
Emotion	66.5 (+/-1)	65.2 (+/-1.3)	63.9 (+/-1.7)	**69.5 (+/-1.4)**
Valence	61.5 (+/-1)	60.4 (+/-0.9)	59.4 (+/-0.9)	**63.8 (+/-1)**
Arousal	48.8 (+/-1.1)	48.1 (+/-0.9)	48.2 (+/-1)	**51 (+/-1.1)**

Table 4.10: Iemocap Evaluation

Note that the + - values presented in parentheses are the standard deviation of the odds in the different test folds.

From the table above we can first observe that when we use a part of fasttext embeddings and not the whole pre-trained model, then we lose about 1% in performance compared to using the whole fasttext. This was to be expected, as trimming the fasttext we have fewer words in the dictionary, while at the same time we lose the advantage of finding word representations through their ngrams. As a result, the number of words outside the vocabulary (OOV) increases.

In contrast to fasttext, we also try bert embeddings. This gives us the advantage of not having any word outside the vocabulary but at the same time to take advantage of the whole environment of a word to produce its representation, something that does not happen with fasttext. This explains the increase in efficiency that we observe in the last column regarding the training of an svm using bert embeddings. This increase amounts to 3-4% for emotion and valence and to 2% for arousal. Finally, we make a comparison of the SVM model with an xgboost (again with bert embeddings) and notice that the SVM is superior. Therefore, this will be our final model that will be used by the system for all three classification tasks.

The best parameter values that gave the best model are shown below:

- **Emotion**:
  Parameters of best svm model: 'SVM C': 5.0, 'SVM gamma': 'auto', 'pca n_components': 0.98
- **Valence**:
  Parameters of best svm model: 'SVM C': 1.0, 'SVM gamma': 'auto', 'pca n_components': 'mle'
- **Arousal**:
  Parameters of best svm model: 'SVM C': 0.5, 'SVM gamma': 'auto', 'pca n_components': None

Below we can study in more detail the confusion matrices for the models in the last column, as well as the performance metrics per class. Note that the confusion matrix here is calculated by adding all sorted samples from all test folds and from all iterations. Therefore, samples from the 5*3=15 different classifications made during repeated stratified cross validation have been added. This practically means that the sum of the samples of the whole table does not correspond to the real total number of samples, i.e. the absolute values are not representative, but the normalized values can show the proportions correctly.

Emotion

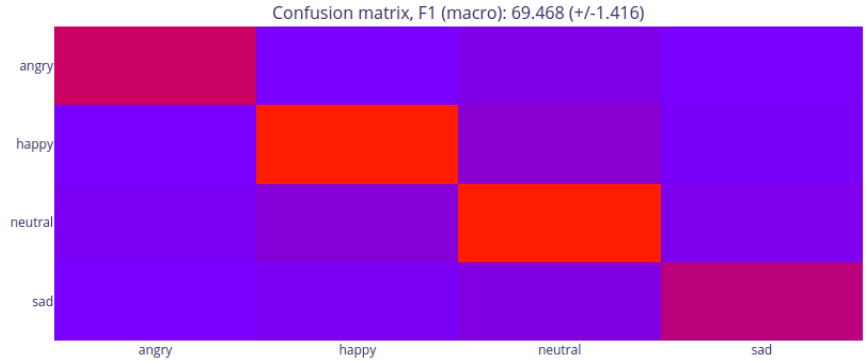


Figure 4.50: Confusion Matrix Emotion

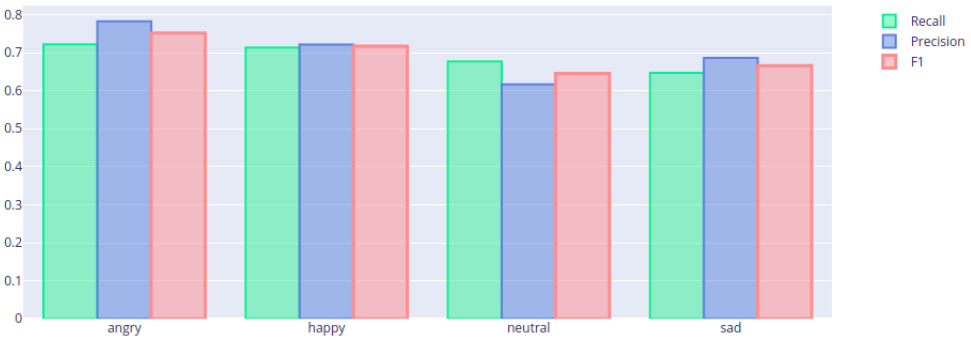


Figure 4.51: Performance Metrics per Emotion Class

Valence

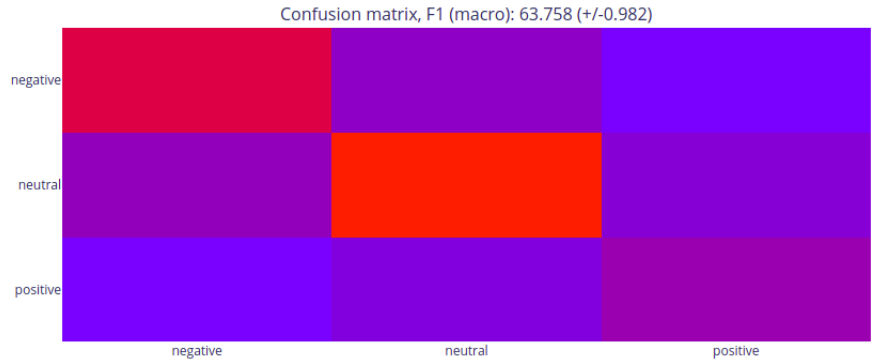


Figure 4.52: Confusion Matrix Valence

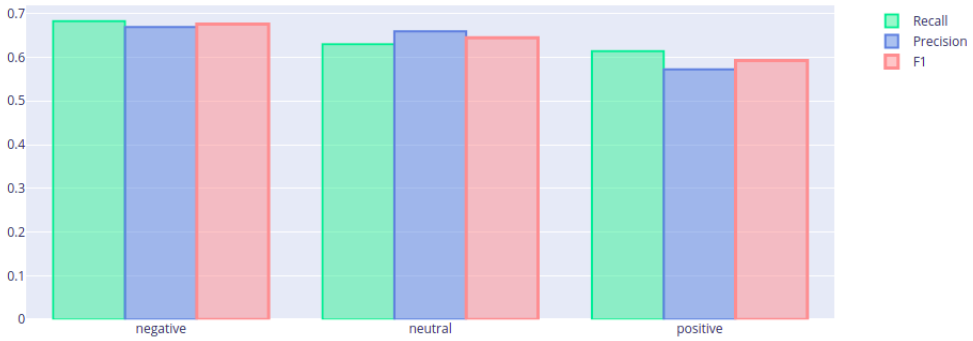


Figure 4.53: Performance Metrics per Valence Class

Arousal

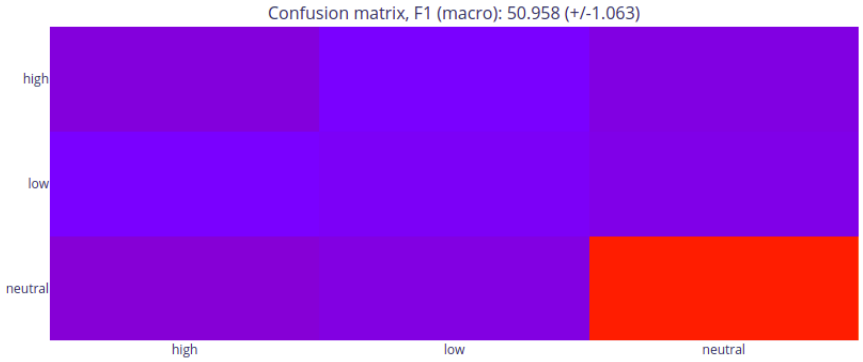


Figure 4.54: Confusion Matrix Arousal

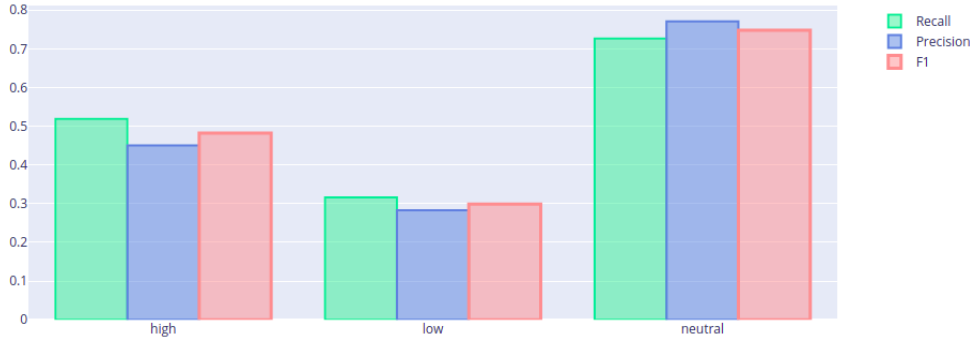


Figure 4.55: Performance Metrics per Arousal Class

From the above images we observe that in both emotion and valence, the returns per class are similar. From the emotion task, the neutral class is the one with the smallest precision, while the sad class is the smallest recall. However, all classes have an f1 score between the range 65% and 75%. On the other hand, in the valence task it seems that the positive class is the one that is doing worse than the other two, which is perceived both by the confusion matrix and by the lower precision and recall that it presents in the graph of the performance metrics per class. In this case too, however, the differences between the classes are not large and the f1 score ranges from 59% to 68%.

Regarding the arousal task, a worse picture appears, as the neutral class seems to be superior to the other two (73% recall and 77% precision). In contrast, the high class has about 52% recall and 45% precision, while the low class does even worse as it has 32%

recall and 28% precision. Therefore, the model does very well in separating/classifying the neutral and moderate class into the high class, while the low class cannot be separated at all, since its performance in this class is equivalent to a random classifier, if not worse.

CHAPTER 5

Recording-Level Analysis

Having analyzed the parts of the system related to segment-level analysis, i.e. those that use segment classifiers, both for audio and text, the next step is to move to recording-level analysis. It is named so because it concerns the total recording, i.e. the total speech and not just some parts of it. Our goal from the beginning was to evaluate the quality of the speech, therefore, we will use the information we extracted from the sections, in the previous chapters, to achieve this purpose, but also adding some additional information / features.

Essentially, in this chapter the rest of the architecture will be analyzed again separately for the sound and separately for the text (the green elements in the shapes 3.1 and 3.2).

5.1 Audio Analysis

The following two subchapters will be presented the features used for total recording/speaking.

5.1.1 Aggregation of Class Posteriors

The audio signal of the speech has first passed through the analysis at the section level, i.e. it has been divided into sections and for each of these sections the class posteriors have been produced from the models we selected in the chapter 4.1.6. That is, we have three labels for each section that deal with emotion, valence and arousal respectively. But we are interested in characterizing the total signal, that is, the total speech. Therefore, we do a **aggregation** of the class posteriors. This aggregation is calculated by taking the average for each tag. That is, the number of sections classified in this tag is calculated and divided by the total number of sections. Thus, we end up with a percentage per tag, which in essence is the probability that this speech belongs to the respective tag.

For example, an audible signal can be characterized by the following percentages / probabilities:

- As for the emotion:
  p(emotion=sad)=30%, p(emotion=neutral)=40%, p(emotion=happy)=10%, p(emotion=angry)=20%
- As for the valence:
  p(valence=positive)=60%, p(valence=neutral)=40%, p(valence=negative)=0%

- Regarding the arousal:
  p(arousal=high)=10%, p(arousal=neutral)=90%, p(arousal=low)=0%

The above probabilities are used as characteristics of the overall speech, which will then help the classifiers to evaluate the quality of the speech. Therefore, we end up with 10 attributes (as many as the total tags), which have emerged from the segment classifiers, for the overall audio file.

Apart from these features, however, some other features of total speech are added, which are analyzed in the next subchapter.

5.1.2 High Level Features

High level features, as we call them, arise after locating the speech segments within the audio signal. This signal contains both parts of silence and parts of speech. The simplest way to separate speech from quiet is based on signal energy, as it is known that quiet has low energy while voice/speech has high.

Voice Activity Detection

We follow a process of detecting speech segments with the help of the pyAudioAnalysis [56] library. The steps for detecting these sections are as follows:

- Divide the audio signal into short-term windows just as we analyzed in chapter 4.1.1.
- For each of these windows we export the 33 audio features we have mentioned before (4.1.3).
- From these characteristics we select the energy, for all samples/windows. We make a classification of this feature in ascending order. Next, we calculate the number of 10% of the total windows. We take this number of lower energy windows and calculate the average energy in these windows. We do the same for this number of higher energy windows too. Thus, we end up with two energy values, one resulting from the lower energy windows and one from the higher energy windows, which are the threshold of low and high energy respectively.
- Next, we get all the characteristics (not just the energy) for the samples/windows that have an energy value below the low energy threshold and respectively for the samples/windows that have an energy value above the high energy threshold.
- From the previous step, we end up with two types of samples: the low energy samples and the high energy samples, or otherwise the quiet/silence samples and the speech/voice samples respectively. Therefore, we have samples of two classes with their corresponding characteristics which we can feed into a model so that it can learn to separate them.
- Making a normalization of the above features, we train a linear svm classifier with the samples of the two classes of the previous step.
- After training the classifier, we use it to predict the odds for each sample/window. This means that we will produce a probability that each sample belongs to the positive class, which in our case is the "high energy" class or otherwise the speech.
- We classify these probabilities in ascending order and then calculate the average of 10% of the total number of probabilities, which have the lowest values and the average of 10% of the total number of probabilities, which have the highest values. We add these two averages weighing 0.5 to each. This means that the two averages have equal influence on the sum. This sum is ultimately the probability threshold above which the samples are considered speech.

- Finally, from all the possibilities of all the sound windows, we select the ones with a value higher than the threshold of the previous step. Respectively we select the windows that correspond to these probabilities.

- From these windows, we "throw" those that are very small and specifically that have a duration of less than or equal to 0.2 seconds. So finally, we come up with a list of short term windows of the original signal, which have been identified as speech windows. That is, the windows of silence have been removed from the signal.

If the initial signal is less than 5 seconds, then we do not apply the above procedure, but we consider the whole signal as "speech".

We apply the process described above for 2 different values of short-term windows: windows of length 0.5 and step 0.25, windows length 1.0 and step 0.25. In this way, we split the sound into both smaller and larger windows as these two techniques can give us different information. For example, there is a probability a larger window that can be classified as "speech" to encapsulate smaller silence pauses, which will be detected when the sound is split into smaller windows.

From the above, we end up with two lists of speech segments: a list of short segments and a list of long segments. Our goal, however, is to export some "high level" features to the overall audio signal.

Features from Voice Activity Detection

We will use the information of the above lists to create the following attributes:

1. Average silence duration which is calculated from the average duration of the silence intervals, i.e. the intervals that are inserted between the speech intervals.

2. Number of pauses/silences per minute (silence segment per minute), which is calculated by dividing the total number of pauses by the minute.

3. The standard deviation of the duration of silence (std of silence duration).

4. The speech ratio which is calculated from the total speech duration divided by the total duration of the entire audio signal.

5. The word rate in speech which is calculated from the number of parts of speech divided by the total length of speech. Here we consider that the speech segments are so small that each one looks like a word.

The above characteristics, as we have mentioned, are calculated twice: one for the speech segments that have resulted from shorter duration short-term windows and one for the speech segments that have resulted from longer duration short-term windows. Therefore, we end up with 10 high-level features for each audio file.

5.2 Text Analysis

Again in the following two subchapters the features used for the total recording/speech will be presented, this time in terms of text.

5.2.1 Aggregation of Class Posteriors

After we have converted the speech into text through speech to text, as we have analyzed before, this text of the speech, has first passed through the analysis at the level of section,

that is, it has been divided into sections (in our case sentences) and for each of these sections the class posteriors have been produced from the models we selected in chapter 4.2.5. That is, we have three labels for each section that deal with emotion, valence and arousal respectively. But we are interested in characterizing the overall text of the speech. Therefore, we do a **aggregation** of the class posteriors. This aggregation is calculated by taking the average for each tag. That is, the number of sections classified in this tag is calculated and divided by the total number of sections. Thus, we end up with a percentage per tag, which in essence is the probability that the text belongs to the respective tag.

For example, a speech text can be characterized by the following percentages/odds:

- As for the emotion:
  p(emotion=sad)=30%, p(emotion=neutral)=40%, p(emotion=happy)=10%, p(emotion=angry)=20%
- As for the strength:
  p(valence=positive)=60%, p(valence=neutral)=40%, p(valence=negative)=0%
- Regarding the arousal:
  p(arousal=high)=10%, p(arousal=neutral)=90%, p(arousal=low)=0%

The above odds are used as features of the overall speech text, which will then help classifiers evaluate the quality of the speech. Therefore, we end up with 10 attributes (as many as the total tags), which have emerged from the section classifiers, for the overall text.

Apart from these features, however, some other features of the whole text are added, which are analyzed in the next subchapter.

5.2.2 High Level Features

The high-level features, as we call them, that concern the whole text, are the following:

1. The word rate, which is calculated by dividing the number of words in the whole text by the duration in minutes (words / minute).
2. The unique word rate. The number of unique words is calculated by taking the appearance of each word once, i.e. double or more occurrences of the same word are not included. We divide this number by the duration and we take the rhythm of unique words.
3. 10-bin histogram of word frequencies. To calculate this attribute, we count for each word in the text, how many appearances it has, i.e. its frequency in the text. Next, we normalize these frequencies by dividing by the total number of words in the text. Finally, we create a 10-point and range 0 to 0.1 histogram for these normalized word frequencies. Each value of the histogram is normalized by the sum of all values. Therefore, we end up with 10 features that correspond to 10 value ranges, each of which indicates the probability or otherwise the percentage of words that have a frequency within the respective value range.

   For example, if we have the text:
   This is a text. The text is short. We are beautiful. Let me explain. Happy birthday to you. Yes, I understand.

   The normalized frequencies per word are as follows:
   This: 0.048, is: 0.095, a: 0.048, text: 0.095, The: 0.048 , short: 0.048, We: 0.048 , are: 0.048, beautiful: 0.048 , Let: 0.048, me: 0.048, explain: 0.048. Happy: 0.048 ,

birthday: 0.048, to: 0.048 , you: 0.048, Yes: 0.048, I: 0.048, understand: 0.048.

The normalized 10-point histogram is as follows:

0,...,0.01 : 0.
0.01,...,0.02: 0.
0.02,...,0.03: 0.
0.03,...,0.04: 0.
0.04,...,0.05: 0.89
0.05,...,0.06: 0.
0.06,...,0.07: 0.
0.07,...,0.08: 0.
0.08,...,0.09: 0.
0.09,...,0.1 : 0.11

We have chosen a frequency range from 0 to 0.1 as a maximum of 0.1 means that a word will appear 1 in 10 times, a frequency that is sufficient.

From the above we conclude with 12 new features of total text that will be added to the 10 features that emerged from the section classifiers, in the previous subchapter, to be used by speech evaluation models.

Reference-based Features

At this point, it is useful to mention the optional addition of some more features. If the speech is based on a reference text, i.e. the speaker was asked to read/recite a predefined text, then some more features can emerge from comparing the text of the speech, i.e. what the speaker finally said, with the reference text.

More specifically, from such a comparison three characteristics can emerge:

- Recall
  Recall in this case reflects how much the user said from what was in the reference text. This metric takes into account and counts possible omissions that the user may have made.
- Precision
  Precision in this case reflects how much of what the user said was in the text. This metric takes into account possible additions that the user may have made.
- F1 score
  F1 score which as we have seen in theory is calculated from recall and precision as follows: $2 * precision * recall/(precision + recall)$.

In order to calculate the above metrics/characteristics, the speech text must first be aligned with the reference text. This practically means that we need to know at all times what the user said which word corresponded to the reference text, even if he did not say it exactly as it appears in the reference text, even if he omitted some words at some point or added his own. Two examples of alignment are presented below:

Example 1:
Reference text: You are very pretty
Speech text: You - very pretty

Example 2:
Reference text: You have - eyes
Speech text: You have pretty eyes

In the first example we can see that the speaker omitted the word "are" and in the second that he added the word "pretty" himself, while it was not in the original reference text. Thus, we see that the alignment of the two texts must be done correctly, even for peculiar cases, as in the above examples.

This alignment can be done by calculating all the possible combinations, i.e. all the possible word matches between the spoken text and the reference text and finally choosing the combination, which is more efficient, i.e. gives the highest score. The score here is the total score from the matching of all words. More specifically, when two words are the same and we match them together, then this match gives score = 1. Conversely, when two words are not the same, then the score is calculated based on the Leveshtein ratio (mismatching words: score = Leveshtein ratio). The Levenshtein ratio is a simple word spacing metric that compares two words letter to letter. For example, if we have the words ab and ac then the Leveshtein ratio will give 0.5 score since half the letters of the two words are the same (a with a) and the other half not (b with c).

Finding the best combination, i.e. the one with the maximum score, is done using a dynamic algorithm, which calculates a two-dimensional array. This table has as many rows as the number of words in the spoken text and as many columns as the number of words in the reference text. Thus we run the table for each word of the speech text that can be combined with each word of the reference text. In each repetition we consider three cases, either to get the score by matching the two words of this repetition, or to get the score by putting a space in the speech text (omitting a word), or by putting a space in the reference text (adding a word by the speaker). In each of these cases we add the score to the score we have collected with the combinations so far and finally select the case that gives the maximum final score to refresh the table in place of the specific repetition. At the end, the last position of this table, gives the maximum total score, i.e. the score that has resulted from the optimal combination of all words. Finally, the alignment corresponding to this score is selected, and based on this alignment, the three metrics mentioned above (recall, precision and f1) are calculated.

These three features can be added to the rest of the full text features to be used by the final models. Nevertheless, in this dissertation we examine the quality of free speech. Thus, it would not be useful to compare this speech with a predefined text. Therefore, we will not use these three additional features. They exist, however, as an option in the system, to enable the use in other speech analysis processes such as for example the detection/evaluation of dyslexia, where the user/speaker is called to read predefined text and in the evaluation plays an important role finding the mistakes he made based on the reference text

CHAPTER 6

Data Collection and Annotation

By the end of the previous chapter, we have now collected all the recording-level features, whether they came from segment classifiers or were subsequently added as high-level features. The ultimate goal of all these features is to be used by recording level classifiers, who will be able to evaluate the quality of the overall speech.

For the training and evaluation of these classifiers, a process of data collection, annotating and agreement was followed. Since there is no open data on the internet, which is related to the quality of speech and which is annotated (supervised learning), this process of data collection was performed end to end, in the context of the research for this dissertation.

The following subsections present all the steps that have been taken to fully create an annotated speech data set.

6.1 Data Collection

For the purpose of collecting data and creating a database, a website was created using the Javascript / html / css programming languages for the front-end part, as well as nodejs for the communication with the server, ie the back-end part. This site was uploaded to an online server in the okeanos cloud service.

The site contained information about the research purpose of the process, some demographic characteristics that the users / candidate speakers were asked to fill in and then the main process of the speech recording.

Demographic characteristics were as follows:

- Name of the user.
- Gender.
- Age.
- English level / fluency
- How many times the speaker speaks in english or reads an english text.
- If he feels anxious when speaking in public.
- If he is introvert or extrovert (on whether practicing comunicational skills).

- The level of his sense of humor
- How much easy it is for him to find examples/argue/justify.
- What his attitude is towards conflicts. That is, if he avoids them, he accepts them or causes them.

All of the above questions aim to create a profile of the speaker, giving some additional information that might then be able to help justify the results or contribute to their production process.

Regarding the main process of speech recording, 40 short English texts (4-5 lines each) were collected from the internet and mainly from the medium, on different topics (such as politics, books, machine learning, etc) which inspire a strong reading / style in the speaker. 20 general questions were also selected that can be answered shortly, such as "what is the best season?", "What do you think will be the biggest challenge for humanity in the coming decades?", "What do you like most about your current job?" etc.

The reason why two types of categories have been created, that of predefined texts and that of free questions, is on the one hand to give users the opportunity to read a specific text, thus providing them with greater ease, but adapting it to the corresponding expressiveness, tone and pronunciation and on the other hand to examine free speech, which does not depend on any reference text, but is spontaneous and instantaneous. These two axes give also the possibility of comparison, as different results are expected in the free text than in the predefined one, mainly in terms of the information provided by the text part of the system (3.2).

The set of these 50 texts / questions are displayed to the user in random order and he is asked to read in the case of the texts or to answer in the case of the questions, while recording himself. Then he has the opportunity to listen to his recordings and choose to upload (save to the server), the one he considers to be the best in terms of the quality of his speech.

This site was shared with both men and women, as well as both people of Greek descent and people of other nationalities, so that there is a variety of pronunciation and data to be as general as possible.

Specifically, a total of 695 recordings / speeches were collected, from 42 different individuals, of which 26 were female and 16 male.

6.2 Data Classes

After the collection of the data described above, the process of their labeling/annotation, ie application of labels in each sample, is necessary and ancillary. If we use supervised learning, the data must be accompanied by their labels, so that the models that will be trained, can learn to classify the samples in the appropriate label-class, by observing some patterns of their characteristics.

There are three tasks chosen for annotation: expressiveness, ease of following the speech and how enjoyable it is.

More specifically, expressiveness is defined as how active, emotional or passionate the speech is, regardless of its content. The marking/annotation in this class is done using 5 labels:

1. Not expressive at all
2. Not that expressive
3. Somehow expressive

4. Quite expressive
5. Very expressive

As ease of following is defined the evaluation of verbal clarity, fluency and rate of speech, for the specific content described. It is noted that fluency, clarity and rate can be interconnected and correlated, e.g. one speaker can make his speech very easy to follow, despite the fact that he speaks very fast, while another speaker can speak slow and yet not making his speech easy to follow. The marking/annotating of this class is done again using 5 tags/labels:

1. Very hard to follow
2. A bit hard to follow
3. Relatively easy to follow
4. Mostly easy to follow
5. Very easy to follow

Enjoyment defines the listener's / annotator's personal view of whether the speech was exciting, entertaining or motivating. Also in this case 5 labels are considered:

1. I hated it
2. Not really
3. It was ok
4. I quite liked it
5. I loved it

The above three tasks were chosen in such a way that they are independent of each other, ie the annotator / listener scores each one independently, without one being connected to the other. This practically means that a speech could be expressive but not enjoyable, or it could be enjoyable but not so easy to follow. Therefore, we are talking about three independent processes that will be performed by three different classifiers.

References: [104],[105].

6.3 Data Annotation

Having defined the three axes on which each speech will be evaluated, another UI (user interface) was created, this time with the help of python/html and flask. It was also uploaded to an online server in the okeanos cloud service and then various listeners/annotators were asked to rate through this platform / website the speeches that were collected earlier.

More specifically, each user registered on the site, then started the annotation process where speeches were displayed in random order, asked to listen to them and finally to choose a tag that suits the respective speech for each of the three tasks (expressiveness, ease of following, enjoyment).

In total, out of the 695 samples, 689 were annotated. Also, the total annotations made by all users were 2687 (for each of the 3 tasks), while the number of users / annotators was 14. The following figure shows more in detail the number of annotations per user:

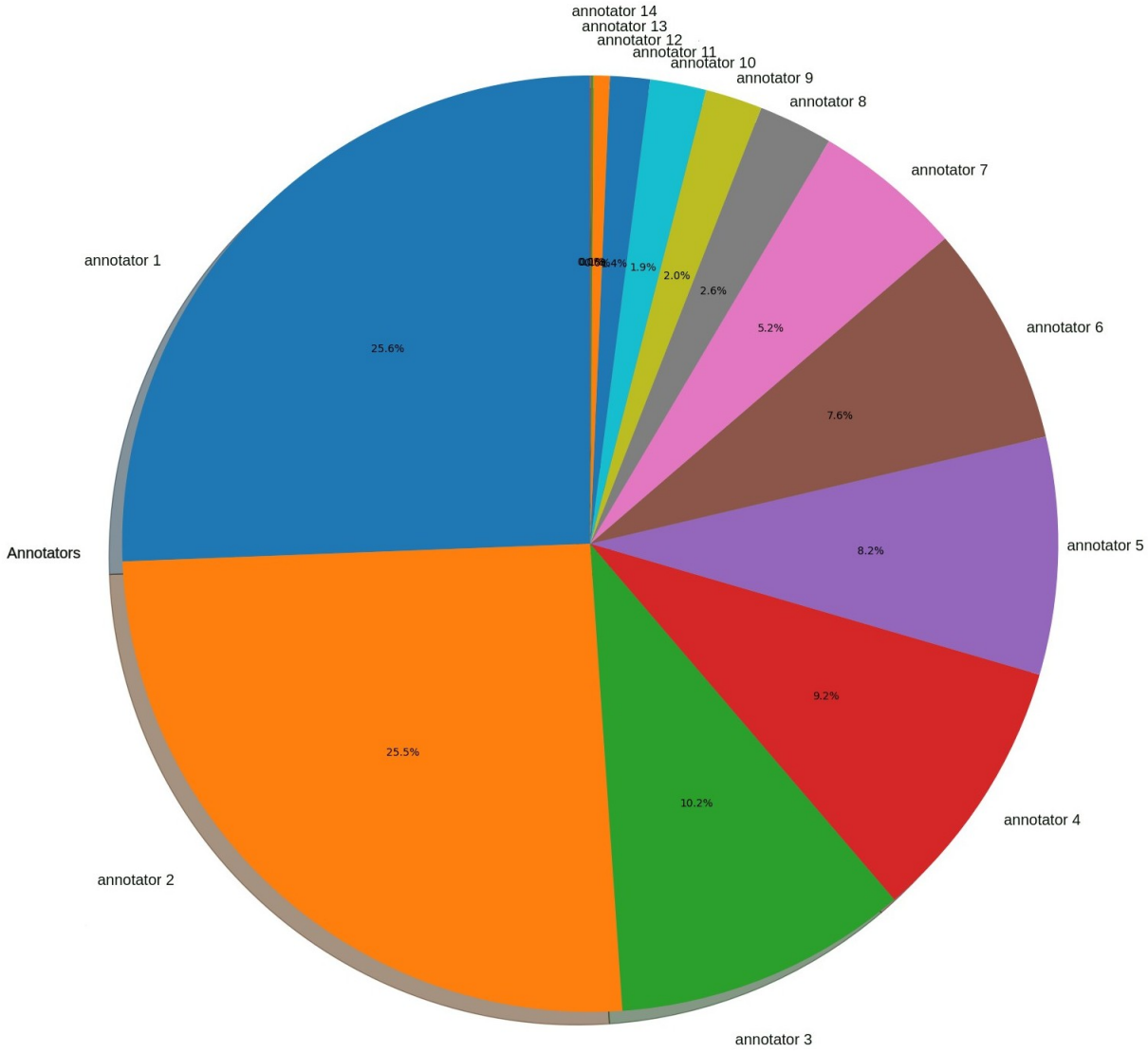


Figure 6.1: Annotation's distribution per user

6.4 Annotation Agreement and Generation of Final Data

After data annotation process, the aggregation and agreement follow. Each sample should end with a single label for each task. In order for this to happen, an agreement must be reached between the tags set by the users / annotators.

For the purpose of aggregating the annotations, the procedure was performed separately for the samples / speeches that had a female speaker and for those with a male speaker. This is because, as we have seen above, female speakers outnumber males and therefore the female samples is much larger than that of males. Thus, in order not to make the model too biased in terms of gender, ie to avoid reducing the problem to gender recognition, it was decided to examine the two sexes separately.

Production of Binary Labels

Although the labels are distinct (5 labels / values per task), they have a continuous, scalable form as the smaller label (1) corresponds to the worst grade and the larger label (5) corresponds to the best one. Therefore, instead of aggregating the labels of a sample based

on majority vote, averaging will be used.

More specifically, for each task (expressiveness, ease of following, enjoyment) and for each gender (female, male), 2 classes / labels will be exported: one negative and one positive, instead of the original 5. This is due to a lack of data, as there are not enough samples for all tags. The export of these two classes is made by the following constraints / steps:

1. The first limitation is that only samples with three or more annotators are retained. This is because for samples with 2 or even worse with only 1 listener / evaluator no valid conclusions can be drawn for the final label, as it will be based on the personal perspective of one or two people.

2. Another limitation is the mean thresholds. The mean value of the annotations given by different annotators in each sample is calculated, and then if this value is below a certain threshold, then the sample is considered negative and is marked accordingly. Conversely, if the mean value is above a certain threshold, then the sample is considered positive and labeled accordingly. If the mean value is in between the threshold values, this sample is discarded and will not be used in the final process. For example, if we have a lower threshold of 2.0 and a higher threshold of 4.0, this means that a sample with a mean value of annotations less than or equal to 2 will receive the final label "negative", while a sample with a mean value of annotations greater than or equal to 4, will receive the final label "positive". Thus, if for example 3 users have rated the specific sample in the expressiveness task as "not that expressive" (value 2), "not that expressive" (value 2), "quite expressive" (value 4), the average value of these notes is $(2 + 2 + 4)/3 = 2.7$ and the sample will be rejected as it does not satisfy either of the two threshold inequalities ($<= 2$ (negative) or $>= 4$ (positive)).

3. In addition to the mean, a deviation threshold is considered. Specifically, the median absolute deviation of the annotations of a sample is calculated, which is the median versus the absolute deviations from the median. In practice, it is the average deviation that results from the deviations of each annotation from the average. Thus, for each sample this value is required to be less than or equal to a predetermined threshold. If this inequality is not satisfied then the sample is discarded. For example, if 5 users have rated a sample in the ease of following task as "mostly easy to follow" (Value 4), "mostly easy to follow" (Value 4), "mostly easy to follow" (Value 4), "mostly easy to follow" "(value 4)," very easy to follow"(value 5), then the average value of these annotations is 4.2 and the deviation is 0.32. In order to maintain this sample, 0.32 must be less than or equal to the deviation threshold, and the mean value of 4.2 must satisfy one of the two inequalities of the thresholds of the previous step.

Agreement of Annotations

The average disagreement of users / evaluators is defined as the average value of the median absolute deviations from all samples. In practice, the users' disagreement is calculated for each sample separately, taking the median absolute deviation of the annotations described above, and then the average value of all these disagreements is calculated to produce an average total disagreement over the entire dataset and annotations. In this way, we have a sense of how well labeled the samples may eventually be. If the average disagreement is high then users had different views on the rating and therefore the samples will not have a "clear" valid label. Conversely, if the average disagreement is low, it means that in most samples the user ratings agreed and therefore the final tags are better defined.

In addition to the total disagreement for all data, the average disagreement for each user is also calculated. Thus, for each sample that the user evaluated, the deviation of the label/annotation he/she set is calculated from the average value of all the annotations of

this sample. Then, taking the average value of these deviations, we have the average user disagreement. This is practically an indicator of the user's validity, which can show us how close the ratings of the specific user are, in comparison with the rest of the users, ie how good his judgment is. When a user's average disagreement is very high compared to the rest, then that user's ratings are rejected and not taken into account in the final labeling process.

6.5 Final Datasets

Based on the above, we will see how the data with their labels were finally configured, for each task separately.

6.5.1 Expressiveness

The table below shows the threshold values used for the expressiveness task, as well as the number of samples per class, after each constraint has been applied. That is, first the average value thresholds are applied, then the deviation threshold is applied and finally the limit of the minimum number of annotators per sample is applied (3 in this case). Finally, the table shows the average disagreement of the annotators. These results are presented for each gender separately:

	Expressiveness			
	Female		**Male**	
	Positive	**Negative**	**Positve**	**Negative**
Mean Threshlod	μ>=4	μ<=2	μ>=3.1	μ<=2
Number of samples after Mean Thresholding	72	80	70	67
Deviation Threshold	σ<0.75	σ<0.75	σ<0.75	σ<0.75
Number of samples after Deviation Thresholding	63	71	48	60
Number of samples after Minimum Annotators	52	53	41	50
Average Disagreement	**0.52**		**0.53**	

Table 6.1: Definition of Expressiveness Dataset

The following pictures show in more detail the distribution of the number of samples per class, 1. before applying the averaging of annotations, 2. after applying the mean value threshold, 3. after applying the deviation threshold and finally 4. after applying the minimum number of annotators.

Female

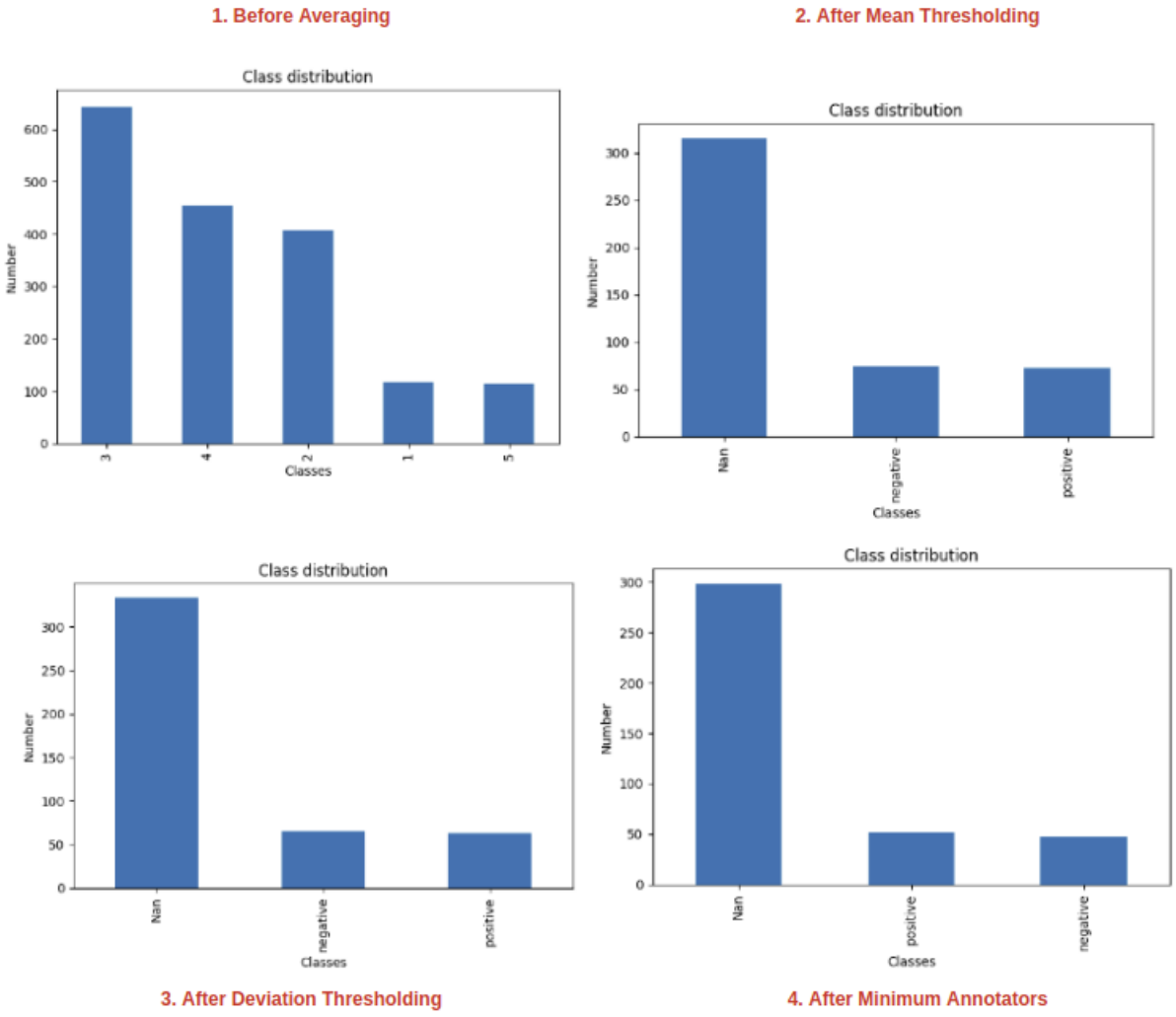


Figure 6.2: Samples Distribution in Classes | Female Expressiveness

Male

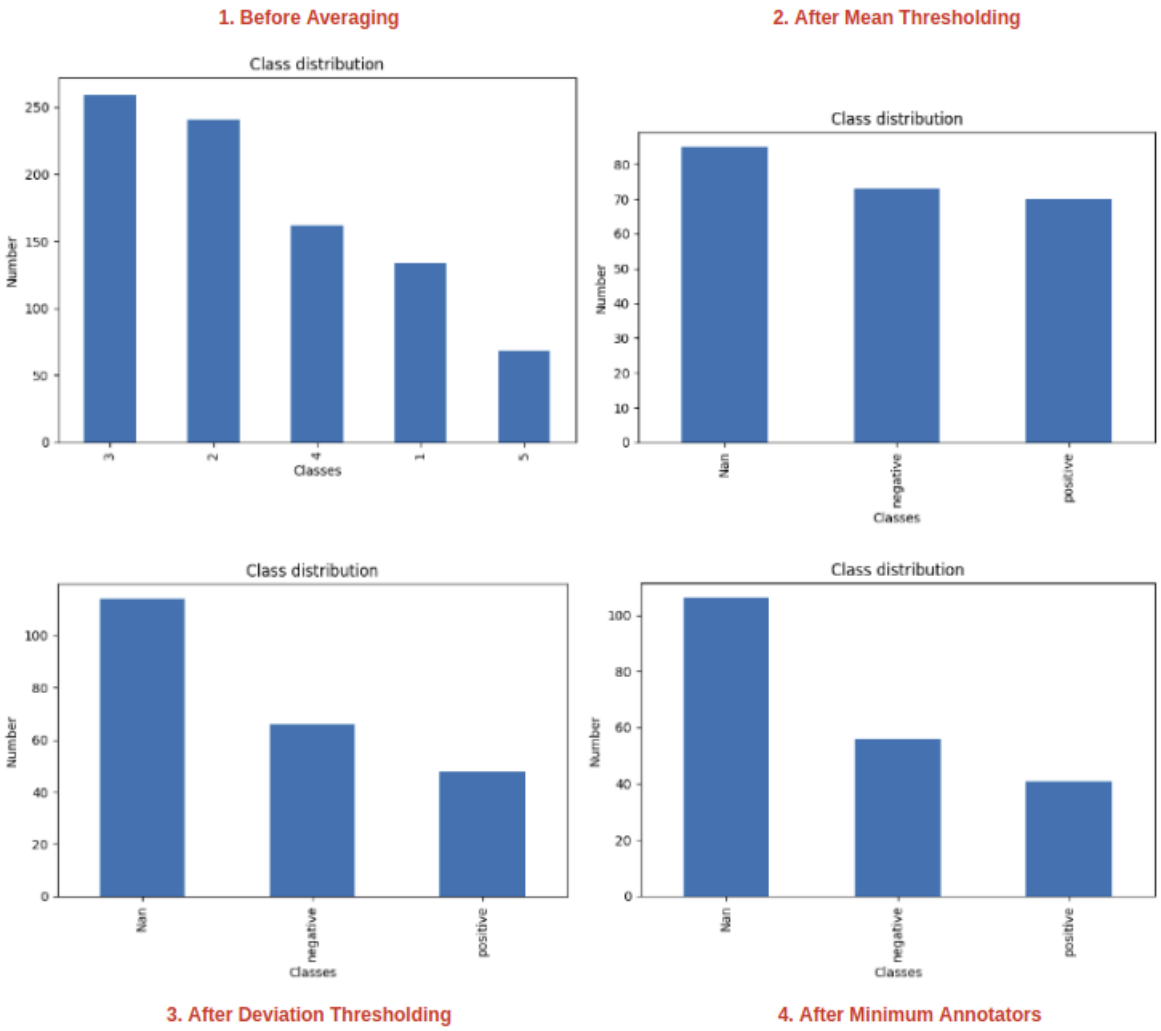


Figure 6.3: Samples Distribution in Classes | Male Expressiveness

It is noted that in the above results, for females, one annotator was rejected who had an average disagreement of 1.03, while for males, three annotators were rejected who had disagreement values of 1.5, 0.79 and 0.73 respectively. These disagreements were high compared to the other annotators and so the annotations of these users were considered invalid.

Regarding the threshold of the deviation (σ), it is observed that the same value has always been chosen (0.75), as it was considered appropriate to keep it in strict limits (without increasing), so that only the samples in which the users / annotators agreed to a reasonable degree, survive.

It is also observed that for females the mean value threshold for the positive class is stricter than for males. This is because the male samples were fewer and in order to create a dataset that has sufficient samples and is balanced in both classes, the range of values accepted for the positive class would have to be expanded. In practice, a looser threshold means that the samples may not be so much separable in the two classes, because one class is closer to the other.

So finally, for **female** we end up with 52 samples in the positive class and with 53 in the negative. Listening to these final classifications, 5 more samples were removed from the

positive class which did not appear to be good enough in terms of expressiveness. Therefore, the final number of samples is **47** samples in class **positive** and **53** in class **negative**.

Respectively for **male**, we end up with 41 samples in the positive class and 50 in the negative class. Finally, 2 additional samples were removed from the negative class, which did not seem to be bad enough in terms of expressiveness. Thus, we ended up with **41** samples in the class **positive** and **48** in the class **negative**.

6.5.2 Ease of Following

	Ease of Following			
	Female		Male	
	Positive	Negative	Positve	Negative
Mean Threshlod	$\mu >= 4.5$	$\mu <= 2.8$	$\mu >= 3.6$	$\mu <= 2.2$
Number of samples after Mean Thresholding	57	82	66	49
Deviation Threshold	$\sigma < 0.75$	$\sigma < 0.75$	$\sigma < 0.75$	$\sigma < 0.75$
Number of samples after Deviation Thresholding	54	60	49	38
Number of samples after Minimum Annotators	44	54	43	33
Average Disagreement	0.57		0.58	

Table 6.2: Definition of Ease of Following Dataset

The following pictures show in more detail the distribution of the number of samples per class, 1. before applying the averaging of annotations, 2. after applying the mean value threshold, 3. after applying the deviation threshold and finally 4. after applying the minimum number of annotators.

Female

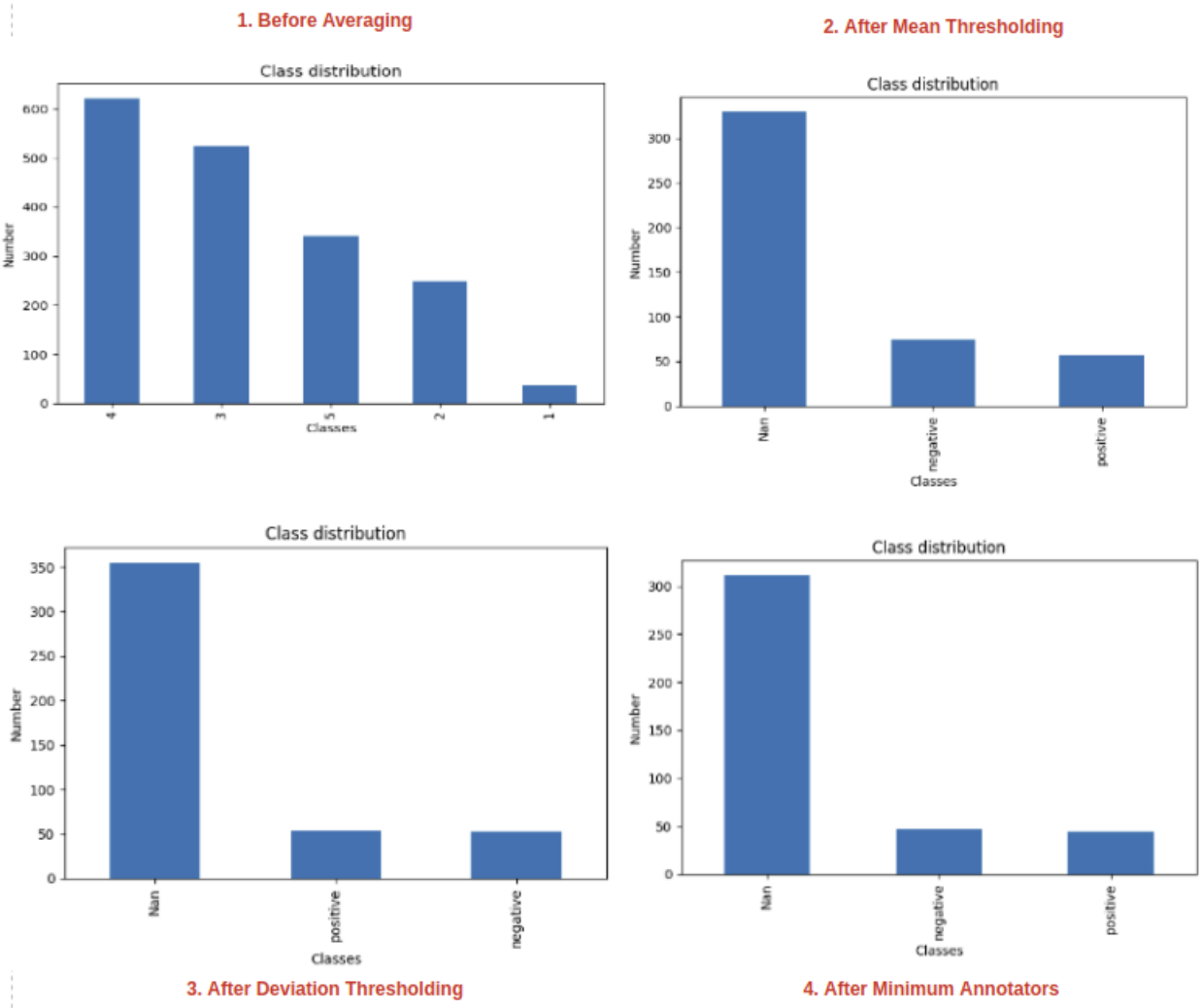


Figure 6.4: Samples Distribution in Classes | Female Ease of Following

Male

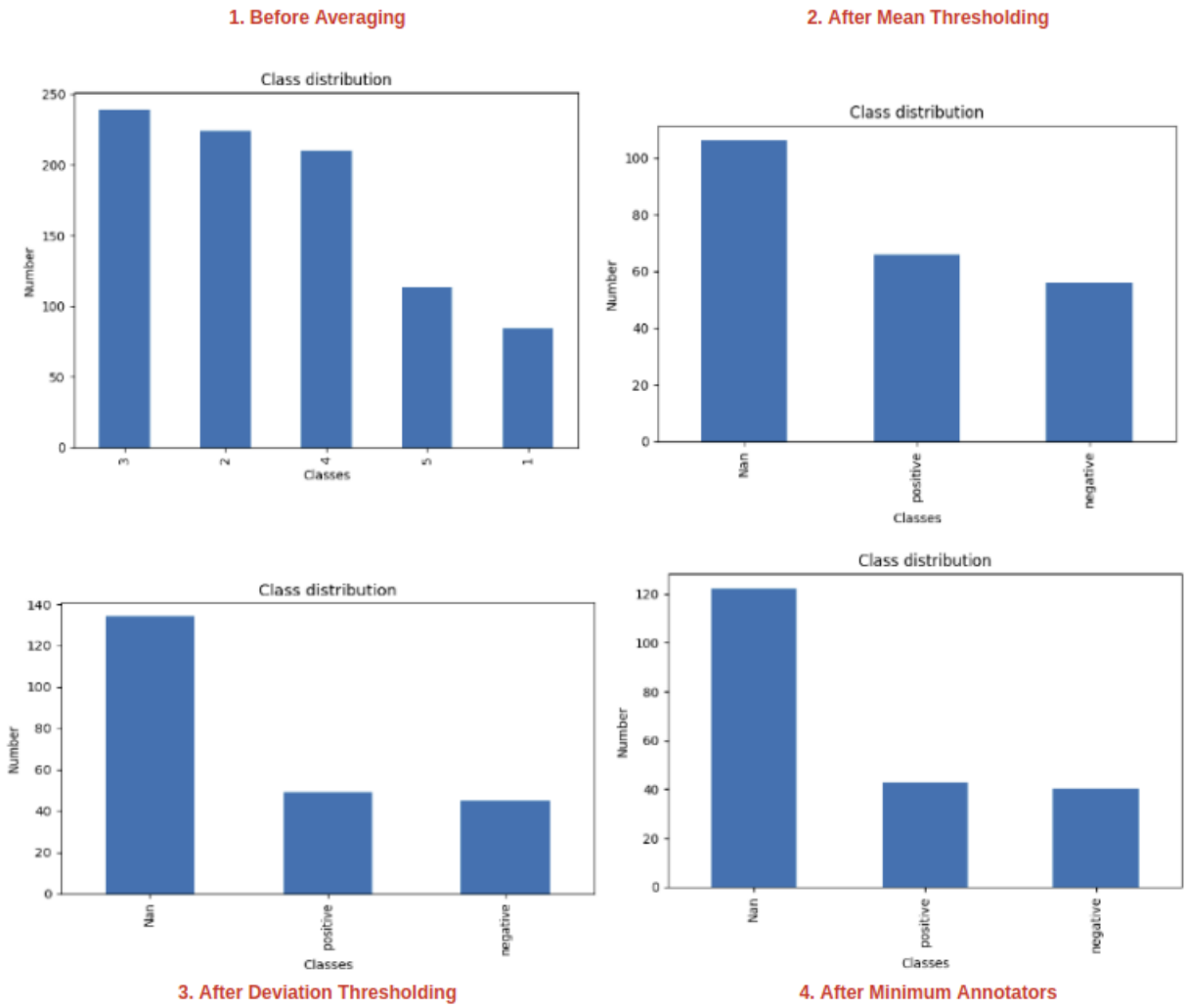


Figure 6.5: Samples Distribution in Classes | Male Ease of Following

It is noted that in the above results, for females, one annotator was rejected who had an average disagreement of 1.08, while for males two annotators were rejected who had disagreement values of 0.76 and 0.75.

The deviation threshold (σ) remained stable in this case as well (0.75), while in terms of the mean value thresholds, there is a much more relaxed value in the positive class of males than in the positive class of females, while the negative class of males carries a slightly stricter value compared to the negative class of females. This shows us that women have more positive signs in terms of ease of following.

So finally, for **females** we end up with 44 samples in the positive class and 54 in the negative. Listening to these final classifications, 6 more samples were removed from the negative class which did not appear to be bad enough in terms of ease of following. Therefore, the final number of samples is **44** samples in the **positive** class and **48** in the **negative** class.

For **males**, the samples were retained exactly as shown by the above restrictions, ie **43** samples for the **positive** class and **33** for the **negative** class.

6.5.3 Enjoyment

	Enjoyment			
	Female		Male	
	Positive	Negative	Positve	Negative
Mean Threshlod	μ>=4.0	μ<=2.4	μ>=3.2	μ<=2.0
Number of samples after Mean Thresholding	71	86	72	64
Deviation Threshold	σ<0.75	σ<0.75	σ<0.75	σ<0.75
Number of samples after Deviation Thresholding	67	73	54	58
Number of samples after Minimum Annotators	60	61	50	51
Average Disagreement	0.5		0.54	

Table 6.3: Definition of Enjoyment Dataset

The following pictures show in more detail the distribution of the number of samples per class, 1. before applying the averaging of annotations, 2. after applying the mean value threshold, 3. after applying the deviation threshold and finally 4. after applying the minimum number of annotators.

Female

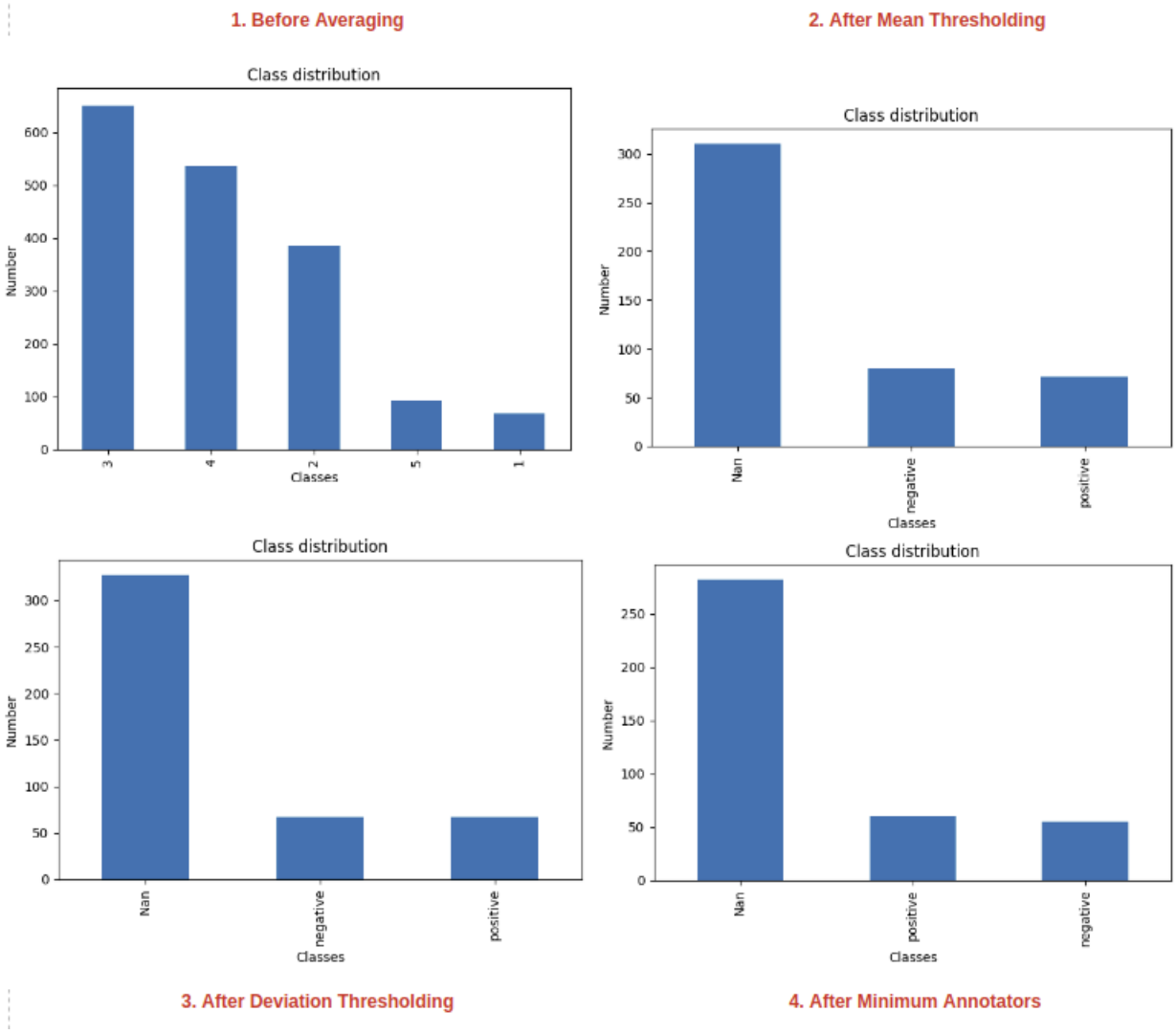


Figure 6.6: Samples Distribution in Classes | Female Enjoyment

Male

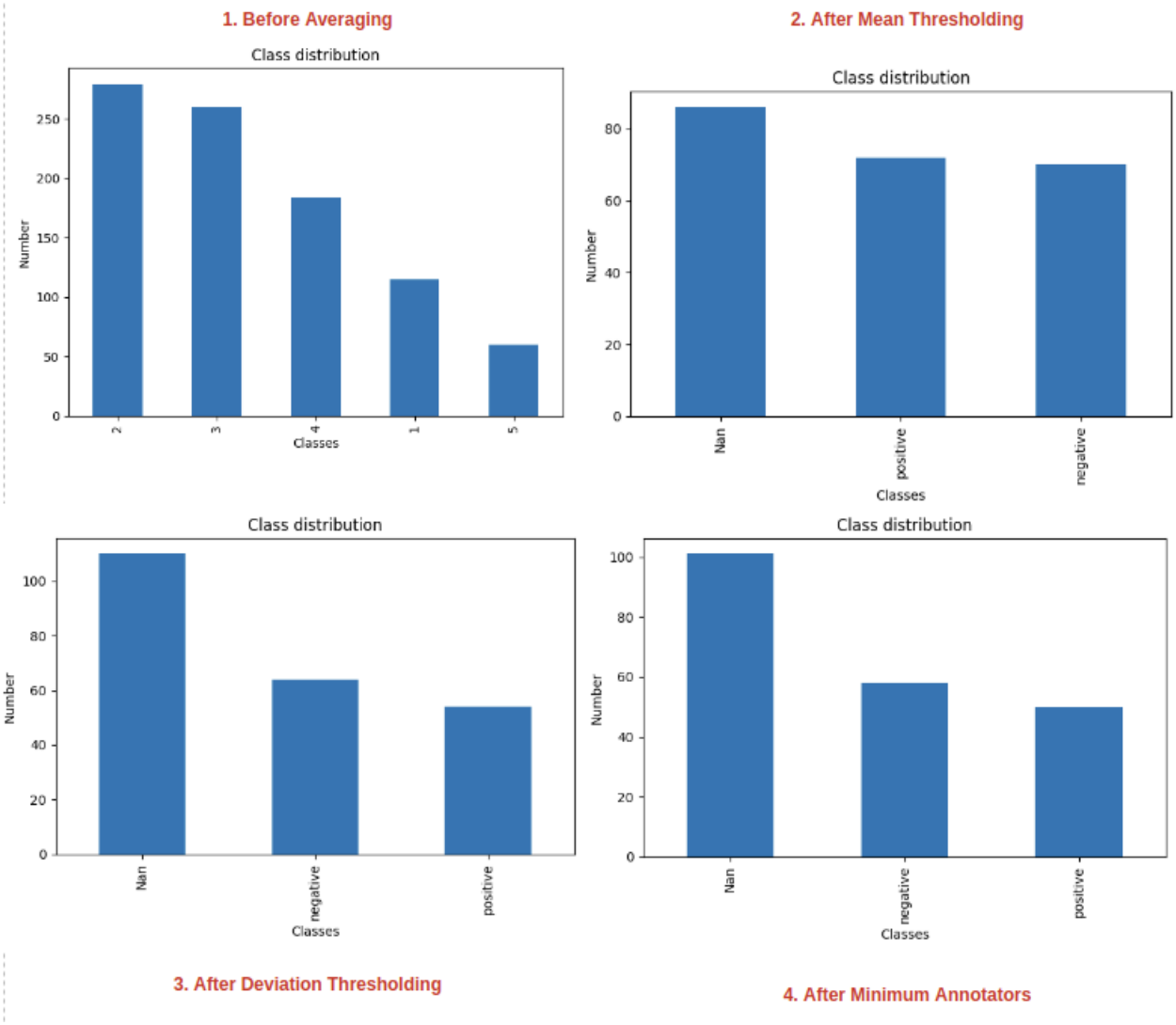


Figure 6.7: Samples Distribution in Classes | Male Enjoyment

It is noted that in the above results, for females, one annotator was rejected who had an average disagreement of 0.79 and for males one who had a disagreement of 0.72.

The deviation threshold remains stable here at 0.75, while in terms of the mean value thresholds, the same behavior is observed as in the previous task of ease of following.

Finally we end up for the **female** with **60** samples in the **positive** class and **61** in the **negative** class, while for the **male** with **50** samples in the **positive** class and **51** in the **negative** class.

6.5.4 Free Text

Two others small indicative datasets were created, which include only samples that are independent of a reference text, that is, only speeches that resulted from answering questions and not from a predefined text that the speaker read.

The reason why these datasets were produced is to examine possible differences in the part of the system that concerns the text analysis. It is obvious that the textual features extracted from different, free-form, speeches, accumulate much more information than the features extracted from speeches of the same reference text.

These datasets were created on the two tasks of expressiveness and enjoyment.

For the expressiveness task, females and males were combined and finally only the samples of them, which come from free speech, were selected. The reason why the sexes were pooled is the lack of data.

Finally, out of the 88 (47 female, 41 male) samples of the positive class and the 101 (53 female, 48 male) of the negative class, the ones that remained as free speech were **22** samples in the class **positive** and **16** in class **negative**. Of course, it is obvious that this number of samples is small for the training of a model, nevertheless it can be used as an indication.

Respectively, from the task of enjoyment, the samples from female and male that come from free speech were reunited.

Finally, out of the 110 (60 female, 50 male) samples of the positive class and the 112 (61 female, 51 male) of the negative class, the ones that remained as free speech were **34** samples in the class **positive** and **21** in class **negative**.

The following table shows the distribution of samples by class, for these two new datasets:

	Free Text	
	Positive	**Negative**
Expressiveness	22	16
Enjoyment	34	21

Table 6.4: Samples Distribution of Free Text Datasets

CHAPTER 7

Experiments and Results

7.1 Experiments

For the final recording level experiments, the data sets of expressiveness and enjoyment were selected as they were the ones that had the least discrepancy in the labels.

For each of the tasks, 8 different types of experiments were performed, in terms of the type of features used. These types are listed below:

1. **Meta Audio (MA)**:
   In these experiments, the sound features derived from the segment-level classifiers (see 4.1.6) were used, as well as the high-level sound features. The total number of these features is 20 and they are analyzed in chapter 5.1. It is noted that for the extraction of features which are fed to the trained models-segment classifiers, mid-term windows of 3 second with step 3 and short-term windows of 0.05 seconds, with step 0.05 were used, the same that also happened during the training of these models.

2. **Text (T)**:
   In these experiments, text features derived from segment-level classifiers (see 4.2.5) were used, as well as high-level text features. The total number of these features is 22 and they are analyzed in chapter 5.2. It is noted that for the export of features which are fed to the trained models-segment classifiers, windows of predefined duration and specifically 3-second windows were used as segments. The best segmentation method that could be applied to the text might have been segmentation per sentence, as each sentence is independent and can contain its own information. Nevertheless, it was observed that in the texts produced by the google speech to text api, punctuation marks and sentences are rarely found. Therefore, the segmentation chosen here was the time windows.

3. **Low Level Audio (LLA)**:
   Low level sound features were used in these experiments. More specifically, as low-level we define those features used for segment classifiers, ie those presented in chapter 4.1.3. This is 136 features per midterm window (average and standard deviation of characteristics in short-term windows). Of course, since the final classifiers are related to the total sound/speech, the long-term averaging of the characteristics in the midterm windows is calculated and we end up with 136 recording level characteristics.

The windows used in this case were 3-second windows with a step of 3 and short-term windows of 0.05 seconds with a step of 0.05.

The reason why such experiments are performed is because comparing their results with the results of the step 1 experiments presented above will be very interesting and important. It is of particular importance to see whether it is beneficial to use segment classifiers and high-level features over direct use of low-level features for speech analysis.

4. **MA + T**:
   In these experiments the features of steps 1 and 2 are used together, i.e. joined together. In essence, these kinds of experiments combine the two systems (audio and text) (3.3), So that the overall information can be used by the recording level classifiers to assess speech quality.

5. **Early Fusion of MA and LLA**:
   In these experiments, the features of steps 1 and 3 are collapsed / combined. That is, a feature vector is used that contains both the high-level features and the low-level audio features.

6. **Late Fusion of MA and LLA**:
   In this case the features of steps 1 and 3 are used again, but this time they are not merged into a common feature vector by fusion (early fusion). What is done is to use a separate classifier for type 1 features and a separate classifier for type 3 features and then aggregate their decisions. More specifically, the average is calculated from the probabilities produced by the two classifiers and then the class whose average probability is higher is selected.

7. **Early Fusion of MA + T and LLA**:
   In this case, the features of steps 1,2 and 3 or otherwise of steps 3 and 4 are combined into a common feature vector by fusion (early fusion).

8. **Late Fusion of MA + T and LLA**:
   In this case the features of step 4 are used to extract a decision from one classifier, the features of step 3 are used to extract a decision from another classifier and then the two decisions are added together by taking the average probabilities in each class and selecting the class with the highest average probability.

For model training, gridsearch is used to find the best parameters. The following parameter values are considered for the svm rbf algorithm: 'gamma': ['auto', 'scale'] and 'C': [0.001, 0.01, 0.5, 1.0, 5.0, 10.0, 20.0]. For the Gaussian Naive Bayes algorithm the following parameter values are considered: 'var smoothing': 100 equal spaced points in space [1, 10^{-9}] and finally for the logistic regression algorithm the following: 'penalty': [' l1 ',' l2 '],' C ': 20 equal points in space $[10^{-4}, 10^{4}]$ and 'solver ': [' liblinear '].

In gridsearch a standardscaler is applied to normalize the features, while the Leave One Group Out is used as a cross validation method. This method uses groups of samples that are already separated and in each iteration, trains the model in all groups except one, which is used for testing. The groups we use are divided based on the speaker. That is, each group contains samples/speeches of a single speaker. Thus, the total number of splits in cross validation is equal to the number of speakers, since in each split a new group-speaker is used as a test set. In this way, samples are tested by speakers that the model has never seen before during training. In other words, we manage to implement a speaker independent evaluation, which gives us more realistic and correct results, as we have said before, speech

is directly dependent on the speaker and if the model is tested on familiar speakers, it can give misleading results.

To select the best model, ie the parameters that give the best performance, we calculate the aggregated confusion matrix. The aggregated confusion matrix is defined as the confusion matrix that has resulted from all cross validation splits, ie by summing the classifications from each test set. The f1 macro metric of this confusion matrix is then calculated and the parameters that gave this maximum metric are finally selected.

Finally, an additional step is applied for the calculation of the metric auc (area under the curve), as well as the graphical representation of the roc curve. Specifically, after the optimal parameters have been found, a cross validation is performed with these parameters constant, using again leave one group out, which in our case means leave one speaker out, and all the predicted probabilities are collected from each test set. From these concentrated probabilities, the metric auc is calculated and the roc curve is created.

The following tables show in detail some experiments performed using different algorithms-classifiers, from the performances of which, the best algorithm was finally selected on a case by case basis. It is noted that as f1, cm and auc are presented the aggregated f1, confusion matrix and auc that have resulted from the aggregation of test sets, as described above. Also, in the confusion matrix, the negative class is denoted as 0 and the positive class as 1.

Female Expressiveness						
	SVM RBF		Gaussian Naive Bayes		Logistic Regression	
	MA	LLA	MA	LLA	MA	LLA
f1	53.9	51	57.8	56.3	**67**	**71**
auc	41.2	47.6	54.3	49.9	**71**	**77.3**
cm	0 1 0 29 24 1 22 25	0 1 0 24 29 1 20 27	0 1 0 21 32 1 9 38	0 1 0 22 31 1 12 35	**0 1 0 34 19 1 14 33**	**0 1 0 37 16 1 13 34**

Table 7.1: Female Expressiveness

Male Expressiveness						
	SVM RBF		Gaussian Naive Bayes		Logistic Regression	
	MA	LLA	MA	LLA	MA	LLA
f1	60.6	66.3	**74.7**	54.7	69.5	**76.4**
auc	57	63.5	**69.4**	67.6	74.1	**71**
cm	0 1 0 29 19 1 16 25	0 1 0 31 17 1 13 28	**0 1 0 40 8 1 14 27**	0 1 0 33 15 1 24 17	0 1 0 34 14 1 13 28	**0 1 0 35 13 1 8 33**

Table 7.2: Male Expressiveness

Female Enjoyment						
	SVM RBF		Gaussian Naive Bayes		Logistic Regression	
	MA	LLA	MA	LLA	MA	LLA
f1	56.2	54.7	50.6	44.5	**52**	**55.8**
auc	19.6	22.9	44.5	31.9	**44.3**	**57**
cm	0 1 0 21 40 1 10 50	0 1 0 23 38 1 15 45	0 1 0 19 42 1 15 45	0 1 0 14 47 1 16 44	**0 1 0 29 32 1 26 43**	**0 1 0 51 10 1 40 20**

Table 7.3: Female Enjoyment

Male Enjoyment						
	SVM RBF		Gaussian Naive Bayes		Logistic Regression	
	MA	LLA	MA	LLA	MA	LLA
f1	65.3	64.3	53.9	65.1	**59.3**	**64.3**
auc	18.9	17.6	56.3	61.5	**57.8**	**70.4**
cm	0 1 0 35 16 1 19 31	0 1 0 35 16 1 20 30	0 1 0 33 18 1 28 22	0 1 0 43 8 1 26 24	**0 1 0 32 19 1 22 28**	**0 1 0 30 21 1 15 35**

Table 7.4: Male Enjoyment

From the above results, we will select the models that give the highest performance in a combination of metric f1 and auc.

Based on the above, we observe that the logistic regression algorithm is the one that gives, in most cases (tasks / gender), the best performances, both for MA and LLA experiments. It is noted that exceptionally for the male expressiveness task for the case of MA, the Gaussian Naive Bayes algorithm is chosen, as it may give a slightly lower auc compared to the logistic regression, but has a better f1. It seems that other algorithms like svm rbf are very complex for the problem we are going to solve and so we need a simpler conventional algorithm. Therefore, for the next more detailed experiments we choose to go with the algorithms that performed better depending on the task and the type of features (in all logistic regression, except for the male expressiveness MA which will be Gaussian Naive Bayes).

It is noted that for the late fusion methods, gridsearch was followed again with leave one speaker out, separately for the MA or MA + T features and separately for the LLA features. For both of the above feature cases, the best parameters for the logistic regression algorithm were found. Then, a cross validation was followed with leave one speaker out, where in each split both the MA (or MA + T) model and the LLA model were trained in the same train set, with the best parameters found previously. In the same test set, predictions were also made by both models. From these predictions / probabilities, of the two models, the average for each sample was calculated and then the class with the maximum average prediction probability was selected. All these final predictions / classes, from all the test sets, were put together into a common (aggregated) confusion matrix. Also, the average probabilities, from all the test sets, were combined together into one vector, from which the aggregated value auc (area under the curve) was derived, as well as the roc curve graphs.

Below are presented in detail the metrics f1, area under the curve and confusion matrix, calculated in the way described above, for each task and for each method (type of features) separately:

Female Expressiveness								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	67	56	71	65	67	69	66	68
auc	71	36.8	77.3	65.6	77	76	74.6	74.9
cm	0 1 0 34 19 1 14 33	0 1 0 45 8 1 32 15	0 1 0 37 16 1 13 34	0 1 0 34 19 1 16 31	0 1 0 35 18 1 15 32	0 1 0 35 18 1 13 34	0 1 0 35 18 1 16 31	0 1 0 35 18 1 14 33

Table 7.5: Female Expressiveness

Male Expressiveness								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	74.7	49.6	76.4	62.9	73.7	-	66.3	-
auc	69.4	40.6	71	71	74.9	-	78.9	-
cm	0 1 0 40 8 1 14 27	0 1 0 32 16 1 27 14	0 1 0 35 13 1 8 33	0 1 0 40 8 1 23 18	0 1 0 39 9 1 14 27	-	0 1 0 42 6 1 22 19	-

Table 7.6: Male Expressiveness

The reason why dashes are presented in the table above in early fusion experiments is because different model algorithms are used for MA and LLA respectively, so it does not make sense to combine these features using a common algorithm.

Female Enjoyment								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	52	63.3	55.8	57.1	51	55.8	56.4	56.3
auc	44.3	64.8	57	50.7	43.8	57	51.2	62.3
cm	0 1 0 29 32 1 26 34	0 1 0 40 21 1 23 26	0 1 0 51 10 1 40 20	0 1 0 29 32 1 19 40	0 1 0 27 34 1 25 35	0 1 0 51 10 1 40 20	0 1 0 29 32 1 20 39	0 1 0 51 10 1 39 20

Table 7.7: Female Enjoyment

Male Enjoyment								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	58.2	59.9	64.3	61.9	59.3	64.3	66.3	65.1
auc	57.4	48.3	70.4	59.9	64.3	70.5	74	66.1
cm	0 1 0 33 18 1 24 26	0 1 0 19 32 1 6 44	0 1 0 30 21 1 15 35	0 1 0 37 14 1 24 26	0 1 0 32 19 1 22 28	0 1 0 30 21 1 15 35	0 1 0 36 15 1 19 31	0 1 0 34 17 1 21 29

Table 7.8: Male Enjoyment

Free Text Expressiveness								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	67.6	67.6	80.9	84	81.3	78.4	86.6	80.9
auc	74.7	64.8	86.7	91.2	86.7	84.4	93.5	86.1
cm	0 1 0 10 6 1 6 16	0 1 0 10 6 1 6 16	0 1 0 12 4 1 3 19	0 1 0 14 2 1 4 18	0 1 0 13 3 1 4 18	0 1 0 12 4 1 4 18	0 1 0 14 2 1 3 19	0 1 0 12 4 1 3 19

Table 7.9: Free Text Expressiveness

Free Text Enjoyment								
	MA	T	LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
f1	62.1	57.7	47.7	81.1	65.3	65.3	55.3	70.1
auc	71.2	57.3	55.7	86.1	65.6	61.8	74.5	66.5
cm	0 1 0 12 9 1 11 23	0 1 0 13 8 1 14 20	0 1 0 6 15 1 11 23	0 1 0 17 4 1 6 28	0 1 0 12 9 1 9 25	0 1 0 11 10 1 10 24	0 1 0 9 12 1 11 23	0 1 0 14 7 1 11 23

Table 7.10: Free Text Enjoyment

7.2 Final Results

Here will be a summary of the results we saw in the above tables, for all tasks together, using only the metric auc, to make it easier to comment. We choose to rely on this metric rather than f1, as the data is scarce and f1 can be easily changed, even if one of the samples is sorted differently. On the contrary, the metric auc is a more realistic value that shows us how the models go at different operation points, ie different probability thresholds. Therefore, with this metric we have a picture of the performance of the classifiers regardless of the specific operation point.

	Individual Modalities			Fusion Methods				
	Meta Audio MA	Text T	Low Level Audio LLA	MA + T	MA and LLA Late Fusion	MA and LLA Early Fusion	MA + T and LLA Late Fusion	MA + T and LLA Early Fusion
Female Expressiveness	71	37	**77**	66	**77**	76	75	75
Male Expressiveness	69	41	**71**	71	75	-	**79**	-
Female Enjoyment	44	**65**	57	51	44	57	51	**62**
Male Enjoyment	57	48	**70**	60	64	70	**74**	66
Free Text Expressiveness	75	65	**87**	91	87	84	**93**	86
Free Text Enjoyment	**71**	57	56	**86**	66	62	75	67

Table 7.11: Final Performances (auc metric)

7.2.1 Commentary

The table above shows the individual experiments on the left, ie the experiments that use one type of feature, without any merging, while on the right shows the experiments with all the methods of merging features presented above. In both cases, it is outlined in bold letters, the greater performance.

Initially, we see that in all cases (rows), the best fusion method has increased the performance compared to the best individual method or in the worst case has kept it the same. The only exception is Female Enjoyment in which there is a slight deterioration of 3 points (65-62 = -3), which however can be considered negligible. On the contrary, in Male Expressiveness we see that the performance has increased by 8 points, with the use of fusion, in Male Enjoyment and in Free Text Expressiveness by 6 points, while in Free Text Enjoyment it has increased by 15 points, a very important difference.

Another important observation is that the combination of Meta Audio with Text features (MA + T), seems to have increased performance compared to individual MA and individual T, in the tasks of Male Expressiveness and Male Enjoyment. This improvement becomes much more evident in free text tasks, where we see the MA + T combination, having increased performance by 16 in Expressiveness (compared to individual MA) and by 15 in Enjoyement (compared to individual MA). This fact shows that the textual information can significantly help the model to distinguish the two classes (negative, positive), since with a simple combination of the two types of features (audio + text), the performance is automatically increased. Of course, this phenomenon was expected to occur mainly in experiments involving free text, as then the text differs from sample to sample and is no longer predetermined.

In addition, it is observed in most cases (4 out of 6 cases), the combination of information from all features (meta audio, text and LLA) to be the one that gives the best result. Also, most of the time (4 out of 6 times), late fusion does better than early fusion. This shows that late fusion introduces a normalization factor, which is missing from the early fusion method. For example, the number of features is something that can affect their utilization. So, if the LLA has far more features than the MA, it probably means that some good features from the MA will be "lost" in the "crowd" and will not be used properly. On the contrary, such an event can be avoided with late fusion, where the two types of features are utilized by separate models and decisions are combined at the end (late).

One last remark that could be mentioned is that in the Free Text Enjoyment task, the MA + T features do better than any other combination containing the LLA information. Thus, it seems that low level audio features (LLA) such as MFCCs or Chroma do not provide information on the quality of a speech, unlike high level audio and text features, which are derived from internal emotion classifiers, which seem to be very important in speech analysis.

7.2.2 Graphs

Here are the graphs of the final results, which include the aggregated confusion matrix, the performance per class, the precision vs recall and the roc curve for the positive class, giving a more complete view.

Female Expressiveness

In this task the best result seemed to be in the method **MA and LLA Late Fusion** (same redult as in the individual LLA). Therefore, the graphical representations from this method are presented:

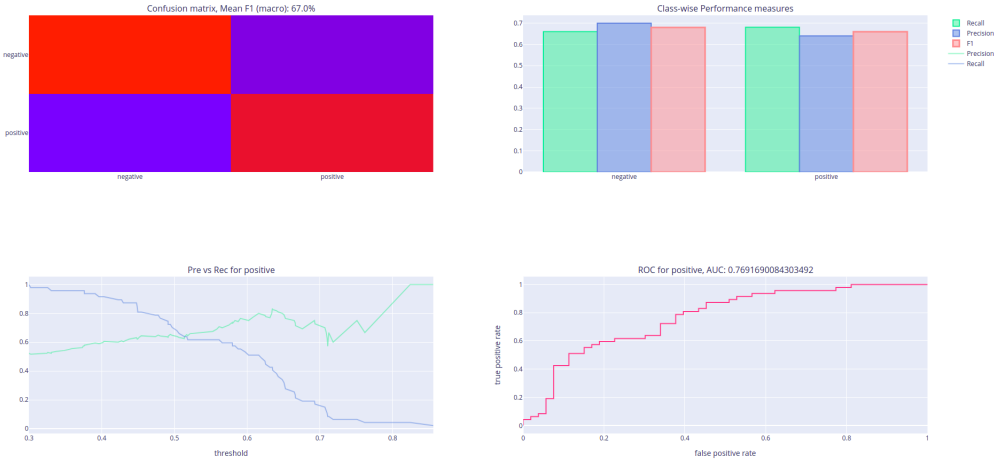


Figure 7.1: Female Expressiveness | Best Results

Male Expressiveness

In this task the best result seemed to be in the method **MA + T and LLA Late Fusion**. Therefore, the graphical representations from this method are presented:

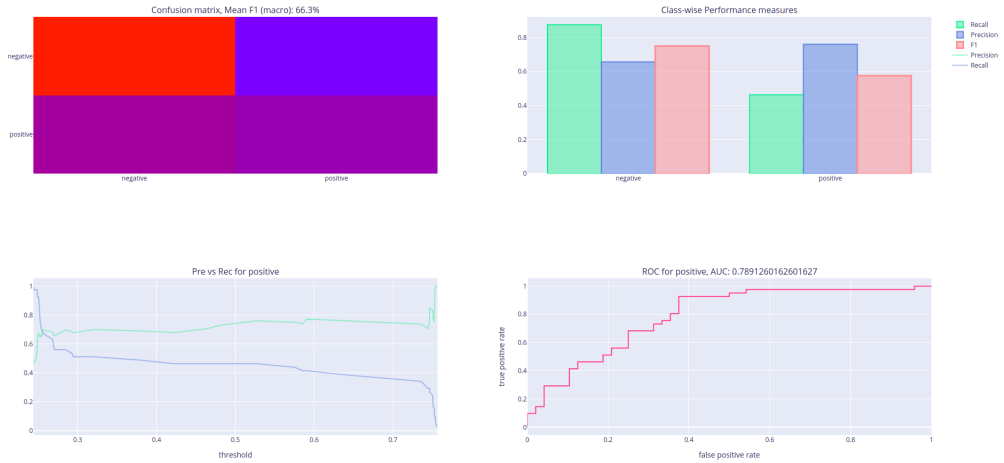


Figure 7.2: Male Expressiveness | Best Results

Female Enjoyment

In this task the best result seemed to be in the **T** method. Therefore, the graphical representations from this method are presented:

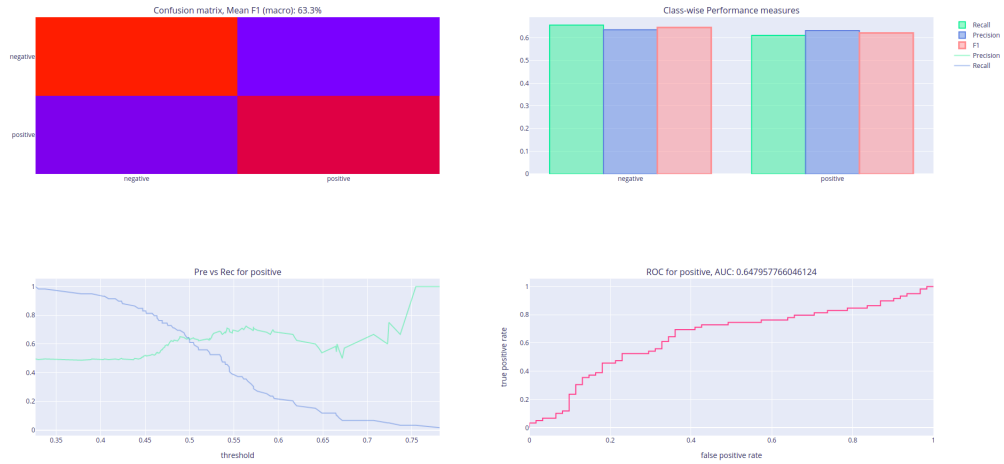


Figure 7.3: Female Enjoyment | Best Results

Male Enjoyment

In this task the best result seemed to be in the method **MA + T and LLA Late Fusion**. Therefore, the graphical representations from this method are presented:

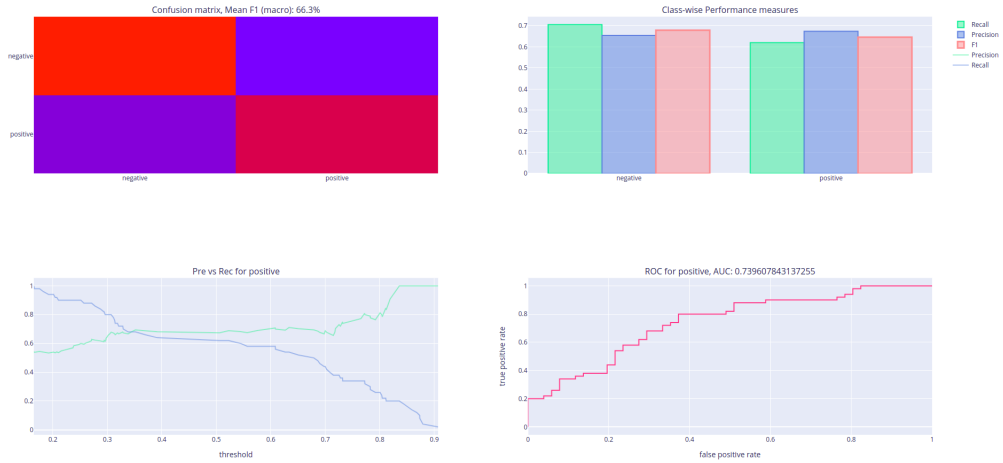


Figure 7.4: Male Enjoyment | Best Results

Free Text Expressiveness

In this task the best result seemed to be in the method **MA + T and LLA Late Fusion**. Therefore, the graphical representations from this method are presented:

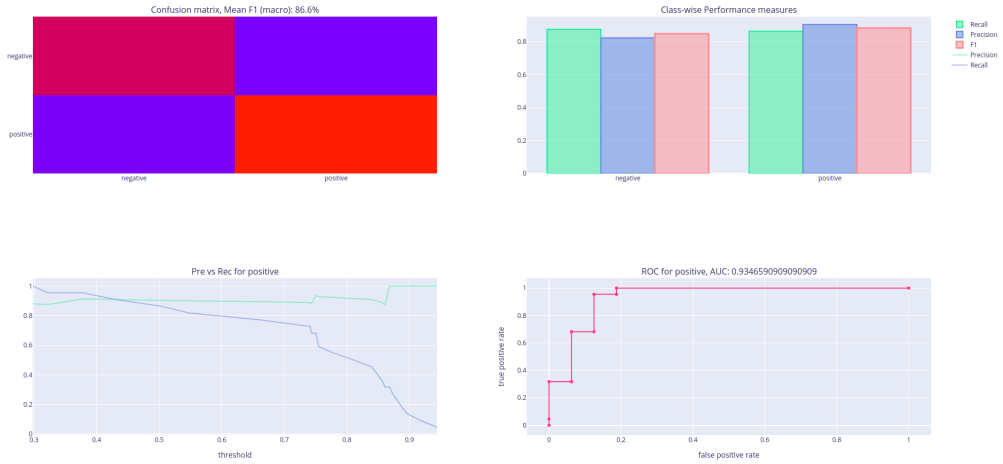


Figure 7.5: Free Text Expressiveness | Best Results

Free Text Enjoyment

In this task the best result seemed to be in the **MA + T** method. Therefore, the graphical representations from this method are presented:

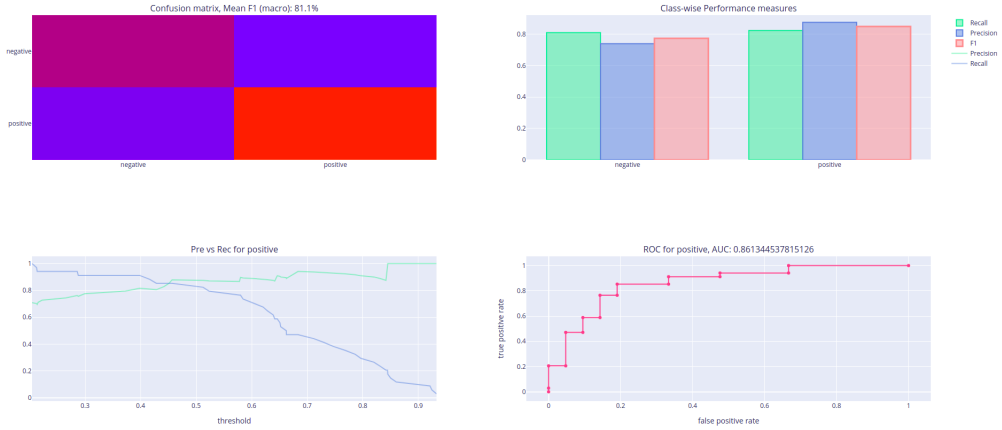


Figure 7.6: Free Text Enjoyment | Best Results

CHAPTER 8

Conclusion

8.1 System Summary

In summary, a speech analytics system was created. This system was divided into two levels: segment-level analysis and recording-level analysis. These two levels applied to both audio and textual speech information. For the first level, 3 classifiers were trained on axes of emotion, arousal and valence respectively. The features used for these classifiers are spectral and temporal characteristics in the case of audio and bert embeddings in the case of text. The decisions made by these classifiers are segmental, ie they concern each window / section separately. For the analysis at the recording level, the aforementioned decisions were summed up using averaging per class and thus created the aggregated posteriors, which in practice are the probabilities or percentages that indicate whether the total speech belongs to each class (emotion, arousal and valence). These aggregated decisions, along with the addition of some high level features of the overall speech, were used to train the final quality assessment models.

For the final models, a data collection and annotation process was followed, as well as an agreement of the annotations. Two types of speeches were collected, those based on predefined text and those with free text. These were then annotated by different commentators/annotators on three axes: expressiveness, ease of following and enjoyment. This was followed by the agreement of the annotations with three restrictions: the mean value, the deviation and the number of annotators per sample. From this agreement emerged two classes for each of the three axes: negative and positive, while some statistics were produced such as the mean disagreement. In the end, only the axes of expressiveness and enjoyment were chosen to be retained, as the axis of ease of following had the greatest average disagreement of annotators.

Finally, experiments were performed, with the above data, on 3 individual types of features: the recording level audio features as described above (MA), the corresponding recording level text features (T) and finally some low level audio features (LLA) which are the same spectral and temporal sound features used for segment-level classifiers, but this time averaged for the entire speech (long-term averaging). In addition, experiments were performed with combinations of the above types using either early fusion, ie feature vector fusion or late fusion, ie subsequent fusion of decisions from separate models.

8.2 Results Discussion

The system developed to assess speech quality seems to be functional. The results we saw in the previous chapter are quite satisfactory, while the proposed fusion methods seem to be an innovation that greatly improves the result.

It is important to mention that this system can be used for any speech analysis process, apart from assessing its quality, such as the recognition of learning difficulties, dyslexia or autism, the retrieval of signals from telephone exchanges, etc.

Despite the fact that the analysis of speech in general, but also the speech emotion recognition in particular, is a difficult problem, as it depends on many factors such as circumstances, speaker, gender, data set, etc. , the system developed in the present dissertation and the experiments performed, were carried out in as unbiased and independent conditions as possible. Thus, the results presented are realistic.

It is noteworthy that high-level features, such as those extracted from segment classifiers that are related to emotion, usually make more sense and provide more information than simpler, low-level features such as MFCCs. In addition, these features enable explainability, as they are able to explain the result. For example, if a speaker has low expressiveness, this is probably related to the high percentage of "low arousal" he / she showed in his / her speech. Thus, there can be a justification of the results and in this way, the speaker understands what he needs to improve or in another case what he is doing well and gives him good speech quality. This can not happen with low-level features that are not explanatory and immediately exploitable.

Finally, an important part of the research is the creation of data set from speeches, as well as their annotation, which was carried out in an impartial and objective manner. Making a supervised data set (containing labels) is an equally important part of data science and machine learning.

8.3 Implementation

The open source code of the system of this dissertation, ie a speech analytics python tool for speech quality assessment, can be found at github [106].

8.4 Future Work

Future extensions and changes that could be made to the system of the present diploma thesis are:

- Domain adaptation in terms of the tasks of emotion, arousal and valence. These models, trained in our system, used open source data without having come into contact with data similar to those ultimately used to assess speech quality (those that resulted from data collection). For this reason, an adjustment can be made in the field of the specific data, either by introducing in the training process of these models, some samples similar to the ones we finally use, or by following an unsupervised method, where we enter the information of this data at the feature level, so that the normalization of the features takes into account the actual range. In this way phenomena such as representing a sample of positive expressiveness as "anger" in terms of emotion or as "negative" in terms of valence, due to misinterpretation of its features, could be avoided.

- The introduction of more powerful and robust models-segment classifiers, which can increase performance. We refer to deep learning methods, such as CNNs, LSTMs and

Transformers. In addition, it is important to study how the performance of long-term recording-level models can be affected by improving the performance of segment-level classifiers. For example, if an emotion classifier increases its performance by 5% at the segment-level, how much better will the models perform at the recording-level.

- The use of neural network (LSTM or other sequential model) in the recording-level part, ie after the posteriors of the segment classifiers. In this way we could take advantage of the sequential information, as we work with sections / windows that have a time sequence.
- The extension of recording-level experiments to more data annotated, for a better generalization of the result, as few data were retained after aggregation / agreement of annotations.
- The application of transfer learning from unsupervised temporal models. In this case, knowledge acquired while solving problems which are using sequential information and corresponding models (eg LSTM) in unmarked data / samples, is stored and this knowledge is applied to the specific problem we try to solve.
- The application of learning methods that take into account the annotation confidence, ie the validity of the annotation of each sample, which can also be defined as the agreement between the annotations.

Bibliography

[1] Public Speaking, Wikipedia - The Free Encyclopedia.
Available at: https://en.wikipedia.org/wiki/Public_speaking, Access 02/03/2021

[2] Designing an Automated Assessment of Public Speaking Skills Using Multimodal Cues. Available at: https://files.eric.ed.gov/fulltext/EJ1126866.pdf, Access 02/03/2021

[3] Artificial Intelligence, Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Artificial_intelligence, Access 02/03/2021

[4] Moral Machine Platform. Available at: https://www.moralmachine.net/, Access 03/03/2021

[5] Bonnefon, Jean-François; Shariff, Azim; Rahwan, Iyad (2016-06-24). "The social dilemma of autonomous vehicles". Science. 352 (6293): 1573–1576. arXiv: https://arxiv.org/abs/1510.03346. Bibcode: https://ui.adsabs.harvard.edu/abs/2016Sci...352.1573B. doi: https://science.sciencemag.org/content/352/6293/1573. ISSN: https://www.worldcat.org/title/science/oclc/1156775655. PMID: https://pubmed.ncbi.nlm.nih.gov/27339987/

[6] Attitudes towards the impact of digitisation and automation on daily life, Special Eurobarometer 460, Available at: https://ec.europa.eu/commfrontoffice/publicopinion/index.cfm/Survey/getSurveyDetail/instruments/SPECIAL/surveyKy/2160, Access 02/03/2021

[7] Self-driving cars will have to decide who should live and who should die. Here's who humans would kill., washingtonpost.com,Carolyn Y. Johnson (2018), Available at: https://www.washingtonpost.com/science/2018/10/24/self-driving-cars-will-have-decide-who-should-live-who-should-die-heres-who-humans-would-kill/, Access 02/03/2021

[8] Machine Learning Definition, Nvidia. Available at: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/, Access 03/03/2021

[9] Machine Learning Definition, Stanford . Available at: https://www.coursera.org/learn/machine-learning, Access 03/03/2021

[10] Machine Learning Definition, McKinsey & Co. Available at: https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/an-executives-guide-to-machine-learning#, Access 03/03/2021

[11] Machine Learning Definition, University of Washington. Available at: https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf, Access 03/03/2021

[12] Machine Learning Definition, Carnegie Mellon University. Available at: http://www.cs.cmu.edu/~tom/pubs/MachineLearning.pdf, Access 03/03/2021

[13] The Hundred-Page Machine Learning Book, by Andriy Burkov, Quebec City, Canada (2019), p.7 ,ISBN: 978-1999579517,doi: 10.1080/15228053.2020.1766224

[14] "What is the kernel trick? Why is it important?",Medium.com, Grace Zhang (2018). Available at: https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d, Access 10/03/2021

[15] "Introduction to Machine Learning Algorithms: Linear Regression", towardsdatascience.com, Rohith Gandhi (2018). Available at: https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a, Access 10/03/2021

[16] "Logistic Regression — Detailed Overview", towardsdatascience.com, Saishruthi Swaminathan (2018). Available at: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc, Access 10/03/2021

[17] "Support Vector Machine — Introduction to Machine Learning Algorithms", towardsdatascience.com, Rohith Gandhi (2018). Available at: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47, Access 10/03/2021

[18] "Machine Learning Basics with the K-Nearest Neighbors Algorithm", towardsdatascience.com, Onel Harrison (2018). Available at: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761, Access 10/03/2021

[19] "A loss function analysis for classification methods in text categorization" ,F. Li and Y. Yang (2003), in ICML 03-063, pp. 472–479.

[20] "Decision Tree Algorithm, Explained" ,By Nagesh Singh Chauhan (2020), Available at: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html, Access 12/03/2021

[21] Bootstrap aggregating , Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Bootstrap_aggregating, Access 12/03/2021

[22] Boosting (machine learning) , Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Boosting_(machine_learning), Access 12/03/2021

[23] XGBoost: A Scalable Tree Boosting System, Tianqi Chen, Carlos Guestrin (2016), arXiv : 1603.02754v3, Available at: https://arxiv.org/pdf/1603.02754.pdf, Access 25/04/2021

[24] "Underfitting and Overfitting in Machine Learning", Datascience Foundation/,by Mayank Tripathi (2020). Available at: https://datascience.foundation/sciencewhitepaper/underfitting-and-overfitting-in-machine-learning, Access 16/03/2021

[25] "Machine Learning Crash Course", Google Developers. Available at: https://developers.google.com/machine-learning/crash-course/, Access 18/03/2021

[26] Receiver operating characteristic , Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Receiver_operating_characteristic, Access 18/03/2021

[27] Confusion matrix , Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Confusion_matrix, Access 18/03/2021

[28] Hyperparameter optimization , Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Hyperparameter_optimization, Access 18/03/2021

[29] "3.1. Cross-validation: evaluating estimator performance", scikit-learn 0.24.1, python package. Available at: https://scikit-learn.org/stable/modules/cross_validation.html, Access 18/03/2021

[30] Cross-validation (statistics) , Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Cross-validation_(statistics), Access 18/03/2021

[31] Activation Functions in Neural Networks , towardsdatascience.com, SAGAR SHARMA (2017), Available at: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, Access 19/03/2021

[32] A Comprehensive Tutorial to learn Convolutional Neural Networks from Scratch (deeplearning.ai Course #4), Analytics Vidhya, PULKIT SHARMA (2018), Available at: https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/, Access 27/04/2021

[33] Convolutional Neural Network, towardsdatascience.com, (2019), Available at: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529, Access 27/04/2021

[34] Applied Deep Learning - Part 4: Convolutional Neural Networks, towardsdatascience.com, Arden Dertat (2017), Available at: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2, Access 27/04/2021

[35] Understanding LSTM Networks, colah' s blog (2015), Available at: http://colah.github.io/posts/2015-08-Understanding-LSTMs/, Access 15/05/2021

[36] Long Short-term Memory , Hochreiter & Schmidhuber (1997), DOI:10.1162/neco.1997.9.8.1735, Available at: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory, Access 15/05/2021

[37] File:Long Short-Term Memory.svg, Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/File:Long_Short-Term_Memory.svg, Access 15/05/2021

[38] How Transformers Work The Neural Network used by Open AI and DeepMind, towardsdatascience, Giuliano Giacaglia (2019), Available at: https://towardsdatascience.com/transformers-141e32e69591, Access 14/05/2021

[39] Attention Is All You Need, Ashish Vaswani, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin (2017), arXiv: 1706.03762, Available at: , Access 14/05/2021

[40] Cloud Speech-to-Text , Google Cloud, Available at: https://cloud.google.com/speech-to-text,

[41] Introduction to Audio Analysis A MATLAB Approach, by Theodoros Giannakopoulos | Aggelos Pikrakis, (2014) , pp.24-27,60-92 ,ISBN: 978-0080993881

[42] Introduction to Speech Processing, Aalto University Wiki,by Tom Bäckström (2021), chapter: 2.b, Available at: https://wiki.aalto.fi/display/ITSP/Introduction+to+Speech+Processing, Access 21/03/2021

[43] An Introduction to the Discrete Fourier Transform, Technical Article - All About Circuits, by Steve Arar (2017), Available at: https://www.allaboutcircuits.com/technical-articles/an-introduction-to-the-discrete-fourier-transform/, Access 01/04/2021

[44] Discrete Fourier transform, Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Discrete_Fourier_transform, Access 01/04/2021

[45] From Frequency to Quefrency: A History of the Cepstrum, by Alan V. Oppenheim and Ronald W. Schafer (2004). Available at: https://www.fceia.unr.edu.ar/prodivoz/Oppenheim_Schafer_2004.pdf, Access 03/04/2021

[46] Cepstrum, Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Cepstrum, Access 03/04/2021

[47] Mel scale, Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Mel_scale, Access 03/04/2021

[48] Mel Frequency Cepstral Coefficient (MFCC) tutorial , practicalcryptography.com. Available at: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/#eqn2, Access 03/04/2021

[49] The dummy's guide to MFCC , medium.com, by Pratheeksha Nair (2018), Available at: https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd, Access 03/04/2021

[50] Pitch (music), Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Pitch_(music), Access 04/04/2021

[51] Octave, Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Octave, Access 04/04/2021

[52] Pitch class, Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Pitch_class, Access 04/04/2021

[53] Equal temperament, Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Equal_temperament, Access 04/04/2021

[54] Chroma and tonality, Juan Pablo Bello, EL9173 Selected Topics in Signal Processing: Audio Content Analysis NYU Poly. Available at: https://s18798.pcdn.co/jpbello/wp-content/uploads/sites/1691/2018/01/6-tonality.pdf, Access 04/04/2021

[55] Chroma feature, Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Chroma_feature, Access 04/04/2021

[56] pyAudioAnalysis: An Open-Source PythonLibrary for Audio Signal Analysis, by Theodoros Giannakopoulos (2015). DOI:10.1371/journal.pone.0144610 Available at: researchgate: https://www.researchgate.net/publication/286637817_pyAudioAnalysis_An_Open-Source_Python_Library_for_Audio_Signal_Analysis, pypi.org: https://pypi.org/project/pyAudioAnalysis/, github: https://github.com/tyiannak/pyAudioAnalysis

[57] EMOVO Corpus: an Italian Emotional Speech Database, by Giovanni Costantini, Iacopo Iadarola, Andrea Paoloni, Massimiliano Todisco1, Department of Electronic Engineering, University of Rome "Tor Vergata", Rome, Italy. (2009) Available at: https://core.ac.uk/download/pdf/53857389.pdf

[58] A database of German emotional speech, by F. Burkhardt1, A. Paeschke, M. Rolfes, W. Sendlmeier, B. Weiss (2005). Available at: https://www.researchgate.net/publication/221491017_A_database_of_German_emotional_speech

[59] Surrey Audio-Visual Expressed Emotion (SAVEE) database, by Philip J. B. Jackson, Sanaul Haq, University of Surrey (2011). Available at: http://personal.ee.surrey.ac.uk/Personal/P.Jackson/SAVEE/

[60] The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English, by Steven R. Livingstone, Frank A. Russo (2018). DOI:10.1371/journal.pone.0196391 Available at: https://www.researchgate.net/publication/325187111_The_Ryerson_Audio-Visual_Database_of_Emotional_Speech_and_Song_RAVDESS_A_dynamic_multimodal_set_of_facial_and_vocal_expressions_in_North_American_English

[61] IEMOCAP: Interactive emotional dyadic motion capture database, by Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh,Emily Mower, Samuel Kim, Jeannette N. Chang, Sungbok Lee and Shrikanth S. Narayanan (2007). Available at: https://ecs.utdallas.edu/research/researchlabs/msp-lab/publications/Busso_2008_5.pdf

[62] Theodoros Giannakopoulos, Principal Researcher NCSR Demokritos. Available at: https://scholar.google.com/citations?user=BeIoqhwAAAAJ&hl=el

[63] Seemo: A Computational Approach to See Emotions, by Zhe Liu, Anbang Xu, Yufan Guo, Jalal U. Mahmud, Haibin Liu, Rama Akkiraju, IBM Research - Almaden San Jose, CA, USA (2018), DOI:10.1145/3173574.3173938. Available at: https://www.researchgate.net/publication/324664655_Seemo_A_Computational_Approach_to_See_Emotions

[64] Richard J Gerrig, Philip G Zimbardo, Andrew J Campbell, Steven R Cumming and Fiona J Wilkes.(2015). Psychology and life. Pearson Higher Education.

[65] Andrew Ortony and Terence J Turner. 1990. What's basic about basic emotions? Psychological review, 97, 3 (1990), 315.

[66] Paul Ekman. 1992. An argument for basic emotions. Cognition & emotion, 6, 3-4 (1992), 169-200

[67] Grekow, J. (2018). From Content-Based Music Emotion Recognition to Emotion Maps of Musical Pieces. Studies in Computational Intelligence, Vol. 747. Springer, Warsaw, Poland.

[68] Wilhelm Max Wundt. 1907. Outlines of psychology.W. Engelmann

[69] Ja Ressel. 1980. A circumplex model of affect. J. Personality and Social Psychology, 39 (1980), 1161-1178

[70] James A Russell and Lisa Feldman Barrett. 1999. Core affect, prototypical emotional episodes, and other things called emotion: dissecting the elephant. Journal of personality and social psychology, 76, 5 (1999), 805

[71] Margaret M Bradley, Mark K Greenwald, Margaret C Petry and Peter J Lang. 1992. Remembering pictures: Pleasure and arousal in memory. Journal of experimental psychology: Learning, Memory, and Cognition, 18, 2 (1992), 379-390.

[72] David Watson and Auke Tellegen. 1985. Toward a consensual structure of mood. Psychological bulletin, 98, 2 (1985), 219.

[73] Zhao, J., Mao, X., Chen, L. (2019).Speech emotion recognition using deep 1D 2D CNN LSTM networks. Biomedical Signal Processing and Control, 47, 312-323, DOI: 10.1016/j.bspc.2018.08.035.

[74] Fayek, H. M., Lech, M., Cavedon, L. (2017). Evaluating deep learning architectures for Speech Emotion Recognition. Neural Networks, 92, 60-68, DOI: 10.1016/j.neunet.2017.02.013.

[75] Yoon, S., Byun, S., Jung, K. (2018, December). Multimodal speech emotion recognition using audio and text. In 2018 IEEE Spoken Language Technology Workshop (SLT) (pp. 112-118). IEEE, DOI:10.1109/SLT.2018.8639583.

[76] D. Kollias, et. al.: "Deep Affect Prediction in-the-wild: Aff-Wild Database and Challenge, Deep Architectures, and Beyond". International Journal of Computer Vision (IJCV), 2019, arXiv:1804.10938v5.

[77] D. Kollias, et. al.: "Analysing Affective Behavior in the First ABAW 2020 Competition". IEEE FG, 2020, arXiv:2001.11409v2.

[78] D. Kollias, et. al.: "Analysing Affective Behavior in the second ABAW2 Competition", 2021, arXiv:2106.15318v2.

[79] D. Kollias, et. al.: "Distribution Matching for Heterogeneous Multi-Task Learning: a Large-scale Face Study", 2021, arXiv:2105.03790v1.

[80] D. Kollias, S. Zafeiriou: "Expression, Affect, Action Unit Recognition: Aff-Wild2, Multi-Task Learning and ArcFace". BMVC, 2019, arXiv:1910.04855v1.

[81] Deep Audio Features, python library. Available at: https://github.com/tyiannak/deep_audio_features

[82] SMOTETomek, python library attribute. Available at: http://glemaitre.github.io/imbalanced-learn/generated/imblearn.combine.SMOTETomek.html

[83] StandardScaler, python library attribute. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[84] VarianceThreshold, python library attribute. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html

[85] PCA, python library attribute. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[86] RepeatedStratifiedKFold, python library attribute. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RepeatedStratifiedKFold.html

[87] Word embedding, Wikipedia - The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Word_embedding, Access 14/04/2021

[88] Creating text features with bag-of-words, n-grams, parts-of-speach and more, UC Business Analytics R Programming Guide. Available at: http://uc-r.github.io/creating-text-features#bag, Access 14/04/2021

[89] Word2Vec and FastText Word Embedding with Gensim , towardsdatascience.com, Kung-Hsiang, Huang (Steeve) (2018), Available at: https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c, Access 14/04/2021

[90] An Intuitive Understanding of Word Embeddings: From Count Vectors to Word2Vec, Analytics Vidhya, (2017). Available at: https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/, Access 15/04/2021

[91] Word2Vec and FastText Word Embedding with Gensim, towardsdatascience.com, Kung-Hsiang, Huang (2018). Available at: https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c, Access 15/04/2021

[92] Co-occurrence matrix & Singular Value Decomposition(SVD), medium.com, Apar Garg (2020). Available at: https://medium.com/@apargarg99/co-occurrence-matrix-singular-value-decomposition-svd-31b3d3deb305, Access 15/04/2021

[93] Intuitive Guide to Understanding GloVe Embeddings, towardsdatascience.com, Thushan Ganegedara (2019). Available at: https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010, Access 15/04/2021

[94] GloVe: Global Vectors for Word Representation, Jeffrey Pennington, Richard Socher, Christopher D. Manning, Computer Science Department, Stanford University. Available at: https://nlp.stanford.edu/pubs/glove.pdf, Access 16/04/2021

[95] Enriching Word Vectors with Subword Information, Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov (2017), arXiv: 1607.04606v2. Available at: https://arxiv.org/pdf/1607.04606v2.pdf, Access 21/04/2021

[96] A Visual Guide to FastText Word Embeddings, Amit Chaudhary (2020). Available at: https://amitness.com/2020/06/fasttext-embeddings/, Access 21/04/2021

[97] Optimize Computational Efficiency of Skip-Gram with Negative Sampling, Pythonic Excursions (2019). Available at: https://aegis4048.github.io/optimize_computational_efficiency_of_skip-gram_with_negative_sampling#eq-27, Access 21/04/2021

[98] Wiki word vectors, fasttext.cc. Available at: https://fasttext.cc/docs/en/pretrained-vectors.html, Access 23/01/2021

[99] FNV Hash. Available at: http://www.isthe.com/chongo/tech/comp/fnv/, Access 21/04/2021

[100] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (2019), arXiv:1810.04805v2, Available at: https://arxiv.org/pdf/1810.04805.pdf, Access 16/05/2021

[101] BERT Explained: State of the art language model for NLP, towardsdatascience.com, Rani Horev (2018). Available at: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270, Access 16/05/2021

[102] NLP: Contextualized word embeddings from BERT, towardsdatascience.com, Andreas Pogiatzis (2019). Available at: https://towardsdatascience.com/nlp-extract-contextualized-word-embeddings-from-bert-keras-tf-67ef29f60a7b, Access 16/05/2021

[103] The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), Jay Alammar, Available at: http://jalammar.github.io/illustrated-bert/, Access 16/05/2021

[104] RoboCOP: A Robotic Coach for Oral Presentations, H.Trinh, R.Asadi, D.Edge, T.Bickmore, Available at: https://dl.acm.org/doi/epdf/10.1145/3090092, Access 15/06/2021

[105] Effects of Good Speaking Techniques on Audience Engagement, Keith Curtis, Gareth J. F. Jones, Nick Campbell (2015) Available at: https://dl.acm.org/doi/10.1145/2818346.2820766, Access 15/06/2021

[106] Implementation of Speech Analytics Python Tool for Speech Quality Assessment, Sofia Eleftheriou, Theodoros Giannakopoulos, Panagiotis Koromilas. Available at: https://github.com/tyiannak/readys