National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Speech and Language Processing Group

Depression Detection from Social Media Posts

Diploma Thesis

of

ILIAS TRIANTAFYLLOPOULOS

Supervisor: Alexandros Potamianos
Associate Professor, NTUA

Athens, March 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Speech and Language Processing Group

Depression Detection from Social Media Posts

Diploma Thesis

of

ILIAS TRIANTAFYLLOPOULOS

Supervisor: Alexandros Potamianos
Associate Professor, NTUA

Approved by the examination committee on 2nd March 2022.

(Signature)	*(Signature)*	*(Signature)*
....	....	....
Alexandros Potamianos Associate Professor, NTUA	Constantinos Tzafestas Associate Professor, NTUA	Giorgos Stamou Professor, NTUA

Athens, March 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Speech and Language Processing Group

(Signature)

....
Ilias Triantafyllopoulos

Electrical & Computer
Engineer

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the National Technical University of Athens.

Περίληψη

Στην παρούσα διπλωματική εργασία μελετάμε την κατάθλιψη και πώς μπορεί να ανιχνευθεί μέσω μοντέλων Βαθιάς Μηχανικής Μάθησης. Η κατάθλιψη είναι ένα από τα πιο κοινά ψυχικά νοσήματα που ταλανίζει πολλούς ανθρώπους σε όλο τον κόσμο. Ευτυχώς, υπάρχει πληθώρα φαρμακευτικών αγωγών που δύνανται να αντιμετωπίσουν την κατάθλιψη. Ωστόσο, η διάγνωση της κρίνεται ακόμη επίπονη, καθώς τα συμπτώματα της κατάθλιψης είναι πολλαπλά και αρκετά κοινά με άλλες ψυχικές ασθένειες και, επομένως, δυσχεραίνουν την διαδικασία διάγνωσης.

Αρχικά, ακολουθώντας μια εκτενή αναφορά στην μέχρι τώρα έρευνα που έχει πραγματοποιηθεί γύρω από την διαδικασία ανίχνευσης κατάθλιψης στον τομέα της Επεξεργασίας Φυσικής Γλώσσας, επιλέγουμε να πραγματοποιήσουμε την δική μας έρευνα σε αυτή την εργασία χρησιμοποιώντας ένα σύνολο δεδομένων που προέρχεται από την διαδικτυακή πλατφόρμα Reddit. Το σύνολο δεδομένων μας, μάς αναγκάζει να στραφούμε σε μοντέλα δύο επιπέδων, καθώς θα χρειαστούμε πρώτα μια επεξεργασία των δεδομένων σε επίπεδο δημοσίευσης και στη συνέχεια σε επίπεδο χρήστη. Η πρώτη μας ενέργεια είναι να ανιχνεύσουμε διάφορες εκδηλώσεις της ανθρώπινης συμπεριφοράς και να τις συνδυάσουμε κατάλληλα με τις ¨με-συμφραζόμενα’ αναπαραστάσεις που εξάγονται από το Bert. Χρησιμοποιούμε κατάλληλους ανιχνευτές για το συναίσθημα, την ειρωνεία και τα χαρακτηριστικά της προσωπικότητας για την κάθε δημοσίευση του χρήστη.

Έπειτα, επιλέγουμε να εστιάσουμε στον ανιχνευτή συναισθήματος και να προσπαθήσουμε να βελτιώσουμε την γενικότερη επίδοση της αρχιτεκτονικής μας. Αναπτύσσουμε διαφορετικές μεθόδους ενσωμάτωσης της πληροφορίας που λαμβάνουμε από τον ανιχτευτή στις αναπαράστασεις που εξάγονται από το Bert. Συγκεντρωτικά, αναλύουμε τρεις μεθόδους που πραγματοποιούνται σε επίπεδο χρήστη, όπου συνδυάζουμε δύο διαφορετικές αναπαραστάσεις του χρήστη, και τρεις μεθόδους σε επίπεδο δημοσίευσης, όπου χρησιμοποιούμε τον μηχανισμό Προσοχής κατάλληλα για να συνδυάσουμε τις δύο αναπαραστάσεις της κάθε δημοσίευσης και συνολικά να πάρουμε μια αναπαράσταση για τον χρήστη. Ύστερα από μία επιπλέον ανάλυση μας στο σύνολο δεδομένων, καταλήγουμε σε δύο σημαντικά χαρακτηριστικά που μπορούν να βοηθήσουν περαιτέρω στην επίδοση του μοντέλου μας, η χρήση υβριστικού και ηθικού λεξιλογίου. Εισάγουμε κατάλληλα αυτά τα χαρακτηριστικά στο μοντέλο μας και καταφέρνουμε μια βελτίωση της τάξης του 5% σε σχέση με το μοντέλο χωρίς αυτά και 15% σε σχέση με τα μέχρι πρότινος καλύτερα μοντέλα.

Τέλος, επεκτείνουμε τις ιδέες και την αρχιτεκτονική μας σε δύο ακόμα σύνολα δεδομένων: ένα ακόμα σχετιζόμενο με την κατάθλιψη και ένα που αφορά στην ανίχνευση δημοσιεύσεων που περιέχουν σημάδια άγχους. Και τα δύο σύνολα δεδομένων αναγκάζουν την αρχιτεκτονική μας να ¨κατέβει’ ένα επίπεδο, προκειμένου να εφαρμοστεί, έχοντας σε αυτές τις περιπτώσεις ένα επίπεδο λέξεων και ένα επίπεδο δημοσιεύσεων. Η ικανότητα γενίκευσης του μοντέλου μας, καθώς και η χρήση ενός ανιχνευτή συναισθήματος και η εισαγωγή επιπλέον χαρακτηριστικών, σχετιζόμενων με το υβριστικό και ηθικό λεξιλόγιο, κρίνονται αποτελεσματικά και στις δύο περιπτώσεις.

Λέξεις Κλειδιά

Βαθιά Μάθηση, Ανίχνευση Κατάθλιψης, Επεξεργασίας Φυσικής Γλώσσας, Bert, Ανιχνευτής Συναισθήματος, Μέθοδοι Ενσωμάτωσης Πληροφορίας, Υβριστικό Λεξιλόγιο, Ηθικό Λεξιλόγιο, Ανίχνευση Άγχους

Abstract

In this Diploma Thesis, we study Depression and how to detect it through Deep Learning models. Depression is one of the most common mental health illnesses that afflict many people around the world. Fortunately, there is a plethora of medical treatments that can treat depression. However, its diagnosis is still considered difficult, as the symptoms of depression are multiple and quite common with other mental illnesses and therefore, they complicate the process of diagnosis.

Firstly, following an extensive analysis to the research done so far on the task of depression detection in the field of Natural Language Processing, we choose to conduct our own research in this task using a set of data derived from the online platform Reddit. In particular, a set of users has been collected, along with their posts on various platform threads and they have been classified as depressive or healthy. Our dataset forces us to switch to two-levels models, as we will need to first process data at a post-level and then at a user-level. Our initial action is to detect various aspects of human behavior and to combine them appropriately with the contextualised representations extracted from Bert. We use appropriate detectors for emotion, irony, and personality traits for each user' post.

Next, we choose to focus on the emotion detector and try to improve the overall performance of our architecture, following observations of the performance of our models. We develop different methods of integrating the information we receive from the detector into the representations extracted from Bert. Collectively, we develop three methods performed at the user-level, where we combine two different representations of the user: one from Bert and one from the emotion detector, and three methods at the post-level, where we use the Attention mechanism appropriately to combine the two representations of each post and create a generalised representation for the user. After an additional analysis of our dataset, we decide to use two handcrafted features that can further help in the performance of our model, the usage of profanity and morality in the vocabulary of the posts. We properly introduce these features in our model and we achieve an improvement of 5% compared to the model without them and 15% compared to the previous best model.

Finally, we extend our ideas and architecture to two more datasets: one related to depression and one that detects stress into posts. Both datasets force our architecture to "go down" a level in order to be implemented, having in these cases a word-level and a post-level. The ability of our model to generalize, as well as the use of an emotion detector and the introduction of additional handcrafted features, related to profanity and morality, are considered effective in both cases.

Keywords

Deep Learning, Depression Detection, Natural Language Processing, Bert, Emotion Detection, Fusion Methods, Profanity, Morality, Stress Detection

To My Family,
Panagiota, Antonios, and Alexandra

Ευχαριστίες

Η εκπόνηση αυτής της διπλωματικής εργασίας αποτελεί ένα σημείο ορόσημο στην ακαδημαϊκή πορεία μου, καθώς μέσω αυτής ολοκληρώνεται ένα μεγάλο κεφάλαιο της ζωής μου.

Πρώτα από όλα, οφείλω ενα αληθινό ευχαριστώ στον επιβλέποντα μου, τον Καθηγητή Αλέξανδρο Ποταμιάνο, που μου έδωσε την ευκαιρία να γίνω μέλος της ομάδας του εργαστηρίου του, μιας ομάδας που μου έδινε διαρκώς νέα ερεθίσματα γνώσης και μάθησης. Χωρίς την καθοδήγηση και τις συμβουλές του δεν θα ήταν δυνατή η περάτωση της παρούσας εργασίας. Οφείλω φυσικά και ένα μεγάλο ευχαριστώ σε όλη την ομάδα του και κυρίως στον Γιώργο Παρασκευόπουλο, Ευθύμη Γεωργίου και Στέφανο Αχλάτη που είχαν πάντα την διάθεση να με συμβουλέυσουν, να με κατευθύνουν και να συζητήσουμε εκτενώς τις ιδέες και τα εμπόδια που παρουσιάζονταν.

Στη συνέχεια, θέλω να ευχαριστήσω βαθιά τους γονείς μου Παναγιώτα και Αντώνιο και την αδερφή μου Αλεξάνδρα. Χωρίς την πλήρη στήριξη και την ανιδιοτελή αγάπη τους που λαμβάνω καθ΄ όλη την διάρκεια της ζωής μου, δεν θα τα είχα καταφέρει. Ευχαριστώ που είστε δίπλα μου.

Θα ήθελα, επίσης, να ευχαριστήσω ειλικρινά τους Βασίλη Μπάλα, Κατερίνα Μάμαλη και Ελεάννα Κουλέτου, που ήταν πάντα πρόθυμοι να ακούσουν και να με βοηθήσουν με τις σκέψεις και τις προτάσεις τους στην ολοκλήρωση αυτής της διπλωματικής εργασίας.

Τέλος, φυσικά, δεν μπορώ να παραλείψω να ευχαριστήσω όλους τους φίλους μου που ήταν εκεί για εμένα σε όλες μου τις στιγμές και με στηρίζουν διαρκώς. Μου έχουν χαρίσει μερικά από τα πιο όμορφα συναισθήματα και στιγμές που έχω βιώσει. Για λόγους συντομίας, δεν θα αναφερθώ ονομαστικά, αλλά γνωρίζουν οι ίδιοι πόσο σημαντική είναι για μένα η σχέση μου με τον/την καθένα/καθεμία τους ξεχωριστά.

Ηλίας Τριανταφυλλόπουλος

Αθήνα, Μάρτιος 2022

Contents

List of Figures

List of Tables

Chapter 0

Εκτεταμένη Ελληνική Περίληψη

0.1 Εισαγωγή

Στην παρούσα διπλωματική εργασία ασχολούμαστε με το πρόβλημα της ανίχνευσης κατάθλιψης από δημοσιεύσεις στα μέσα κοινωνικής δικτύωσης. Η κατάθλιψη είναι μία σύνηθης ψυχική διαταραχή που ταλανίζει περίπου 322 εκατομμύρια ανθρώπους παγκοσμίως, το οποίο αντιστοιχεί στο 4.4% του παγκόσμιου πληθυσμού [13]. Ακόμα και αν υπάρχει μία πληθώρα φαρμακευτικών αγωγών για την αντιμετώπιση της κατάθλιψης, η διάγνωση της κρίνεται ακόμα επίπονη, εξαιτίας των πολλαπλών συμπτωμάτων της, τα οποία σε μεγάλο μέρος είναι δυνατό να σχετίζονται με λοιπές ψυχικές ασθένειες. Επιπλέον, ένας τεράστιος αριθμός ανθρώπων χρησιμοποιεί τα μέσα κοινωνικής δικτύωσης στις μέρες μας, τα οποία εμπιστεύονται για να μοιράζονται προσωπικές τους πληροφορίες και να εκφράζονται. Η μελέτη της κατάθλιψης σε αυτή την εργασία αποτελεί μια πολυεπίπεδη διεργασία, η οποία πραγματοποιείται με χρήση μηχανικής μάθησης και νευρωνικών δικτύων. Η ανίχνευση γίνεται σε επίπεδο χρηστών και επομένως θα πρέπει πρώτα να επεξεργαζόμαστε τις δημοσιεύσεις του εκάστοτε χρήστη και μετά μέσω αυτών να επεξεργαζόμαστε τον χρήστη σαν οντότητα. Λαμβάνοντας το σύνολο των δεδομένων μας από το Reddit, αρχικά προτείνουμε μία μελέτη της ανίχνευσης κατάθλιψης που θα εμπεριέχει την ανίχνευση διαφορετικών στοιχείων της ανθρώπινης συμπεριφοράς και συγκεκριμένα την ανίχνευση του συναισθήματος, της ειρωνείας και της προσωπικότητας. Στη συνέχεια και ύστερα από τα συμπεράσματα μας από την παραπάνω αρχιτεκτονική, εστιάζουμε στον ανιχνευτή συναισθήματος και προσπαθούμε να αναπτύξουμε διαφορετικές μεθόδους ενσωμάτωσης της πληροφορίας που λαμβάνουμε από αυτον στο βασικό μας μοντέλο. Τα αποτελέσματα μας δείχνουν ότι η συναισθηματική πληροφορία μπορεί να γίνει σημαντική για το μοντέλο μας και να το βοηθήσει στην εργασία της ανίχνευσης κατάθλιψης, αναπτύσσοντας κατάλληλες μεθόδους. Επιπλέον, διεξάγουμε μία ανάλυση δεδομένων προκειμένου να εξάγουμε χρήσιμα χαρακτηριστικά που μπορούμε να ενσωματώσουμε στο μοντέλο μας χειροκίνητα, ώστε να αυξήσουμε την απόδοση του. Αναλύουμε την σημασία δύο τέτοιων χαρακτηριστικών: το υβριστικό και το ηθικό λεξιλόγιο. Τέλος, επεκτείνουμε τις ιδέες μας σε δύο επιπλέον σύνολα δεδομένων, όπου και επιβεβαιώνουμε τα ευρήματα μας.

0.2 Θεωρητικό Υπόβαθρο

0.2.1 Αναδρομικά Νευρωνικά Δίκτυα

Ένα αναδρομικό νευρωνικό δίκτυο είναι ένας τύπος νευρωνικών δικτύων που εμπεριέχουν εσωτερική μνήμη. Είναι αναδρομικό, καθώς η έξοδος σε κάθε βήμα αντιγράφεται και στέλνεται πίσω στο αναδρομικό δίκτυο και χρησιμοποιείται σαν είσοδος στο επόμενο βήμα. Ο μηχανισμός αυτός είναι πολύ σημαντικός για εφαρμογές, όπου απαιτείται η απομνημόνευση της ιστορίας των προηγούμενων εισόδων και εξόδων. Στο 1 απεικονίζεται μία τέτοια διάταξη. Η κρυφή κατάσταση σε κάθε χρονικό βήμα και η έξοδος υπολογίζονται ως παρακάτω:

$$\mathbf{h_t} = f_h\left(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x_t} + \mathbf{b_h}\right)$$
$$\mathbf{y_t} = f_y\left(\mathbf{W}_{yh}\mathbf{h_t} + \mathbf{b_y}\right)$$

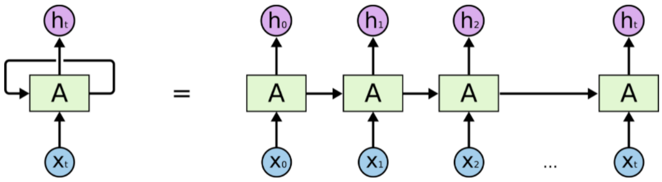


Figure 1. *Ένα ξετυλιγμένο ΑΝΔ, όπου X_t είναι το διάνυσμα εισόδου που περιέχει ακολουθίες χαρακτήρων από μία λέξη και h_t είναι το διάνυσμα εξόδου. Πηγή: colah.github.io*

Στην παρούσα εργασία, χρησιμοποιούμε το **Gated Recurrent Unit (GRU)**, το οποίο αποτελεί ένα παραγόμενο των ΑΝΔ και αντιμετωπίζει εν μέρει το βασικό πρόβλημα αυτών, το "gradient vanishing", προσθέτοντας δύο επιπλέον πύλες, την πύλη Ανανέωσης και την πύλη Αναίρεσης. Η πύλη Ανανέωσης είναι εκείνη που αποφασίζει πόση από την παρελθοντική πληροφορία χρειάζεται να περάσει στο μέλλον. Η πύλη Αναίρεσης πόση από την προηγούμενη πληροφορία θα πρέπει το μοντέλο να ξεχάσει.

0.2.2 Μηχανισμός Προσοχής

Το πέρασμα της πληροφορίας μέσα από μία σειρά αναδρομικών συνδέσεων προκαλούν απώλεια πληροφορίας. Επιπλέον, η ακολουθιακή φύση των ΑΝΔ δυσχαιρένουν τον παράλληλο υπολογισμό. Μία σύγχρονη λύση σε αυτά ήρθε από τον Μηχανισμό της Προσοχής, το οποίο προτάθηκε πρώτα από τους Bahdanau et al. [14]. Το σημαντικότερο στοιχείο του μηχανισμού είναι ότι βοηθάει τα μοντέλα να δώσουν περισσότερη προσοχή σε συγκεκριμένα σημεία κατά την επεξεργασία των δεδομένων.

Αυτο-Προσοχή Η Αυτο-Προσοχή είναι ένας μηχανισμός προσοχής που μπορεί, επίσης, να εφαρμοστεί σε μία πρόταση, όπου δεν υπάρχει κάποια επιπλέον πληροφορία και απεικονίζεται στο 2.

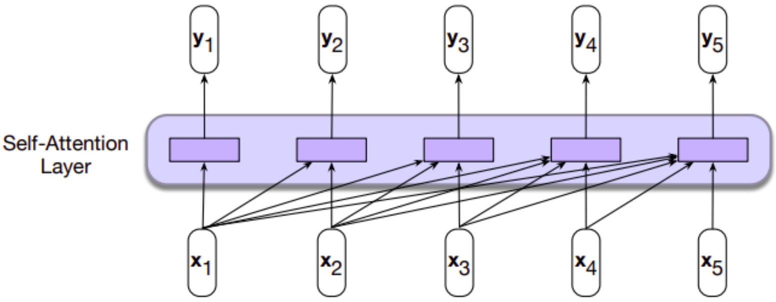


Figure 2. *Η ροή της πληροφορίας σε έναν μηχανισμό Αυτο-Προσοχής. Σε αντίθεση με τα ΑΝΔ, οι υπολογισμοί σε κάθε χρονικό βήμα είναι ανεξάρτητοι όλων των υπολοίπων βημάτων και επομένως μπορούν να γίνουν παράλληλα. Πηγή: [1]*

Κάθε είσοδος x_i μπορεί να παίξει τρεις διαφορετικούς ρόλους στην διαδικασία. Μπορούμε με βάση αυτό, να δώσουμε τους ορισμούς των "query", "key" και "value" που αποτελούν και τις πιο σημαντικές έννοιες του μηχανισμού της προσοχής:

1. **query** είναι το στοιχείο-στόχος της διαδικασίας. Αυτό το στοιχείο συγκρίνεται με κάθε στοιχείο εισόδου που προηγείται αυτού.

2. **key** είναι κάθε στοιχείο εισόδου που προηγείται του στοιχείου-στόχου και συγκρίνεται με αυτό.

3. **value** είναι το στοιχείο που χρησιμοποιείται για τον υπολογισμό της τελικής εξόδου.

Προσοχή Πολλαπλών-Κεφαλών. Η προσοχή Πολλαπλών-Κεφαλών είναι μία άλλη αρχιτεκτονική προσοχής, η οποία αποτελείται από μερικά επίπεδα που τρέχουν παράλληλα. Ο μηχανισμός αυτός επιτρέπει στο μοντέλο να προσέχει την πληροφορία από διαφορετικούς υποχώρους αναπαραστάσεων σε διαφορετικές θέσεις. Έχει αποδειχθεί ότι δουλεύει καλύτερα από την Προσοχής Μίας-Κεφαλής, καθώς εφαρμόζει τον συνηθισμένο μηχανισμό Προσοχής σε πολλαπλά κομμάτια παράλληλα και έπειτα συνενώνει τα αποτελέσματα.

0.2.3 BERT Αρχιτεκτονική

Το BERT, το οποίο προτάθηκε από την Google AI [15], είναι ένα προεκπαιδευμένο μοντέλο από βαθείς transformers 2-κατευθύνσεων για την κατανόηση της γλώσσας, το οποίο στη συνέχεια επανεκπαιδεύεται για μια εργασία. Είναι, επομένως, ένα γλωσσικό μοντέλο το οποίο έχει εκπαιδευτεί στο BooksCorpus (800εκ. λέξεις) και στην αγγλική Wikipedia(2,500εκ λέξεις) πραγματοποιώντας δύο εργασίες:

- Masked Language Model: Αντί να προβλέπεται η επόμενη λέξη, τοποθετείται μία μάσκα σε τυχαίες λέξεις στην πρόταση και προβλέπονται αυτές. Αυτό επιτρέπει στο μοντέλο να χρησιμοποιήσει την πληροφορία και των δύο κατευθύνσεων.

- Next Sentence Prediction: Αυτή η εργασία προβλέπει αν η πρώτη πρόταση που δίνεται στο μοντέλο προηγείται της δεύτερης.

0.2.4 Μέθοδοι Ένωσης

Πολλά προβλήματα απαιτούν την ενσωμάτωση διαφορετικών πηγών πληροφορίας. Μερικές φορές αυτά τα προβλήματα περιλαμβάνουν πολλές, ξεχωριστές αναπαραστάσεις πληροφορίας, όπως για παράδειγμα κείμενο, εικόνα, ήχος κοκ., όπου θα πρέπει να συνδυαστούν κατάλληλα για να πραγματοποιήσουν μία συγκεκριμένη εργασία, όπως το να καταλάβουν την σκηνή μιας ταινίας. Άλλες φορές, αυτά τα προβλήματα περιλαμβάνουν πολλαπλές πηγές του ίδιου είδους εισόδου, όπως για παράδειγμα όταν προσπαθούμε να εξάγουμε μία ενιαία περίληψη πολλαπλών κειμένων. Σε αυτές τις περιπτώσεις, λοιπόν, χρειάζεται να αναπτύξουμε μεθόδους ένωσης της πληροφορίας ή με κάποιον τρόπο να εισάγουμε μια δευτερεύουσα πληροφορία μέσα στην πρωτεύουσα. Έτσι, έχουν αναπτυχθεί διάφορες μέθοδοι. Η πιο συχνή μέθοδος εξ' αυτών είναι η απλή συνένωση (concatenation) όπως φαίνεται και στο σχήμα 3. Υπάρχουν και άλλες κοινοί μέθοδοι, όπως η χρήση της δευτερεύουσας πληροφορίας για να γίνει κάποια απολέπιση (scaling) της πρωτεύουσας πληροφορίας ή/και να προσθέσουμε κάποια κλίση (bias) σε αυτή. Η ένωση μπορεί να γίνει σε οποιοδήποτε επίπεδο της αρχιτεκτονικής και συχνά συνίσταται ο πειραματισμός με διαφορετικά επίπεδα ενσωμάτωσης. Τέλος, η δευτερεύουσα πληροφορία που ενσωματώνεται μπορεί να είναι το οτιδήποτε θεωρούμε εμείς χρήσιμο. Σε πολλές εφαρμογές αποτελεί απλά μία εξωτερική γνώση που μπορεί να βοηθήσει το εκάστοτε μοντέλο να βελτιώσει την απόδοση του.

0.3 Σχετική Βιβλιογραφία

Τα μέσα κοινωνικής δικτύωσης αποτελούνται από χρήστες που ανεβάζουν διάφορες δημοσιεύσεις τους. Ως αποτέλεσμα, υπάρχουν διαφορετικές προσεγγίσεις κάθε εργασίας που βασίζεται σε κάποια online πλατφόρμα. Ειδικότερα, μπορούμε να χρησιμοποιήσουμε είτε τις δημοσιεύσεις είτε τους χρήστες, οι οποίοι αποτελούνται από δημοσιεύσεις, για μία εργασία.

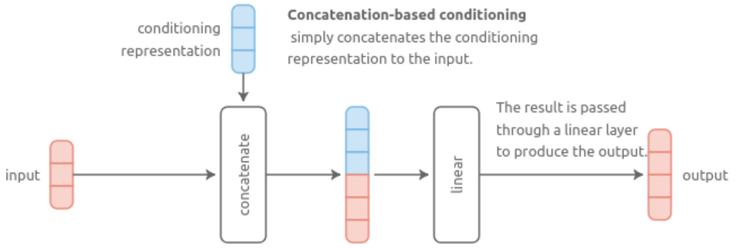

conditioning representation
Concatenation-based conditioning
simply concatenates the conditioning representation to the input.
input
concatenate
linear
The result is passed through a linear layer to produce the output.
output

Figure 3. *Απλή συνένωση (concatenation) αναπαραστάσεων.*

Όσον αφορά την πρώτη περίπτωση, έχουμε την εργασία ταξινόμησης μιας δημοσιεύσης σε μία κλάση και επομένως το πρόβλημα ανάγεται σε ένα απλό πρόβλημα ταξινόμησης κειμένου, όπου η κάθε δημοσίευση θεωρείται αυτόνομη και ανεξάρτητη των υπολοίπων. Οι πιο συνήθεις αρχιτεκτονικές που χρησιμοποιούνται σε αυτή την περίπτωση είναι ΑΝΔ, CNNs και transformers ([16], [17], [18]). Επιπλέον, είναι δυνατό να χρησιμοποιηθούν και κάποια υβριδικά μοντέλα, τα οποία συνδυάζουν δύο ή παραπάνω από τα παραπάνω μοντέλα ([19]). Τέλος, υπάρχει μία έντονη κατεύθυνση και προς την μη-επιβλεπόμενη μάθηση, χρησιμοποιώντας Variational Autoencoders [20], Adversarial Training [21] και Reinforcement learning [22].

Σχετικά με την δεύτερη περίπτωση, σκοπός μας είναι να ταξινομήσουμε έναν χρήστη σε μία ή σε περισσότερες κατηγορίες, όταν ο χρήστης έχει δημοσιεύσει μία σειρά από κείμενα στα μέσα κοινωνικής δικτύωσης. Στην περίπτωση αυτή, υπάρχουν δύο δυνατές προσεγγίσεις του προβλήματος. Η πιο συχνή προσέγγιση είναι να σπάσουμε το πρόβλημα σε δύο επιπέδα και συνεπώς να αποκτήσουμε ένα επίπεδο-δημοσιεύσεων και ένα επίπεδο-χρήστη. Τότε, θα πρέπει αρχικά να εξάγουμε τα πιο χρήσιμα χαρακτηριστικά από κάθε δημοσίευση, λαμβάνοντας μια αναπαράσταση της καθεμίας και στη συνέχεια, χρησιμοποιώντας αυτές τις αναπαραστάσεις όλων των δημοσιεύσεων των χρηστών, να εξάγουμε μία συνολική αναπαράσταση για τον χρήστη. Μία δεύτερη προσέγγιση είναι να θεωρήσουμε τους χρήστες σαν ξεχωριστές οντότητες και να προσπαθήσουμε να αναλύσουμε τις συμπεριφορές και τις σχέσεις μεταξύ διαφορετικών χρηστών (μέσω κοινών στοιχείων των ίδιων ή των δημοσιεύσεων τους). Σε αυτή την προσέγγιση, χρησιμοποιούνται πολύ συχνά τα Graph Neural Networks [23], [24].

Τέλος, ένα σημαντικό στοιχείο στις διάφορες εργασίες που σχετίζονται με θέματα ψυχικής υγείας είναι η εξαγωγή συγκεκριμένων χαρακτηριστικών που θα φανούν χρήσιμα στο μοντέλο μας και σχετίζονται με την εκάστοτε εργασία. Η εξαγωγή τέτοιων χαρακτηριστικών της γλώσσας έχει αποδειχθεί ιδιαίτερα χρήσιμη, καθώς βελτιώνει σημαντικά τις επιδόσεις των μοντέλων. Για την εξαγωγή αυτών, χρησιμοποιούνται συχνά λεξικά, όπως το LIWC [25] και το Emolex [26], τα οποία ταξινομούν ένα σύνολο λέξεων σε συγκεκριμένες κατηγορίες. Επιπλέον, μπορούν να χρησιμοποιηθούν άλλες τροπικότητες (όπως ήχος, εικόνα), ανίχνευση στοιχείων συμπεριφοράς (όπως ειρωνεία, προσωπικότητα), ανίχνευση μεταφορικής γλώσσας, καθώς ακόμα και δημογραφικά χαρακτηριστικά.

0.4 Προτεινόμενα Μοντέλα

Το σύνολο δεδομένων που θα χρησιμοποιήσουμε στην παρούσα διπλωματική εργασία προέρχεται από το Reddit, μία online πλατφόρμα, το οποίο αποκαλείται "Reddit Self-reported Depression Diagnosis (RSDD) dataset" [27]. Αποτελείται από χρήστες που κατέγραψαν μόνοι τους ότι έχουν διαγνωσθεί με κατάθλιψη και περιλαμβάνει σχεδόν 9,000 καταθλιπτικούς χρήστες και 107,000 υγιείς.

0.4.1 "Ανιχνεύοντας τα Πάντα"

Το πρώτο μοντέλο που χρησιμοποιούμε για την ανίχνευση της κατάθλιψης αποτελείται από ένα BERT και τρεις διαφορετικούς ανιχνευτές, ένας Συναισθήματος, ένας Ειρωνείας και ένας Προσωπικότητας. Κάθε ένας από τους παραπάνω ανιχνευτές, έχει εκπαιδευτεί σε μία συγκεκριμένη εργασία, που σχετίζεται με την αντίστοιχο στόχο ανίχνευσης, με ένα συγκεκριμένο διαφορετικό σύνολο δεδομένων και με μία συγκεκριμένη διαφορετική αρχιτεκτονική. Ουσιαστικά, στο πρόβλημα μας, κάνουμε χρήση της τεχνικής της Μεταφοράς Μάθησης, καθώς εκπαιδεύουμε κάποια μοντέλα για διαφορετικούς σκοπούς και μετά τα μεταφέρουμε στο πρόβλημα μας προεκπαιδευμένα.

Η αρχιτεκτονική του μοντέλου απεικονίζεται στο 4. Συγκεκριμένα, σε επίπεδο δημοσίευσης, έχουμε ότι:

$$b_i = \mathrm{BERT}(w_1, w_2, ..., w_{n_i})$$
$$e_i = \mathrm{EMOTION}(w_1, w_2, ..., w_{n_i})$$
$$i_i = \mathrm{IRONY}(w_1, w_2, ..., w_{n_i})$$
$$p_i = \mathrm{PERSONALITY}(w_1, w_2, ..., w_{n_i})$$

όπου $b_i \in \mathbb{R}^{768}$, $e_i \in \mathbb{R}^{300}$, $i_i \in \mathbb{R}^{256}$ και $p_i \in \mathbb{R}^{300}$, ενώ $w_1, w_2, ..., w_{n_i}$ οι λέξεις της εκάστοτε δημοσίευσης. Τελικά:

$$p_i = b_i \circ c_i \circ i_i \circ p_i$$

όπου $\circ$ είναι το σύμβολο της εκάστοτε μεθόδου συνένωσης.

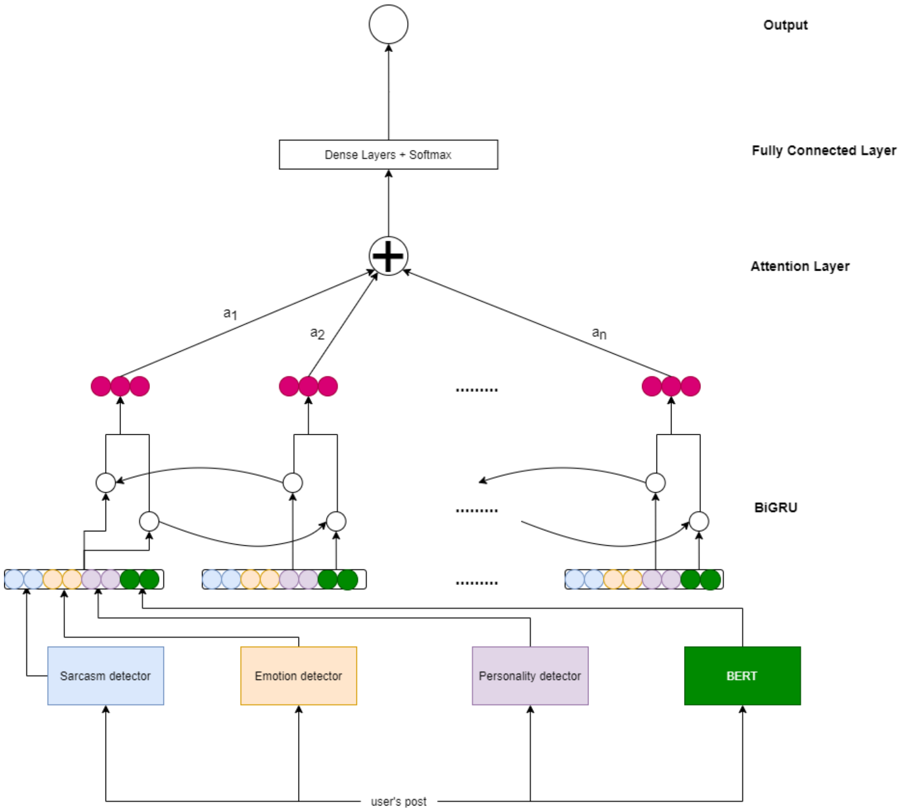


Figure 4. *Η συνολική αρχιτεκτονική του μοντέλου μας "Ανιχνεύοντας τα Πάντα".*

Πειραματιζόμαστε με διαφορετικούς αριθμούς από δημοσιεύσεις και καταλήγουμε σε κάποια χρήσιμα συμπεράσματα. Αρχικά, συμπεραίνουμε ότι όσες περισσότερες δημοσιεύσεις επεξεργαζόμαστε από τον κάθε χρήστη, τόσο καλύτερη επίδοση καταφέρνει το μοντέλο μας, κάτι που είναι λογικό, καθώς περισσότερες δημοσιεύσεις σημαίνει και περισσότερη πληροφορία και επομένως, μπορούμε να εξάγουμε πιο

Model	precision	recall	f1-score
SGL-CNN [28]	0.51	0.56	0.53
MGL-CNN [28]	0.63	0.48	0.54
KFB-BiGRU-Att [6]	0.57	0.51	0.54
KFB-BiGRU-Att-AdaBoost [6]	0.58	0.54	0.56
BERT 1:12	0.41	0.74	0.53
BERT 1:7	0.64	0.53	**0.58**
BERT-EMOTION 1:12 linear 100	0.31	0.81	0.45

Table 1. *Αποτελέσματα όλων των μοντέλων για 600 δημοσιεύσεις ανά χρήστη.*

βέβαια συμπεράσματα. Παρατηρούμε ότι η επίδραση των ανιχνευτών Ειρωνείας και Προσωπικότητας είναι περισσότερο τυχαία. Αυτό οφείλεται στην ανυπαρξία της ειρωνείας σε τέτοια θέματα στο σύνολο δεδομένων και στην ανικανότητα της εργασίας "ανίχνευση προσωπικότητας", καθώς φαίνεται να μην δύναται η εξαγωγή σημαντικών συμπερασμάτων, σε συνδυασμό με το κακό σύνολο δεδομένων που χρησιμοποιείται για την εκπαίδευση στην συγκεκριμένη εργασία, αντίστοιχα. Ο ανίχνευτης Συναισθήματος φαίνεται να έχει μια προβλεπόμενη αντίδραση. Ειδικότερα, κάθε φορά που χρησιμοποιείται ο συγκεκριμένος ανιχνευτής παρατηρούμε μία τάση του μοντέλου να ταξινομήσει πολλούς χρήστες ως καταθλιπτικούς, ανεξάρτητα από το αν ανήκουν όντως σε αυτή την κατηγορία, κάτι που παρατηρείται από το υψηλό Recall σε συνδυασμό με το χαμηλό Precision στις αντίστοιχες περιπτώσεις. Η κατάθλιψη είναι μία ψυχική ασθένεια που σχετίζεται άμεσα με το συναίσθημα και επομένως η προσπάθεια ανίχνευσης συναισθημάτων από την κάθε δημοσίευση του κάθε χρήστη δίνει μία ώθηση στο μοντέλο να συσχετιστεί ο εκάστοτε χρήστης με την κατάθλιψη και επομένως να ταξινομηθεί σε αυτή την κλάση. Πειραματιζόμαστε επίσης με διαφορετικούς τρόπους προσαρμογής της διάστασης της αναπαράστασης των ανιχνευτών, με τον καλύτερο τρόπο να αποτελεί η προσθήκη linear layers που προβάλλουν την διάσταση όλων στα 100. Τέλος, το BERT πετυχαίνει τα καλύτερα αποτελέσματα από όλα και φαίνεται από μόνο του να διαθέτει επαρκή ικανότητα να πραγματοποιήσει την δεδομένη ταξινόμηση. Συνεπώς, το καλύτερο συνολικό αποτέλεσμα (μετρική f1-score) δίνεται από το σκέτο BERT και με μία αναλογία βαρών 1:7 στην συνάρτηση κόστους. Τα αποτελέσματα, για 600 δημοσιεύσεις ανά χρήστη, φαίνονται συνοπτικά στο 1.

Χρησιμοποιώντας ως βάση το καλύτερο μοντέλο μέχρι τώρα (BERT 1:7), προσπαθούμε να δούμε πόσο καλά γενικεύει για διαφορετικούς αριθμούς δημοσιεύσεων ανά χρήστη στην φάση ελέγχου. Μεταβάλλοντας, κατάλληλα τον αριθμό αυτό, ξεκινώντας από 10 δημοσιεύσεις ανά χρήστη και φτάνοντας μέχρι και 2000, τα αποτελέσματα αναπαρίστανται στο 5. Παρατηρούμε ότι όλες οι μετρικές έχουν μία σταδιακή άνοδο μέχρι περίπου τις 600-800 δημοσιεύσεις και ύστερα το Recall μειώνεται, το Precision συνεχώς αυξάνεται και το f1-score μένει κατά βάση σταθερό. Η μείωση του Recall μας δείχνει την γενικότερη μείωση των ταξινομημένων ως καταθλιπτικών χρηστών. Ωστόσο αυξάνεται το Precision και επομένως καταλήγουμε στο ότι κυρίως μειώνονται οι λανθασμένα ταξινομημένοι ως καταθλιπτικοί χρήστες και όχι τόσο οι σωστά ταξινομημένοι. Αυτό επιβεβαίωνεται και από το σταθερό f1-score, το οποίο μας δείχνει, εν τέλει, ότι ο ρυθμός μείωσης των λανθασμένα θετικών δειγμάτων είναι υψηλότερος από τον ρυθμό μείωσης των σωστά θετικών δειγμάτων.

0.4.2 Εναλλακτικό Μοντέλο

Όπως σημειώσαμε παραπάνω, ο ανιχνευτής Συναισθήματος αποδείχθηκε ο πιο "λογικός" ανιχνευτής, καθώς επηρεάζει το μοντέλο κατάλληλα. Συνεπώς, αποφασίζουμε να εστιάσουμε πλήρως σε αυτόν τον ανιχνευτή και να αναπτύξουμε τρόπους, ώστε η πληροφορία που θα εξάγεται από αυτόν να βοηθήσει το μοντέλο-βάση, που θα είναι το BERT. Για αυτό τον σκοπό, εφαρμόζουμε και αναλύουμε 6 διαφορετικές μεθόδους συνένωσης της συναισθηματικής πληροφορίας με την πληροφορία που προέρχεται από

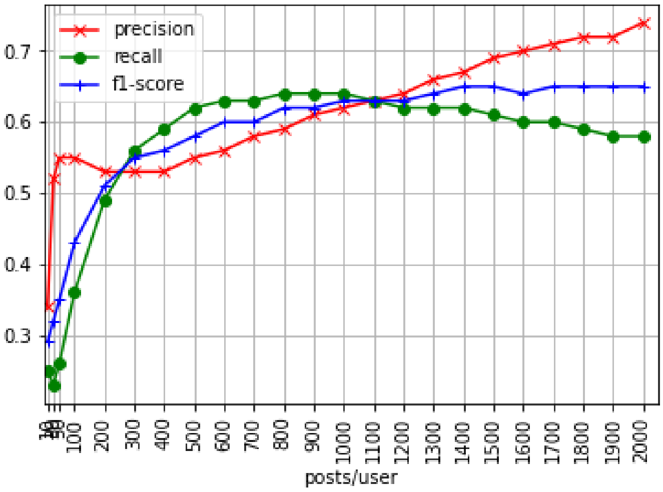
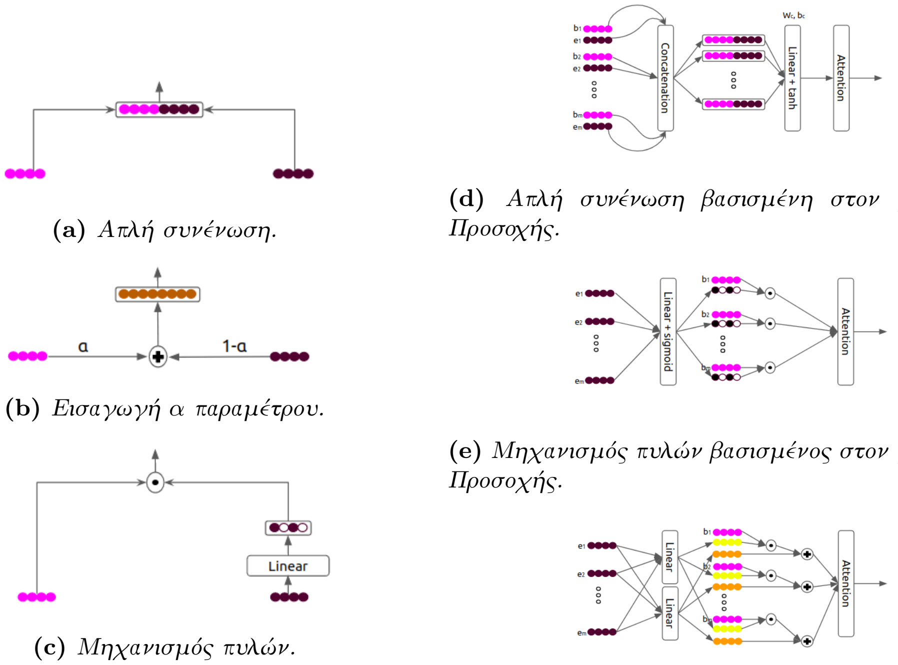


Figure 5. *Precision, Recall, και F1-Score για διαφορετικούς αριθμούς δημοσιεύσεων ανά χρήστη κατά την φάση ελέγχου.*

το BERT. Τρεις από τις μεθόδους θα πραγματοποιούνται σε επίπεδο δημοσίευσης και τρεις σε επίπεδο χρήστη, όπως αυτές διαγράφονται στο 6.



(a) *Απλή συνένωση.*

(b) *Εισαγωγή α παραμέτρου.*

(c) *Μηχανισμός πυλών.*

(d) *Απλή συνένωση βασισμένη στον μηχανισμό Προσοχής.*

(e) *Μηχανισμός πυλών βασισμένος στον μηχανισμό Προσοχής.*

(f) *Αφινικός Μετασχηματισμός βασισμένος στον μηχανισμό Προσοχής.*

Figure 6. *Μέθοδοι ένωσης των αναπαραστάσεων. Αριστερά απεικονίζονται οι μέθοδοι σε επίπεδο χρήστη και δεξιά σε επιπέδο δημοσίευσης.*

Έστω ότι b_j η αναπαράσταση του χρήστη από το BERT και e_j η αντίστοιχη αναπαράσταση από τον ανιχνευτή συναισθήματος. Επίσης, b_i η αναπαράσταση της δημοσίευσης i του χρήστη από το BERT και e_i η αντίστοιχη αναπαράσταση από τον ανιχνευτή συναισθήματος.

Απλή Συνένωση Η τελική αναπαράσταση του χρήστη δίνεται από :

$$u_j = b_j \parallel e_j$$

α παράμετρος Η τελική αναπαράσταση του χρήστη δίνεται από :

$$u_j = \alpha \cdot b_j + (1 - \alpha) \cdot e_j$$

Διαισθητικά, ρυθμίζεται πόσο θα χρησιμοποιηθεί η κάθε αναπαράσταση μέσω της α παραμέτρου.

Μηχανισμός Πυλών Η τελική αναπαράσταση του χρήστη δίνεται από :

$$u_j = b_j \odot \mathrm{sigmoid}(\mathrm{linear}(e_j))$$

Διαισθητικά, δημιουργούμε ένα διάνυσμα μάσκας μέσω της αναπαράστασης του συναισθήματος, το οποίο επιλέγει συγκεκριμένα προεξέχοντα χαρακτηριστικά από την αναπαράσταση του BERT.

Απλή Συνένωση βασισμένη στον μηχανισμό Προσοχής Μαθαίνουμε μία συνάρτηση από την συνένωση των δύο αναπαραστάσεων μίας δημοσίευσης από το BERT και από τον ανιχνευτή συναισθήματος. Συγκεντρωτικά:

$$f_c(b_i, e_i) = \tanh(W_c[b_i \parallel e_i] + b_c)$$

όπου W_c, b_c είναι εκπαιδευόμενες παράμετροι.

Μηχανισμός Πυλών βασισμένος στον μηχανισμό Προσοχής Όμοια με την αντίστοιχη περίπτωση σε επίπεδο χρήστη, μαθαίνεται μία μάσκα χαρακτηριστικών και στη συνέχεια εφαρμόζεται στο BERT. Συνεπώς:

$$f_g(b_i, e_i) = \sigma(W_g b_i + b_g) \odot e_i$$

όπου W_g, b_g είναι εκπαιδευόμενες παράμετροι.

Αφινικός Μετασχηματισμός βασισμένος στον μηχανισμό Προσοχής Ένας αφινικός μετασχηματισμός εφαρμόζεται στον λανθάνων χώρο των κρυφών καταστάσεων. Συγκεκριμένα, χρησιμοποιούμε τις αναπαραστάσεις από τον ανιχνευτή συναισθήματος για να δημιουργήσουμε τα διανύσματα γ και β. Έχουμε:

$$\begin{aligned} f_a(b_i, e_i) &= \gamma(e_i \odot b_i + \beta(e_i)) \\ \gamma(x) &= W_\gamma x + b_\gamma \\ \beta(x) &= W_\beta x + b_\beta \end{aligned}$$

όπου $W_\gamma, W_\beta, b_\gamma, b_\beta$ είναι εκπαιδευόμενες παράμετροι.

Τα αποτελέσματα φαίνονται στον πίνακα 2. Παρατηρούμε ότι όλες οι μέθοδοι ένωσης βελτιώνουν κατά πολύ την απλή περίπτωση συνδυασμού του BERT με τον ανιχνευτή Συναισθήματος που είδαμε προηγουμένως. Επιπλέον, οι μέθοδοι που πραγματοποιούνται σε επίπεδο χρήστη σημειώνουν υψηλότερο Recall, αλλά χαμηλότερο Precision από εκείνες που λαμβάνουν χώρα σε επίπεδο δημοσίευσης. Αυτό είναι λογικό να συμβαίνει, καθώς όταν η συναισθηματική πληροφορία ενώνεται σε επίπεδο χρήστη, βρίσκεται πολύ βαθιά στο μοντέλο και επομένως μπορεί να επηρεάζει περισσότερο την τελική απόφαση του ταξινομητή. Έτσι, όπως σχολιάστηκε και προηγουμένως, αφού η συναισθηματική πληροφορία προκαλεί περισσότερους χρήστες να ταξινομηθούν ως καταθλιπτικοί, όταν αυτή η πληροφορία μπαίνει πιο αργά στο μοντέλο, αυξάνεται το Recall (αυξάνονται οι ταξινομημένοι ως καταθλιπτικοί χρήστες).

Model	precision	recall	f1-score
BERT 1:7	0.64	0.53	0.58
BERT-EMOTION linear 100	0.31	0.81	0.45
Απλή Συνένωση	0.46	0.71	0.56
α παράμετρος	0.49	0.70	0.57
Μηχανισμός Πυλών	0.46	0.69	0.55
Μηχανισμός Πυλών + sigmoid	0.54	0.66	0.59
Απλή Συνένωση βασισμένη στον μηχανισμό Προσοχής	0.51	0.62	0.56
Μηχανισμός Πυλών βασισμένος στον μηχανισμό Προσοχής	0.54	0.59	0.57
Αφινικός Μετασχηματισμός βασισμένος στον μηχανισμό Προσοχής	0.56	0.61	0.58

Table 2. *Τα αποτελέσματα των διαφορετικών μεθόδων ένωσης.*

Χειροποίητα Χαρακατηριστικά Έπειτα, ακολουθούμε μία εκτενής ανάλυση των δεδομένων μας. Η κατάθλιψη είναι μια περίπλοκη ψυχική ασθένεια, η οποία δεν μπορεί να ανιχνευθεί εύκολα από γραπτά κείμενα των ατόμων. Συνίσταται, λοιπόν, να ερευνηθούν τα κείμενα περισσότερο, ώστε να βρεθούν κάποια διαφορετικά λεξιλογικά μοτίβα που μπορούν να φανούν χρήσιμα. Έπειτα από την εύρεση τέτοιων μοτίβων, η προσαρμογή τους στο μοντέλο αρμόζει να γίνει χειροκίνητα, βάζοντας τα άμεσα στο μοντέλο μας. Ύστερα από εκτενή ανάλυση του συνόλου δεδομένων μας, καταλήγουμε να μελετήσουμε τρια λεξικά χαρακτηριστικά:

- **Αριθμός δημοσιεύσεων και λέξεων ανά δημοσίευση.** Συγκεκριμένα, παρατηρήθηκε ότι οι καταθλιπτικοί χρήστες έχουν κατά μέσο όρο τον διπλάσιο αριθμό δημοσιεύσεων από τους υγιείς χρήστες, καθώς και περισσότερες λέξεις ανά δημοσίευση τους σε αναλογία 5:3. Φυσικά, τον αριθμό των δημοσίευσεων δεν μπορούμε να τον χρησιμοποίησουμε στο μοντέλο μας, καθώς δεν έχουμε χρονική πληροφορία και επομένως, ο διπλάσιος αριθμός δημοσιεύσεων μπορεί να γράφτηκε και σε διπλάσια χρονική διάρκεια. Αντιθέτως, τον αριθμό των λέξεων δύναται να τον χρησιμοποιήσουμε.

- **Υβριστικό Λεξιλόγιο.** Αφού παρατηρήσαμε την έντονη χρήση υβριστικού λεξιλογίου από τα καταθλιπτικά άτομα, επιβεβαιωθήκαμε και από τις σχετικές μετρήσεις, καθώς πράγματι οι καταθλιπτικοί χρήστες κάνουν πιο έντονη χρήση υβριστικού λεξιλογίου. Για τον κάθε χρήστη βγάζουμε ένα ενιαίο σκορ για το πόσο υβριστικός είναι και χρησιμοποιούμε αυτό σαν επιπλέον χαρακτηριστικό.

- **Ηθικό Λεξιλόγιο.** Χρησιμοποιώντας την επέκταση του LIWC στις ηθικές διαστάσεις που έκαναν στο [29], δίνοντας έναν αριθμό που ονομάζεται "ηθική δύναμη" και ποσοτικοποιεί την ηθική μίας λέξης, εξάγουμε δύο χαρακτηριστικά για κάθε ηθική διάσταση για τον κάθε χρήστη. Το πρώτο χαρακτηριστικό είναι ο μέσος όρος της "ηθικής δύναμης" των δημοσίευσεων του κάθε χρήστη και το δεύτερο χαρακτηριστικό είναι το ποσοστό των δημοσιεύσεων ενός χρήστη που περιλαμβάνουν τουλάχιστον μία λέξη που περιέχεται στο λεξικό της ηθικής διάστασης. Συνολικά, για τις 5 διαστάσεις, συγκεντρώνουμε 10 χαρακτηριστικά για τον κάθε χρήστη. Παρατηρούμε ότι οι καταθλιπτικοί χρήστες χρησιμοποιούν πολύ πιο συχνά ηθικό λεξιλόγιο, ενώ δεν υπάρχουν σημαντικές διαφορές στην "ηθική δύναμη".

Εφαρμόζουμε τα παραπάνω στο καλύτερο μοντέλο μας, το οποίο είναι ο Μηχανισμός Πυλών σε επίπεδο χρήστη. Τα αποτελέσματα συγκεντρώνονται στον πίνακα 3. Τα χειροποίητα χαρακτηριστικά ενσωματώνονται στο επίπεδο χρήστη του μοντέλου μας, καθώς θέλουμε να επηρεάσουν αρκετά την απόφαση του μοντέλου. Βλέπουμε ότι τόσο η ύβρις όσο και το ήθος είναι πολύ σημαντικά στην απόδοση του μοντέλου και βελτιώνουν αρκετά την επίδοση αυτού, φτάνοντας μάλιστα σε ένα f1-score

	precision	recall	f1-score
Μηχανισμός Πυλών	0.54	0.66	0.59
Μηχανισμός Πυλών + ύβρις + ήθος	0.64	0.61	**0.63**
Μηχανισμός Πυλών + ήθος	0.63	0.60	0.62
Μηχανισμός Πυλών + ήθος + ύβρις	0.71	0.55	0.62
Μηχανισμός Πυλών + ήθος + ύβρις + αριθμός_λέξεων	0.49	0.71	0.58

Table 3. *Τα αποτελέσματα της ενσωμάτωσης των χειροποίητων χαρακτηριστικών στο επίπεδο χρήστη του μοντέλου μας.*

ίσο με 0.63. Όταν, ωστόσο, προστίθενται και τα δύο, δεν παρατηρούμε κάποια επιπλέον βελτίωση. Τέλος, ο αριθμός των λέξεων φαίνεται να μην έχει εν τέλει τα επιθυμητά αποτελέσματα.

0.5 Περισσότερα Σύνολα Δεδομένων

Επεκτείνουμε τις παραπάνω ιδέες μας σε δύο ακόμα σύνολα δεδομένων, ένα σχετιζόμενο ξανά με την κατάθλιψη και ένα με το άγχος. Και στα δύο σύνολα δεδομένων, πραγματοποιούμε κάποιες σημαντικές διαφορετικές στην αρχιτεκτονική:

- Καθώς πλέον η εργασία μας είναι να ταξινομήσουμε μία δημοσίευση και όχι έναν χρήστη, πέφτουμε ένα επίπεδο στην αρχιτεκτονική. Αντί για επίπεδο δημοσίευσης και επίπεδο χρήστη, πλέον έχουμε επίπεδο λέξης και επίπεδο δημοσίευσης.

- Προσθέτουμε μία επιπλέον περίπτωση αφινικού μετασχηματισμού, όπου βάζουμε μία μη γραμμική συνάρτηση (συγκεκριμένα, την υπερβολική εφαπτομένη).

- Αλλάζουμε το σύνολο δεδομένων και την αρχιτεκτονική του ανιχνευτή συναισθήματος.

- Όσον αφορά το χειροποίητο χαρακτηριστικό του ήθους, αλλάζουμε τον τρόπο ανίχνευσης. Αυτή την φορά, δημιουργούμε τα ιστογράμματα και λαμβάνουμε την κατανομή της κάθε δημοσίευσης σε 8 διαστήματα και την περνάμε ως είσοδο για το ήθος. Συνεπώς, αφού έχουμε 5 διαστάσεις, έχουμε ένα διάνυσμα 40 διαστάσεων (5 ηθικές διαστάσεις · 8 διαστήματα) και χρησιμοποιούμε ένα linear layer για να μειώσουμε την διάσταση σε 5.

- Πέρα από την ενσωμάτωση των χαρακτηριστικών σε επίπεδο δημοσίευσης, εξάγουμε τις αντίστοιχες τιμές ανά λέξη και ενσωματώνουμε την πληροφορία και σε επίπεδο λέξης. Τότε, έχουμε 4 διαφορετικές περιπτώσεις: ενσωμάτωση συναισθηματικής πληροφορίας σε επίπεδο λέξης και χειροποίητων χαρακτηριστικών σε επίπεδο λέξης, ενσωμάτωση συναισθηματικής πληροφορίας σε επίπεδο λέξης και χειροποίητων χαρακτηριστικών σε επίπεδο δημοσίευσης, επίπεδο δημοσίευσης - επίπεδο λέξης, επίπεδο δημοσίευσης - επίπεδο δημοσίευσης. Όταν βρισκόμαστε στην πρώτη περίπτωση, τα χειροποίητα χαρακτηριστικά αποτελούν μέρος της συναισθηματικής πληροφορίας (απλή συνένωση αυτών), ενώ στην δεύτερη περίπτωση, τα χειροποίητα χαρακτηριστικά ενσωματώνονται τόσο στο BERT όσο και στην πληροφορία συναισθήματος μέσω της μεθόδου του ΜΗχανισμού Πυλών (καθώς αυτή αποδείχτηκε η καλύτερη).

0.5.1 Κατάθλιψη

Το σύνολο δεδομένων που χρησιμοποιούμε προέρχεται από το [9] και είναι αρκετά μικρό, καθώς περιλαμβάνει 1,293 καταθλιπτικές δημοσιεύσεις και 548 κανονικές. Τα συνολικά αποτελέσματα των μεθόδων που αναλύσαμε, παρουσιάζονται στον πίνακα 4.

	Acc	F1	P	R
LIWC [8]	70	72	74	71
LDA [8]	75	74	75	72
unigram [8]	70	81	71	95
bigram [8]	79	78	80	76
LIWC+LDA+unigram [8]	78	81	84	79
LIWC+LDA+bigram [8]	91	93	90	92
LSTM [7]	87.03	-	90.30	-
Bi-LSTM [7]	86.46	-	88.08	-
BiLSTM+Att [7]	88.59	-	90.41	-
EAN [7]	91.30	-	91.91	-
Απλή Συνένωση με Προσοχή	**93.03**	**95.25**	94.24	**96.35**
Μηχανισμός Πυλών με Προσοχή	91.79	94.29	94.34	94.23
Αφινικός Μετασχηματισμός με Προσοχή	92.87	95.00	**95.92**	94.11
Μη-Γραμμικός Αφινικός Μετασχηματισμός με Προσοχή	92.87	95.01	95.83	94.23
Απλή Συνένωση	88.65	91.95	91.60	92.31
α παράμετρος	89.19	92.48	90.44	94.62
Μηχανισμός Πυλών + sigmoid	90.27	93.18	91.79	94.62

Table 4. *Τα αποτελέσματα των μοντέλων του [7] και [8] στο σύνολο δεδομένων [9], καθώς επίσης και τα αποτελέσματα των μοντέλων μας με τις 7 διαφορετικές μεθόδους ένωσης.*

Παρατηρούμε ότι οι μέθοδοι που περιλαμβάνουν μηχανισμό Προσοχής ξεπερνούν όλα τα state-of-the-art μοντέλα σε επίδοση, ενώ όλες οι μεθόδοι ένωσης ξεπερνούν τα baseline μοντέλα. Παρατηρούμε για μία ακόμη φορά ότι οι μέθοδοι που πραγματοποιούνται σε επίπεδο δημοσίευσης έχουν κατά μέσο όρο υψηλότερο Recall και χαμηλότερο Precision από αυτές που συμβαίνουν σε επίπεδο λέξης. Επιπλέον, πειραματιζόμαστε προσθέτοντας τα χαρακτηριστικά ύβρις και ήθος στο μοντέλο μας, αφού πρώτα έχουμε επιβεβαιώσει τις παρατηρήσεις μας και σε αυτό το σύνολο δεδομένων, ότι δηλαδή υπάρχουν αρκετές διαφορές των δύο κλάσεων ως προς αυτά τα χαρακτηριστικά. Τόσο η ύβρις όσο και το ήθος σημειώνουν σημαντική βελτίωση στην απόδοση των μοντέλων μας, κρίνοντας την χρήση τους πολύ υποσχόμενη. Ως προς το accuracy, και τα δύο χαρακτηριστικά βελτιώνουν την επίδοση 3% ακόμα. Όπως σημειώσαμε και προηγούμενως, έτσι και εδώ, όταν προσθέτουμε και τα δύο χαρακτηριστικά ταυτόχρονα, δεν λαμβάνουμε κάποια ιδιαίτερη βελτίωση. Όταν βάζουμε τα χαρακτηριστικά και την συναισθηματική πληροφορία σε επίπεδο λέξης φαίνεται να σημειώνει χειρότερη επίδοση από όταν τα χαρακτηριστικά μπαίνουν σε επίπεδο δημοσίευσης. Αυτό είναι λογικό να συμβαίνει καθώς χάνεται ένα μέρος της δράσης τους στο βάθος του μοντέλου. Αντιθέτως, όταν η συναισθηματική πληροφορία μπαίνει σε επίπεδο δημοσίευσης, τα χαρακτηριστικά σημειώνουν καλύτερη επίδοση όταν μπαίνουν σε επίπεδο λέξης, πιθανώς επειδή τότε ενσωματώνονται τόσο στο BERT όσο και στο συναίσθημα και στην συνέχεια ενώνονται ξανά και άρα επιδρούν περισσότερο.

0.5.2 Άγχος

Δοκιμάζουμε την εργασία μας σε μία διαφορετική εργασία από αυτή την ανίχνευση κατάθλιψης, αλλά όχι εντελώς μακριά αυτής. Επιλέγουμε την εργασία της ανίχνευσης Άγχους. Χρησιμοποιούμε το σύνολο δεδομένων [12], προερχόμενο από το Reddit, που αποτελείται από 3,553 δημοσιεύσεις. Τα συνολικά αποτελέσματα φαίνονται στον πίνακα 5.

Τα ALL, Ekman, Sentiment αποτελούν διαφορετικά μερίδια του συνόλου δεδομένων. Το ALL περιλαμβάνει όλες τις 27 ετικέτες των δεδομένων, ενώ τα υπόλοιπα δύο είναι αντιστοιχήσεις των 27 ετικετών σε λιγότερες. Παρατηρούμε ότι οι μέθοδοι μας ξεπερνούν την επίδοση των baseline μοντέλων, αλλά όχι του καλύτερου αυτών που είναι το MentalRoBERTa, μία RoBERTa προεκπαιδευμένη σε ιατρικά κείμενα. Οι μέθοδοι μας δουλεύουν καλύτερη στην περίπτωση του ALL,

	Binary F1	Accuracy
RNN [10]	67.58 ±1.22	68.86 ±1.10
BERT [10]	78.88 ±1.09	79.11 ±1.32
MentalBERT [11]	80.04	-
MentalRoBERTa [11]	81.76	-
Multi1 ALL [10]	79.02 ±0.35	**79.72** ±0.69
Multi2 ALL[10]	78.97 ±0.24	78.55 ±0.07
FT ALL [10]	76.40 ±0.50	76.83 ±0.40
πλή Συνένωση ALL	79.09 ±0.88	78.29 ±0.33
α παράμετρος ALL	80.20 ±1.20	78.70 ±0.68
Μηχανισμός Πυλών + sigmoid ALL	**80.76** ±0.38	79.50 ±0.43
Απλή Συνένωση με Προσοχή ALL	79.23 ±0.74	78.00 ±0.35
Μηχανισμός Πυλών με Προσοχή ALL	79.90 ±0.30	77.62 ±0.64
Αφινικός Μετασχηματισμός με Προσοχή ALL	79.74 ±0.33	78.14 ±0.63
Μη-Γραμμικός Αφινικός Μετασχηματισμός με Προσοχή ALL	79.62 ±1.10	78.32 ±0.37
Multi1 Ekman [10]	**80.24** ±1.39	**81.07** ±1.13
FT Ekman [10]	79.44 ±0.29	79.53 ±0.46
Μηχανισμός Πυλών + sigmoid Ekman	79.69 ±0.96	78.48 ±0.40
Multi1 Sentiment [10]	79.46 ±1.05	79.86 ±0.50
FT Sentiment [10]	**79.75** ±0.52	**80.61** ±0.40
Μηχανισμός Πυλών + sigmoid Sentiment	79.11 ±0.97	78.06 ±0.40

Table 5. *Η επίδοση των μοντέλων των [10] και [11] στο σύνολο δεδομένων [12], καθώς επίσης και τα αποτελέσματα των δικών μας μοντέλων.*

δηλαδή όταν χρησιμοποιείται όλο το σύνολο δεδομένων. Πειραματιζόμαστε και σε αυτά τα δεδομένων με τα χαρακτηριστικά ύβρις και ήθος, σε πλήρη αντιστοιχία με την μεθοδολογία που ακολουθήσαμε στο προηγούμενο σύνολο δεδομένων. Τα συμπεράσματα είναι αρκετά παρόμοια, καθώς και εδώ τα χειροποίητα χαρακτηριστικά φέρουν σημαντικές βελτιώσεις στην απόδοση των μοντέλων. Σε αντίθεση με πριν, παρατηρούμε ότι η περίπτωση εσωματώσης των χειροποίητων χαρακτηριστικών σε επίπεδο λέξης φέρνει καλύτερα αποτελέσματα, ανεξάρτητα από το πότε ενσωματώνεται η συναισθηματική πληροφορία.

Chapter 1

Introduction

1.1 Motivation

Depression is a mental health disorder which affects a large portion of society. More specifically, [13] records 322 million people worldwide suffering from depression, which corresponds to 4.4% of the world population. WHO states that over 800,000 suicide deaths are reported each year due to depression, while for 15-29-year-old people, it is the leading factor of death. These facts and statistics make depression a severe mental health disorder that its detection is necessary. The symptoms of depression can be the loss of interest in everyday activities, sleeping and eating disorders, feelings of worthlessness, sadness and exhaustion, or even thoughts of suicide [30]. The diagnosis of depression is not always easy, as there is not stable medical indicators of it.

Depression detection is a task of detecting the signs of a depressive person and if those signs are enough to diagnose a person as depressive. The symptoms vary a lot, as well as the duration and the severity. They can be identified in someone's behavior, speech, facial expressions or even use of language. The procedure of clinical diagnosis of depression is subjective to some extent and the need of developing intelligent systems in order to help decision making and depression studying is considered imperative.

Moreover, social media platforms have been grown up lately, as more and more users are active in a platform and they share a lot of their personal stuff. A lot of open data are provided, nowadays, making the data-driven approaches more valuable, as a plethora of information can be exploited. In particular, the analysis of the data can give important insights that can be used from AI systems. In the task of depression detection, deep learning models can perform better if the observations of data's characteristics are integrated into them or if different modalities of data be combined, such as text, speech, image/video or even medical values(e.g. derived from hematological examinations).

Developing AI systems for a depression detection task can be very helpful for the society. At first, collecting the data and analysing in hand is really time-consuming and there is always a high probability of a human error. AI systems, however, are capable of processing large amounts of data in a short time and that makes them ideal for an assisting role for the doctor's decision. Moreover, as already referred, a huge amount of users has found company in a social media platform, making them trust the virtual world more than reality. This affects their emotional state and the possibilities of curing them. Having this in mind, the detection of a mental health issue through their posts or their activity and the detection of the severity are crucial tasks. This work can replace the visit to a doctor to some extent.

However, even if a depression detection task can be a challenging task with a lot to offer in the medical community, there are several limitations towards this goal. The majority of the data that are provided are unlabelled, as social media platforms can give permission to some of their data but they have not labelled their data in such a specific task. Even the manual annotation of them can be really hard and time-consuming. Moreover, there are some ethical considerations that should be taken into consideration. There are some limitations on the data, due to medical confidentiality and in any case, the application of an AI system, executing a task like that, is suggested only to happen auxiliary.

In this work, we aim to research methods and techniques that can help algorithms deal effectively with the task of depression detection. To this end, we present an analysis of posts derived from social media and provide the algorithms with psychological and linguistic information so as to discriminate the two classes easier and help models generalize more effectively from unseen data. We deepen on the ways that we can integrate some aspects of human behaviour in our model, in order to enhance the performance.

1.2 Thesis Contributions

Depression detection is a multifactorial process. A user who posts continually in a social media platform will have a variety of emotional posts. In other words, a depressive user will have, also, some happy posts and a healthy user will have some emotionally negative posts. This procedure is difficult to be learnt by an artificial intelligence system, as it is difficult for the humans as well. It is thus important to propose methods and systems capable of making right predictions, given more general observations and not only relied on negative emotions.

In the context of this thesis, we will be looking into methods and techniques for detecting depression in users from Reddit. In particular, our key contributions are the following:

- We propose a novel model for depression detection by incorporating representations extracted from different detectors, which detect different aspects of the human behavior (emotion, irony, personality) in the proposed models.

- We focus on the emotion detector and we try to improve the fusion methods of the emotional information and our basis model's information. We propose different fusion methods and we achieve to enhance the general performance of our model.

- We conduct a data analysis of the depressive users that are used in this dataset, aiming to detect some patterns in the usage of language. We incorporate our findings into the model to improve it.

- We extend our ideas in more datasets, as well, including one more dataset related with depression and one related with stress. We conclude that the model can generalize.

1.3 Thesis Outline

Chapter 2: Machine Learning, provides background knowledge to set the stage for the subsequent chapters. First, we provide an overview of technical information that is relevant in order to understand the contents of this thesis. Next, we introduce the reader to machine learning together with its most elementary methods. We subsequently delve into the machine learning models primarily used in this thesis.

Chapter 3: Natural Language Processing, presents the natural language processing background needed to understand this thesis. After briefly presenting popular natural language processing tasks, language modeling is presented, initially in the form of an-gram model based on the Markov assumption and then as a recurrent neural network. Then, the most used transformer (BERT) is presented in detail, as it will be one of the core models in this thesis.

Chapter 4: Depression Detection, is a short survey that introduces the topic of depression detection and how the research community has approached this task. It is based, mainly, on the correlation of depression detection with social media and numerous techniques and methods are summarized.

Chapter 5: Proposed Models, presents a variety of novel architectures, which try to detect every aspect of a depressive user. At first, we introduce a multi-view approach, where BERT, Emotion, Irony and Personality detectors are used. Then, we focus on BERT and Emotion detector and we explore the different ways that we can infuse the emotional information into the representations of BERT.

Chapter 6: Conclusions, contains our conclusion, summarizing our findings and providing an outlook into the future work.

Chapter 2

Machine Learning

2.1 Introduction

Machine Learning is the study of computer algorithms that improve automatically through experience. Applications range from datamining programs that discover general rules in large data sets, to information filtering systems that automatically learn users' interests [31]. Based on Oxford dictionary, Machine Learning can be defined as "a type of artificial intelligence in which computers use huge amounts of data to learn how to do tasks rather than being programmed to do them". Machine learning algorithms build a mathematical model of sample data, known as “training data”, in order to make predictions or decisions. The primary aim is to allow the computers to learn this process automatically, without human intervention or assistance and adjust actions accordingly.

Machine learning algorithms are used in a wide variety of applications, including computer vision, speech recognition and email filtering. In these areas of study, scientists are not capable of developing traditional algorithms and computational methods to find solutions for the needed tasks. Machine learning is closely related to computational statistics, so it focuses on making predictions using computers. Therefore, such tasks otherwise infeasible to deal with can be handled using statistical models.

Machine learning is considered as a subset of Artificial Intelligence, a field that includes algorithms that can derive new knowledge and reason, based on logical inference rules. However, Machine Learning algorithms do not need the formal description of the whole knowledge and that makes them more beneficial. The general aim of Machine Learning is to learn a mapping function from the input space to the output space or to learn a representation, something that is called representation learning.

2.2 Types of Machine Learning

In ML, tasks are generally classified into four basic categories. These are supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. In the first case, models learn the mapping between inputs and outputs, as there is information about the categories of the output, known as "labels". In the second case models have to find themselves structure within the input data, as there is lack of labels and the third cases consists a combination of the two referred. Finally, in the fourth case the training system is treated as an agent that aims to maximize a profit, defined by a Reward Function, by interacting with a dynamic environment. Some new methods are appeared in the latest years, as Meta-learning where the model learns how to learn.

2.2.1 Supervised Learning

In supervised learning, there are input variables X and an output variable Y. The goal is to build an algorithm that learns the mapping function from the input to the output.

$$Y = f(X)$$

During training, both X and their corresponding y are provided; thus, the training data are labeled. At inference, we expect that the mapping function can successfully predict the output for every given x provided from the same distribution as the training sample.

Tasks in supervised learning are distinguished into classification and regression. Classification is the task of identifying to which class a sample belongs to. On the other hand, regression is the procedure of mapping a sample to a continuous value.

2.2.2 Unsupervised Learning

In unsupervised learning, there are only the input variables X, without having an output variable that would declare the label of the sample. The goal is to find some patterns between the samples that will be adequate to distinguish them and assign them to a class or a value.

A large subclass of unsupervised tasks is the problem of clustering. Clustering refers to grouping observations together in such a way that members of a common group are similar to each other, and different from members of other groups. Generative models are also a subclass of unsupervised learning models. Generative models mimic the process of generating training data and hence, they aim to generate new samples that will look similar to training samples. This type of learning is unsupervised because the process that generates the data is not directly observable.

2.2.3 Semi-Supervised Learning

In semi-supervised learning, there are the input variables X and an output variable Y for only a subset of the samples. It is a combination of Supervised and Unsupervised learning, usually referred as an unsupervised learning method with a bias. In other words, semi-supervised algorithms use the initially labeled samples to label the whole dataset, introducing pseudo-labels for the unlabeled samples. After that, all the labels are combined to create the fully labeled dataset. It can be used in various cases, where it is infeasible to label every sample. An example is internet content classification, where human only label a subset of the content, but it is impossible everything to get labeled.

2.2.4 Reinforcement Learning

In reinforcement learning, there are an environment and an agent which interacts with it. The agent performs actions based on observations, and then receives a reward from the environment. The behavior of the agent depends on a function that maps the observations of the environment to actions. The machine uses trial-and-error, in order to learn. It starts with random trials and having as aim to optimise a reward, it progresses to adnvanced techniques and abilities. An important issue of reinforcement learning is the trade-off between exploration, in which the system experiments with new kinds of actions to check how profitable or harmful they are, and exploitation, in which the system follows a more conservative approach of selecting already known actions to achieve a high reward. [32]

2.3 Classification Models

2.3.1 Introduction

In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data. Having the input samples, the goal is a model to classify them to a class label, according to some characteristics of them, a processing of them or even randomly. This model is called a classifier.

There exist various classification algorithms for modeling classification tasks. Some of the most common classification tasks are : (a) given a sentence to predict if its sentiment is positive or negative, (b) given medical examinations' values to predict if a tumor is malignant or benign, (c) given an image to predict if it is depicted a cat or a dog. Yet, not all classifiers can be successfully applied to all tasks, and vice versa. To this end, it is recommended to conduct experiments and discover which method results in the best performance for a given task.

2.3.2 Naive Bayes

Naive Bayes is a classifier that is based on **Bayes Theorem**. According to Bayes Theorem, the posterior probability of a sample to belong to a specific class is equal to the product of the likelihood of this sample's generation in this class with the class prior probability, divided by the predictor prior probability as depicted below :

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}, \text{ for a class } c$$

Yet, $P(x)$ is identical for all classes, and therefore can be ignored. Approximations are commonly used, such as using the simplifying assumption that features are independent given the class. This yields the naive Bayes classifier NB defined by discriminant functions [33]:

$$f_i{}^{NB}(\mathbf{x}) = \prod_{j=1}^{n} P(\mathbf{X} = \mathbf{x_j}|c = i)P(c = i)$$

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods. One of the main advantages of Naive Bayes Classifier is that it does not require a lot of training data to estimate the parameters.

According to the assumptions that are made on distribution of the features, there are different models, that are called "event models" of the Naive Bayes Classifier. Some of the most popular are Gaussian Naive Bayes and Multinomial Naive Bayes.

Gaussian Naive Bayes is the classifier when the data follow a normal (gaussian) distribution. Where there are continual values there is a typic assumption that the values associated with each class are distributed according to a normal (or Gaussian) distribution. This can be written as $X \sim \mathcal{N}(\mu, \sigma^2)$, where μ is the mean and σ is the variance. Then, the probability density of u, given a class $c = k$ is :

$$p(x = u|c = k) = \frac{1}{\sigma_k\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{u - \mu_k}{\sigma_k}\right)^2\right)$$

Multinomial Naive Bayes is the classifier where the samples (feature vectors) have been generated by a multinomial $(p_1, p_2, ..., p_n)$ where p_i is the probability that event i occurs (or K such multinomials in the multiclass case). The MNB calculates the class prior $P(c = k)$ by dividing the number of events belonging to class k by the total number of events. The likelihood is calcualted as :

$$p(\mathbf{x} \mid c = k) = \frac{(\sum_{i=1}^{n} x_i)!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} {p_{ki}}^{x_i}$$

MNB is primarily used in Natural Language Processing and particulary in document classification. In this case, the MNB calculates the probability of a tag for a given sample and then gives the tag with the highest probability as output.

2.3.3 Support Vector Machines

A support vector machine (SVM) is a computer algorithm that learns by example to assign labels to objects. In essence, an SVM is a mathematical entity, an algorithm (or recipe) for maximizing a particular mathematical function with respect to a given collection of data. The basic ideas behind the SVM algorithm, however, can be explained without ever reading an equation. Indeed, I claim that, to understand the essence of SVM classification, one needs only to grasp four basic concepts: (i) the separating hyperplane, (ii) the maximum-margin hyperplane, (iii) the soft margin and (iv) the kernel function. [34].

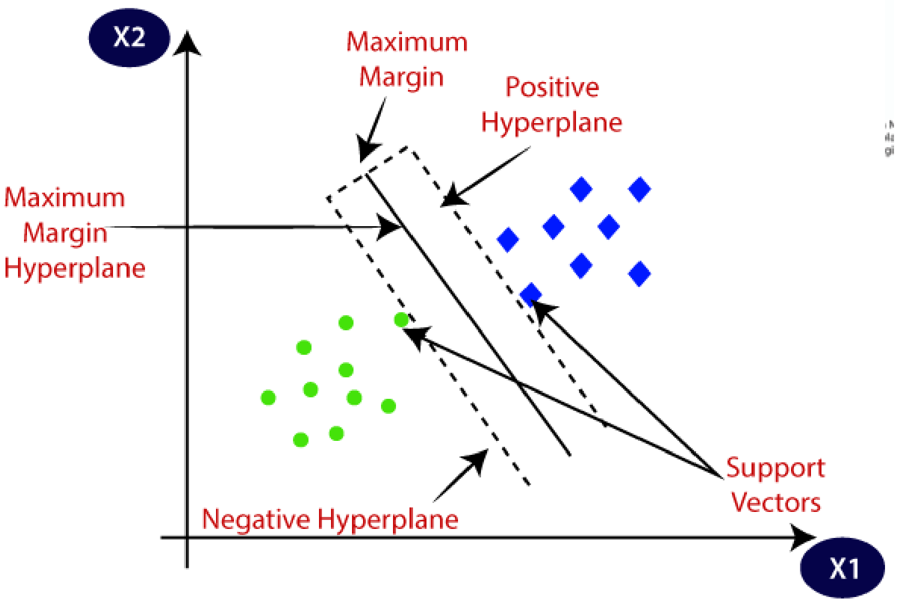


Figure 2.1. *The basic concepts of a SVM visualised*

Suppose we have a two-class classification problem using linear models of the form:

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

where $\phi(\mathbf{x})$ denotes a fixed feature-space transformation, and we have made the bias parameter b explicit. The training dataset comprises N input vectors $x_1, x_2, ..., x_N$ with corresponding target values $y_1, y_2, ..., y_N$ where $y_i \in \{-1, 1\}$, and new data points x are classified according to the sign of $f(\mathbf{x})$. We shall assume for the moment that the training data set is linearly separable in feature space, so that by definition there exists at least one choice of the parameters $\mathbf{w}$ and b such that a function of the above form satisfies $f(\mathbf{x}) > 0$ for points having $y_i = +1$ and $f(\mathbf{x}) < 0$ for points having $y_i = -1$, so that $y_i \cdot f(\mathbf{x}) > 0$ for all training points.

The separating hyperplane is a hyperplane that separates the two classes. In the case of the two classes, where the hyperplane is a line, the separating hyperplane is a linear function that make the two classes distinguishable. There can be infinite separating hyperplane in a problem. In 2.1, three separating hyperplanes are depicted : the positive, the negative and the maximum-margin. The aim of SVM is to find the separating line for which the minimum distance between the two classes is as wide as possible. This final hyperline is depicted by the **maximum-margin hyperplane**.

If $f(\mathbf{x})$ separates the data, the geometric distance between a point $\mathbf{x_i}$ and a hyperplane $f(\mathbf{x}) = 0$ is $\frac{|f(\mathbf{x_i})|}{\|\mathbf{w}\|}$. Furthermore, we are only interested in solutions for which all data points are correctly classified, so that $y_i \cdot f(\mathbf{x_i}) > 0$ for all i. In order to maximize the distance $\frac{1}{\|\mathbf{w}\|}$, we need to minimize the norm $\|\mathbf{w}\|$ subject to $y_i \cdot (\mathbf{w} \cdot \mathbf{x_i} b_i) > 1$, for i between 1 and M.

Intuitively, we would like the SVM to be able to deal with errors in the data by allowing a few anomalous expression profiles to fall on the 'wrong side' of the separating hyperplane. To handle cases like these, the SVM algorithm has to be modified by adding a **'soft margin'**. Essentially, this allows some data points to push their way through the margin of the separating hyperplane without affecting the final result.

Furthermore, it is possible that the given data points are not linearly separable in the space. In that case, a non-linear classification is occurred, as it is needed to move in a higher dimensional space. More specifically, SVM use a **kernel function** $K(x, y)$ to map the input vectors to a space where they can be separable. The most common kernel functions are :

1. Polynomial kernel : $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$

2. Gaussian radial basis kernel : $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$, for $\gamma > 0$

2.3.4 Loss Function

The goal of any Supervised Learning algorithm is to return a function $f()$ which accurately matches the input examples to the corresponding labels. To quantify the loss (error) of the model, a Cost Function is used that predicts $\hat{y}$ when the actual label is y. Usually, the Cost Function $L(\hat{y}, y)$ assigns a numeric value to the predicted output $\hat{y}$ given the actual output y. It must have an infimum, which means that the lower the error value, the better the prediction. Function parameters are set in order to minimize L loss in the training examples.

Given a train set $(x_{1:n}, y_{1:n})$, a cost function L per sample and a function $f(x; \theta)$, we define the total loss as the average loss on all training data:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(f(x; \theta), y_i\right)$$

The goal is to find the optimal parameters θ that minimize the total error:

$$\hat{\theta} = \arg_\theta \min \mathcal{L}(\theta) = \arg_\theta \min \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(f(x; \theta), y_i\right)$$

Some standard cost functions are the following:

Mean Squared Error (MSE): MSE calculates the mean squared prediction error:

$$J(\partial) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - P_i)^2$$

Where the prediction error is the difference between the true value (Y_i) and the predicted value (P_i) for an instance, and ∂ is the parameter vector of the network. MSE is used with regression models.

Mean Absolute Error (MAE): MAE calculates the mean of the absolute prediction error:

$$J(\partial) = \frac{1}{n}\sum_{i=1}^{n} |Y_i - P_i|$$

Where Y_i is the true value and P_i is the predicted value for an instance, and ∂ is the parameter vector of the network.

Cross Entropy: Cross-entropy loss function uses the concept of cross-entropy. Cross-entropy is mathematically defined as:

$$H(p, q) = -\sum_{k} p_k \log q_k$$

Where p and q are the true and the predicted probability distributions, respectively, the more the two distributions differ, the higher the value of the cross-entropy. The cross-entropy loss function is widely used in classification problems. Based on the definition of cross-entropy, the goal of the Cross-entropy loss function is to minimize the cross-entropy between the model's distribution and the distribution of the given data.

2.4 Deep Learning

2.4.1 Introduction

Deep learning (DL) is part of the broader sector of Machine Learning and is a set of learning methods that imitate the workings of human brain in processing data. The basic elements of deep learning are artificial neural networks which are usually stacked in multiple levels and form different neural architectures. Algorithms in deep learning extract high-level features from raw data by propagating the input through the consecutive levels of such architectures. Each level serves as a function that learns to transform the input data into a representation. Deep learning has been applied to numerous fields of study, including computer vision, speech recognition, natural language processing, bioinformatics and medical image analysis. The capability of DL algorithms to find solutions to complex tasks usually surpasses the human performance.

2.4.2 Concepts

2.4.2.1 Underfitting, Overfitting, Regularization and Dropout

One of the most important challenges in Deep (and Machine) Learning is the need the proposed algorithm to perform well in new samples. Generalization is the ability of the model to generalize its performance to newly unseen data. As it is already discussed, every model in Machine Learning is trained on an initial dataset that is called training dataset. After that, it is tested in a different dataset, that is called testing dataset, and it may include samples that the model has not seen ever before. It is important that the model generalizes in those new data.

Underfitting occurs when the algorithm cannot capture the trend of the training data. More specifically, a training error is calculated in every epoch of the training process that depicts how close in the real world the model comes. This error is needed to be as low as possible. A model is said to have underfitting when this error is extremely high, as it is shown in 2.2.

Overfitting occurs when the algorithm cannot capture the trend of the testing/new data in the same degree as the training data. More specifically, a test error is also defined in the same way as the train error. If the gap between the train and the test error is too large, then the model is said to have overfitting, as it is shown in 2.2.

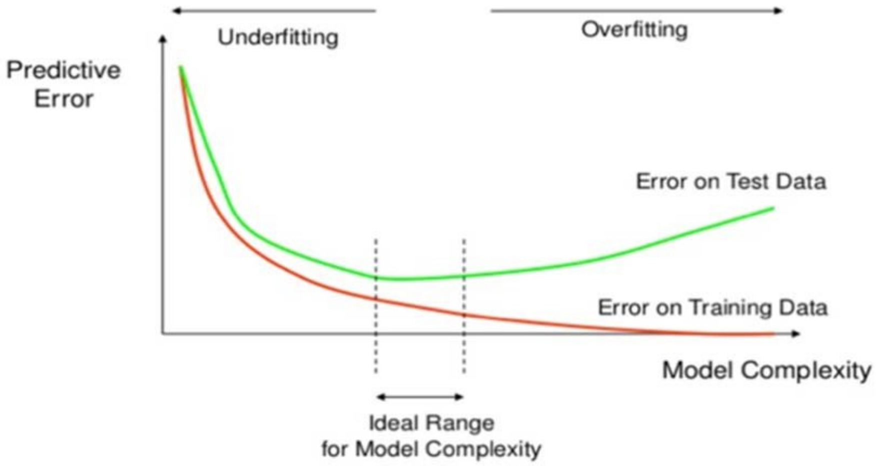


Figure 2.2. *Training and test errors behave differently. At the left end of the graph, training error and generalization error are both high. This is the underfitting regime. As we increase complexity, training error decreases, but the gap between training and generalization error increases. Eventually, the size of this gap outweighs the decrease in training error, and we enter the overfitting regime, where complexity is too large.*

There are several ways to overcome the problem of overfitting and improve the generalization. The two most common ones are Regularization and Dropout.

Regularization is the most common way to mitigate overfitting. To face the potential loss of generalization, we impose restrictions on the form of the solution by forcing the model to choose the smallest - in order of parameters- solution. This is done by implying a term in the loss equation then penalizing the size of the model. Thus, the loss function takes the following form:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta) = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L\left(f\left(\mathbf{x}_i; \theta\right), y_i\right) + \lambda R(\theta)$$

The regularization term considers the parameter values and scores their complexity. We then look for values that have both a low loss and low complexity. What regularization inherently intends to do is penalize complex models and favor simpler ones. λ is a value that must be set manually, based on the classification performance on a development set (called hyperparameter). The Regularizers R measure the norms of the parameter matrices and opt for solutions with low norms. The two most common regularization norms are L_2 and L_1.

Dropout. An effective technique for preventing neural networks from overfitting the training samples is dropout training. Dropout is designed to prevent the network from learning to rely on specific weights. The term "dropout" refers to dropping out units (hidden and visible) in a neural network. By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections, as shown in 2.3. The choice of which units to drop is random. In the simplest case, each unit is retained with a fixed probability p independent of other units, where p can be chosen using a validation set or can simply be set at 0.5, which seems to be close to optimal for a wide range of networks and tasks. For the input units, however, the optimal probability of retention is usually closer to 1 than to 0.5 [2]. The dropout technique is one of the key factors contributing to the robust results of neural networks.

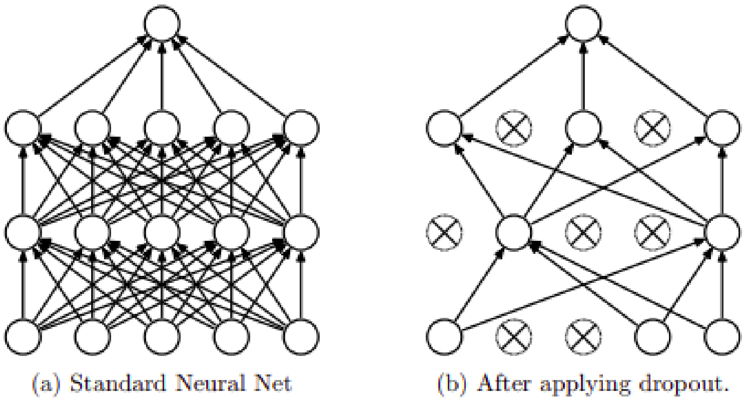


Figure 2.3. *Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped. [2]*

2.4.2.2 Activation Function

In artificial neurons, the activation function is a mathematical equation that determines the output of the neuron. It serves as a mathematical gate between the neuron's input or set of inputs, and the output that will be transmitted to the next layer. In its simplest form, it can be a binary function that turns the neuron on and off, depending on the input. It can also help normalize the output to a range between -1 and 1, or transform the input signals into output signals. Activation functions can be either linear or non-liner and each neuron in a network can have a different activation function. Some of the most popular functions are presented next.

Sigmoid Function

The sigmoid non-linearity has the mathematical form:

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

Its formula limits the output in the range between 0 and 1 and is thus used especially when a model has to predict the probability as an output. It is graphically illustrated in 2.4.

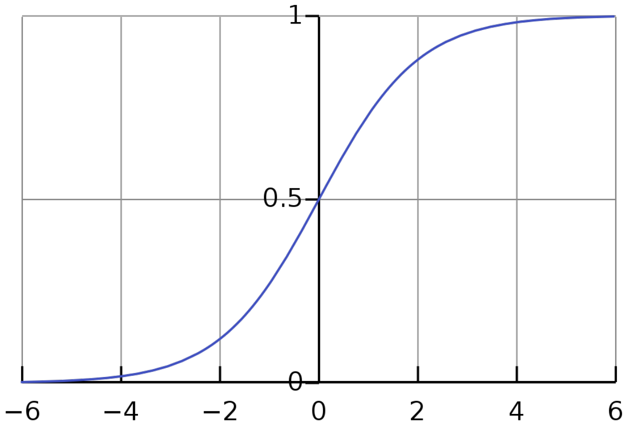


Figure 2.4. *The sigmoid function*

Softmax Function

The softmax function takes as input a vector of real numbers and normalizes it into a probability distribution. The distribution consists of a probability value in the range between 0 and 1, for each input number, and the components add up to 1. Softmax function is largely used in neural networks, where it is needed to map the outputs of the networks to a probability distribution over multiple predicted classes. The function for each $i-th$ value in the initial input vector follows this function:

$$f(x_i) = \frac{e^{x_i}}{\sum_{n=1}^{\infty} e^{x_n}}$$

It is graphically illustrated in 2.5.

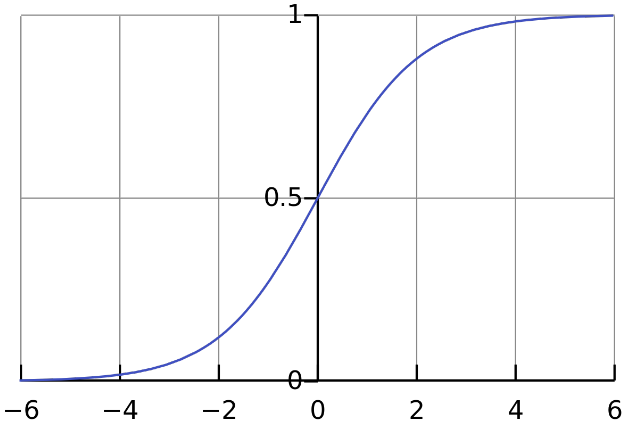


Figure 2.5. *The softmax function*

Hyperbolic Tangent (tanh)

Tanh or Hyperbolic Tangent function is shown in 2.6 and is defined as the ratio of the hyperbolic sine and hyperbolic cosine functions. Tanh follows the function:

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

As an activation function, *tanh* is mostly used to model inputs that have strongly negative and positive values as it is zero-centered. Its outputs range between -1 and 1 and is basically a rescaled sigmoid function.

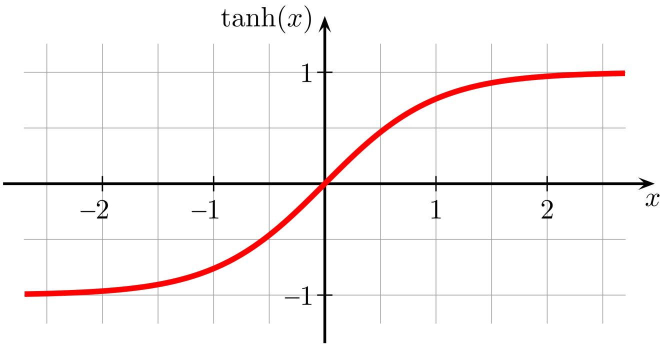


Figure 2.6. *The tanh function*

Rectified Linear Unit (ReLU)

ReLU is a non-linear function and is the most commonly used activation function in neural networks. It is a simple calculation that returns the value of the input, or 0, if the input value is less than 0. Thus, it can be defined as:

$$f(x) = max(x, 0)$$

It is graphically depicted in 2.7. It is linear for values greater than zero and non-linear for negative values, as they are always output as zero.

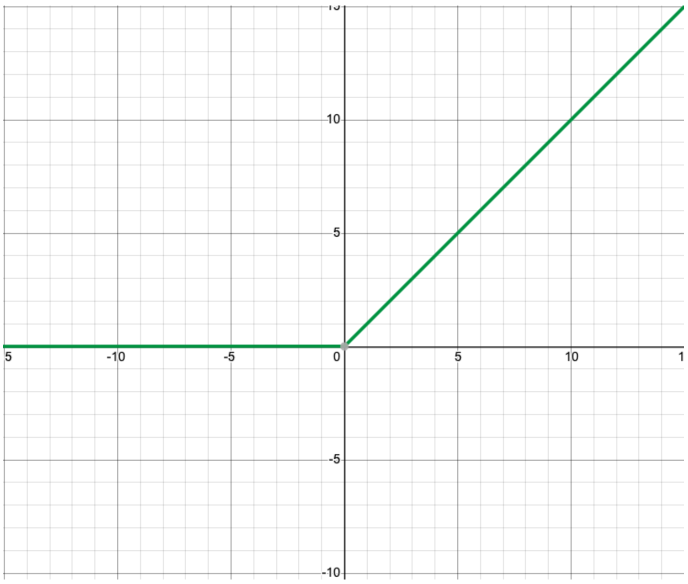


Figure 2.7. *The ReLU function*

2.4.2.3 Gradient Descent

Gradient descent (GD) is one of the most popular algorithms to perform optimization in neural networks. It computes the gradient of the cost function concerning the parameters θ for the entire dataset. The learning rate (η) is a hyperparameter that controls the extent to which the model parameters are adjusted concerning the loss gradient. GD is formally defined as:

$$\theta = \theta - \eta \nabla_{\theta} J(\theta)$$

Stochastic Gradient Descent (SGD) in contrast performs a parameter update for each training example $\mathbf{x}_i$ and label y_i :

$$\theta = \theta - \eta \nabla_{\theta} J\left(\theta; \mathbf{x}_i; y_i\right)$$

Gradient descent performs redundant computations for large datasets, as it recomputes gradients for similar examples before each parameter update. SGD does away with this redundancy by performing one update at a time. It is, therefore, usually much faster and can also be used to learn online.

2.4.2.4 Backpropagation

In order to minimize the cost function in a artificial neural network, using optimum set of values for the parameters θ (weights, also noted as w), we have to compute the gradient. Even if the computation follows the chain rule, it may get complex and difficult in more complicated networks. Fortunately, the gradients can be computed through backpropagation algorithm [35] [36]. Backpropagation computes the derivatives of a complex mathematical relation, using the chain rule and saving the results in the interim. The aim is to update the weights of the network in the optimal way after each computation of the loss function. The computation is performed for one layer of the network at a time, starting from the last layer and iterating backwards. The gradients computed can help us understand how quickly the loss function changes when the weights change and thus how well the network is performing. In this way, it is easier to fine tune the weights so as to further minimize the model's loss and improve the overall performance.

2.4.3 Models

2.4.3.1 Artificial Neural Networks

An Artificial Neural Network (ANN) is a biologically inspired computational model patterned after the network of neurons present in the human brain. The area of ANNs has been initially inspired by modeling biological neural systems but has since diverged and become a matter of engineering and achieving good results in Machine Learning tasks. Thus, we first introduce a very brief and high-level description of the biological system that has influenced a large portion of deep learning.

The basic computational unit of the brain is a neuron. Billions of neurons can be found in the human nervous system. Figure 2.8 shows the comparison between a biological neuron and its mathematical notation. Each neuron receives input signals from its dendrites and produces output signals along its (single) axon. The axon connects via synapses to the dendrites of other neurons. In the computational model of a neuron, the signals that travel along the axons (e.g., x_0) interact multiplicatively (e.g., $x_0 w_0$) with the dendrites of the other neuron based on the synaptic strength at that synapse (e.g., w_0). The idea is that the synaptic strengths (the weights w) are learnable and control the strength of influence of one neuron on another. In the basic model, the dendrites carry the signal to the cell body, where they all get summed. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon. In the computational model, we assume that only the frequency of the firing communicates information. We thus model the neuron's firing rate with an activation function f, which represents the frequency of the spikes along the axon. A standard activation function is the sigmoid function σ, since it takes a real-valued input and squashes it to a range between 0 and 1.

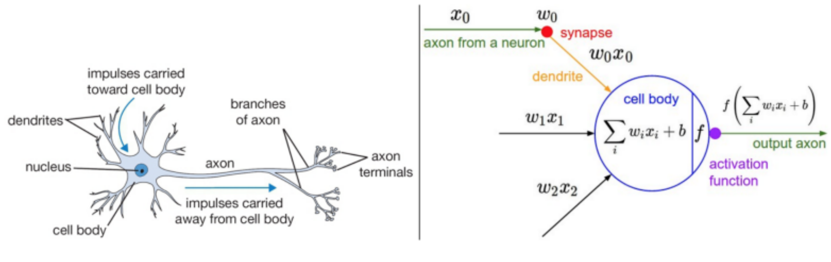


Figure 2.8. *Perceptron Structure. Source: cs231n.github.io*

In order to learn complex non-linear functions, architectures that combine several artificial neurons can be designed and implemented. Such architectures are called Multi-Layer Perceptrons (MLPs). A multilayer perceptron (MLP) is a class of feedforward artificial neural networks (FFNN). An MLP consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a non-linear activation function. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable. In Figure 2.9 we visualize the difference between a simple neural network and a deep neural network (such as an MLP or an FFNN). A deep neural network consists of more than one hidden layer.

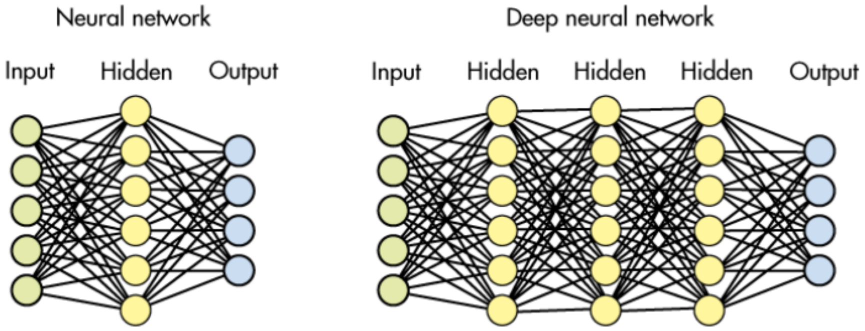


Figure 2.9. *Multilayer perceptron (MLP) Structure. Source: electronicdesign*

Each neural network is composed of the following layers:

Input layer. This layer accepts the input data. It provides information from the outside world to the network without any further computation. Nodes pass on the information to the hidden layer.

Hidden layer(s). More than one hidden layer can be used to preprocess the inputs obtained by the previous layer. They extract the required features from the input data. When moving to higher hidden layers, higher-level features are constructed.

Output layer. After data preprocessing, a decision is made by the network in this layer.

2.4.3.2 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a type of neural network that have an internal memory. It is recurrent as the output of every step is copied and sent back into the recurrent network and thus it is fed as input to the next step. Thus, the output of the current step depends on the past computations. This "memory" mechanism of RNNs is implemented using an internal hidden layer that produces a hidden state, which remembers all information about what has been calculated. The mechanism is very important for tasks as natural language generation and speech recognition, where the model needs to remember the previous words in the sentence so as to understand the context or generate a new word. It makes RNNs applicable to tasks that require to remember the history of previous inputs and outputs. The unfolded equivalent structure of a recurrent neural network is depicted in 2.10. As it is observed, the RNN takes the first input x_0 from the given sequence and produces an output h_0, which is known as the hidden state. In the next timestep, the network is given both the next input x_1 along with h_0. The procedure is continued for all timesteps in the given sequence. The hidden state at each timestep and the output are calculated as following:

$$\mathbf{h_t} = f_h\left(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x_t} + \mathbf{b_h}\right)$$
$$\mathbf{y_t} = f_y\left(\mathbf{W}_{yh}\mathbf{h_t} + \mathbf{b_y}\right)$$

where h_t is the hidden state at time step t, x_t is the input vector at time step t, y_t is the output vector at time step t, b_h is the bias for h, by is the bias for y and f_x, f_h are the activation functions for x and h respectively. They are three separate matrices of weights: W_{hx} (input-to-hidden weights), W_{hh} (hidden-to-hidden), and W_{yh} (hidden-to-output).

Long Short-Term Memory (LSTM). A subcategory of Recurrent Neural Networks (RNNs) described above, are the LSTMs. They were originally proposed by Hochreiter and Schmidhuber in 1997 [37]. They were proposed in order to overcome the vulnerability of RNNs. In particular, when training a RNN using backpropagation, the gradients which are back-propagated can "vanish"

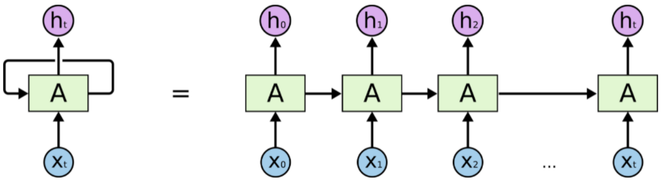


Figure 2.10. *An unrolled up RNN where X_t is the input vector containing sequences of characters of a words while h_t is as output vector. Source: colah.github.io*

(that is, they can tend to zero) or “explode” (that is, they can tend to infinity), because of the computations involved in the process, which use finite-precision numbers, a phenomenon called as "gradient vanishing". Adding the LSTM to the network is like adding a memory unit inside the network that can remember context from the very beginning of the input. LSTM is denoted in Figure 2.11.

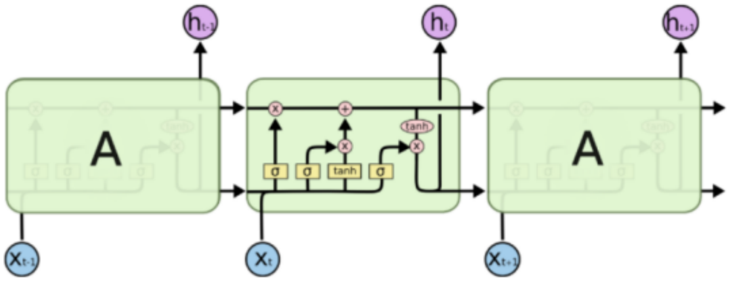


Figure 2.11. *The repeating module in an LSTM. Source: colah.github.io*

Given a sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_n$ of vectors of an input sequence of length n, for vector $\mathbf{x}_t$, with inputs $\mathbf{h}_{t-1}$ and $\mathbf{c}_{t-1}, \mathbf{h}_t$ and $\mathbf{c}_t$ are computed as follows:

$$\begin{aligned}
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f\right) \\
\mathbf{i}_t &= \sigma\left(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o\right) \\
\mathbf{u}_t &= \tanh\left(\mathbf{W}_u \mathbf{x}_t + \mathbf{U}_u \mathbf{h}_{t-1} + \mathbf{b}_u\right) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{u}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right)
\end{aligned}$$

Forget gate (f_t)**.** This gate decides what information should be thrown away or kept. Information from the previous hidden state h_{t-1} and information from the current input x_t is passed through the sigmoid activation function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

Input gate (fi_t)**.** The previous hidden state and current input are passed into a sigmoid function that decides which values will be updated by forcing the values to be between 0 and 1 (0 means unimportant and 1 means important). The hidden state and current input are also passed to the tanh function to squish values between -1 and $1\,(u_t)$. Finally, the tanh output is multiplied with the sigmoid output $(i_t \odot u_t)$. The sigmoid output will filter the important information of tanh.

Cell state (c_t)**.** The cell state gets pointwise multiplied by the forget vector. This can drop values in the cell state if it gets multiplied by values near 0. Then, we take the output from the input gate and do a pointwise addition that updates the cell state to new values that the neural network finds relevant.

Output gate (o_t). The output gate decides what the next hidden state should be. As the hidden state contains information on previous inputs, it is also used for predictions. First, the previous hidden state and the current input are passed into a sigmoid function. Then, the newly modified cell state is passed to the tanh function. We multiply the tanh output with the sigmoid output ($o_t \odot \tanh(\mathbf{c}_t)$) to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step.

Gated Recurrent Unit (GRU). The Gated Recurrent Unit (GRU) [38] is a variant of an LSTM, in the sense that it includes fewer parameters and a forget gate [39] yet it lacks an output gate and a cell state. The performance of the two types of networks is found to be generally equivalent. GRUs effectively solve the vanishing gradient problem using the update and reset gates, which decide the information that should pass to the output.

Update gate (z_t). The update gate acts similar to the forget gate and input gate of an LSTM. The input x_t at timestep t is added to the hidden state from the previous timesteps and the result is passed through a sigmoid activation function. The result is thus squashed between 0 and 1. To this end, the update gate determines how much of the previous information needs to be passed along to the future.

Reset gate (r_t). The reset gate determines how much of the past information should the model forget. The procedure is the same as the update gate, in the sense that the current input is added to the previous hidden state.

The equations that describe the function of a GRU are the following:

$$\begin{aligned}
\mathbf{z}_t &= \sigma\left(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{x}_t + \mathbf{b}_z\right)\\
\mathbf{r}_t &= \sigma\left(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{x}_t + \mathbf{b}_r\right)\\
\mathbf{u}_t &= \tanh\left(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h\right)\\
\mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \mathbf{u}_t
\end{aligned}$$

where matrices W and U contain the weights of the input and recurrent connections of the network.

2.4.3.3 Attention Mechanisms

Even if LSTMs solved the problem of gradients vanishing partially, the passing of the infromation through a series of recurrent connections evoke information loss. Moreover, the inherently sequential nature of recurrent networks makes it hard to do computation in parallel. A modern solution to this came by Attention Mechanism, which was firstly proposed by Bahdanau et al. [14]. Attention has become enormously popular within the Artificial Intelligence community as an essential component of neural architectures for many applications in Natural Language Processing, Speech, and Computer Vision. The key contribution of the mechanism is that it helps models to direct their focus and pay greater attention to certain factors when processing the data. Attention is the key module of that is called "transformers" and we will see in the next section.

Self-attention. Self-attention, also known as intra-attention, is an attention mechanism Attention can be also applied in a single sentence when there is no additional information, by allowing it to attend to itself using self-attention and it is depicted in 2.12.

For the computation of a specific output, we need the corresponding input and everything preceded that. For example, if we need to compute y_3 as it is depicted in 2.12, we need x_1, x_2, x_3 and to calculate the similarity of x_3 with x_1, x_2 and x_3 itself. The simplest form of comparison

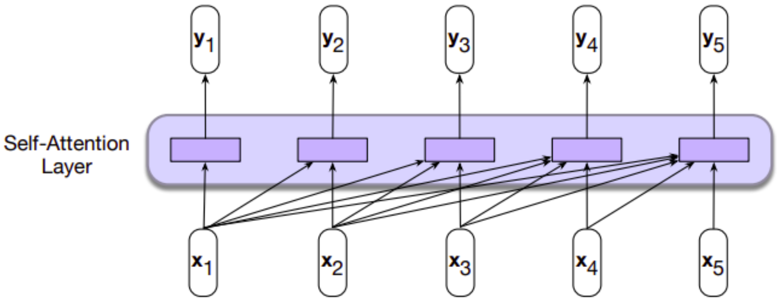


Figure 2.12. *Information flow in a causal (or masked) self-attention model. In processing each element of the sequence, the model attends to all the inputs up to, and including, the current one. Unlike RNNs, the computations at each time step are independent of all the other steps and therefore can be performed in parallel. Source: [1]*

between elements in a self-attention layer is a dot product. In every "step", there is an element-focus (which in the above example is x_3). After the calculation of the scores of every set, we use a softmax layer that indicates the proportional relevance of each input to the target element. Finally, given these scores, we can compute our final output by taking the average of the inputs that influence this output weighted by the scores generated by softmax. Formally, we have :

$$\text{score}(x_i, x_j) = x_i \cdot x_j$$
$$\alpha_{ij} = \text{softmax}(\text{score}(x_i, x_j)) = \frac{\exp(\text{score}(x_i, x_j))}{\sum_{k=1}^{i} \exp(\text{score}(x_i, x_k))}, \forall j \leq i$$
$$y_i = \sum_{j \leq i} \alpha_{ij} x_j$$

As we can notice, every input embedding (x_i) can play three different roles in the process. According to this, we can define the key, query, value definitions that are the key concepts of the attention mechanism.

1. **query** is the target element of the process. This element is then compared to every preceding input.

2. **key** is every preceding input that is compared to the current target element.

3. **value** is the element that is used to compute the final output.

In order to incorporate the above in the attention mechanisms, three matrices are introduced : $\mathbf{W^Q}$, $\mathbf{W^K}$ and $\mathbf{W^V}$. The process is parallelized and therefore the matrices' multiplication take place at the same time. Having the input sequence into a single matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$, we can produce matrices $\mathbf{Q} \in \mathbb{R}^{N \times d}$, $\mathbf{K} \in \mathbb{R}^{N \times d}$ and $\mathbf{V} \in \mathbb{R}^{N \times d}$ by :

$$\mathbf{Q} = \mathbf{XW^Q}; \mathbf{K} = \mathbf{XW^K}; \mathbf{V} = \mathbf{XW^V}$$

A typical approach for the computation of the score function is to multiply the $\mathbf{Q}$ with $\mathbf{K^T}$ and divide it by the dimensionality of the key and query vectors, in order to avoid the large values of the exponential that can lead to an effective loss of gradients during training. Finally, self-attention is computed as :

$$\text{SelfAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{QK^T}}{\sqrt{d_k}})\mathbf{V}$$

Multi-head attention. Multi-head attention is an another attention architecture, which consists of several attention layers running in parallel. This mechanism allows the model to jointly attend to information from different representation subspaces at different positions. It is found that multi-head attention works better than single-head, as it applies the usual attention mechanism to multiple chucks in parallel, and then concatenates the results. To implement this, we have several heads and different query, key, value matrics for every head : $\mathbf{W_i^Q}$, $\mathbf{W_i^K}$ and $\mathbf{W_i^V}$, where i denotes the head. This is illustrated by 2.13.

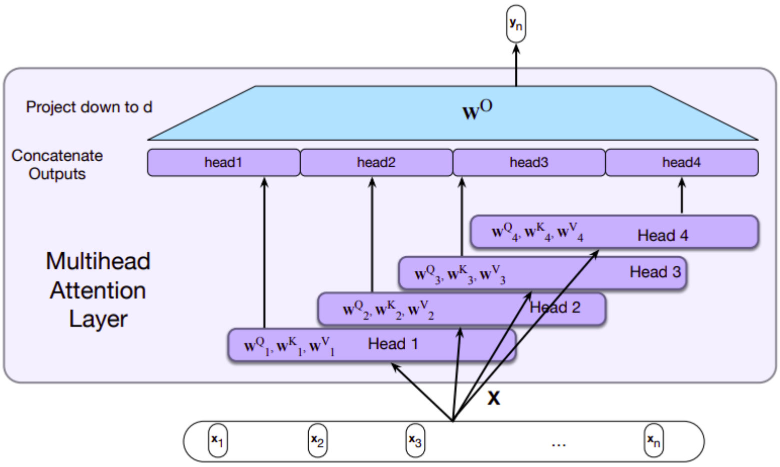


Figure 2.13. *Multihead self-attention: Each of the multihead self-attention layers is provided with its own set of key, query and value weight matrices. The outputs from each of the layers are concatenated and then projected down to d, thus producing an output of the same size as the input so layers can be stacked. Source: [1]*

2.4.3.4 Transformers

The Transformer is a deep learning model introduced by Vaswani et al. [40] used primarily in Natural Language Processing. The Transformer is based in the Attention Mechanism idea, discussed in the previous subsection 2.4.3.3. Like RNNs, Transformers are designed to handle sequential data, however do not require that the sequential data be processed in order. Since their introduction, Transformers have become the model of choice for tackling many problems in NLP. As Transformers allows for much more parallelization than RNNs during training, it has enabled training on larger datasets and led to the development of pretrained systems such as BERT (Bidirectional Encoder Representations from Transformers) that is discussed later, which have been trained with huge general language datasets, and can be fine-tuned to specific language tasks.

The Transformer is an encoder decoder architecture. The encoder consists of a set of encoding layers that processes the input iteratively one layer after another and the decoder consists of a set of decoding layers that does the same thing to the output of the encoder. Encoders and decoders, consists of identical subsystems, that have different parameter values. Encoders and decoders accept inputs from the lower layers and carry them to the higher ones. Each encoder consists of two subsystems. The first is a self-attention layer, that allows the encoder to hold the dependencies that the input under study may have with the other inputs in the sequence. The second is a feed forward neural network for additional processing of the outputs, and residual connections and layer normalization units. Similar is the decoders architecture with an addition of a intermediate encoder-decoder attention mechanism. This subsystem is responsible to identify and focus on specific elements of the input that has already been encoded by the encoder. The simple form of the transformer is presented in 2.14.

In Transformer, there are three types of attention in terms of the source of queries and key-value

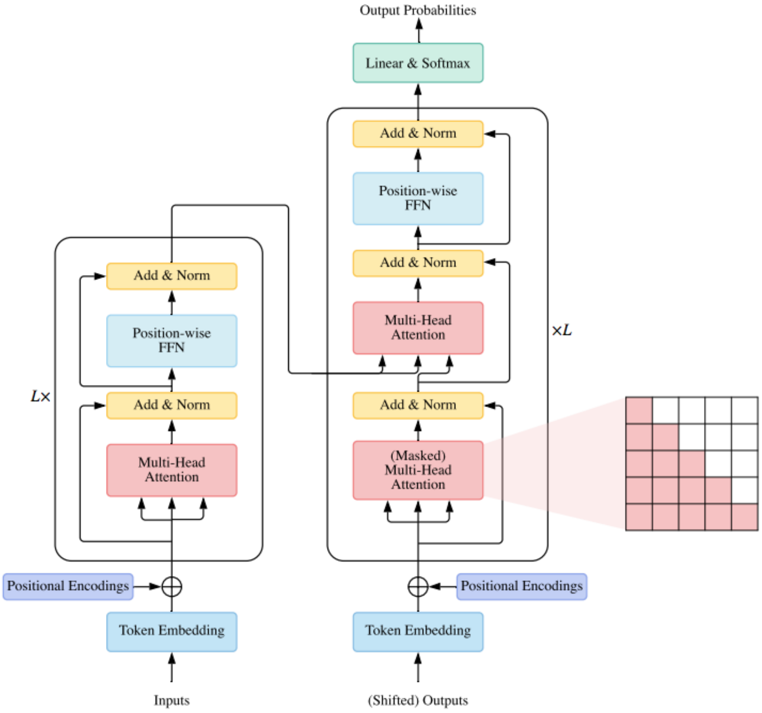


Figure 2.14. *Overview of the simple Transformer architecture. Source: [3]*

pairs:

1. **Self-attention.** In Transformer encoder, we set $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{X}$, where $\mathbf{X}$ is the outputs of the previous layer.

2. **Masked Self-attention.** In the Transformer decoder, the self-attention is restricted such that queries at each position can only attend to all key-value pairs up to and including that position. To enable parallel training, this is typically done by applying a mask function to the unnormalized attention matrix $\hat{A} = \exp\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)$, where the illegal positions are masked out by setting $\hat{A}_{ij} = -\infty$ if $i < j$. This kind of self-attention is often referred to as autoregressive or causal attention.

3. **Cross-attention.** The queries are projected from the outputs of the previous (decoder) layer, whereas the keys and values are projected using the outputs of the encoder.

Another issue to be addressed is the position of inputs. Some vectors need to be introduced which will give the system a sense of order in the input sequence. This vectors are known as positional embeddings. Those embeddings are added to the corresponding input embeddings, and converts the input to vectors with internal representation of time characteristics. The first encoder takes positional information and embeddings of the input sequence as its input, rather than encodings.

2.5 Dimensionality Reduction

There are also plenty of cases where feature vectors contain redundant information. Our goal is to get compact and informative representations that contain all the useful properties of the data, while having the less possible dimensions. This is achieved by transforming the given features to a new set of less features, which will exhibit high information packing properties. This procedure is referred to as **dimensionality reduction**. There are many dimensionality reduction techniques, such as Principal Component Analysis, Independent Component Analysis, Singular Value Decomposition etc. Here, we will focus on Principal Component Analysis, which is one of the most widely used.

2.5.1 Principal Component Analysis

PCA is a linear orthogonal transformation, whose goal is to give lower-dimensional data, while preserving as much of the data's variation as possible. In general, a desirable property of features is to be mutually uncorrelated, in order to avoid information redunduncies. In this way, PCA is basically a transform that wants to ensure the newly generated features are uncorrelated.

Let x be a zero-mean random variable. Suppose we want the direction w such that the projection of x along this direction has maximum variance:

$$\max_{w}(w'x) \quad \text{st.} \quad w = 1.$$

We have

$$(w'x) = w'xx'w = w'\Sigma w.$$

The Lagrangian is

$$L = w'\Sigma w + \lambda(w'w - 1).$$

The stationary condition is

$$\frac{\partial L}{\partial w} = 2\Sigma w - 2\lambda w = 0, \Sigma w = \lambda w.$$

Thus w is an eigenvector of Σ. Since

$$w'\Sigma w = w'(\lambda w) = \lambda,$$

the direction with maximum variance is the largest eigenvector. This procedure can be iterated to get the second largest variance projection (orthogonal to the first one), and so on. For a set of data points, we use the ML estimate of the covariance matrix.

High-Dimensional Data Let X be the $d \times n$ matrix of high-dimensional points ($d > n$). Instead of explicitly estimating the covariance matrix, we use the following trick:

$$\begin{aligned}\Big(\frac{1}{n}XX'\Big)W &= W\Lambda \\ \Big(\frac{1}{n}X'X\Big)X'W &= X'W\Lambda \\ \Big(\frac{1}{n}X'X\Big)V &= V\Lambda\end{aligned}$$

We've reduced the problem to finding the eigen-decomposition of $\frac{1}{n}X'X$, which is $n \times n$, instead of $d \times d$. By re-projecting,

$$\begin{aligned}\frac{1}{n}X(X'X)F &= XFV \\ \Big(\frac{1}{n}XX'\Big)(XF) &= (XF)V\end{aligned}$$

so XF are the eigenvectors of $\frac{1}{n}XX'$. Since

$$\begin{aligned}\frac{1}{n}X'XF &= FV\\ (F'X')(XF) &= nF'FV\\ (XF)'(XF) &= nV\end{aligned}$$

we have $E = \frac{1}{\sqrt{n}}XFV^{-1/2}$.

Maximum Information Preservation Let

$$y = w'x,$$

where $w = 1$.

We want to maximize $I(X;Y)$ or $H(Y)$ under the constraint on w:

$$J = 1/2(\log(2\pi e)^D + \log|Cy|) + aw'w.$$

$$Cy = [w'xx'w] = w'[xx']w = w'Cw.$$

Thus we maximize

$$\begin{aligned}L &= \log|w'Cw| + aw'w\\ dL/dw &= 2Cw/|w'Cw| + 2aw\\ Cw &= a|w'Cw|w\end{aligned}$$

Thus, w is an eigenvector of the covariance matrix C of x, and the maximum occurs with the eigenvector with the largest eigenvalue.

Maximum Likelihood The data matrix is modeled as linear combinations of a small set of basis vectors plus noise.

$$X = UV + Y$$

X is DxN, where each column is a datum. U is DxK, where each column is a factor. V is KxN, where each column is a vector of coefficients. Y is DxN noise.

Assuming normal noise with equal variance for all points, ML estimation of UV gives a least squares cost function.

$$J = (X - UV)^2.$$

The stationary conditions are

$$\begin{aligned}dJ/dU &= (X - UV)V' = 0\\ dJ/dV &= U'(X - UV) = 0\\ XV' &= UVV'\\ U'X &= U'UV\end{aligned}$$

To make the solution unique against rotations and scalings of U and V, we constrain U'U = I and V'V diagonal:

$$\begin{aligned}V &= U'X\\ XX'U &= SU\\ 1/NXX'U &= S/NU\end{aligned}$$

The basis functions are the eigenvectors of the covariance matrix, assuming X is zero mean .

Generalized PCA To generalize PCA to other data the noise distribution can be changed and a link function added:

$$X \sim p(f(g(A)h(B))).$$

2.6 Transfer Learning

In machine learning (and in particular deep learning), a ubiquitous problem is that models which solve complex problems need vast amounts of data. Nevertheless, getting these amounts of data for supervised models can become unfeasible due to time restrictions or computational limitations. Moreover, models that are trained on small, specific datasets face a performance drop, when they are used to tackle a different task, which might still be similar to the one they were trained on. The goal of transfer learning is to improve learning in the target task by leveraging knowledge from the source task.

In transfer learning, the concepts of a domain and a task are used. A domain $\mathcal{D}$ is composed of two parts, i.e., a feature space $\mathcal{X}$ and a marginal distribution $P(X)$. In other words, $\mathcal{D} = \{\mathcal{X}, P(X)\}$. And the symbol X denotes an instance set, which is defined as $X = \{\mathbf{x} \mid \mathbf{x}_i \in \mathcal{X}, i = 1, \ldots, n\}$. A task $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and a decision function f, i.e., $\mathcal{T} = \{\mathcal{Y}, f\}$. The decision function f is an implicit one, which is expected to be learned from the sample data.

Given some observation(s) corresponding to $m^S \in \mathbb{N}^+$ source domain(s) and task(s) (i.e., $\left\{(\mathcal{D}_{S_i}, \mathcal{T}_{S_i}) \mid i = 1, \ldots, m^S\right\}$), and some/an observation(s) about $m^T \in \mathbb{N}^+$ target domain (s) and task (s) (i.e., $\left\{(\mathcal{D}_{T_j}, \mathcal{T}_{T_j}) \mid j = 1, \ldots, m^T\right\}$), transfer learning utilizes the knowledge implied in the source domain(s) to improve the performance of the learned decision functions $f^{T_j}(j = 1, \cdots, m^T$) on the target domain(s).

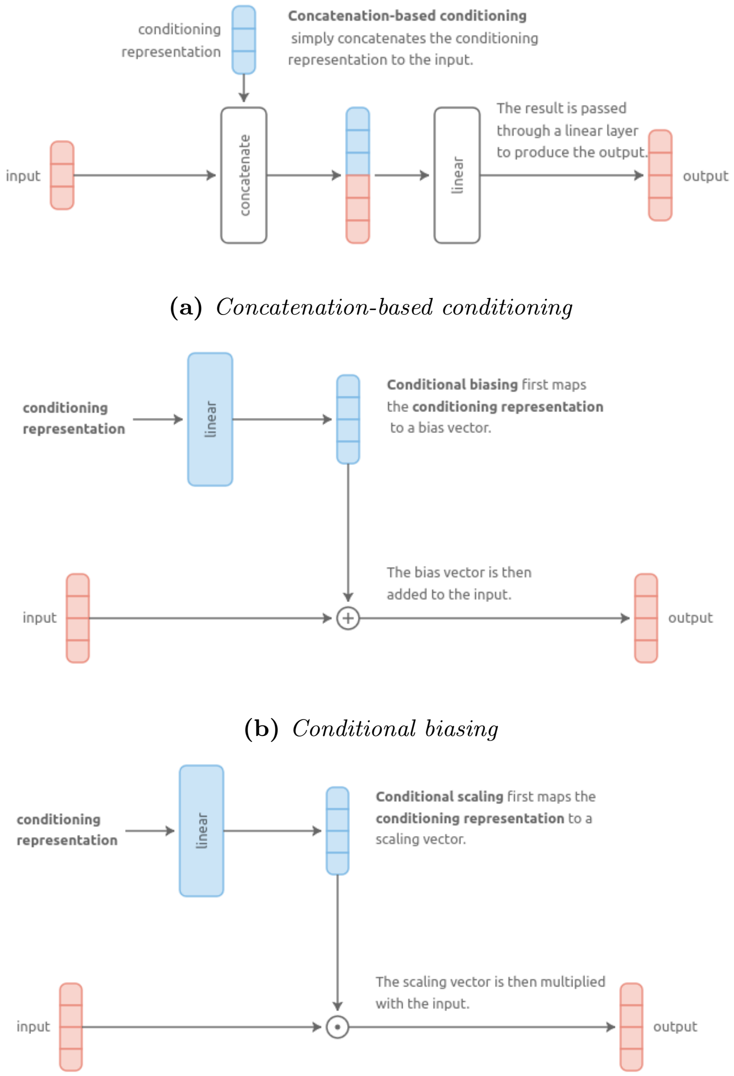
If m^S equals 1 , the scenario is called single-source transfer learning. Otherwise, it is called multi-source transfer learning. Besides, m^T represents the number of the transfer learning tasks where in the majority of scenarios $m^T = 1$ [41].

2.7 Conditioning

Many real-world problems require integrating multiple sources of information. Sometimes these problems involve multiple, distinct modalities of information—vision, language, audio, etc.—as is required to understand a scene in a movie or answer a question about an image. Other times, these problems involve multiple sources of the same kind of input, i.e. when summarizing several documents or drawing one image in the style of another. When approaching such problems, it often makes sense to process one source of information in the context of another; for instance, in the right example above, one can extract meaning from the image in the context of the question. In machine learning, we often refer to this context-based processing as conditioning: the computation carried out by a model is conditioned or modulated by information extracted from an auxiliary input [4].

Finding an effective way to condition on or fuse sources of information is an open research problem. These conditioning methods are often called feature-wise transformations. In the language of multi-task learning, where the conditioning signal is taken to be a task description, feature-wise transformations learn a task representation which allows them to capture and leverage the relationship between multiple sources of information, even in remarkably different problem settings.

The most common method of conditioning is concatenate a representation of the conditioning information to an input, hidden, or output layer of a deep neural network. This approach is quite parameter-efficient, as we only need to increase the size of the layer's weight matrix. However, this approach makes the implicit assumption that we know where the model needs to use the conditioning information. It could be in the any layer of the network. Because this operation is cheap, we might as well avoid making any such assumptions and concatenate the conditioning representation to the input of all layers in the network. This method is called concatenation-based conditioning and it is illustrated in 2.15a. Another efficient way to integrate conditioning information into the network is via conditional biasing (2.15b), namely, by adding a bias to the hidden layers based on the conditioning representation. Interestingly, conditional biasing can be thought of as another way to implement concatenation-based conditioning. The two methods are equivalent. Another efficient way to integrate class information into the network is via conditional scaling, i.e., scaling hidden layers based on the conditioning representation. This method is illustrated in 2.15c.


conditioning representation
input
concatenate
linear
output
conditioning representation
linear
input
output
conditioning representation
linear
input
output

Concatenation-based conditioning simply concatenates the conditioning representation to the input.

The result is passed through a linear layer to produce the output.

(a) *Concatenation-based conditioning*

Conditional biasing first maps the **conditioning representation** to a bias vector.

The bias vector is then added to the input.

(b) *Conditional biasing*

Conditional scaling first maps the **conditioning representation** to a scaling vector.

The scaling vector is then multiplied with the input.

(c) *Conditional scaling*

Figure 2.15. *Feature-wise transformations. Source : [4]*

Chapter 3

Natural Language Processing

3.1 Introduction

Natural Language Processing (NLP) is a subfield of Artificial Intelligence that concerns the interaction between computers and human language. NLP is the technology used to help computers understand, interpret and even generate natural language data. NLP is a challenging and quite demanding task. This is due to the nature of human language that makes the task difficult. The rules on which languages are based are not easy to be understood by machines. Some of these rules can be low-leveled, yet some others are more abstract. To this end, in order to understand the language, machines have to know not only the words but also the whole meaning behind them.

The hierarchical levels of language that NLP examines are:

Phonology. This level deals with the interpretation of speech sounds within and across words. There are, in fact, three types of rules used in the phonological analysis: 1) phonetic rules – for sounds within words; 2) phonemic rules – for variations of pronunciation when words are spoken together, and; 3) prosodic rules – for fluctuation in stress and intonation across a sentence.

Morphology. This level deals with the componential nature of words composed of morphemes, the smallest units of meaning.

Lexical. At this level, humans, as well as NLP systems, interpret the meaning of individual words. Several types of processing contribute to word-level understanding, the first of these being assignment of a single part-of-speech tag to each word. In this processing, words that can function as more than one part-of-speech are assigned the most probable part-of-speech tag based on the context in which they occur.

Syntactic. This level focuses on analyzing the words in a sentence to uncover the grammatical structure of the sentence.

Semantic. Semantic processing determines the possible meanings by focusing on the interactions among word-level meanings in the sentence. This level of processing can include the semantic disambiguation of words with multiple senses, in an analogous way to how syntactic disambiguation of words can function as multiple parts-of-speech is accomplished at the syntactic level. Semantic disambiguation permits one and only one sense of polysemous words to be selected and included in the semantic representation of the sentence.

Pragmatic. This level is concerned with the purposeful use of language in various situations. The goal is to explain how extra meaning is read into texts without being encoded in text. This level requires world knowledge, including the understanding of intentions, plans, and goals.

Discourse. While syntax and semantics work with sentence-length units, the discourse level of NLP works with units of text longer than a sentence. It does not interpret multi-sentence texts as just concatenated sentences, each of which can be interpreted singly. Rather, discourse focuses on the properties of the text that convey meaning by making connections between component sentences.

3.2 Applications

Natural Language Processing provides implementations for any task that utilizes text. The most common applications of NLP are:

Information Retrieval and Web Search: the science of searching for documents, for information within documents, and metadata about documents, as well as searching databases and the World Wide Web.

Information Extraction (IE): the recognition, tagging, and extraction into a structured representation, certain critical elements of information, e.g., persons, companies, locations, organizations, from extensive collections of text.

Text Summarization: the process of distilling the essential information from a source in order to produce an abridged version.

Question Answering (QA): the response from documents to extracted or generated answer.

Machine Translation (MT): the use of computer software in order to translate text or speech from one natural language to another.

Speech Recognition and Synthesis: the extraction of a textual representation of a spoken utterance.

Text Generation: A method for generating sentences from "keywords".

Natural Language Understanding and Generation (NLU, NLG): NLG system is like a translator that converts a computer-based representation into a natural language representation.

Natural Language Inference (NLI) is the task of determining whether a "hypothesis" is true (entailment), false (contradiction), or undetermined (neutral) given a "premise".

3.2.1 Sentiment Analysis

Sentiment analysis refers to the usage of natural language processing, computational linguistics, text analysis and biometrics to identify, analyze and extract subjective information (people's opinions, sentiments, emotions and evaluations) towards entities as products, businesses, organizations, topics or other people. It detects polarity within a given text, which means positive, neutral or negative feelings. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

Sentiment analysis is considered, nowadays, one of the hottest areas in computer science. This is because of the advantages in understanding the sentiment in a piece of text as it can contain the sentimental intention of a person towards something (information that can be exploited into recommendation systems etc). There is a huge explosion today of 'sentiments' available from social media including Twitter, Facebook, message boards, blogs, and user forums. These snippets of text are a gold mine for companies and individuals that want to monitor their reputation and get timely feedback about their products and actions. Sentiment analysis offers these organizations the ability to monitor the different social media sites in real time and act accordingly. Marketing managers, PR firms, campaign managers, politicians, and even equity investors and online shoppers are the direct beneficiaries of sentiment analysis technology [42].

3.2.2 Emotion Recognition

Emotion Recognition is the process of identifying human emotion through the expression, such as fear, anger, happiness, sadness, surprise and disgust, and is closely related to Sentiment Analysis. Emotions can be detected in human non-verbal cues, such as facial expressions from video, on spoken expressions from audio and on written expressions from text. Hence, extracting and understanding emotion is of high importance to the interaction between human and machine communication. Although it is a recent field of research, it has gained much popularity due to the ability of machines to make decisions based on objective rules, in contrast to humans who vary widely in their accuracy at recognizing emotions. In the case of conversations, emotion recognition extracts opinions between participants from conversational data which can be found on social media platforms.

3.2.3 Sarcasm Detection

Sarcasm is considered one of the most useful NLP tasks in Sentiment Analysis. The goal of Sarcasm Detection is to determine whether a sentence is sarcastic or non-sarcastic. Sarcasm is a type of phenomenon with specific perlocutionary effects on the hearer, such as to break their pattern of expectation. Consequently, correct understanding of sarcasm often requires a deep understanding of multiple sources of information, including the utterance, the conversational context, and, frequently some real world facts. Sarcasm poses a major challenge for sentiment analysis models, mainly because sarcasm enables one speaker or writer to conceal their true intention of contempt and negativity under a guise of overt positive representation. Thus, recognizing sarcasm and verbal irony is critical for understanding people's actual sentiments and beliefs. The figurativeness and subtlety inherent in its sentiment display, a positive surface with a contemptuous intent (e.g., "He has the best taste in music!"), or a negative surface with an admiring tone (e.g. , "She always makes dry jokes!"), makes the task of its identification a challenge for both humans and machines.[43]. Moreover, the figurative language is a good indicator for a variety of different tasks, as also depression detection, because it is usual for depressive people to use sarcasm in order to describe their daily routine.

3.2.4 Personality Detection

Personality detection is the task of deciding the personality of a person according to five different axes/traits. Personality is a combination of an individual's behavior, emotion, motivation, and thought pattern characteristics. Our personality has great impact on our lives; it affects our life choices, well-being, health, and numerous other preferences. That is the reason why the detection of our personality is really important nowadays. Having this knowledge, the products and services recommended to a person can be affected by our personality traits, comparing the people with other people who prefer these and have similar traits. Personality detection is considered, also, a subfield of sentiment analysis, as it is a helpful task of deciding the sentiment. Personality is typically formally described in terms of the Big Five personality traits,3 which are the following binary (yes/no) values [44]:

- **Extroversion (EXT).** Is the person outgoing, talkative, and energetic versus reserved and solitary?

- **Neuroticism (NEU).** Is the person sensitive and nervous versus secure and confident?

- **Agreeableness (AGR).** Is the person trustworthy, straightforward, generous, and modest versus unreliable, complicated, meager, and boastful?

- **Conscientiousness (CON).** Is the person efficient and organized versus sloppy and careless?

- **Openness (OPN).** Is the person inventive and curious versus dogmatic and cautious?

3.3 Word Representation

One of the most challenging concepts in NLP is the word representation. The input of NLP models is usually words and it is necessary to represent them in numbers that can indicate their meaning as a word or as a member of a set, a sentence. The role of context of a sentence is really important in the word representation as words in similar contexts tend to have similar meanings also. This is a field that has developed over the years and the methods have been evolved towards a direction where every vector of a word can include more meaningful representation of that word involving different aspects, like context, semantics etc.

There are two ways that Linguists think about meaning, one is through "Denotational Semantics" which is the concept of representing an idea as a symbol (a word or a one-hot vector), and the other is through "Distributional Semantics" which is the concept of representing the meaning of a word based on the context in which it usually appears. Distributional Semantics include methods that are really sparse i.e. mostly zeros, but a lot of work has been done to dense vectors that capture useful semantic properties.

3.3.1 Denotational Representation

3.3.1.1 Vocabulary IDs

Let us assume that we have a vocabulary V in a given task, containing all the words in the training, development, and test dataset. The most naive approach is to match all the words of the vocabulary with a unique ID symbol, starting from 1 and ending to the dimension of V, that equals to the total of different words that are included in the sets. For example for the given vocabulary $V = \{'I',' got',' covid'\}$, we can define the following mapping:

$$w^{\mathrm{I}} = 1, w^{got} = 2, w^{covid} = 3$$

This approach does not encapsulate the word's meaning through the representation, and neither a notion of similarity and difference between words nor the ambiguity problem is solved. Moreover, this representation is computationally expensive as it uses $|V|$ different IDs. Finally, this approach is not capable of any further improvement.

3.3.1.2 One-Hot-encoding

One-Hot-Encoding is equivalent to the Vocabulary IDs, where we represent every word with an $\mathbb{R}^{1\times 1}$ vector. The only difference is the dimensionality augmentation of the subspace to achieve further improvement through compression techniques. In One-Hot-Encoding, every word is represented as an $\mathbb{R}^{|V|\times 1}$ vector with all 0 and one 1 at the index of that word in the sorted English language. So, for example, word vectors in this type of encoding would appear as the following:

$$w^{\mathrm{I}} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, w^{got} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, w^{covid} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

We represent each word as a completely independent symbol. As we previously discussed, this word representation does not directly give us a notion of similarity and difference between words, nor the ambiguity problem is solved. For instance,

$$\left(w^{\text{covid}}\right)^T w^{\text{H1N1}} = \left(w^{\text{fever}}\right)^T w^{\text{covid}} = 0$$

The corresponding matrix that describes the method will be of $|V| \times |V|$ dimension and in the following form:

$$\begin{bmatrix} w^{\mathrm{I}} \\ w^{\mathrm{got}} \\ w^{\mathrm{covid}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Motivated by the fact that this matrix is sparse, we can apply many compression techniques such as Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF), but the final result should also be classified as "Denotational Semantics".

3.3.2 Distributional Semantics

As it is referred to [1] the idea of semantics is to represent a word as a point in a multidimensional space that is derived from the distributions of word neighbors. Vectors for representing words are called **embeddings**, although the term is sometimes more strictly applied only to dense vectors rather that sparse.

3.3.2.1 Sparse word vectors

In sparse word vectors, the representation of a word is very sparse, i.e. including mostly zeros. The two most commonly used models are tf-idf and pmi.

Term Frequency–Inverse Document Frequency (TF-IDF) Representation. The general idea of TF-IDF representation is that the meaning of a word is defined by a simple function of the counts of nearby words. In particular, there is a paradox that TF-IDF tries to solves : Words that occur nearby frequently (maybe pie nearby cherry) are more important than words that only appear once or twice. Yet words that are too frequent—ubiquitous, like the or good— are unimportant [1].

TF proposed by Luhn [45] is the frequency of the word in the document. Usually, we use the count as the tf:

$$\mathrm{tf}_{t,d} = \mathrm{count}(t, d)$$

or a more squash term:

$$\mathrm{tf}_{t,d} = \log_{10}(\mathrm{count}(t, d) + 1)$$

The idf proposed by Sparck Jones [46] is defined as N/df_t , where N is the total number of documents, and df_t is the number of documents in which term t appears. The fewer documents in which a term occurs, the higher this weight. The lowest weight of 1 is assigned to terms that occur in all the documents. Usually, we use a more squash term for idf:

$$\mathrm{df}_t = \log_{10}(N/\mathrm{df}_t)$$

The tf-idf weighted value $w_{t,d}$, and the corresponding word vector, is the product of this two terms:

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

Using tf-idf we represent every word with an $\mathbb{R}^{|D|\times 1}$ vector, where $|D|$ is the number of different documents in the collection and in general case $|D| < |N|$. Thus, the word representation is smaller than One-Hot-Vector and should contain some information about the word meaning, but tf-idf does not give us directly neither a notion of similarity and difference between words nor the ambiguity problem is solved.

Pointwise Mutual Information (PMI) Representation. Pointwise mutual information proposed by Fano [47] is one of the essential concepts in NLP. It is a measure of how often two events x and y occur, compared with what we would expect if they were independent:

$$I(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

The pointwise mutual information, proposed by Church and Hanks [48] is then defined as:

$$\text{PMI}(w, c) = \log_2 \frac{P(w, c)}{P(w)P(c)}$$

The numerator tells us how often we observed the two words together. The denominator tells us how often we would expect the two words to co-occur assuming they each occurred independently. Thus, the ratio gives us an estimate of how much more the two words co-occur than we expect by chance. PMI is a useful tool whenever we need to find words that are strongly associated.

3.3.2.2 Dense word vectors

Dense word vectors, usually called as embeddings, are short with the dimension ranging from 50 to 1000. The values of the vectors will be real-valued numbers that can be negative [1]. The key idea behind Dense word vectors is that words that are similar or can be used to a similar context should have similar word vectors, that follow the distributional hypothesis but also a dense vector for every word is adopted to include the total amount of information. The main benefits of the dense vectors are : firstly, a smaller dimension of vectors helps models to have less parameters/weights, but also helps to avoid overfitting; secondly, dense vectors tend to capture better the synonymy. Different methods all create supervised training instances in which the goal is to either predict the word from its context, or predict the context from the word. Perhaps the most important set of pretrained embedding vectors is **word2vec**. Word2vec is an approximation of language modeling, applied to a fixed word window.

Word2Vec. Word2Vec was first proposed by Mikolov et al. [49]. Word2Vec is a shallow, two-layer neural network that is trained to reconstruct linguistic contexts of words. Given an input of a large corpus of words, it produces a vector space, typically of some hundred dimensions, with each word in the corpus being assigned a corresponding vector in the space. Word vectors are located in the space such that words that share common contexts in the corpus are located close to one another in the space. Word2Vec has two forms, the continuous bag of words (CBOW) model and the Skip-Gram model as presented in Figure 3.1. When the feature vector assigned to a word cannot accurately predict that word's context, the components of the vector are adjusted. The vectors of words judged similar by their context are nudged closer together by adjusting the numbers in the vector.

Continuous Bag Of Words (CBOW) model is trained considering a window around our target words, having as a result to move to a new space where the context of the current target word is learned. Formally, as it is depicted in 3.1, if we need to predict the $w(t)$ the input to our model could be $w(t-2), w(t-1), w(t+1), w(t+2)$, declaring the context of the word we need to predict.

Skip-Gram is the exact opposite of CBOW, as in that case we need to predict the surroundings of a word having as input the same word. Formally, taking as input the $w(t)$, the output could

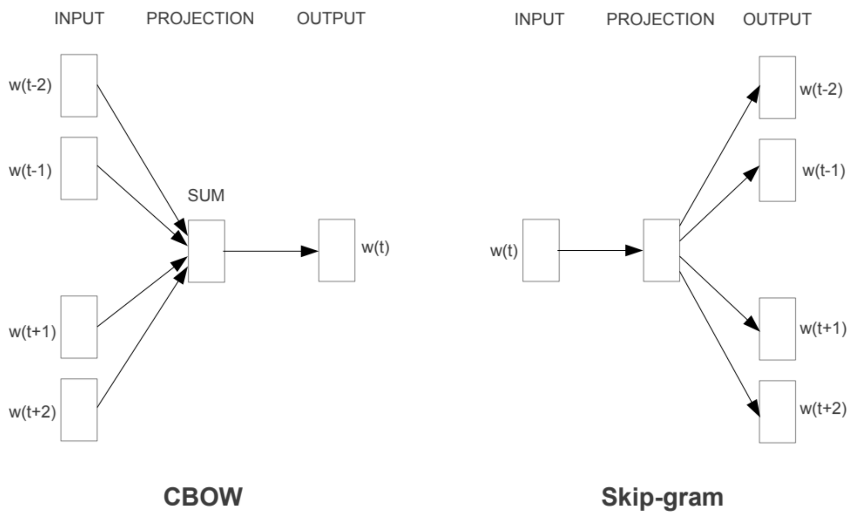


Figure 3.1. *The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.*

be $w(t-2), w(t-1), w(t+1), w(t+2)$. So, the task is to learn the embeddings by predicting the context of a word.

A serious problem of Word2Vec is the existence of some words in the test dataset that never appeared during training set. A possible solution to this came by **fasttext**, that was proposed by [50]. Fasttext solves this problem by using subwords models, representing each word as itself plus a bag of constituent n-grams. Then a skipgram embedding is learned for each constituent n-gram, and the word is represented by the sum of all of the embeddings of its constituent n-grams. Unknown words can then be presented only by the sum of the constituent n-grams.

GloVe Embeddings. GloVe, proposed by Pennington et al. [51], is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. GloVe is essentially a log-bilinear model with a weighted least-squares objective. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. The central intuition underlying the model is the observation that ratios of word-word co-occurrence probabilities can encode some form of meaning. The GloVe model is trained on the non-zero entries of a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given corpus. Populating this matrix requires a single pass through the entire corpus to collect the statistics.

Contextual Embeddings. The word2vec methods and GloVe are fast, efficient to train, and easily available online with code and pretrained embeddings. These are static embeddings, meaning that the method learns one fixed embedding for each word in the static embeddings vocabulary. However, NLP has evolved towards learning dynamic contextual embeddings like the popular family of BERT representations, in which the vector for each word is different in different contexts. That is a better solution to a variety of problems that static embeddings have, as the differentiation of words that have more senses than one. For example, in the following sentences: "I dream of surfing the perfect wave." and "Will there be another wave of illness in the fall?", the word "wave" has a pretty different meaning. It is further dicussed through BERT analysis in 3.5.

3.4 Language Modeling

Language Models (LM) are model that assign probabilities to sequences of words. The aim is to develop Language Models that assign higher probabilities to sequences that are either more grammatically correct or have more sense to occur. For example, the sequence "I had fever and therefore I got covid" is more possible than "I had fever and therefore I got ice cream" and therefore, it is needed a higher probabibility to be assigned to that.

Moreover, prediction of the next word is the most important task in Language Models. More specifically, we need to compute the co-occurence of all the words of the sentence, including the predicted word for all the possible predicted words. Finally, we keep the word with the highest probability. Formally, if we have n words, the probabiblity is :

$$P(w_1, w_2, \ldots, w_n)$$

This probability is equal to the following probability, using the chain rule:

$$P\left(w_1, w_2, \ldots, w_n\right)=\prod_{i=1}^n P\left(w_i \mid w_{i-1}, w_{i-2}, \ldots, w_1\right)=P(w_1)P(w_2 \mid w_1)\ldots P(w_n \mid w_1, \ldots, w_{n-1})$$

3.4.1 N-gram Language Models

Let us assume the sentence "I had fever and therefore I got" and we need to compute the probability the next word to be "covid". That means we need to calculate the probability :

$$P(\text{covid} \mid \text{I had fever and therefore I got})$$

One way to estimate this probability is from relative frequency counts: take a very large corpus, count the number of times we see "I had fever and therefore I got", and count the number of times this is followed by "covid". This would be answering the question “Out of the times we saw the history h, how many times was it followed by the word w”, as follows:

$$P(\text{covid} \mid \text{I had fever and therefore I got})=\frac{\text{count(I had fever and therefore I got covid)}}{\text{count(I had fever and therefore I got)}}$$

For a better LM we have to look back at Bayesian probability theory, and most precisely on the Markov condition, sometimes called the Markov assumption, which is an assumption made in Bayesian probability theory, that every node in a Bayesian network is conditionally independent of its non-descendants, given its parents. A more general assumption is the Causal Markov (CM) condition, which states that conditional on all its direct causes, a node is independent of all variables that are not direct causes or direct effects of that node. By using these theorems, we can construct Bigram LM, Trigram LM, and more complex LM.

A Bigram Language Model can formally be expressed as:

$$P\left(w_1, w_2, \cdots, w_n\right)=\prod_{i=2}^n P\left(w_i \mid w_{i-1}\right)$$

In order to train a Bigram language model given a set of corpora C of $|N|$ total words, we have to calculate the following probability for each word w_i in the training corpora:

$$P\left(w_i \mid w_{i-1}\right)=\frac{\operatorname{count}(w_i, w_{i-1})}{\sum_{w \in C} \operatorname{count}(w_{i-1}, w)}, \forall w_i \in C$$

Finally, a Trigram Language Model can formally be expressed as:

$$P\left(w_1, w_2, \cdots, w_n\right)=\prod_{i=2}^{n} P\left(w_i \mid w_{i-1}, w_{i-2}\right)$$

In order to train a Trigram language model given a set of corpora C of $|N|$ total words, we have to calculate the following probability for each word w_i in the training corpora:

$$P\left(w_i \mid w_{i-1}, w_{i-2}\right)=\frac{\operatorname{count}(w_i, w_{i-1}, w_{i-2})}{\sum_{w \in C} \operatorname{count}(w_{i-2}, w_{i-1}, w)}, \forall w_i \in C$$

Typically, the actual number of words in the history is based on how much training data are available. Trigram language models are commonly used, which consider a two-word history to predict the third word. Language models may also be estimated over 2-grams (bigrams), single words (unigrams), or any other order of n-grams.

3.4.2 Neural Language Models

Non-linear neural network models allow conditioning on increasingly large context sizes with only a linear increase in the number of parameters. For example, a neural probabilistic language model was popularized by Bengio et al. [5].

This model takes as an input vector representations of a word window of n previous words, looked up in a table C. These vectors are now known as word embeddings. These word embeddings, $C(w) \in \mathbb{R}^{d_w}$, are then concatenated and fed into a hidden layer, whose output is provided to a softmax layer as shown in Figure 3.2. This neural language model can be mathematically described as follow:

$$\begin{gathered}
\mathbf{x}=\left[\mathbf{C}\left(w_1\right) ; \mathbf{C}\left(w_2\right) ; \ldots ; \mathbf{C}\left(w_n\right)\right] \\
\hat{y}=P\left(w_i \mid w_{1: k}\right)=L M\left(w_{1: k}\right)=\operatorname{softmax}\left(\mathbf{h} \mathbf{W}_2+\mathbf{b}_2\right) \\
\mathbf{h}=g\left(\mathbf{x} \mathbf{W}_1+\mathbf{b}_1\right) \\
\mathbf{x}=\left[\mathbf{C}\left(w_1\right) ; \mathbf{C}\left(w_2\right) ; \ldots ; \mathbf{C}\left(w_n\right)\right] \\
\mathbf{C}(w)=\mathbf{E}_{[w]}
\end{gathered}$$

where $w_i \in \mathcal{V}, \mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d_w}, \mathbf{W}_1 \in \mathbb{R}^{n \cdot d_w \times d_{\text{hid}}}, \mathbf{b}_1 \in \mathbb{R}^{d_{\text{hid}}}, \mathbf{W}_2 \in \mathbb{R}^{d_{\text{hid}} \times|\mathcal{V}|}, \mathbf{b}_2 \in \mathbb{R}^{|\mathcal{V}|}$.

V is a finite vocabulary. The vocabulary size $|V|$, ranges between 1,000 - 1,000,000 words, with the common size being around 70,000 unique words. Feedforward neural networks have been replaced with recurrent neural networks [52] for language modeling.

3.5 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT), proposed by Google AI [15], is a pre-trained model of deep bidirectional transformers for language understanding, which is then fine-tuned for a task. More specifically, BERT is consisted of a number of stacked encoders of transformers. During self-attention layer, there is no mask in the future information of the sentence, allowing the model to contextualize each token using information from the entire input. Everything else is the same as the encoder's model, using as input subwords tokenization combined with positional embeddings and passing them through a series of standard transformer blocks consisting of self-attention and feedforward layers augmented with residual connections and layer

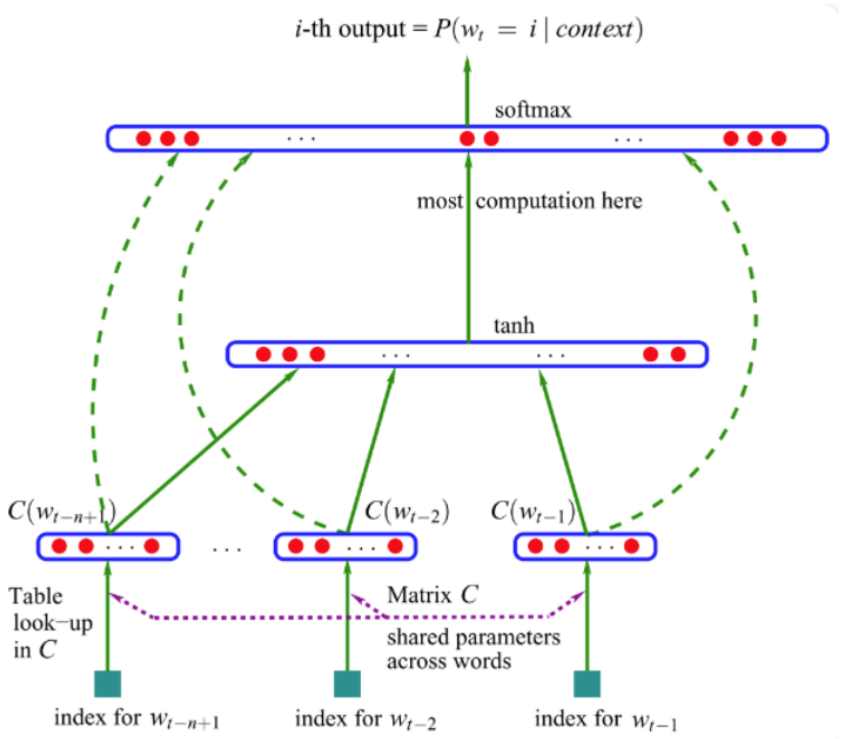


Figure 3.2. *A feed-forward neural network language model. Source: [5]*

normalization. There are different versions of BERT, but the initial suggestion included a subword vocabulary of 30,000 tokens, size of hidden layers equal to 768 and 12 layers of encoders, having multi-head attention of 12 heads. This resulted to a model with over 100M parameters. The model was trained in two unsupervised tasks :

Masked Language Model. BERT is allowed to see the complete sentence context. Therefore training BERT with a a predict-next-word task is trivial, since the answer is directly available from the context. It was then proposed to mask random words in a sentence and try predict those masked words instead. WordPiece Tokenization is used to generate the sequence of tokens where rare words are split into sub-tokens. Then, masking of 15% of the Wordpiece Tokens is performed. Masking replaces the words with [MASK] tokens. However, instead of constantly replacing the selected words with a [MASK] token, after a token was chosen, there were three ways of dealing with it:

- It is replaced with a token [MASK]
- It is replaced with another random token from the vocabulary
- It is left unchanged

The probability of choosing every one way of the above is 80% for the first one and 10% for each of the next two. Performing prediction on only 15% of all words instead of performing prediction on all words would entail that BERT would be much slower to converge. However, BERT showed immediate improvements in absolute accuracy while converging slightly slower than traditional unidirectional left-right models.

Next Sentence Prediction. This task entails predicting whether the first sequence provided immediately precedes the next. This task allows the Transformer to perform better on several downstream tasks such as question-answering, Natural Language Inference that involve understanding the relationship between two input sequences. The dataset used for training had a balanced 50/50 distribution created as follows: choosing an actual pair of neighboring sentences for positive examples and a random choice of the second sentence for the negative examples. The input sequence for this pair classification task is generated as:

$$\text{[CLS]} < \text{SentenceA} > \text{[SEP]} < \text{SentenceB} > \text{[SEP]}$$

Where sentences A and B are two sentences after performing the masking operations. The [CLS] token is the first token used to obtain a fixed vector representation that is consequently used for classification, and [SEP] is used to separate the two input sequences. As a result, the authors were able to achieve an impressive accuracy of $97 - 98\%$ in the next sentence prediction task.

Fine-Tuning. The power of pretrained language models lies is their ability to extract generalizations from large amounts of text—generalizations that are useful for myriad downstream applications. Fine-tuning is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks, whether they involve single text or text pairs, by swapping out the appropriate inputs and outputs. The pattern of the input and output sequence varies depending on the type of the task. The tasks can be broadly divided into four categories:

- Single Sentence Classification Tasks: The input of this task is taken considering the sentence A as the Single Sentence and the sentence B as $\emptyset$.
- Sentence Pair Classification Tasks: The input of this task is taken considering the two sentences A and B as the sentence pair.
- Question Answering Tasks: The input of this task is taken considering the sentence A as the question and the sentence B as the answer.
- Single Sentence Tagging Tasks: The input of this task is taken considering the sentence A as the Single Sentence and the sentence B as $\emptyset$.

The output is usually the token [CLS] that carries the information of the whole sentence or sentences and then fed into classification layers for tasks like sentiment analysis or entailment. For token level tasks, such as sequence tagging or question answering, the token representations are fed into an output layer.

3.6 Evaluation Metrics

Evaluation metrics are metrics used to measure the quality of the statistical or machine learning model. They provide models with feedback so as to improve until they reach a desirable performance. The performance is usually evaluated on the test set. The most common metrics used in classification tasks are accuracy, recall, precision and F1-score.

Accuracy. It is mostly used in binary classification tasks. Accuracy divides the number of true predictions by the total number of observations and outputs a percentage score. Its mathematical formula is:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP, TN, FP, FN are the number of true positive, true negative, false positive and false negative predictions.

Recall. Recall devides the number of true predictions by the sum of true positive and false negative predictions :

$$\text{Recall} = \frac{TP}{TP + FN}$$

Therefore, recall calculates the proportion of the correct predictions of one class by the total samples of that class. We have to rely on Recall for measuring our model, when the false negative predictions are really important to our task. For example, for sick patient detection, if the patient is actually sick but is predicted as healthy, then it can be catastrophic.

Precision. Precision devides the number of true predictions by the sum of true positive and false positive predictions :

$$\text{Precision} = \frac{TP}{TP + FP}$$

Therefore, precision calculates the proportion of the correct predictions of one class by the total samples that were predicted belonging to that class. We have to rely on Precision for measuring our model, when the false positive predictions are really important to our task. For example, for spam mail detection, we care more for the false positive predictions, meaning for predicting a non-spam mail as a spam, because in that case we may lose important mails.

The two metrics are depicted alltoghether in 3.3.

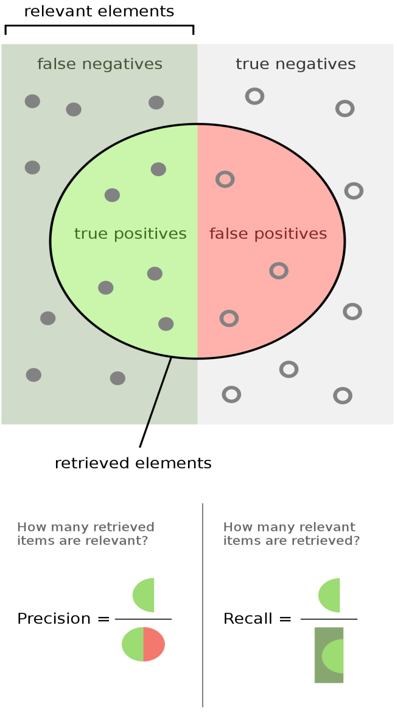


Figure 3.3. *Precision and Recall.*

F1-Score. This metric is used for multi-class classification or for binary classification where the number of observations is not balanced across the two classes. F1-score is the harmonic mean of Precision and Recall and is defined as follows :

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Chapter 4

Depression Detection

4.1 Introduction

Depression is a mental health disorder which affects a large portion of society. More specifically, [13] records 322 million people worldwide suffering from depression, which corresponds to 4.4% of the world population. Interestingly, depression is affecting more and more people every year, as the number of people with depression increased by 18.4% in the last ten years. To this end, detecting mental health issues and especially depression is proven crucial nowadays, as it aims to prevention of worse things like suicide. It is true that the most of the people do not attend a therapist and it is vital to detect their depression status, so as to have a better view of their suicidal thoughts.

In our research, we will attempt to detect depression from social media posts. To achieve this, a model for text classification needs to be built, by taking the utmost advantage of the novel architectures of Natural Language Processing (NLP). We will attempt to improve the performance of this task, using common sense and conclusions from the science of psychology. [53] showed that the metadata of the posts can be useful for our task, introducing the idea of the multimodal approach. Moreover, as our task is related with sentiment analysis, we will use some techniques frequently applied in this field. External knowledge is going to be incorporated, as it seems to be crucial in similar tasks ([54]). We will start our analysis, exploiting some useful techniques of deep learning and after that we will incorporate some handcrafted features that will be derived from data analysis and interepretability methods.

4.2 Social Media

According to [55] and [56], it is reported that there is a close association between depressive young adults and the excessive use of social media. However, it is noted that it is not sure if the young adults in question use social media more because of their depression, or if the use of social media causes their depressive symptoms. In any case, social media is proved to be a good indicator for our task, detecting depression from them. [57] shows that users of social media affected by depression tend to have different online activity and behavior than those who aren't, proving once again that the task of distinguishing them is feasible. [58] reports the numerous applications in NLP when we extract information from social media, but in our case we focus on health care applications entirely.

A lot of different social media platforms have been used in various NLP tasks, mainly detecting sentiment of posts or users that lead to mental health applicationg. The most usual platforms that are used are Twitter, as it had an open API to obtain data easily and Reddit, as there are many datasets free based on it. The source of the corpus can be also just writing samples, like in [59], where they assessed suicide risk in pediatric populations. Some more scientific records have also been allowed to be used, like the Electronic health Records [60], which are narratve text and it was expolited to predict depression severity or to predict the suicide risk of active duty military personnel [61]. The research that is usual in this field (NLP on mental health) includes the extraction of handcrafted features from posts or texts, as they may contain some core information that differs them from healthy posts. Therefore, it tries to intedify linguistic features, e.g. in [62] they intedify the features that are characteristics of early stage dementia. Finally, the usage of lexicons is very often in social media, as they organize the words in different categories that can be good indicators of many mental health situations. LIWC (Linguistic Inquiry Word Count) [25] is the most common lexicol, containing numerous categories and a huge vocabulary of words.

4.3 Related Work

Social media consists of users who upload several posts. As a result, there are some different approaches of any task that is based on an online platform. Specifically, we can employ either only posts or users, containing from posts. The second case involves the first one, while we have to edit firstly posts and then to move to a user level.

4.3.1 Posts Classification

The posts' classification is reduced in a text classification task, where the every text/post is considered autonomous and independent of other posts. This classification's category may be possible in cases where a post can capture enough information, e.g. classification of a post as stressed or non-stressed, but it may be inappropriate to classify it on a mental health situation. Towards the second occasion, the text classification uses longer texts, like mental health records or clinical interviews, in order to conentrate more information.

There are several works that used text classification as a main task for an application of NLP in mental health. [63] separated out LiveJournal posts that discuss depression and related topics, while [57] utilized Twitter data to predict depression in one of the most affective research in the area. [64] host a shared task that also attempts to make clinical diagnoses from Twitter data; in this case, for depression and posttraumatic stress disorder. [16] built a hierarchical neural network with attention mechanisms in both the word level and the turn level to detect depression from transcribed clinical interviews. External knowledge was also injected and the summary provided by the dataset was used.

As the deep learning models become the state-of-the-art models in the most of the NLP tasks, the text classification is the basic task that is affected by them, diminishing the importance of handcrafted features. The majority of works in this field use RNNs and Attention mechanisms as we have already discussed. The evolution of these, transformers, have been proposed to beat the best models in many text classification tasks. XLNet [17] integrates the idea of autoregressive models like OpenGPT and bi-directional context modeling of BERT. XLNet makes use of a permutation operation during pre-training that allows context to include tokens from both left and right, making it a generalized order-aware autoregressive language model. On another perspective of the task, [18] propose a graph-CNN-based DL model to first convert text to graph-of-words, and then use graph convolution operations to convolve the word graph. They show through experiments that the graph-of-words representation of texts has the advantage of capturing non-consecutive and long-distance semantics, and CNN models have the advantage of learning different level of semantics. [19] built a Hybrid Model, i.e. a Convolutional LSTM (C-LSTM) network. C-LSTM utilizes a CNN to extract a sequence of higher-level phrase (n-gram) representations, which are fed to a LSTM network to obtain the sentence representation. Finally, unsupervised learning has been proven crucial recently, as more and more Variational Autoencoders [20], Adversarial Training [21] and Reinforcement learning [22] is used.

4.3.2 Users Classification

Users' classification task is a two-levels task containing a post-level and a user-level. The problem is to classify a user in one or more categories, when a user has published numerous posts in social media. Towards this goal, there are two paths to follow: we can either analyze the total of the posts of the user, extract the most useful features from them and then extract the most important features from the interconnections between the posts of the same user, or we can analyze the behaviors and relationships between different users and, combining with the features of their posts, to conclude about the annotation of a specific user. There are several techniques that are followed in both cases and the sources of the data are similar like them in the post's classification task. In this work, we focus on data derived from social media and more specifically, Twitter, Reddit and Flickr are the most used.

[65] manually evaluated 200 college students' posts of one year, in order to conclude about their depression status. A study using data from Weibo, a Chinese social network platform [66] analyzed the social attitudes of people publicly commenting on others who made public their intention to commit suicide. As referred earlier, it is usual to utilize the connections between users, in order to extract important conclusion than just simply extracting their post's information. In [67] they did a study on TrevorSpace, a social network popular amongst lesbian, gay, transgender and bisexual communities about mental health status. The study did not use the text but only the connections amongst individuals and showed that some features are predictive of distress or mental-ill health. On the other hand, the details that can be derived from the users' interactions can lead to important conclusions and results. In this scenario, the most usual way to represent users' activity is using Graph Neural Networks. As it is obvious, the connections of users in social media, e.g. having someone as a friend or retweeting someone's post, can be thought as a huge graph, where the nodes are the users and the edges connect users that are related. In [24] they build a multiple relation graph from a twitter dataset, in order to detect anomalous users. [23] tries to detect personality of the users combining GNNs and Transformers. In particular, they build a graph for every user, which contains posts, words and categories of the words as nodes and it initializes it with embeddings extracted from BERT.

4.4 Datasets' Overview

There are several datasets related to depression that have been used in a variety of research works. As discussed earlier, the most common among them are the ones derived from social media through an API. A lot of researchers are used to create their own dataset, which consists of the information they need, acquiring permission from a platform to use its API, e.g. [68] exploit the Flickr API to extract useful features from users' posts in Flickr. The Reddit Self-reported Depression Diagnosis (RSDD) dataset ([27]) created by annotating users from the Reddit dataset, which is publicly available. This is going to be the main dataset which will be used in this work. No noteworthy results have been accomplished and this is a challenge for us. This dataset is the one of the biggest in this area, as it includes over 380 million posts in the training set. Self-Reported Mental Health Diagnoses (SMHD) dataset ([69]) includes users from Reddit who have been diagnosed with one mental health condition among the following: ADHD, Anxiety, Autism, Bipolar, Depression, Eating, OCD, PTSD, Schizophrenia. The extraction of both the diagnosed and control users was carried out by the same procedure. DAIC-WoZ dataset, which is part of the DAIC dataset ([70]) is multimodal, including audio, video and text from the sessions between a client and a virtual agent serving as a therapist. This dataset differs from the above in the length of the text we are forced to process, as is the case of this dataset a whole interview between two entities is available.

The usage of different APIs is occurred detecting various keywords that are related to the mental health - target. Most of the works try to find users in discussions of a mental health topic, like depression, and then they analyze these users (or their posts individually). Twitter, also, provides the opportunity of hashtags, which make searching more easily. [71] tried a different approach, combining two methods. At first, they interviewed some users, in order to use those interviews as their main corpus, but then they took, also, their twitter's content, so as to analyze their preferences. Finally, in [72] they deployed a corpus-based approach. More specifically, they obtained an existent dataset, which had been acquired through the Twitter API and they did their own annotation, labelling posts as depressive or not, taking advantage of their knowledge about the depression's symptoms.

4.5 Features Selection

Even if deep learning models have been the state-of-the-art models in a plethora of tasks, mental health tasks exploit, still, the creation of handcrafted features, according to the careful examination of the dataset. These features are capable to achieve a better general performance, because they make the classes more distinguishable in the cases like depression detection. They can also be combined with deep learning models or to incorporate them as external knowledge. The most usual method is the usage of lexicons. Lexicons are dicitionaries that assign words to one or more category. The categories can be more general, like if a word is a verb or pronoun, but they can be more specific, like if a word declares anxiety or fear. In the task of depression detection, where an association between depression and lexical features has been noticed, the finding of such features is required. The most common lexicons in those applications are: the psycholinguistic dimensions of LIWC (Lexicon Inquiry Word Count) [25] which includes 73 dimensions and a vocabulary of 18,504 words, the Emolex (NRC Emotion Lexicon) [26] which consists of 14,182 words assigned to 19 dimensions related to emotion, and Subjectivity Lexicon (MPQA) [73] with 6,886 words in 4 sentimental dimensions.

There are several lexical features, which can be extracted or detected in order to assist in the classification task. [16] uses external knowledge, which was injected in both the word-level analysis of turns and the summary provided by the dataset. [68] develops a multimodal approach for detecting the users with mental health issues. They detect different features from every modality separately and they combine it altogether. This approach also proved that the psychological research about what people with a mental health disorder usually prefer is verified by their activity on social media. [74] introduced a different perspective of processing social media posts as they noticed that a lot of misclassified samples may include various views such as emotion, sarcasm, personality and sentiment which are capable of distinguishing a disorder but aren't captured by current models. Consequently, multiview architecture is proposed, which consists of a task-agnostic view and NLU view, including the above four views and fusing all together in the end. After that, attention was given to the sarcasm view and figurative language. So, in [75] there was an attempt to detect depressive symptoms from twitter posts, building a 2-task problem: detecting depressive symptoms and detecting figurative language, designing soft-parameter sharing between tasks and layers. However, a basic problem of social media posts had to be faced: most of them are only a few words and, as a result, they don't have a lot of context. In the direction of solving this problem, [76] proposes a model combining CNN and Bi-GRU with multi-attention mechanism. Some other studies, e.g. [77] have focused on detecting some demographic features of the people in a dataset and how those features are related to some mental health illnesses like depression or PTSD. Taking advantage of those studies, we can make some important conclusion about the profiles of users belonging to a specific class and exploit this information to achieve better performance.

Chapter 5

Proposed Models

5.1 Dataset

In this work we use one main dataset for the task of depression detection. The dataset is called "Reddit Self-reported Depression Diagnosis (RSDD) dataset" as it was proposed by [27]. As it is declared from the title it is a dataset created by users of Reddit who self-reported that they have been diagnosed as depressive. The dataset is really useful for our task, as it includes depression-based annotations.

More specifically, the RSDD dataset consists of Reddit posts for approximately 9,000 users who have declared they had been diagnosed with depression and approximately 107,000 matched control users. Users annotated as "depressive" are users who match a high precision diagnosis pattern, e.g. "I was just diagnosed with depression". However, there were some exclusive condition that were applied to those users. If the users had less than 100 posts on Reddit before their diagnosis posts, they were discarded. Moreover there were two exclusive conditions related to the posts of the users :

1. The mention of depression was required to be no more than 80 characters away from the part of the post matching a diagnosis pattern; posts that did not satisfy this constraint were ignored. (A mention of depression is defined as the occurrence of one of the following strings: "depression", "depresion", "depressive disorder", "major depressive", or "mild depressive".)
2. The users were then viewed by annotators and only the users with at least two positive annotations were finally included in the dataset. The most common false positives included hypotheticals (e.g., “if I was diagnosed with depression”), negations (e.g., “it’s not like I’ve been diagnosed with depression”), and quotes (e.g., “my brother announced ‘I was just diagnosed with depression’ ”).

Finally, it is clarified that every post that contained a keyword related to depression was removed by the dataset of the depressive users.

The selection of control users followed a different process. An initial pool of control users was selected removing all the users who had ever posted in a subreddit related to mental health or used a keyword related to depression or metntal health. Control users were chosen by matching candidate control users with diagnosed users. Each diagnosed user was greedily matched with the 12 control users who had the smallest Hellinger distance between the diagnosed user’s and the control user’s subreddit post probability distributions, excluding control users with 10% more or fewer posts than the diagnosed user.

Every user (every value of the list) contains his or her posts, an id that is specific for every user and the label ”control” or ”depression”, depending on the category he or she is involved. The posts and the length of the posts of every user are very large, so we only show an example containing a posts of a user (an element of the list of a user) :

Label	training	validation	testing	total
control	35753	35746	35775	107274
depression	3070	3070	3070	9210

Table 5.1. *Number of samples for every class.*

	mean number of	
training set	posts of healthy users	895.4270
	posts of depressive users	1739.5355
	words of healthy users	24.3848
	words of depressive users	40.7978
testing set	posts of healthy users	908.1891
	posts of depressive users	1741.0013
	words of healthy users	24.3858
	words of depressive users	40.8768

Table 5.2. *Average number of posts and words per user for both classes.*

1451878990, "One time I was sitting in my apartment doing homework and this fly kept buzzing around my head and every time I swatted at it it would just come closer. Finally, I got fed up and I went to try and grab it and I ended up catching it by its wings with my index finger and thumb. I felt like the Karate Kid but cooler.

Also, the format of the data of every user is a dictionary:

[{ "posts" : [post_id, post], ..., [post_id, post], "id" : user_id, "label" : label}]

After the unzipping of the dataset files, we save every file in a corresponding list that is constituted by lists. Every sub list represents the post and the total number of lists represent the user. We save in a different file the list of labels for every user. In this way, the processing becomes easier. Finally, a lot of users have as label "None". We decided to remove totally these users from our data as they cannot provide us any useful information. The final number of samples for every data file is illustrated in the 5.1:

Both the number of posts made by each user and the length of each post vary widely, as shown in 5.1. As shown in Figure 5.1, approximately 20% of users have more than 1500 posts and 20% of posts over 250 words. A more analytical view about the mean numbers is presented in 5.2.

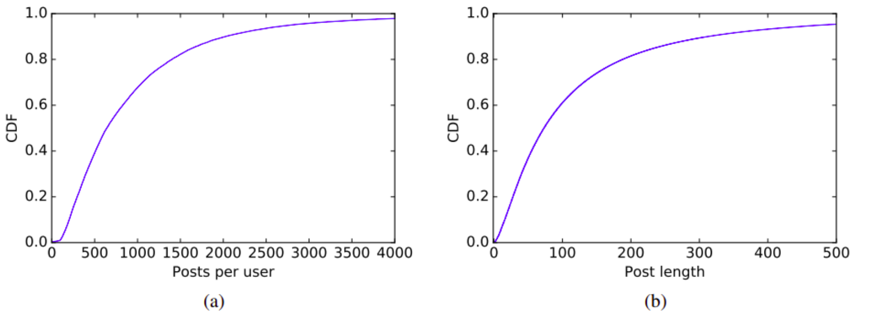


Figure 5.1. *Empirical cumulative distribution functions (CDF) of the number of posts per user (a) and the post length (b).*

5.2 Related Work

There are several works, which used the RSDD dataset, in order to detect depression or to analyse the characteristics of the users. [28] proposed an architecture, which consists of the classical two-level system: a post-level and a user-level. In every level, they use a corresponding operation, that is either a single-gated LeakyReLU or a multi-gated LeakyReLU operation. The overview of the proposed architecture is illustrated in 5.2.

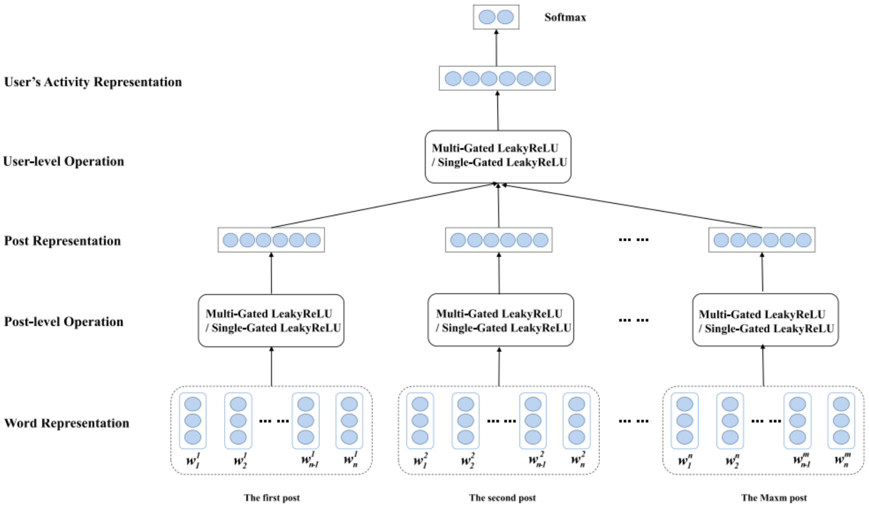

Softmax
User's Activity Representation
User-level Operation
Multi-Gated LeakyReLU / Single-Gated LeakyReLU
Post Representation
Post-level Operation
Multi-Gated LeakyReLU / Single-Gated LeakyReLU
Multi-Gated LeakyReLU / Single-Gated LeakyReLU
Multi-Gated LeakyReLU / Single-Gated LeakyReLU
Word Representation
w^1_1 w^1_2 w^1_{n-1} w^1_n
w^2_1 w^2_2 w^2_{n-1} w^2_n
w^n_1 w^n_2 w^m_{n-1} w^m_n
The first post
The second post
The Maxm post

Figure 5.2. *Each target user has up to m posts and each post consists of n words. w_n^m stands for the n-th word in the m-th post.*

It is noteworthy that they do not use pretrained embeddings but they train new embeddings for the words of this specific task. The difference of this work is that they use gated cotemporal convolutions instead of the typical recurrent networks that are more usual. Both the post-level and the user-level operation of SGL-CNN and MGL-CNN consists of two convolutional layers and a global average pooling. The difference between SGL and MGL is the number of gating weights generated. In the first convolutional layer, two filters are applied to derive feature values. Then, a second filter is applied to obtain a gating weight. Finally, an element-wise product between the products of the first filters and those of the second is used in order to get an enhancement feature map, which it gets through an average pooling layer afterwards.

In [6], they use BERT to get pretrained embeddings of the posts and then they use GRU-Attention in order to extract the user's representation. The architecture is illustrated in 5.3. A knowledge fusion layer is exploited at first to inject knowledge triples into original posts and in this way, a more informative initial representation of the post is created. Then, BERT takes as input the posts' representations and outputs the corresponding embeddings of them. AFter that, they are combined through a BiGRU layer with Attention to capture the user's representation. Finally, the classification is ocurred through dense layers. They also use an AdaBoost ensemble algorithm in order to adjust the weights of the samples.

[74] introduced a different perspective of processing social media posts as they noticed that a lot of misclassified samples may include various views such as emotion, sarcasm, personality and sentiment which are capable of distinguishing a disorder but aren't captured by current models. Consequently, multiview architecture is proposed, which consists of a task-agnostic view (BERT embeddings) and NLU view, including the above four views and fusing all together in the end. This idea inspired us to detect the respective aspects for our task, detecting depression, as it is

explained in a later subsection. In a following work, theye gave attention to the sarcasm view and figurative language. So, in [75] there was an attempt to detect depressive symptoms from twitter posts, building a 2-task problem: detecting depressive symptoms and detecting figurative language, designing soft-parameter sharing between tasks and layers. However, a basic problem of social media posts had to be faced: most of them are only a few words and, as a result, they don't have a lot of context. In the direction of solving this problem, [76] proposes a model combining CNN and Bi-GRU with multi-attention mechanism.

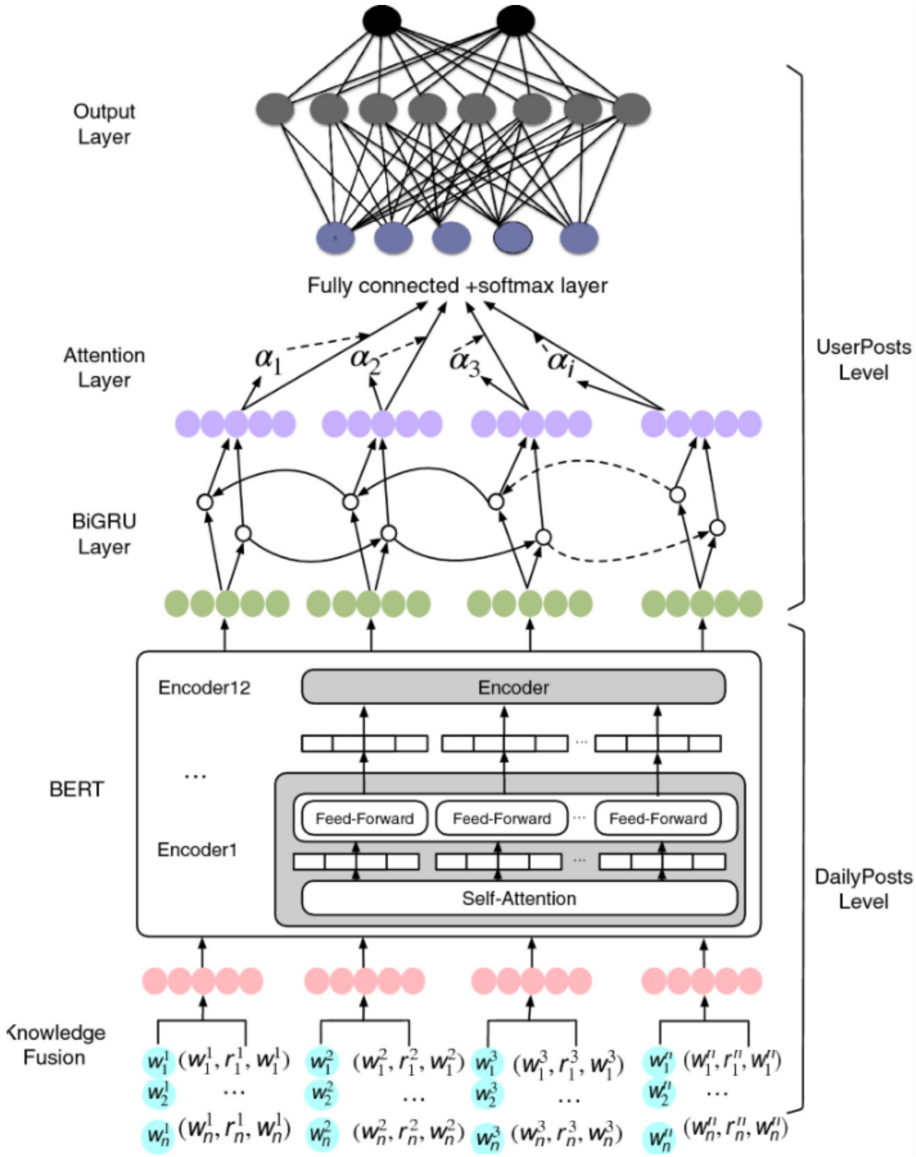


Figure 5.3. *The architecture of [6].*

5.3 Baselines

The problem is consisted of two levels: a post-level and a user-level. Theoretically, we need to compute two different representations. Firstly, we have to calculate a representation for every post of the user and in a second stage, to aggregate those representation, in order to extract a unified representation for the user. It is important to run some baselines, before we continue to the rest implementations. The baselines that are created are:

- **BoW-SVM** and **BoW-MNB** classifiers [78]. The set of features in this case is the post itself represented as sparse bag of words features. SVM were explained in 2.3.3 and Multinomial NB was explained in 2.3.2.
- **Feature-rich-SVM** and **Feature-rich-MNB** classifiers. This set of features includes bag of words features encoded as sparse weighted vectors, external psycholinguistic features captured by LIWC [79], and emotion lexicon features [80].
- **User-model CNN** as it is proposed by the creators of the dataset [27]. At post's level, every post is processed through a CNN and then there is merge layer for combining all them together. At user's level, there is an output layer consisting of dense layers with dropout.
- **Long Short Term Memory** as it was described in 2.4.3.2. It works in two levels, as well. At first, it take as input the words of a post and produces the post's representation and after that, it take all the posts as a sequence to produce user's representation.
- **Bi-directional Long Short-Term Memory**. It works in the same way as the simple LSTM but it captures bidirectional semantic dependency.
- **GRU - Attention** as it was described in 2.4.3.2 and 2.4.3.3. It works also in a two-level system, where in every level it is used as an encoder model to output post's and user's representation. It is used also, combined with the rest variants as **LSTM-Attention** and **Bi-LSTM-Attention**.
- **CIFG-LSTM** is just a variant of LSTM. It works in the same exact way, but it is designed to couple the input gate and the forget gate as one uniform gate [81].
- **Tree-LSTM** [82]. It is used in order to avoid the sequential way and achieve the representation of words to sentences over parse tree structures.
- **Bert-LSTM** as BERT is described in 3.5. At first, BERT is used to extract post's representations and then LSTM is used to capture user's representation.

5.4 Detectors

Every model of the following is trained to detect an aspect of the Natural Language, as all of them are part of sentiment analysis. This can help us to have a more spherical knowledge of the user's mental health situation and personality. Having this knowledge will help us to our task of depression detection, because the depressive user differs from a healthy user in terms of his/her emotions and personality. Moreover, the sarcasm/irony detector is considered important as the depressive users tend to be more sarcastic about life.

5.4.1 Emotion Detector

Aiming to detect the emotion of the user, we will use a pretrained model that will examine a post on more fine-grained emotions, i.e. valence, arousal and dominance. In this work, in order to extract the emotion from posts, we leverage a system similar to the CNN Network that is proposed by [83] and we utilize the EmoBank-2017 dataset [84] to train the model for fine-grained analysis.

EmoBank-2017 dataset consists of 10,548 (after filtering, 10,062 remained) unique sentences/ posts, annotated in terms of valence, arousal and dominance in a 5-point scale. The model that will be trained in this dataset is simply a variant of the CNN architecture and is depicted in 5.4.

We use the pretrained Google word2vec word embeddings vectors, as it has been proven to achieve a better performance than training embeddings without any initial information [85]. At a

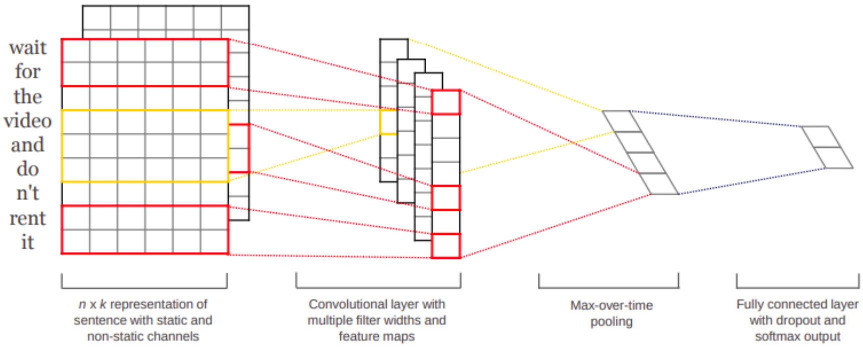

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 5.4. *Model architecture with two channels for an example sentence.*

first stage, we preprocess the whole dataset of the sentences and then the model is trained in the above architecture. During the preprocessing, we use 50 as maximum sentence length and 10,000 as the maximum number of words to be used in the model. Validation split is on 20% of the train dataset. The model consists of an embedding layer, three convolutional layers with max pooling layers, a concatenation layer that combines the outputs of the three previous layers, a dropout layer and finally dense layers for the final regression in the three dimensions (valence, arousal and dominance). We train the model for 25 epochs and batch size = 50.

After training the model, we move to the inference stage, where we insert our RSDD dataset. Here, we do not need to take the VAD (Valence - Arousal - Dominance) values of the posts, but just a representation of them on a higher dimensional space that will be a good indicator of the emotional state of the post. Therefore, without finetuning the model, we extract the representation of the post after the dropout layer, moving in a 300d space. In simple words, it is like getting the embeddings of a post, relying on the emotion of it. We keep the parameters equals to them during training stage (maximum sentence length = 50 and maximum nb words = 10,0000). We extract and save the embeddings for all posts for all users.

5.4.2 Irony Detector

Social Media texts often contain sarcasm and figurative language (FL), in general. In sentiment analysis, the presence of FL such as sarcasm in a text can work as an unexpected polarity reverser, which may undermine the accuracy of the system, if not addressed adequately [86]. It is possible for depressive users to utilize sarcasm so as to express their emotions. For example, a depressed user's post contain *"... It goes from pain to slight discomfort.. I cant move. Great way to start the day !"* So, it may be beneficial to detect the sarcasm of the texts. To do so, we will use the pretrained model of [87]. They created an architecture of two CNNs, a Bi-LSTM and a DNN layer to extract the sarcasm of Twitter texts. The final dataset that was collected by them contain 40K posts, 18K of them are labelled as sarcastic. Five more datasets are used for intentifying the context of the tweets, as also the LIWC dictionary [25] for obtaining AnalyzeWords.com dimensions. They also exploit Google word2vec embeddings to extract the embeddings of the words of the post firstly. The overall architecture is depicted in 5.5.

Following the same process as in the emotion detection, we freeze the model and extract the post's representation after the dense layer. This gives us a 256d-vector of embeddings of the post, which representates the sarcasm disposition that it contains. We extract those embeddings for the posts of RSDD, as it is described in a next subsection.

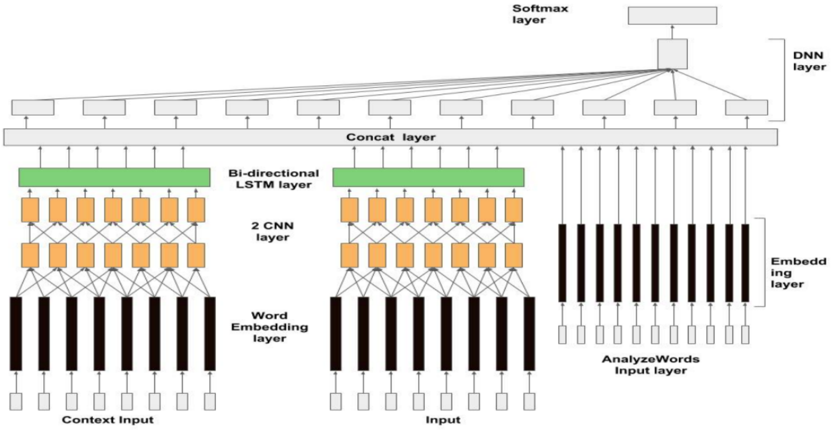


Figure 5.5. *A Neural Architecture for Detecting Sarcasm in Contextualized Utterances.*

5.4.3 Personality Detector

Personality Detector is used for detection of personality traits in the posts. Our personality and our behaviour impacts many aspects of our life and also our mental health status. It seems important to detect the personality of every user as we suppose that it may influence his/her mental status, contributing to classify them as healthy or depressive. We utilize the same architecture as in the emotion detector (5.4). The dataset for this task is taken by [88], called MyPersonality. It contains only 711 short conversations extracted by the tv show Friends, annotated in a 3-point scale (-1, 0, 1) with the five personality traits as labels : **Agreeableness** (trustworthy, straightforward, generous vs. unreliable, complicated, meager, and boastful), **Conscientiousness** (efficient and organized vs. sloppy and careless), **Extroversion** (outgoing, talkative, and energetic vs. reserved and solitary), **Openness** (inventive and curious vs. dogmatic and cautious) and **Neuroticism** (sensitive and nervous vs. secure and confident). This task is proven really challenging because only text information is given and might not contain enough information to fully understand the social context. Moreover, the dataset is small enough to make it possible to predict the personality with high accuaracy. During the training process, the accuracy does not get higher than 25% and that makes us consider the functionality of this specific detector in the experiments. As in the emotion detector, we use the same hyperparameters and extract a 300d embedding vector for every post in our dataset. Further details are provided in the following section.

5.5 "Detecting Everything" Model

Our task is a user classification task, where the input to the model is the set of the posts of one user and the output is a prediction of the user's depression status. The idea of implementing this architecture is inspired by the capabilities of different kind of detectors as discussed earlier. Our goal is to combine the different architectures in the best possible way so as to achieve the best performance. Various cases of preprocessing the data and combining the detectors are implemented and they are presented in the following subsections.

5.5.1 Proposed Model

Let us assume a post $S_i = (w_1, w_2, ..., w_{n_i})$, where n_i = number of words allowed to the model. The user that consists of the total number of his/her post is represented as $U_j = (S_1, S_2, ..., S_{m_j})$, where m_j = number of posts allowed to the model. The proposed architecture is summarised in 5.6.

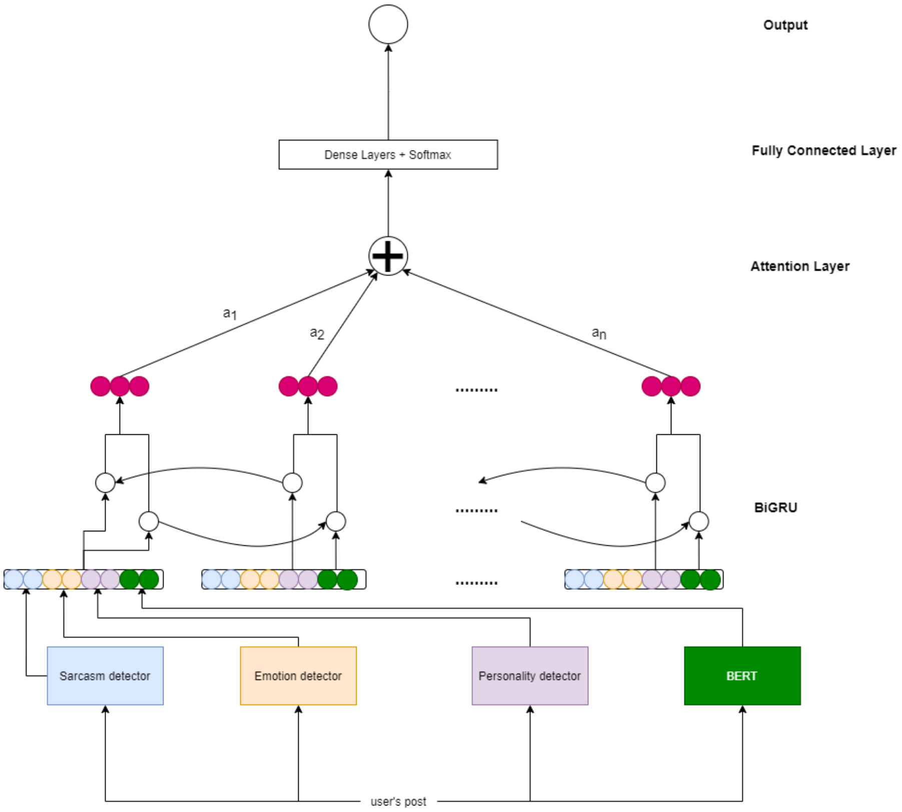


Figure 5.6. *The overall architecture of "Detect Everything" model.*

Every post is getting through every kind of detector, taking as input every time the S_i, for $i = 1, 2, ..., m$, where m = total number of posts allowed to enter into the next stage. The number of words of every post that were allowed to get through every detector (n) is different for each detector. As we have seen in 5.4.1, 5.4.2, 5.4.3, the allowed number is $n = 50$, but in BERT it is $n = 100$, padding in every case or truncating for longer sentences. Having the BERT as a first "detector" gives us the opportunity to capture a more contextualised representation of a post. It is not focused on only an aspect of the post. Instead, it outputs a representation containing something more abstract that can help our model to have a better knowledge of the language that is used and its different hierarchical levels. BERT is implemented as discussed in 3.5 and it outputs a vector with dimensionality of 768 for every post, taking into consideration the mean output of the 12 layers in the token $[CLS]$, a token that is exploited to capture the information of the whole sentence (being the first token of the sentence). Formally, in the post-level:

$$b_i = \text{BERT}(w_1, w_2, ..., w_{n_i})$$
$$e_i = \text{EMOTION}(w_1, w_2, ..., w_{n_i})$$
$$i_i = \text{IRONY}(w_1, w_2, ..., w_{n_i})$$
$$p_i = \text{PERSONALITY}(w_1, w_2, ..., w_{n_i})$$

where $b_i \in \mathbb{R}^{768}$, $e_i \in \mathbb{R}^{300}$, $i_i \in \mathbb{R}^{256}$ and $p_i \in \mathbb{R}^{300}$.

After that level, there is a fusion layer where all the representation have to be combined in a unified representation. It is usual to use simple concatenation, in order to take a total representation of the post, having a final vector with a dimensionality equal to the sum of the individuals ones.

We try some different cases later. Formally, the final post's representation is given by :

$$post_i = b_i \circ c_i \circ i_i \circ p_i$$

where $\circ$ is a symbol of the fusion method.

Sequentially, we move to the user-level architecture, where we use a Bi-GRU and Attention model as they have been described in 2.4.3.2 and 2.4.3.3. Let $\overrightarrow{h_i}$ be the annotation of the i_{th} post obtained by the forward GRU and $\overleftarrow{h_i}$ the corresponding of the backward GRU. The final output of the Bi-GRU is :

$$h_i = \overrightarrow{h_i} \parallel \overleftarrow{h_i}$$

The Attention layer is exploited to capture the distinguishing effect of the posts, giving attention to the most important, and to form the user's representation. The user's representation results as follows:

$$t_i = f(h_i)$$
$$\alpha_i = \frac{e^{t_i}}{\sum_i e^{t_i}}$$
$$u = \sum_i \alpha_i t_i$$

We use a fully connected softmax layer to output the probability distribution over the final classes. The fully connected softmax layer maps a K-dimension user's vector into a 2-dimension vector composed of probabilities between 0 and 1. Meantime, the sum of probabilities equals 1.

5.5.2 Dataset preprocessing

The number of posts that we take into consideration, in order to take a decision about the depression's status of a user is an important hyperparameter that we should tune it properly. For this purpose, we build a pipeline for the data preprocessing before they enter into the detectors. This is illustrated by 5.7.

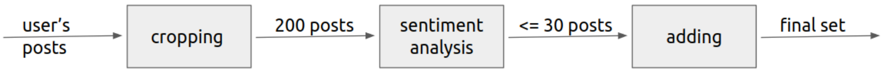


Figure 5.7. *The preprocessing stage of our architecture.*

As we concluded in data analysis' section, the average number of posts of users is about 950 posts/user (around 900 posts for healthy users and 1700 for depressive, resulting a weighted mean of 950 posts/user). It is wasteful in terms of time and sources to try to pass every post of every user through the detectors, as the dataset consists of around 760,000,000 posts in the training and testing set combined. Therefore, the initial idea is to minimize the dataset to the least possible, but without discarding the most valuable information about classifying a user as depressive or not. In simple words, we need to keep a small number of posts, which will contain the majority of the information we need. After that, we can make some serious conclusions about our model and the detectors and decide what is best to continue with.

As it is shown in 5.7, the stage of our preprocessing is threefold:

1. **cropping**: In this step, we crop randomly 200 posts of the user if the number of his/her posts is greater. If it is not, we keep the same amount of posts.

Model	precision	recall	f1-score
BoW-MNB	0.44	0.31	0.36
BoW-SVM	0.72	0.29	0.42
Feature-rich-MNB	0.69	0.32	0.44
Feature-rich-SVM	0.71	0.31	0.44
User model-CNN	0.59	0.45	0.51
LSTM	0.50	0.39	0.44
Bi-LSTM	0.56	0.40	0.47
Tree-LSTM	0.54	0.41	0.47
CIFG-LSTM	0.51	0.37	0.43
Bert-LSTM	0.53	0.49	0.51
GRU-Attention	0.57	0.40	0.47
LSTM-Attention	0.54	0.35	0.42
Bi-LSTM-Attention	0.62	0.39	0.48

Table 5.3. *Baseline models.*

Model	precision	recall	f1-score
SGL-CNN [28]	0.51	0.56	0.53
MGL-CNN [28]	0.63	0.48	0.54
KFB-BiGRU-Att [6]	0.57	0.51	0.54
KFB-BiGRU-Att-AdaBoost [6]	0.58	0.54	0.56

Table 5.4. *State-Of-The-Art models.*

2. **sentiment analysis** : In this step, we use the 'sentiment analysis' pipeline of huggingface [89]. In this pipepline, every sentence is classified as "negative" or "positive", accompanied with a confidence score about the decision. We keep only the posts of 'Negative' sentiment and we sort them, according to the confidence score. After that, we choose the m with the highest score (in the 5.7 $m = 30$).

3. **adding** : If the number of 'negative' posts of the previous stage is less than m, we add random posts from the initial pool of the posts of the user until they reach at m posts.

In the end, we have m posts that we hope they carry the most valuable context for our task. We chose to preprocess our data in this way, as depressive users are highly correlated with negative sentiments. But, the disadvantage of our method is that taking the negative-sentiment posts of a user may create a dataset of users with very similar context and emotion in their posts, as healthy users have, also, written not only positive posts.

5.5.3 Results

At first, we provide the results of the baselines 5.3 that they were run in the whole dataset, containing every post of the user in 5.3. We provide, also, the results of [28] and [6] for comparison in 5.4.

We experiment with two different $m = 30, 100$. As it was previously noted, utilizing as many posts as possible is really time and resources - consuming. Dealing with it, we run different combinations of our model for 30 and 100 posts, we came into conclusion and then we run the whole dataset for our best detectors and architectures. For $m = 30$, the results are depicted in 5.5. The corresponding results for $m = 100$ are presented in 5.6.

Every letter stands for the detector that participates in our model: B stands for Bert, E for

Model	precision	recall	f1-score
B 1x12 linear	0.27	0.56	0.36
	0.61	0.72	0.64
B 1x12 lstm	0.30	0.47	0.37
	0.63	0.69	0.65
BEIP 1x12	0.25	0.70	0.37
	0.61	0.76	0.63
BEI 1x12	0.16	0.87	0.27
	0.57	0.74	0.52
B 1x12	0.34	0.43	0.38
	0.65	0.68	0.66
E 1x12	0.12	0.65	0.21
	0.54	0.63	0.48
BEI 1x4	0.40	0.45	0.42
	0.68	0.69	0.69
BEI 1x1	0.73	0.12	0.20
	0.83	0.56	0.58
BEIP 1x12 pca different	0.13	0.11	0.12
	0.53	0.52	0.53
BEIP 1x12 pca same (5ep)	0.13	0.41	0.19
	0.53	0.58	0.52
BEIP 1x12 pca same (15ep)	0.14	0.38	0.20
	0.54	0.59	0.53
BEIP 1x12 linear 256	0.23	0.73	0.36
	0.60	0.76	0.61
BEI 1x12 linear 256	0.28	0.63	0.39
	0.62	0.75	0.65
BEIP 1x12 linear 100	0.28	0.65	0.39
	0.62	0.75	0.65
BEIP 1x6 linear 100	0.36	0.49	0.42
	0.66	0.71	0.68
BEI 1x12 linear 100	0.22	0.74	0.34
	0.60	0.76	0.60
BEIP 1x12 conv1d 256	0.25	0.56	0.35
	0.60	0.71	0.62
BEI 1x12 conv1d 256	0.20	0.70	0.31
	0.58	0.73	0.58

Table 5.5. *Results of various models for $m = 30$ posts per user.*

Model	precision	recall	f1-score
B 1x12 lstm	0.28	0.68	0.39
	0.62	0.77	0.65
BEIP 1x12	0.27	0.76	0.40
	0.63	0.79	0.65
BEI 1x12	0.31	0.70	0.43
	0.64	0.78	0.67
B 1x12	0.34	0.61	0.44
	0.65	0.75	0.68
BEI 1x4	0.57	0.38	0.45
	0.76	0.68	0.71
BEIP 1x12 linear 256	0.30	0.72	0.42
	0.64	0.79	0.67
BEI 1x12 linear 256	0.30	0.72	0.42
	0.63	0.79	0.66
BEIP 1x12 linear 100	0.31	0.70	0.43
	0.64	0.78	0.67
BEIP 1x6 linear 100	0.32	0.68	0.44
	0.65	0.78	0.68
BEI 1x12 linear 100	0.38	0.58	0.46
	0.67	0.75	0.70
BEIP 1x12 conv1d 256	0.20	0.59	0.30
	0.58	0.69	0.59

Table 5.6. *Results of various models for $m = 100$ posts per user.*

Emotion detector, I for Irony detector and P for personality detector, e.g. BEI means that every detector participates except from Personality detector. In both cases, we run firstly the baseline Bert-LSTM to have a minimum of the desired results. For $m = 30$ we have one more baseline that consists of just a BERT with a classification head (a linear layer). Every row of a model consists of two subrows: the first one is about the depressive class and the second one is for the macro average of both classes. The analogy of the number of samples between the two classes is healthy:depressive = 11.65:1. So, in our loss we use weights to the classes to give more importance to the less represented class and we adjust these weights to a fixed vector, that is declared in every model next to the detectors that participated. E.g. 1x12 stands for fixed weigths vector [1, 12]. In the first 8 rows of the first case and the first 5 rows of the second one, we used concatenation as a fusion method having the initial dimensions of every detector (768 for Bert, 300 for Emotion, 256 for Irony, 300 for Emotion). After that, we experiment a bit differently. Specifically, we train PCA and we use linear or conv1d layers so as to reduce the dimensions in the same number. We use two occasions for PCA : one where we use the same PCAs for training and inference and one where we use different ones (annotated as "same" and "different" in the tables). The dimension in this case is reduced to 30 for every detector, results to a 120d vector when we concatenate all the detectors. In the case of "linear" or "conv1d" method we use linear and conv1d layers before the fusion layer, so as to project every vector to one with a same dimensionality for all the detectors. For the linear method, we experiment with reducing dimensionality to 256 and 100 and for the conv1d method we only experiment with 256. Finally, it is noted that in the second table we experimented only with the cases that were proven better in the first table and the more worse ones are not conducted (like PCAs).

Finally, we run the best models for $m = 600$, extracting the embeddings of the whole dataset from BERT and from Emotion detector, summarized in 5.7.

Model	precision	recall	f1-score
B 1x12 lstm	0.29	0.61	0.39
	0.62	0.74	0.65
B 1x12	0.41	0.74	0.53
	0.69	0.83	0.73
B 1x7	0.64	0.53	0.58
	0.77	0.79	0.78
BE 1x12 linear 100	0.31	0.81	0.45
	0.65	0.83	0.68

Table 5.7. *Results of various models for* $m = 600$ *posts per user.*

Experimental Setup. In the implementation of the Encoder (Bi-GRU), we use 128 hidden size, 1 layer and 0 dropout, resulting to an output of 256. The Attention is a selfAttention mechanism taking key, query and value from the same source. Model parameters are optimized using Adam with 10^3 learning rate. Models are trained for 10 epochs and CrossEntropyLoss is used with fixed weights. For model implementation, Pytorch framework [90] is used. Running BERT or Emotion detector to extract embeddings for all posts of all users take about 10 days.

5.5.4 Conclusions

Carefully noticing the tables 5.5, 5.6 and 5.7 we come into the following conclusions:

- Adding posts, the model predicts more users as depressive, increasing recall but decreasing precision. Using a simple lstm in the classification head, the model of 100 posts/user predicts 2000 more users as depressive than the one of 30 users/post. However, the proposed architecture captures the true positive samples and it increases its recall, keeping precision constant. Having more posts per user, it means that we have more negative posts per user but with less confidence or more positive posts. So, the difference between the two classes can become more distinguishable. It is a fact that our model is able to distinguish more easily when a user is indeed depressive (more true positive samples), but at the same time, the rate of classifying users as false positive (depressive but they are not) remains stable. We guess that more posts may result in better results.

- As it is obvious, increasing the weights' analogy in the loss function, results in higher recall and smaller precision. These two metrics are considered as opposite in this problem. More specifically, our dataset is very unbalanced. Therefore, when there are not weights or the analogy is very small, the model tends to predict the majority of the samples as the more represented class, which is the "healthy" one in our case. So, we do not have so many samples classified as depressive and the recall is very low. On the contrary, a high analogy in weights force the model to take into consideration more the class with the lowest training samples and the precision is increased. Weights seem to be a lot influential in our task and that makes us question the robustness of our model. Moreover, we need balance between these two metrics, so it is important to explore the different values of that.

- PCA seems not to work in our case. That is possibly because of the dimension. More specifically, every user has 30 posts (in the first case) that every one of them is represented by 768d vector of BERT, 300d of Emotion, 300d of Personality and 256d of Irony. Therefore, the new dimension after PCA should be smaller or equal than $min\{30, dim(\text{detector\_vector})\} = 30$. Our representation goes in a very low dimension and it is no more able to capture the low level features we need. Furthermore, using the same PCAs for training and for testing is obvious better than training a new one in every case.

- Trying to move to a common dimension for all detectors, we use different trainable layers before we concatenate the various embeddings. In the table above, there are two techniques presented including 4 linear layers in parallel and 4 conv1d in parallel, correspondingly. The linear layers provide us a lot better results. After adjusting, also, the weights in the loss function, we achieve better results for both precision and recall. The best general model seem to bee the one which combines every detector and uses 1:6 analogy of the weights of two classes.

- Several experiments with different combinations of the detectors were held. It is observed that BERT embeddings (the task-agnostic embeddings) are the most important as they can capture more information, in order to classify the samples. Using BERT, Emotion and Irony embeddings combined seem to make the problem more biased toward the small class, as we can notice that recall is very high in these cases. Adding Personality embeddings, precision is increased and the two metrics are more balanced. Yet, in the case of linear layers before the rest architecture, adding the personality embeddings make the results worse. In general, the influence of the four detectors is not clearly stated and more experiments are needed to make this more lucid.

- The decrease of the dimension in the 4 layers from 256 to 100 in each detector seem to have better results. The explanation of this observation is not clear. It is possible the lower dimension to include more concrete information and when the embeddings are concatenated, it is easier to focus on specific features that make the classes distinguishable.

- Emotion detector, as expected, make the model predict more samples as depressive. Emotion is highly correlated with depression, because depression is considered a mental illness that affects our emotions. It seems to be the most valuable detector of the three and can be used properly to face Bert's inaccuracies. On the other hand, Irony and Personality detectors seem to be more random and not offer something important in our architecture. This may be because of the lack of sarcasm about life in our dataset in irony's case and because of the poor performance of personality detector in the initial dataset [88] as described, also, in 5.4.3. It is obvious that these detectors exacerbate the performance of the proposed architecture.

- In 5.7, BERT achieves a lot higher performance in terms of f1-score compared with the state of the art models (5.4). This is mainly due to the BERT's capabilities, but also because of the hyperparameters tuning and the weights we use in the loss (as we already noticed, the model's performance is changing rapidly with respect to the weights). Putting the Emotion Detector and using two linear layers of output size = 100, the recall increases, but the precision gets lower. For reasons of completeness, we note the confusion matrices of the two cases (only using BERT and using BERT with Emotion) in 5.8.

	healthy	depressive
healthy	32,498	3,277
depressive	789	2,281

(a) *B 1x12 model*

	healthy	depressive
healthy	30,315	5,460
depressive	580	2,490

(b) *BE 1x12 linear 100 model*

Table 5.8. *Confusion Matrices for comparison between two models, when $m = 600$.*

posts/user	precision	recall	f1-score
10	0.34	0.25	0.29
30	0.52	0.23	0.32
50	0.55	0.26	0.35
100	0.55	0.36	0.43
200	0.53	0.49	0.51
300	0.53	0.56	0.55
400	0.53	0.59	0.56
500	0.55	0.62	0.58
600	0.56	0.63	0.60
700	0.58	0.63	0.60
800	0.59	0.64	0.62
900	0.61	0.64	0.62
1000	0.62	0.64	0.63
1100	0.63	0.63	0.63
1200	0.64	0.62	0.63
1300	0.66	0.62	0.64
1400	0.67	0.62	0.65
1500	0.69	0.61	0.65
1600	0.70	0.60	0.64
1700	0.71	0.60	0.65
1800	0.72	0.59	0.65
1900	0.72	0.58	0.65
2000	0.74	0.58	0.65

Table 5.9. *Results of the B 1x7 model for different number of posts during inference.*

5.5.5 Does it generalize?

In this section, we conduct experiments to investigate if our best model till now can generalize and learn the task, having the least possible number of posts during inference process. The best model, as it was proven in the previous section is B 1x7, having 600 posts/user in both training and testing time. In order to notice the different performance of our model, we experiment with various numbers of post in the testing time, as depicted in 5.9.

Every time, the posts are selected randomly. The fluctuations of precision, recall and f1-score are concentrated on a common figure in 5.8.

We notice that both precision and recall have a steady increase till about 600 posts/user and then recall starts falling slowly. On the other hand, precision continue rising gradually, meaning that even if in the beginning we achieve better performance in both metrics increasing the number of posts/user, the model starts classifying less and less users as depressive (recall is decreasing), but it stops classifying mainly the non depressive users as depressive, making it work properly (precision is increasing). Moreover, f1-score remains stable to a value of 0.65 with minor variations. That is because of the fact that the rate of decrease of false positive samples (healthy that were classified as depressive) is higher than the rate of decrease of true positive samples (depressive that were classified as depressive). To verify these conclusions, we provide in 5.10 the confusion matrices in the three points of interest: 10, 600, 2000 posts/user.

5.6 Hybrid Model

At first, we focused on changing the user's level of our architecture, in order to achieve better performance, having stable the B 1x7 model in the post's level. Our task is, then, reduced in a

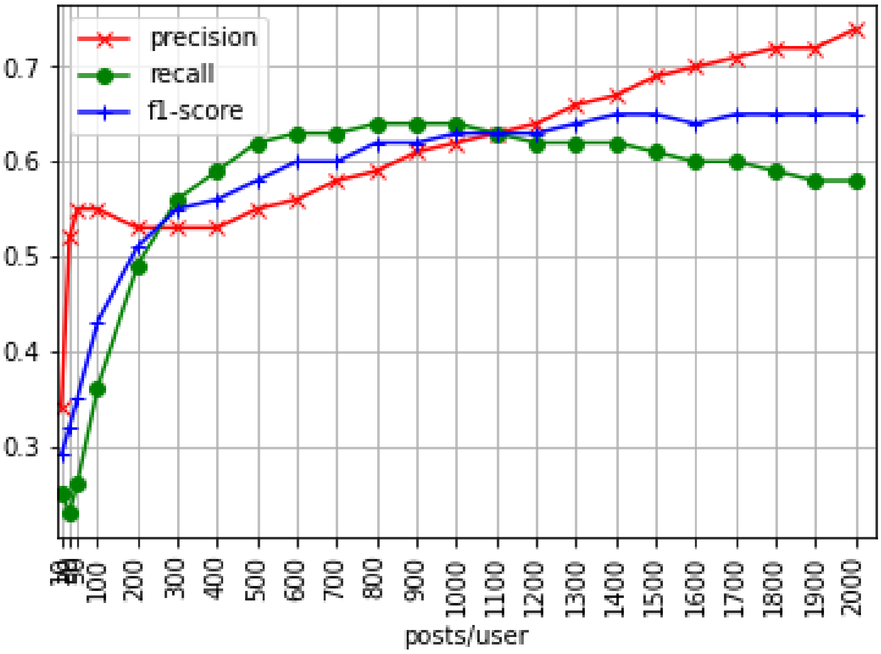


Figure 5.8. *Precision, Recall, and F1-Score for different numbers of posts per user in the testing set.*

	healthy	depressive
healthy	34,311	1,464
depressive	2,315	755

(a) *10 posts/user*

	healthy	depressive
healthy	34,276	1,499
depressive	1,134	1,936

(b) *600 posts/user*

	healthy	depressive
healthy	35,138	637
depressive	1,297	1,773

(c) *2000 posts/user*

Table 5.10. *Confusion Matrices for comparison between three different numbers of posts per user in testing time.*

document classification problem, where every sentence of a document is the post in our problem. [91] builds a hybrid model of CNN and LSTMs. More specifically, it uses multiple LSTMs in a first layer, one for each sentence capturing the semantic meaning of every sentence individually. Secondly, a higher layer LSTM captures the joint meaning created by the sentences. Finally, a CNN is placed over the LSTM for feature extraction. Later, in [92] they introduce an initial embedding layer into the architecture and an attention later before CNN. This is our main inspiration for our model.

5.6.1 Proposed Model

Our proposed model is depicted in 5.9.

We use B 1x7 model as an embedding layer to extract the posts' representations. Having a post $S_i = (w_1, w_2, ..., w_{n_i})$, we take the post' representation as:

$$b_i = \text{BERT}(w_1, w_2, ..., w_{n_i})$$

Let $\overrightarrow{h_i}$ be the annotation of the i_{th} post obtained by the forward GRU and $\overleftarrow{h_i}$ the corresponding of the backward GRU. The final output of the Bi-GRU is :

$$h_i = \overrightarrow{h_i} \parallel \overleftarrow{h_i}$$

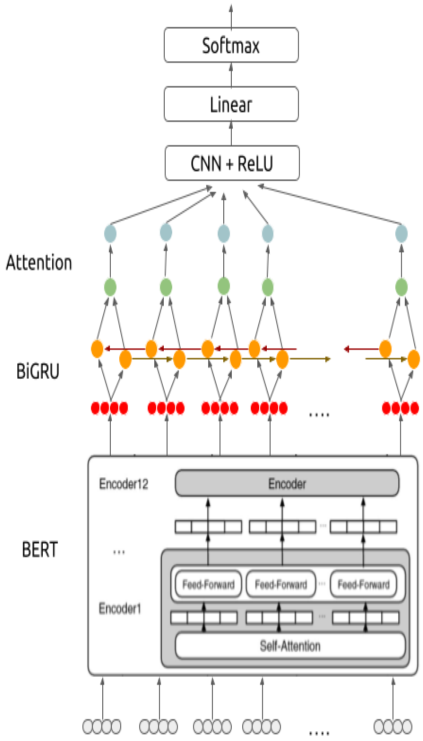


Figure 5.9. *The Hybrid model.*

Model	precision	recall	f1-score
B 1x7	0.64	0.53	0.58
	0.77	0.79	0.78
Hybrid	0.51	0.65	0.57
	0.74	0.80	0.76

Table 5.11. *Results of the Hybrid model compared with our "Detect Everything" model.*

In the Attention layer, we have as input a matrix $H = [h_1, h_2, ..., h_m$ and it outputs a matrix $M = [\alpha_1 h_1, \alpha_2 h_2, ..., \alpha_m h_m]$, where :

$$t_i = f(h_i)$$

$$\alpha_i = \frac{e^{t_i}}{\sum_i e^{t_i}}$$

Finally, CNN takes as input the M and outputs a vector of dimensionality K, which is then goes through the classification head for the final classification of the user as depressive or not.

5.6.2 Results

The results of the hybrid model, as also the results of the "Detect Everything" corresponding model for reasons of comparison, are presented in 5.11. We, also, provide the confusion matrices of both models in 5.12.

Experimental Setup. The B 1x7 model in the post-level is the same as in the "Detect Everything" model. Therefore, we have $m = 600$ and that is the second dimension of the matrix M. In the implementation of the Encoder (Bi-GRU), we use 128 hidden size, 1 layer and 0 dropout, resulting to an output of 256, being the first dimension of M. Model parameters are optimized

	healthy	depressive
healthy	34,858	917
depressive	1,438	1,632

(a) *B 1x7 model*

	healthy	depressive
healthy	33,840	1,935
depressive	1,075	1,995

(b) *Hybrid model*

Table 5.12. *Confusion Matrices for comparison between two models.*

using Adam with 10^3 learning rate. Models are trained for 10 epochs and CrossEntropyLoss is used with fixed weights 1x7. For model implementation, Pytorch framework [90] is used.

5.6.3 Conclusions

In general, hybrid model has a worse performance that the B 1x7 model. Yet, the differences are not so much important, something that make us consider the importance of user-level architecture. It seems that post-level model plays a more significant role for achieving a better general performance. Observing the 5.11, we notice that Recall metric is higher for the hybrid model, but Precision is lower. This can be explained easliy noticing the confusion matrices in 5.12. In particular, Hybrid model has classified more users as depressive than the B 1x7 model. Yet, the rate of increase of the false positive samples is much higher than the rate of the increase of true positive samples (29,1% and 3% respectively) and that results the lower f1-score.

5.7 Alternative Model

As a result of the previous methods, we decide to focus on the post-level. We need to explore the usage of the emotion detector and how this can enhance the performance of BERT in our task. On this purpose, we will explore some fusion methods, regarding condition as it was discussed in 2.7. The difference in our task is that we do not integrate information of different modality, but information that contains a different aspect of language's form. We experiment with various ways to infuse emotion information into our model.

5.7.1 Proposed Model

At first, we move the fusion layer in the user-level. That means that we extract a general representation of every user only from BERT and the respective representation only from the Emotion Detector and then we fuse the two representation before the output layer. The overall architecture is illustrated by 5.10.

In the 5.10, only the concatenation is depicted as a fusion method. Let us assume that we have b_j as user's representation from the left model of 5.10 and e_j from the right one. The final's user represenation is given by:

$$u_j = b_j \parallel e_j$$

However, there are some additional methods that are proposed and are used. Specifically, inspired by [7], we propose a technique where we introduce a floating point alpha $\alpha \in [0, 1]$ that is trained during the training process and it, intuitively, adjusts how much of every detector we will use for our user's representation. The final user's representation is given by:

$$u_j = \alpha \cdot b_j + (1 - \alpha) \cdot e_j$$

Furthermore, inspired by [93] we introduce a case of infusion emotional information, considering it as external knowledge. Therefore, we try to adapt this knowledge auxiliary, without letting it

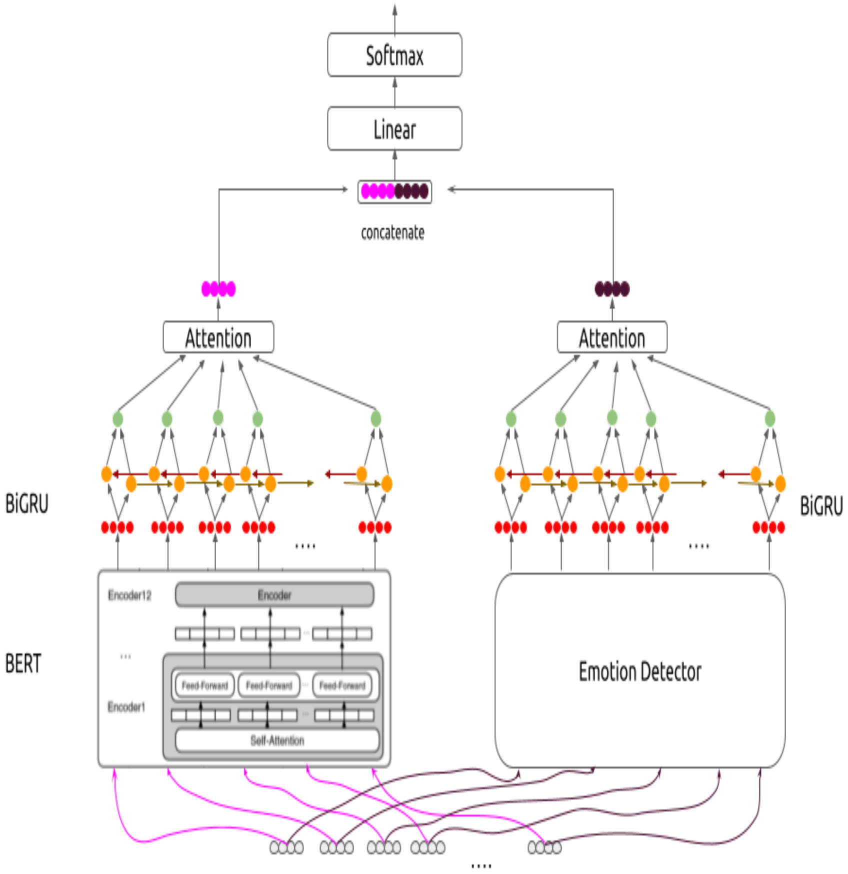


Figure 5.10. *Alternative model. It is the same as that in 5.6, but we have only BERT and Emotion detector, as also a different stage of concatenation.*

influence the model uncontrollably. Towards this direction, we pass the emotional representation through a linear layer and an activation function, using sigmoid function and then we use element-wise multiplication for the combination of the two representations. It, actually, learns a feature mask, which is applied on bert's representation. A gate mechanism with a sigmoid activation function, generates a mask-vector from e_j with values between 0 and 1. Intuitively, this gating mechanism selects salient dimensions (i.e. features) of b_j, conditioned on the emotional information. The output of this method is:

$$u_j = b_j \odot \text{sigmoid}(\text{linear}(e_j))$$

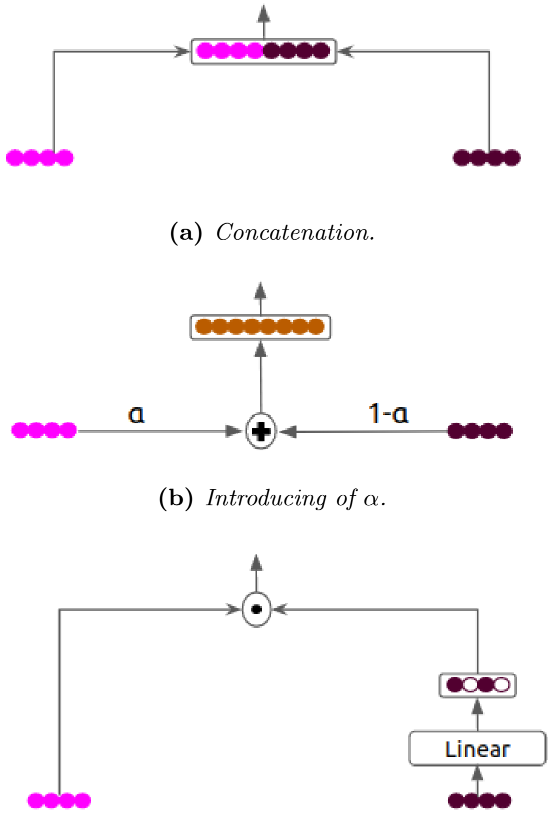
The three different methods are concentrated on 5.11.

Because of the promising results that these methods have in infusing emotional information into the contextualised information we take from Bert, we try to introduce a model that will combine the user's representation of all detectors we had in "Detect Everything" model. We train four hyperparameters $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, which are responsible for the participation of BERT embeddings, Emotion, Irony and Personality respectively. Formally, having as output of each detector the b_j, e_j, i_j, p_j, the final user's representation is given by:

$$u_j = \lambda_1 \cdot b_j + \lambda_2 \cdot e_j + \lambda_3 \cdot i_j + \lambda_4 \cdot p_j$$

The corresponding diagram of the model is in 5.12.

At a second phase, we move the fusion layer back to the post-level again, applying the attention-



(a) *Concatenation.*

(b) *Introducing of α.*

(c) *Element-wise multiplication with the a gate mechanism.*

Figure 5.11. *Fusion layer methods in the user-level.*

based methods of [93]. The general architecture is similar. It differs in the emotion's side, as we add a linear layer after we take the post representations and then we combine them with BiGRU's outputs of the Bert's side. This is better depicted in 5.13.

The fusion layer is now one of the three occasions that are described in [93] and are concentrated on 5.14. In the case of Attentional Concatenation (5.14a), we learn a function of the concatenation of each post representation extracted from BERT b_i, with its corresponding representation from the emotion detector e_i. Concretely:

$$f_c(b_i, e_i) = \tanh(W_c[b_i \parallel e_i] + b_c)$$

where W_c, b_c are learnable parameters.

In Attentional Feature-based Gating (5.14b) a feature mask is learned and then applied to every BERT representation b_i. A gate mechanism with a sigmoid function generates a mask-vector from each e_i with values between 0 and 1. Formally:

$$f_g(b_i, e_i) = \sigma(W_g b_i + b_g) \odot e_i$$

where W_g, b_g are learnable parameters.

In the last case of Attentional Affine Transformation (5.14c), a feature-wise affine transformation is applied to the latent space of the hidden states. Specifically, we use the emotional representations e_i, in order to conditionally generate the corresponding scaling γ and shifting β

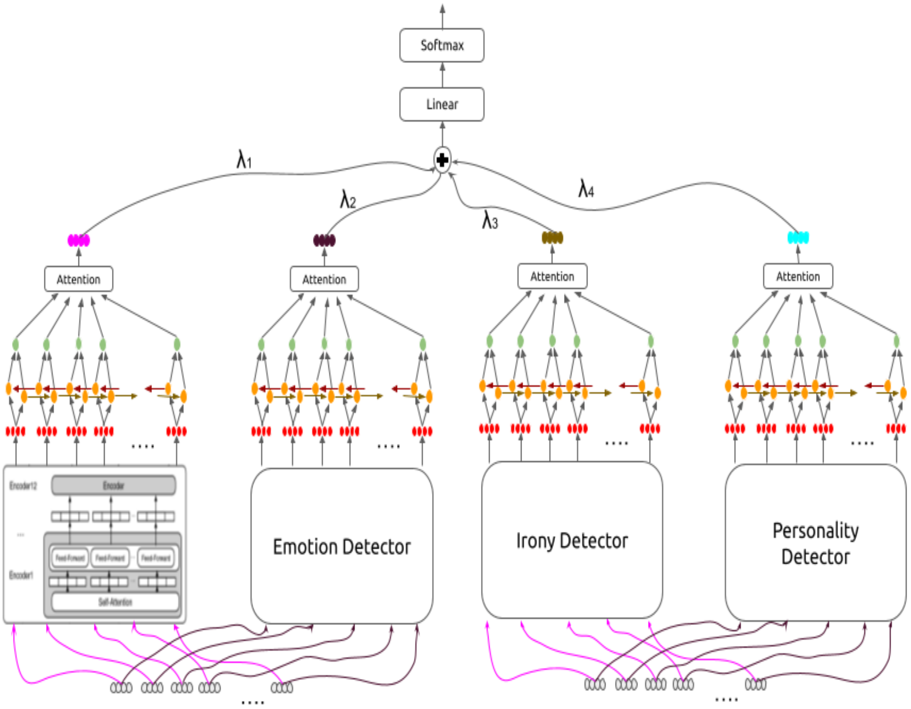


Figure 5.12. *BEIP model. Each detector has its own hyperparameter that is trainable and determine how much it will participate in forming the final representation of the user.*

vectors. Concretely:

$$\begin{aligned} f_a(b_i, e_i) &= \gamma(e_i \odot b_i + \beta(e_i)) \\ \gamma(x) &= W_\gamma x + b_\gamma \\ \beta(x) &= W_\beta x + b_\beta \end{aligned}$$

where $W_\gamma, W_\beta, b_\gamma, b_\beta$ are learnable parameters.

Finally, we try a different architecture where we try to import the emotional information in more stages of the BERT side. Diagrammatically, it is shown in 5.15, as it was inspired by [94] and [95]. More specifically, the two kinds of detector follow the usual process as the input gets through BiGRU and Attention, but one more Attention layer is added, that takes as input the concatenation of the outputs of both BiGRUs. The output of this layer is concatenated with the output of the Attention layer of Emotion-side and the combined representation is moved into a linear layer, imitating the idea of 5.11c. So, in a similar manner, it is combined finally with the representation of Bert-side. In this way, we try to make the emotional information get influenced more by Bert. We will call this model as "Stagefull".

5.7.2 Results

The results of all above models that were described are summarized in 5.13.

Experimental Setup. In the implementation of all models, the Encoder(s) (Bi-GRU) use 128 hidden size, 1 layer and 0 dropout, resulting to an output of 256. Model parameters are optimized using Adam with 10^3 learning rate. Models are trained for 10 epochs, CrossEntropyLoss is used with fixed weights 1x7 and we implement them with the Pytorch framework [90]. About the concatenation model, we concatenated the representation of the both sides, resulting to a 512d vector. In α parameter model, we initialize $\alpha = 0.5$, given the same probability to both sides

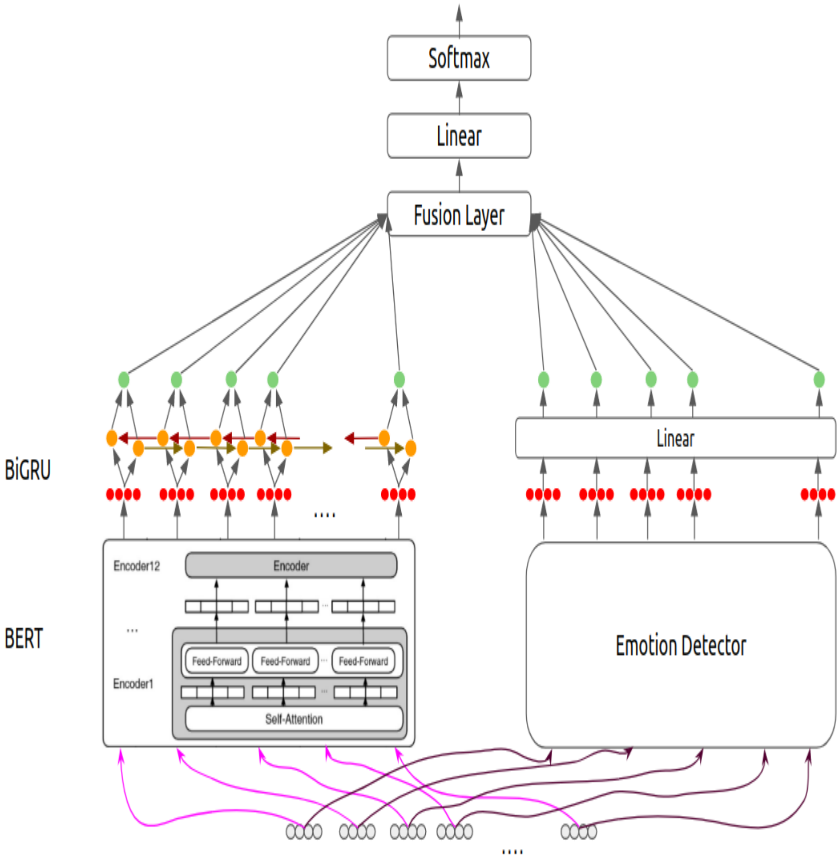


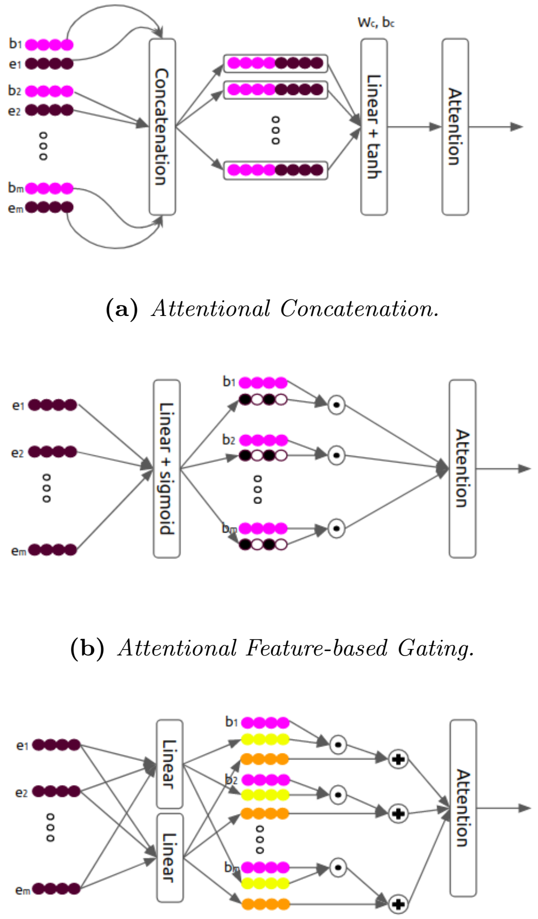
Figure 5.13. *Alternative model, but we combine the representations in the-post level. More specifically, after we take a representation of every post for both BERT and Emotion side (through a BiGRU and a Linear layer respectively) we use a fusion layer that integrate the emotional information.*

participates to the same degree reaching at 0.6931. The variation of α during the 10 epochs are shown in 5.16. In the "BEIP with parameters" model, we initialize $\lambda_1 = 0.5, \lambda_2 = 0.3, \lambda_3 = 0.1, \lambda_4 = 0.1$, reaching at 0.8979, 0.2177, 0.1647, 0.0512 respectively in the 10^{th} epoch, as it is depicted in 5.17. Moreover, in this model, BERT and Emotion processes 600 posts of every user, but the irony and personality processes each only 100 posts per user. This is because of the time consuming that this architecture is. In the attentional modules, we use a similar BiGRU as in the above cases and a linear layer that projects the emotional vector to a 128d space. Finally, in "Stagefull" model, every Attention layer outputs a 256d. Therefore, the linear layer, before the element-wise multiplication, takes as input a vector of 512d and outputs one of 256d.

5.7.3 Conclusions

After observing the above results, we come into the following conclusions:

- The three different methods of fusion that are presented in 5.11 enhance the performance of BE 1x7 linear 100 models, which uses concatenation in a post-level, combining the post representations. The 5.11b method is proven a bit better than the simple concatenation, because of the ability that it gives to the model to consider more Bert's representation, which seem to result in better performance. The 5.11c technique, adding a sigmoid layer is the best of all the three, reaching f1-score near the simple BERT model. That method



(a) *Attentional Concatenation.*

(b) *Attentional Feature-based Gating.*

(c) *Attentional Affine Transformation.*

Figure 5.14. *Fusion layer methods in the post-level.*

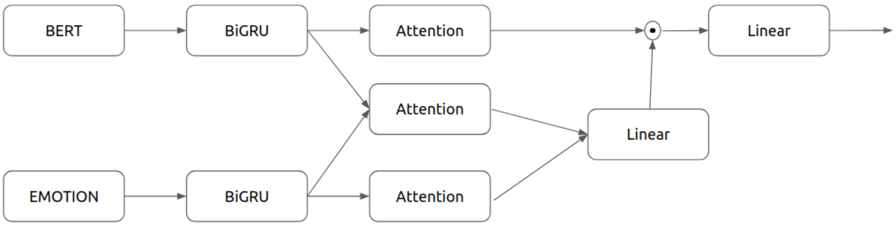


Figure 5.15. *Stagefull model. It contains one more Attention layer, which gives the opportunity BERT embeddings to affect more the emotion's ones.*

get the best of the advantages of the emotion detector getting it involved to the best way. This involvement increases recall score, as more users are classified as "depressive" (because giving importance to emotion, capture a more depressive side of the user), but decreases the precision, as more healthy users are also classified as depressive.

- As we can notice in 5.16 α is gradually rising up. This makes sense, as the most concrete information we can get is from BERT embeddings and therefore it is given more attention to these during training than the emotion detector's embeddings.

Model	precision	recall	f1-score
B 1x7	0.64	0.53	0.58
	0.77	0.79	0.78
BE 1x7 linear 100	0.31	0.81	0.45
	0.65	0.83	0.68
Concatenation (5.11a)	0.46	0.71	0.56
	0.72	0.82	0.75
α parameter (5.11b)	0.49	0.70	0.57
	0.73	0.82	0.76
gate mechanism (5.11c)	0.46	0.69	0.55
	0.72	0.81	0.75
gate mechanism + sigmoid	0.54	0.66	0.59
	0.75	0.81	0.78
BEIP with parameters (5.12)	0.48	0.71	0.57
	0.73	0.82	0.76
Stagefull (5.15)	0.43	0.72	0.54
	0.70	0.82	0.74
Attentional Concat (5.14a)	0.51	0.62	0.56
	0.74	0.78	0.76
Attentional Gate (5.14b)	0.54	0.59	0.57
	0.75	0.77	0.76
Attentional Affine (5.14c)	0.56	0.61	0.58
	0.76	0.79	0.77

Table 5.13. *Results of the different fusion methods and the additional models. B 1x7 and BE 1x7 linear 100 are provided for comparison reasons.*

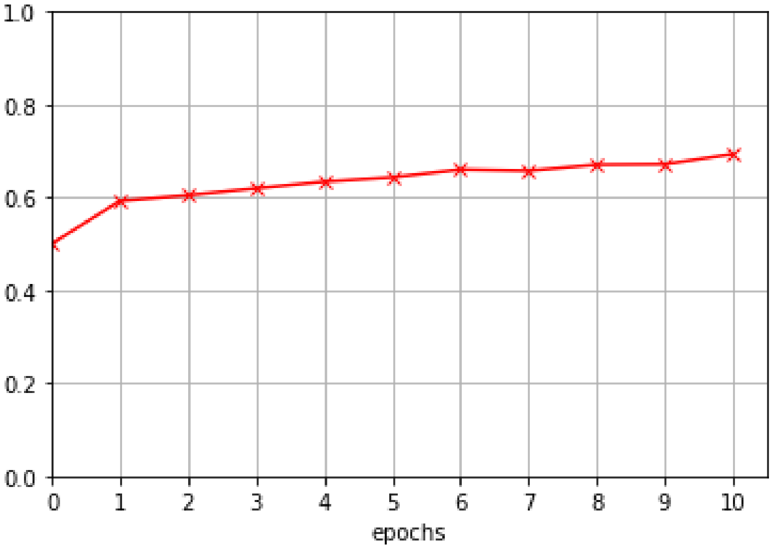


Figure 5.16. *The variation of alpha with respect to the number of epoch.*

- If we observe the diagram 5.17, we can conclude a similar trend as this in the previous note. BERT embeddings get the most attention of all detectors and indeed it continues rising up. Emotional's attention get a marginal decrease, as irony's attention increases in an opposite rate. However, in the last 4 epochs, both of them seem to remain stable, having emotional detector as more important the irony one. Finally, the personality detector is proven bad as we have already discussed. More specifically, it tends to 0, contributing the less to the classification decision.

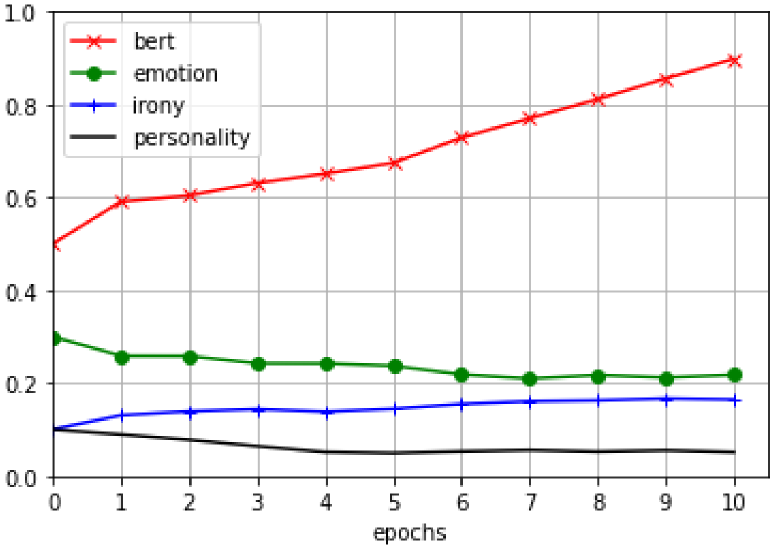


Figure 5.17. *The fluctuations of the four parameters of BEIP with parameters model.*

- Stagefull model seem to achieve a poor general performance. Adding more BERT information into the Emotional embeddings has a negative effect to them and it recant the emotion detector's strength.

- The three fusion methods that are presented in 5.14 and occur in the post level, combining the post's representations extracted from the emotional detector attentionally with the BERT embeddings, enhance, also, the overall performance of the Bert-Emotion model. If we compare these methods with the fusion methods on the user-level, we notice that they achieve a similar general performance (f1 score) but different rates of precision/recall, having lower recall and higher precision. In other words, the early attentional fusion has caused less samples to be classified as depressive keeping the overall ratio constant. That makes sense as the fusion takes place really early and therefore, the bad influence of the emotional information (which used to lead the model towards depression's bias) is less. Yet, the model does not change its ability to distinguish in the same level as before, as both the false positive and true positive samples are diminished. Among the three different techniques, attentional affine transformation method seem to work better.

5.7.4 Handcrafted Features

As analyzed in 4.5, the need to apply handcrafted features in our models came up because of the nature of our task. Depression detection from social media posts is not a simple task, where we have to look for just sad posts. A depressive user has not really important differences than a healthy user about their emotions in the posts they publish. It is possible depressive users to put more emotional words in their writings, but being depressive does not infer that he/she will have only depressive posts. That's why we have to search more about lexical features that can distinguish the depressive users more easily. To this goal, we, firstly, explore the posts of some users to detect possible patterns. We also, try to look some examples about the posts that affected our models' decision. For this purpose, we focus on our best model, B 1x7. Having the attention layer on the top of the architecture, we detect in which posts of every classification it gives more attention, assigning higher attention scores. For ethical reasons, we are not allowed to publish the posts of the users (or even a part of them) or give more information about them. Therefore, we cannot share in this work the posts that were given the most attention. We examined the top 5

	mean number of	
training set	posts of healthy users	895.4270
	posts of depressive users	1739.5355
	words of healthy users	24.3848
	words of depressive users	40.7978
testing set	posts of healthy users	908.1891
	posts of depressive users	1741.0013
	words of healthy users	24.3858
	words of depressive users	40.8768

Table 5.14. *The average number of posts/user and words/post used by either depressive or healthy group.*

	profanity
healthy users	0.13630575
depressive users	0.15048312

Table 5.15. *The mean profanity in the posts of the two groups.*

posts of some users that were classified either correctly or wrongly, trying to find some patterns that led to the specific label. In the following paragraphs, we analyze some features.

Posts and Words Count. One of the most obvious things that anyone can notice from a small portion of the samples is the differences in the numbers of posts and words the two different groups use. The statistics related to those are summarized in 5.14.

We notice that depressive users have published approximately the double posts of the healthy. This feature, however, cannot be used in our architecture, as it is a data-dependent feature and can be changed from dataset to dataset. In other words, if we keep in consideration this characteristic, our model will not be able to generalize in future datasets. On the other hand, we can exploit the number of words used by the two group for every post in average. The analogy of words count between depressive and healthy group is 5:3 and that makes it a considerable difference. In practice, it is a fact that depressive users tend to be more talkative and descriptive and this reflect to their writings. They tend to over analyze the situations and their stories, as well as to express their opinions elaborately. This relationship between depression and rumination is further discussed in [96]. Hence, it could be a future feature in our analysis.

Profanity. After printing the posts with the highest attention scores of the users that were classified as depressive, we observed a lot of swear words in those texts. After that, we decided to explore the profanity of the posts and the users and how this can affect our classification task. In the initial training set, we computed the profanity score (the probability of this sentence to involve swear words) of each sentence and we took a mean of the profanity of all posts, having one number for every user that indicates its profanity level. Taking the average of all the users, the profanity is finally depicted in 5.15. The difference may not seem so important, but it proves our observations and that is a good sign that may help us in our task.

We run also the B 1x7 model and measured the profanity for the True Positive, True Negative, False Positive and False Negative samples. The results are shown in 5.16.

It is noteworthy that the profanity score for the healthy users that were classified as depressive is really high, making the impression that profanity may play a significant role in the classification. Also, as it is known in ethical AI the biases usually become more obvious through the model. This

	profanity
healthy users classified as healthy	0.13616985
healthy users classified as depressive	0.15091686
depressive users classified as healthy	0.14035547
depressive users classified as depressive	0.15014291

Table 5.16. *The mean profanity in the posts of the two groups.*

moral dimension	feature	healthy	depressive
Care/Harm	average	4.07348569	**4.26168489**
	% of posts	6.418146%	**10.233882%**
Fairness/Cheating	average	**7.32488846**	7.19491389
	% of posts	6.464941%	**10.451726%**
Loyalty/Betrayal	average	6.32002966	**6.47052218**
	% of posts	4.722318%	**8.131549%**
Authority/Subversion	average	**6.36485541**	6.35726845
	% of posts	6.495626%	**11.158458%**
Purity/Degradation	average	**6.72920755**	6.6377047
	% of posts	8.199081%	**12.222311%**

Table 5.17. *The mean profanity in the posts of the two groups.*

is verified by the fact that the profanity difference became bigger.

Morality. LIWC lexicon [25] is one of the most popular dictionaries used in tasks related to sentiment analysis. It includes a huge variety of dimensions and this gives us the opportunity to explore more lexical aspects of the language of depressive users. Morality is an aspect highly related to depressive users [97]. This lead us to deepen more in the moral dimensions of LIWC. In [29] they expand the moral dimensions with more words and they give a number in every one of them, naming it "moral strength". This number/score is on 9-scale and every time as higher it is, it approaches the good case of the dimension. For example, in the moral dimension Care/Harm, a word with score 9 belongs totally to "Care" class, whereas a 1-score word belongs to "Harm" class. Taking the 5 moral dimensions, we developed 2 features for each, resulting to 10 features for every sample. The first feature is the average score for every user and the second one is the percentage of the posts containing words belonging to this moral dimension for every user. The statistics of the features (the mean values) for both depressive and healthy users are presented in 5.17.

We observe the the main differences between the two classes are about the percentage of the posts containing moral words. The average score of posts does not differ importantly and that makes sense, as the depressive status of a user does not affect the direction of his/her moral state. Depressive tend to talk and write more morally and they are more aware about morality and it is imprinted into the percentage feature.

Incorporation. The next step to our analysis is to incorporate the above features during training to validate if they enhance our model or not. As the features we extract are for each user, we incorporate them in the user-level of our architecture, concatenating them just before the output layer, as depicted in 5.18.

We experiment under the same circumstances as them described in 5.7.2. We use the gate mechanism + sigmoid model as the base model as it is the one with the best performance till now. The experiments adding the suitable handcrafted features are presented in 5.18. The "profanity

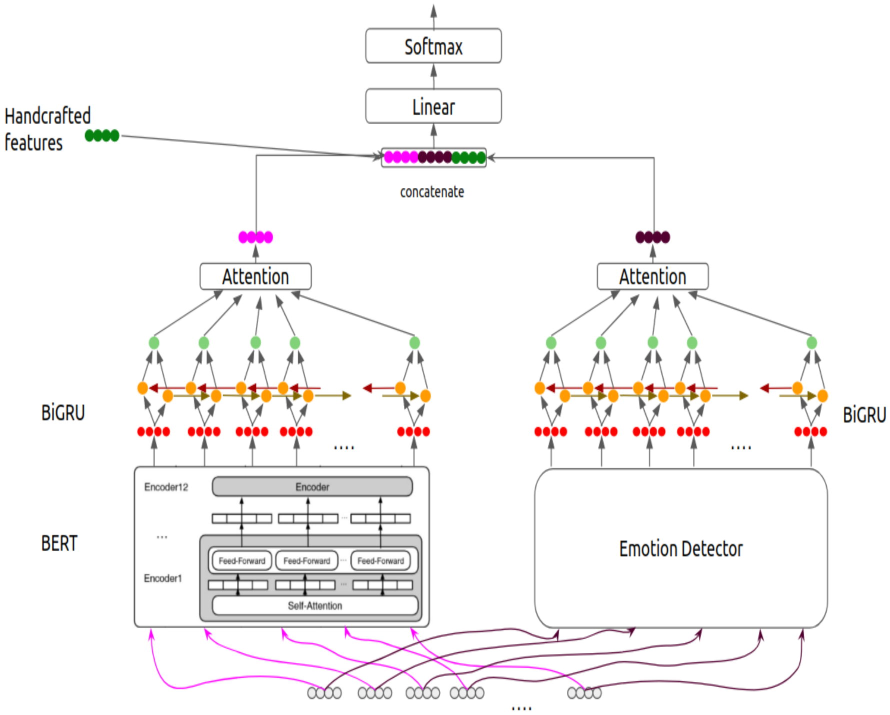


Figure 5.18. *The incorporation of the handcrafted features in the user-level of our architecture, right before the output layer.*

filter" architecture uses the profanity score as a filter score rather than a feature. The core idea is that when we have a larger profanity score, there is the need to classify more users as depressive. Therefore, when the profanity score is above a threshold, then the model follows the path of "gate mechanism + sigmoid" architecture, as it should considerate emotion's embeddings as well. On the contrary, if it is below the threshold, it decides only relied on BERT embeddings. The threshold is defined in 0.143. Profanity simply taken as a feature creates a 1d vector, whereas morality features create a 10d vector, putting the average score and percentage of each of the 5 moral dimensions. Words_count is also a 1d vector.

As we can observe, profanity seem to have the best results among all the possible handcrafted features. It seem to bring precision and recall in a balance and it achieves the best f1-score. The confusion matrices for comparison are provided in 5.19. Putting profanity either as a filter or as a feature, we achieve to reduce the users classified as depressive, in general. Moreover, having profanity as a feature, we enhance our model as both the false positive samples are reduced and the true positive are increased. That gives us the best performance overall. Morality achieves really similar results as profanity. However, combining profanity with morality, does not improve further the performance, but instead it keeps it stable, increasing the precision and diminishing the recall. That means that in that case, more users were classified as healthy. Finally, words_count feature does not provide us the desired results and it seems inappropriate for our task.

5.8 More Datasets

5.8.1 Depression

In this section, we will expand our work to more datasets related to mental health issues. At first, we will extend our application to one more depression detection dataset extracted from

	precision	recall	f1-score
gate mechanism + sigmoid	0.54	0.66	0.59
	0.75	0.81	0.78
profanity filter	0.59	0.59	0.59
	0.78	0.78	0.78
gate + sigmoid + profanity	0.64	0.61	**0.63**
	0.81	0.79	0.80
gate + sigmoid + morality	0.63	0.60	0.62
	0.80	0.79	0.79
gate + sigmoid + morality	0.71	0.55	0.62
+ profanity	0.84	0.76	0.79
gate + sigmoid + morality	0.49	0.71	0.58
+ profanity + words_count	0.73	0.82	0.77

Table 5.18. *Results of the incorporation of handcrafted features in the user-level of the architecture.*

	healthy	depressive
healthy	34,042	1,733
depressive	1,047	2,023

(a) *gate*

	healthy	depressive
healthy	34,498	1,277
depressive	1,257	1,813

(b) *profanity filter*

	healthy	depressive
healthy	34,738	1,037
depressive	1,193	1,877

(c) *gate + profanity*

Table 5.19. *Confusion Matrices for comparison between three different models.*

Reddit. The dataset was built by Inna Pirina et al. [9] and consists of a list of depressed and non-depressed users. Since the users are often active in different subreddits, each group is created by a corresponding random number of messages covering different topics. The data corpus contains depression-indicative posts (1293) and standard posts (548). Depression-indicative posts are collected from relatively large subreddits devoted to depression, where depressed users seek support from an online community. Standard posts written by non-depressed users are collected from subreddits related to a family or friends.

This dataset contains only posts and not users. So, we move our architecture one level down. In particular, the post-level is now considered as a word-level and the user-level is considered as the post-level. That means that both BERT and Emotion Detector extract representation for the words of the post and not a unified representation for the whole post. We will experiment with the 7 fusion methods that were described in the previous section (the Attentional Affine Transformation method is divided in two: we use a non-linear tanh function as an extra experiment). Therefore, we will use the architectures of 5.10 and 5.13. To implement the emotion detector, we will finetune a BERT for 3 epochs in the dataset of GoEmotion [98], including all 27 emotions. This dataset consists of 58,009 Reddit comments labeled by crowd workers. We then use transfer learning as usually, freezing all the layers of the emotion detector. We also use the classical BERT freezed (only for inference) to extract the BERT embeddings. The state-of-the-art models in this dataset are taken from the works of [7] and [8] and they are concentrated on 5.20.

Finally, our results, including the seven fusion methods that are depicted in 5.11 and 5.14 are presented in 5.20. Moreover, in each one of the seven fusion methods, we infuse the handcrafted features of profanity and morality. About profanity, the statistics of the dataset are depicted in 5.21. We observe a marginal difference between the two profanity scores and hence, we guess that it can produce some helpful outcomes. About morality, we follow a different approach. We produce the histograms of both depressive and healthy group for the five moral dimensions as shown in 5.19. Then, we integrate the corresponding distribution of every post into the final post's representation

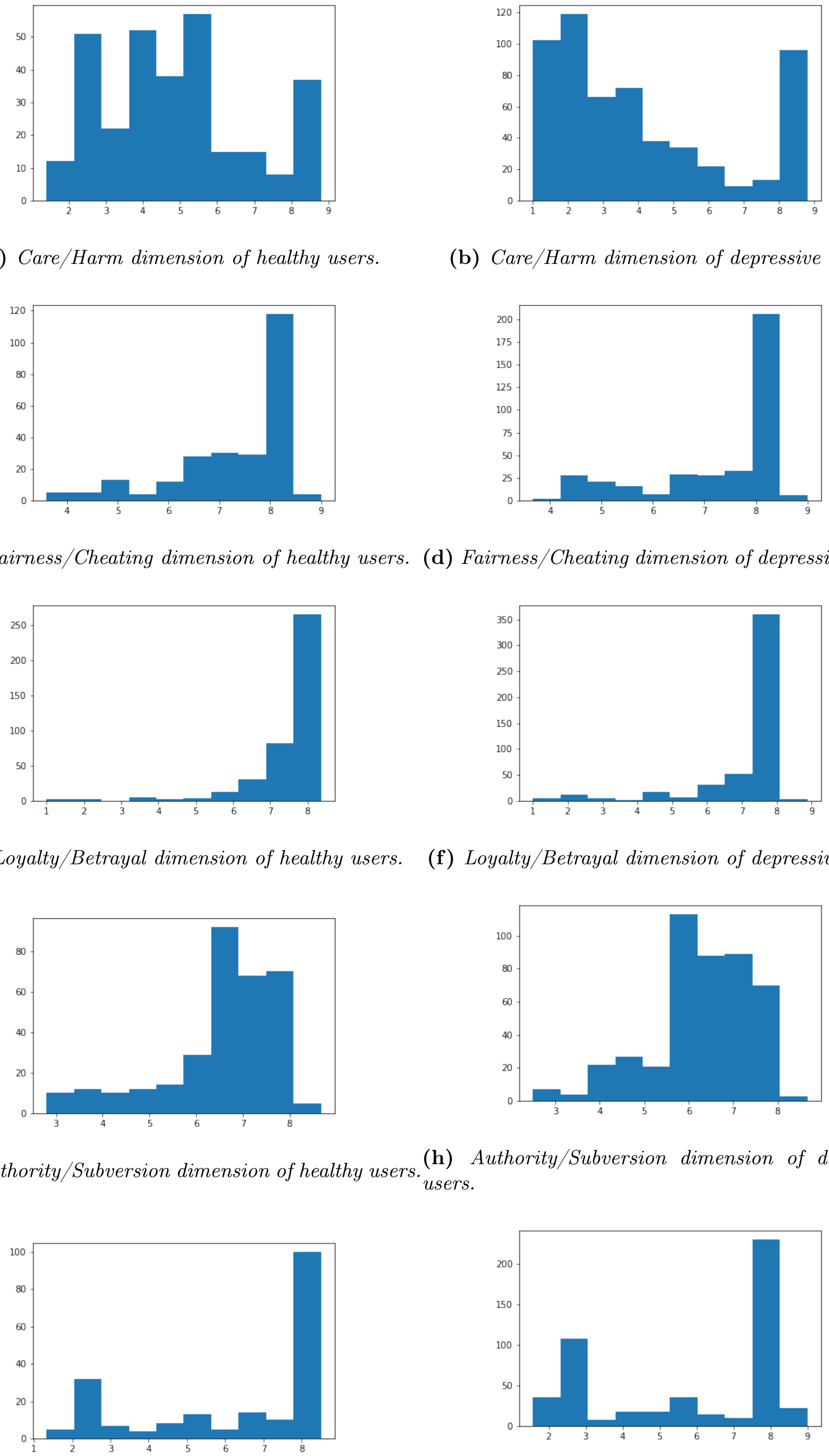
	Acc	F1	P	R
LIWC [8]	70	72	74	71
LDA [8]	75	74	75	72
unigram [8]	70	81	71	95
bigram [8]	79	78	80	76
LIWC+LDA+unigram [8]	78	81	84	79
LIWC+LDA+bigram [8]	91	93	90	92
LSTM [7]	87.03	-	90.30	-
Bi-LSTM [7]	86.46	-	88.08	-
BiLSTM+Att [7]	88.59	-	90.41	-
EAN [7]	91.30	-	91.91	-
concat 5.14a	**93.03**	**95.25**	94.24	**96.35**
gate 5.14b	91.79	94.29	94.34	94.23
affine 5.14c	92.87	95.00	**95.92**	94.11
non-linear affine 5.14c	92.87	95.01	95.83	94.23
Concatenation 5.11a	88.65	91.95	91.60	92.31
α parameter 5.11b	89.19	92.48	90.44	94.62
gate + sigmoid 5.11c	90.27	93.18	91.79	94.62

Table 5.20. *The performance of the models of [7] and [8] in the dataset [9], as also our model with the seven different architectures: the first four fuse the two kinds of representation in the word-level and the last three combine the representations in a post-level, right before the output layer.*

	profanity
healthy users	0.16583891
depressive users	0.30781266

Table 5.21. *The mean profanity in the posts of the two groups.*

before the output layer. The 5.23 shows the results of every case. This time, we also try to infuse the handcrafted features in a different level in our architecture, namely in the word-level, instead of only the post-level. More specifically, for the cases where the fusion of the emotional information occurs in the word-level (i.e. the cases in 5.14), the handcrafted features are integrated as a part of the emotional information. After the emotional representations are passed through the linear layer, they are concatenated with the handcrafted features before the attentional fusion. When the fusion occurs in the post-level (i.e. the cases in 5.11), the handcrafted features are infused in the word-level in both the bert's and emotion's representations. The fusion is the attention-based gating method, as it was proved as the best method among the attention-based methods. This incorporation's technique is noted as "WL" in 5.23.



(a) *Care/Harm dimension of healthy users.*

(b) *Care/Harm dimension of depressive users.*

(c) *Fairness/Cheating dimension of healthy users.*

(d) *Fairness/Cheating dimension of depressive users.*

(e) *Loyalty/Betrayal dimension of healthy users.*

(f) *Loyalty/Betrayal dimension of depressive users.*

(g) *Authority/Subversion dimension of healthy users.*

(h) *Authority/Subversion dimension of depressive users.*

(i) *Purity/Degradation dimension of healthy users.*

(j) *Purity/Degradation dimension of depressive users.*

Figure 5.19. *Histograms of morality dimensions.*

	Acc	F1	P	R
concat	93.03	95.25	94.24	96.35
+ profanity	93.51	95.56	96.27	94.85
+ profanity WL	94.05	95.65	93.80	**97.58**
+ morality	**96.22**	**97.30**	**97.67**	96.92
+ morality WL	92.43	94.66	93.94	95.38
+ profanity + morality	93.51	95.49	96.21	94.78
+ profanity + morality WL	92.43	94.66	93.94	95.38

(a) *Results of integrating handcrafted features into concatenation fusion method in the word-level.*

	Acc	F1	P	R
gate	91.79	94.29	94.34	94.23
+ profanity	94.05	95.85	96.95	94.78
+ profanity WL	92.97	94.90	92.37	**97.58**
+ morality	92.97	94.98	95.35	94.62
+ morality WL	92.97	95.02	94.66	95.38
+ profanity + morality	**94.59**	**96.24**	**96.97**	95.52
+ profanity + morality WL	91.35	93.89	93.18	94.62

(b) *Results of integrating handcrafted features into gating fusion method in the word-level.*

	Acc	F1	P	R
affine	92.87	95.00	95.92	94.11
+ profanity	94.59	96.24	**96.97**	95.52
+ profanity WL	94.05	95.62	94.49	96.77
+ morality	**95.68**	**96.92**	96.92	**96.92**
+ morality WL	92.97	95.02	94.66	95.38
+ profanity + morality	94.05	95.85	96.95	94.78
+ profanity + morality WL	93.51	95.42	94.70	96.15

(c) *Results of integrating handcrafted features into affine fusion method in the word-level.*

	Acc	F1	P	R
non-linear affine	92.87	95.01	95.83	94.23
+ profanity	**95.14**	**96.65**	**97.74**	95.59
+ profanity WL	94.05	95.62	94.49	**96.77**
+ morality	94.05	95.79	95.42	96.15
+ morality WL	92.97	95.02	94.66	95.38
+ profanity + morality	94.05	95.85	96.95	94.78
+ profanity + morality WL	93.51	95.42	94.70	96.15

(d) *Results of integrating handcrafted features into non-linear affine fusion method in the word-level.*

	Acc	F1	P	R
Concatenation	88.65	91.95	91.60	92.31
+ profanity	89.19	92.31	92.31	92.31
+ profanity WL	**91.89**	**94.25**	**93.89**	**94.62**
+ morality	90.27	93.18	91.79	**94.62**
+ morality WL	91.35	93.89	93.18	**94.62**
+ profanity + morality	89.73	92.66	93.02	92.31
+ profanity + morality WL	91.35	93.89	93.18	**94.62**

(a) *Results of integrating handcrafted features into concatenation fusion method in the post-level.*

	Acc	F1	P	R
α parameter	89.19	92.48	90.44	94.62
+ profanity	87.03	91.04	88.41	93.85
+ profanity WL	**91.89**	94.25	**93.89**	94.62
+ morality	87.03	91.04	88.41	93.85
+ morality WL	**91.89**	94.25	**93.89**	94.62
+ profanity + morality	87.57	91.64	89.05	93.85
+ profanity + morality WL	**91.89**	**94.30**	93.23	**95.38**

(b) *Results of integrating handcrafted features into alpha fusion method in the post-level.*

	Acc	F1	P	R
gate + sigmoid	90.27	93.18	91.79	94.62
+ profanity	91.35	94.12	92.75	**95.52**
+ profanity WL	**91.89**	**94.25**	93.89	94.62
+ morality	90.81	93.77	92.09	**95.52**
+ morality WL	**91.89**	94.21	**94.57**	93.85
+ profanity + morality	90.81	93.77	92.09	**95.52**
+ profanity + morality WL	91.35	93.89	93.18	94.62

(c) *Results of integrating handcrafted features into gating fusion method in the post-level.*

Table 5.23. *The results of the seven fusion methods, when the profanity and morality are infused into them in the post-level.*

Initially, we observe in 5.20 that all our seven proposed fusion methods beat the baseline models and the word-level fusion methods beat the state-of-the-art models, making the 5.14a the best model in the terms of Accuracy, F1 score and Recall, while the 5.14c is the best in terms of Precision. The results show that the infusion of the emotional information can only enhance the performance, independently of how this fusion takes place. The three cases that make the fusion on the post-level have a general higher recall, but lower precision than the cases of the attentional fusion in the word-level. That can be explained by the fact that putting the emotional information in a later stage of our architecture make our classification results rely on the emotion more and as a consequence to classify more posts as depressive, even if they are not and they just have a negative emotion. 5.21 and 5.19 show the importance of handcrafted features, as there are some important differences between the two classes. Profanity and some moral dimensions (mainly Care/Harm and Purity/Degradation) are suitable to make the classes more distinguishable and hence, it makes sense to try integrating them to our models. This is, also, verified by the results in 5.23. In the four cases of attentional fusion in the word-level, profanity and morality enhance remarkably the performance, reaching even at an accuracy of 96.22% in the case of attentional concatenation. The "WL" cases seem to have a bit lower performance (mainly in terms of accuracy and f1-score), compared to their respective case in the post-level. This makes sense, because when we put the handcrafted features later in the model, we make it give them more importance and be affected by them. In a similar way, the handcrafted features seem to be important in the three cases of late emotional fusion. One interesting trend here is that the WL techniques achieve a better general performance. We assume that this occurs because the features are integrated in both BERT and emotion detector in the word-level and then they are combined again in the post-level. As a result, we consider their influence twice. Contrariwise, in the previous case, we considered the influence of the handcrafted features only once, as they were infused in the bert's representations as a part of the emotional information. In both cases, we notice that the combination of the profanity and morality does not improve necessarily the results. This cannot be explained intuitively. One possible reason is the dataset to contain depressive posts that are distinguishable either in terms of profanity or in terms of morality but not both. On the contrary, the posts that have a higher profanity and a morality similar to one of a depressive post, may have equal probability to belong to either healthy or depressive posts. Finally, profanity and morality have an equal importance to our models.

Experimental Setup. Emotion detector: We finetuned the pretrained BERT of huggingface for 3 epochs, with a batch size of 16. Binary Cross Entropy loss was used as a criterion loss and the Adam optimizer with learning rate $2 \cdot 10^{-5}$ and epsilon equal to 10^{-8}.

General architecture: For every moral dimension, we extracted the histograms in 8 bins and we took the distribution of every post as an input. Then having a vector of 40d (5 dimensions x 8 bins) we use a linear layer to reduce the dimension to 5. The BiGRU of Bert's side has hidden size 128 and then the output is a vector of 256d. We use also dropout of probability 0.2. The linear layer of emotion's side project the information to a 128d space. In "WL" cases, we pass the profanity through a linear layer, in order to project the vector in a 5d space. We split the dataset into 90/10 train/test set and we use cross-validation during training with 10 folds, trained for 5 epochs. Finally, we use Cross Entropy loss and Adam optimizer with learning rate $5 \cdot 10^{-5}$ and epsilon equal to 10^{-8}.

5.8.2 Stress

Our final contribution in this thesis was to move on another task that is similar to ours (belonging to mental health issues), but includes different characteristics in its data. Stress detection is a task that can be affected a lot of the emotional information that is included in the posts or text and therefore, we chose this task to check our model of infusing the emotional information in a classic BERT model.

For this purpose, we decided to use the Dreaddit dataset [12], which is online available and consists of 3,553 segments of Reddit posts from various support communities where the authors believe posters are likely to express stress. The stress detection problem as expressed in this dataset is a binary classification problem, with crowdsourced annotations aggregated as the majority vote from five annotators for each data point.

For the training of the emotion detection, we will use the GoEmotions dataset, as exactly in the previous subsection. The dataset is given also with some class mappings: one where labels are clustered together into the Ekeman 6 basic emotions (anger, disgust, fear, joy, sadness, surprise, neutral) ([99]), and one into simple polarity (positive, negative, ambiguous, neutral). We run our experiments with each version of this dataset. The state-of-the-art models are derived from [10] and [11]. In [10], they have a similar approach, where they use GoEmotion dataset auxiliary in three different ways: a simple multi-task learning where the stress detection and emotion detection tasks are learned concurrently and symbolized as Multi1, one where a BERT trained firstly in the GoEmotions dataset and then the emotion labels that are extracted, are used as "silver data" in stress detection task, symbolized as Multi2, and finally one BERT that is fine-tuned in both

	Binary F1	Accuracy
RNN [10]	67.58 ±1.22	68.86 ±1.10
BERT [10]	78.88 ±1.09	79.11 ±1.32
MentalBERT [11]	80.04	-
MentalRoBERTa [11]	81.76	-
Multi1 ALL [10]	79.02 ±0.35	**79.72** ±0.69
Multi2 ALL[10]	78.97 ±0.24	78.55 ±0.07
FT ALL [10]	76.40 ±0.50	76.83 ±0.40
Concatenation ALL 5.11a	79.90 ±0.88	78.29 ±0.33
α parameter ALL 5.11b	80.20 ±1.20	78.70 ±0.68
gate + sigmoid ALL 5.11c	**80.76** ±0.38	79.50 ±0.43
concat ALL 5.14a	79.23 ±0.74	78.00 ±0.35
gate ALL 5.14b	79.90 ±0.30	77.62 ±0.64
affine ALL 5.14c	79.74 ±0.33	78.14 ±0.63
non-linear affine ALL 5.14c	79.62 ±1.10	78.32 ±0.37
Multi1 Ekman [10]	**80.24** ±1.39	**81.07** ±1.13
FT Ekman [10]	79.44 ±0.29	79.53 ±0.46
gate + sigmoid Ekman 5.11c	79.69 ±0.96	78.48 ±0.40
Multi1 Sentiment [10]	79.46 ±1.05	79.86 ±0.50
FT Sentiment [10]	**79.75** ±0.52	**80.61** ±0.40
gate + sigmoid Sentiment 5.11c	79.11 ±0.97	78.06 ±0.40

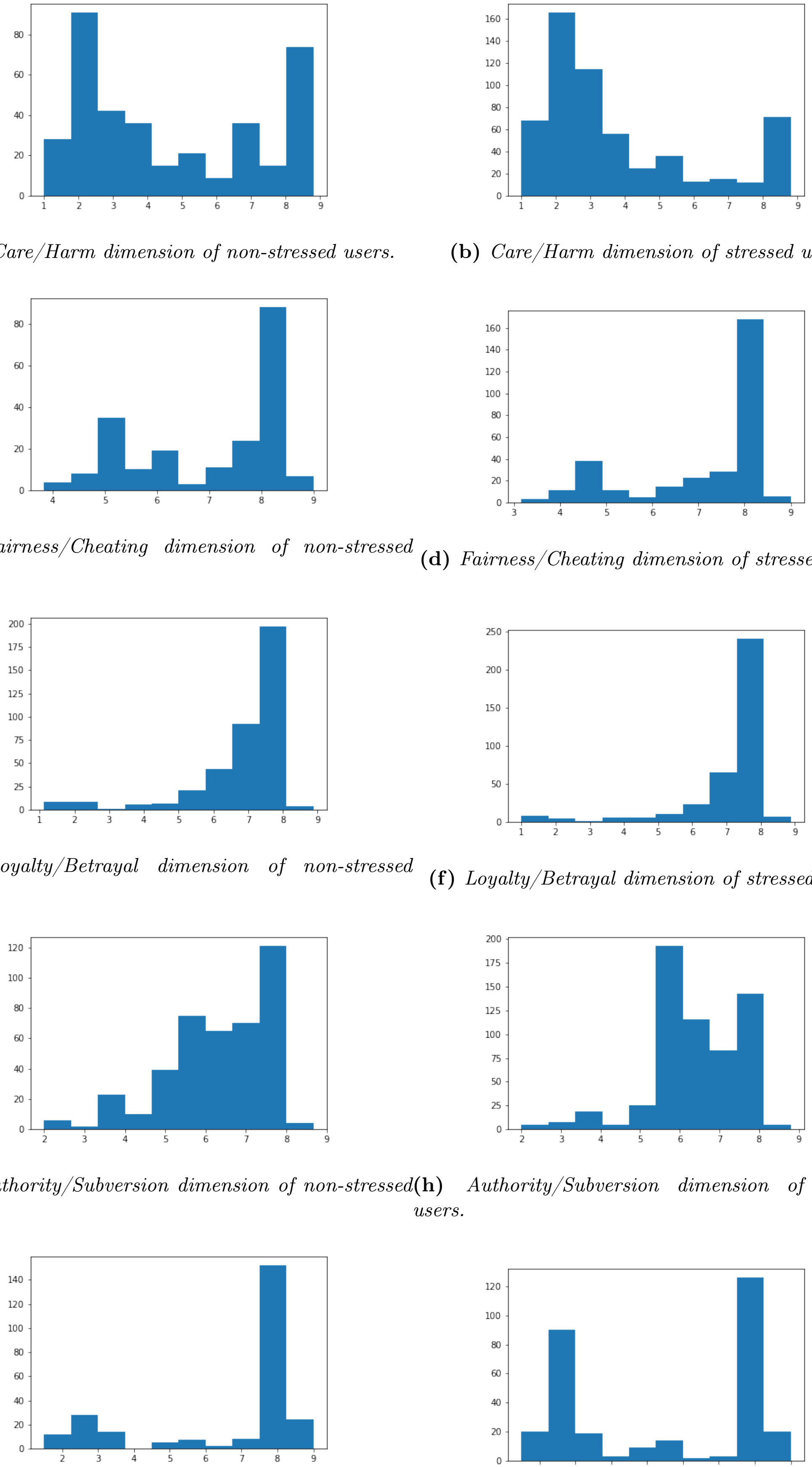
Table 5.24. *The performance of the models of [10] and [11] in the dataset [12], as also our models' results. The table is divided in 5 sections. In the first one, there are the baselines. In the second one, the models of [11] are included, while in the next three the rest of the models are presented, according to which subset of the GoEmotions has been used : ALL, Ekman or Sentiment subset.*

	profanity
non-stressed users	0.09232399
stressed users	0.19555190

Table 5.25. *The mean profanity in the posts of the two groups.*

dataset, symbolized as FT. In [11], they pretrain a BERT and a RoBERTa with Reddit posts and they use the stress detection task (among with some others) as a downstream task. The state-of-the-art models, combined with our results are summarized in 5.24.

Moreover, in each one of the seven fusion methods, we infuse the handcrafted features of profanity and morality, as exactly we did in the previous subsection. About profanity, the statistics of the dataset are depicted in 5.25. We observe a marginal difference between the two profanity scores and hence, we guess that it can produce some helpful outcomes. About morality, we follow the previous approach with the histograms. The histograms for the five moral dimensions as shown in 5.20. Then, we integrate the corresponding distribution of every post into the final post's representation before the output layer. We also apply the "WL" technique we explained in the previous subsection. The 5.26 shows the results of every case.



(a) *Care/Harm dimension of non-stressed users.*

(b) *Care/Harm dimension of stressed users.*

(c) *Fairness/Cheating dimension of non-stressed users.*

(d) *Fairness/Cheating dimension of stressed users.*

(e) *Loyalty/Betrayal dimension of non-stressed users.*

(f) *Loyalty/Betrayal dimension of stressed users.*

(g) *Authority/Subversion dimension of non-stressed users.*

(h) *Authority/Subversion dimension of stressed users.*

(i) *Purity/Degradation dimension of non-stressed users.*

(j) *Purity/Degradation dimension of stressed users.*

Figure 5.20. *Histograms of morality dimensions.*

	Acc	F1
concat	78.00	79.23
+ profanity	76.41	78.61
+ profanity WL	**78.04**	**80.05**
+ morality	77.76	79.80
+ morality WL	77.06	79.03
+ profanity + morality	77.62	78.26
+ profanity + morality WL	76.92	78.54

(a) *Results of integrating handcrafted features into concatenation fusion method in the word-level.*

	Acc	F1
gate	77.62	79.90
+ profanity	78.18	79.72
+ profanity WL	77.48	**80.10**
+ morality	**78.32**	79.79
+ morality WL	77.62	79.38
+ profanity + morality	77.90	80.00
+ profanity + morality WL	77.76	80.05

(b) *Results of integrating handcrafted features into gating fusion method in the word-level.*

	Acc	F1
affine	78.14	79.74
+ profanity	78.60	80.02
+ profanity WL	76.50	78.95
+ morality	77.48	**80.20**
+ morality WL	**79.16**	80.05
+ profanity + morality	78.46	80.10
+ profanity + morality WL	77.50	79.04

(c) *Results of integrating handcrafted features into affine fusion method in the word-level.*

	Acc	F1
non-linear affine	78.32	79.62
+ profanity	77.20	79.23
+ profanity WL	78.46	78.31
+ morality	**78.88**	**80.57**
+ morality WL	78.32	80.15
+ profanity + morality	78.04	80.25
+ profanity + morality WL	78.18	79.90

(d) *Results of integrating handcrafted features into non-linear affine fusion method in the word-level.*

	Acc	F1
concatenation	78.29	79.90
+ profanity	78.04	79.90
+ profanity WL	78.04	79.53
+ morality	77.34	79.34
+ morality WL	**79.02**	**80.42**
+ profanity + morality	78.18	79.90
+ profanity + morality WL	78.32	79.14

(e) *Results of integrating handcrafted features into concatenation fusion method in the post-level.*

	Acc	F1
α parameter	**78.70**	80.20
+ profanity	78.32	79.90
+ profanity WL	77.76	79.48
+ morality	78.32	79.95
+ morality WL	77.20	78.75
+ profanity + morality	77.48	79.39
+ profanity + morality WL	78.46	**80.31**

(f) *Results of integrating handcrafted features into alpha fusion method in the post-level.*

	Acc	F1
gate + sigmoid	**79.50**	80.76
+ profanity	78.04	79.90
+ profanity WL	78.46	**80.80**
+ morality	78.04	79.64
+ morality WL	77.48	80.10
+ profanity + morality	78.32	79.95
+ profanity + morality WL	78.04	80.55

(g) *Results of integrating handcrafted features into gating fusion method in the post-level.*

Table 5.26. *The results of the seven fusion methods, when the profanity and morality are infused into them in the post-level.*

Initially, we observe in 5.24 that our methods does not improve the models remarkably. The pretrained models in medical text that were finetuned in Dreaddit ([11]) achieve the best binary F1 score of all, while the best accuracy is noted by the Multi1 model with using the Ekman mapping in GoEmotions. However, the "gate+sigmoid" model (5.11c) achieve a noteworthy performance beating the rest similar cases that use the assistance of emotional information. That means that this specific fusion method helps in our task, but the idea of integration of the emotional information seems to be beaten easily by other models in the task of stress detection. 5.21 and 5.19 show the importance of handcrafted features, as there are some important differences between the two classes. Profanity and the most of moral dimensions are suitable to make the classes more distinguishable and hence, it makes sense to try integrating them to our models. This is, also, verified by the results that are presented in 5.26 as they follow a similar pattern with those that were noted in 5.23. The trends in the results are similar with those explained in the previous dataset. The "WL" cases seem to work better in this dataset in both of the kinds of fusion level. In general, profanity and morality seem to be important for our model and this is an important conclusion for the relationship between those and the task of stress detection.

Experimental Setup. Emotion detector: We finetuned the pretrained BERT of huggingface for 3 epochs, with a batch size of 16. Binary Cross Entropy loss was used as a criterion loss and the

Adam optimizer with learning rate $2 \cdot 10^{-5}$ and epsilon equal to 10^{-8}. The same finetuning was for Ekman and Sentiment class mapping.

General architecture: For every moral dimension, we extracted the histograms in 8 bins and we took the distribution of every post as an input. Then having a vector of 40d (5 dimensions x 8 bins) we use a linear layer to reduce the dimension to 5. The BiGRU of Bert's side has hidden size 128 and then the output is a vector of 256d. We use also dropout of probability 0.2. The linear layer of emotion's side project the information to a 128d space. In "WL" cases, we pass the profanity through a linear layer, in order to project the vector in a 5d space. We split the training set into 90/10 train/validation set and we train the model with early stopping for maximum of 20 epochs, having patience = 5 and tolerance = 0.0001. Finally, we use Cross Entropy loss and Adam optimizer with learning rate $5 \cdot 10^{-5}$ and epsilon equal to 10^{-8}.

Chapter 6

Conclusions

6.1 Discussion

In this work, we investigate methods and architectures in order to tackle the task of depression detection from social media posts. We perform text-based classification, divided in two levels, a post-level and a user-level. Our approaches enhance the state-of-the-art models, which have been tested in the specific dataset and they achieve a better performance than the existent baselines.

At first, we conducted a full ablation study about the influence of some aspects of human behaviour to the depression detection. These can be considered as symptoms of the depressive users that are declared to the language they use. The aspects that were exploited are Emotion, Sarcasm and Personality. We use BERT to extract contextualised representations of every post of the user, as it has been proven state-of-the-art in many tasks. We conclude that BERT can achieve itself the best results, incorporating a simple deep learning architecture in the user-level. To leverage the emotional information that is extracted from the emotion detector we apply, we further augment our system with the emotion detection in the post-level. Then, we experiment with multiple fusion layers that are applied in both the post-level and user-level. More specifically, we propose four different methods of incorporating the emotional representations of the posts in the corresponding Bert's ones and then they are trained together (in a unified representation) having a common user-level architecture. On the other hand, we try to extract a user's representation from BERT and Emotion detector and we combine those representation with three different ways, before proceeding to the output layer. Our results show that the Emotion detector is the most useful detector, as it detects the most emotional posts, making our model to detect the depressive users more easily. We propose the integration of emotional posts' representations into the bert's embeddings. This form of conditioning on the attention distribution, enforces the contribution of the most salient features for the task at hand. The emotional information reinforces the respective posts that are important in the classification problem. Moreover, we propose the user-level methods of fusion, in order to make the most salient features of the user's representation to contribute in the final decision. The results show that our methods increase the performance of our baseline models.

Secondly, we perform a thorough data analysis, so as to find some patterns the language use of depressive users that will assist us in the detection task. We focus on two major patterns, the usage of profanity and the moral language that depressive users often utilize. We incorporate these handcrafted features in our best model till now, which is a combination of BERT and emotion detector. It is shown that the results of the model, in which the handcrafted features are infused, are better than every result we have achieved in this specific task with this specific dataset. However, the combination of the features, putting all together, evoke inappropriate results, making us consider them as opposite asssists. To clarify, the profanity may not be used by different users than them who use a lot of moral words, as in that case, the combination of them would result a better performance. On the contrary, the combination of profanity and morality keep the users that classified as depressive constant, but make the depressive users that do not use much of both of the above to be classified as healthy, increasing the precision and decreasing the recall.

6.2 Future Work

Through the end of this thesis, we wish to open some new roads and create new aspects to explore the capabilities of a depression detection model. We suggest the following points be explored in future work:

- **Exploring the interpretability.** We started with the basics of interpretability in our effort to find useful handcrafted features that could be incorporated in our model. It is proposed to deepen more and find not only the posts, but also the words that contribute at most in the classification task.
- **Exploring the robustness.** We started checking the robustness of our model by experimenting with the genearlization's capabilities of it. We suggest to make our model more robust, mainly in adversarial attacks by creating some general triggers. One of the main ideas that can be applied to our task is the one of [100], where they built some universal triggers in order to trick the model.
- **Different handcrafted features.** One of the main ideas for the future work is to explore how useful some different handcrafted feature can be and which exactly will be. One core idea that is used in similiar occasions is the utilization of LIWC lexicon. Moreover, we can extract some characteristics of the depressive people' language, like the usual usage of past tense or their intense interest in more social and political issues.
- **Graph Neural Networks.** We can exploit the recent developments of Graph Neural Networks as they are recommended in datasets related to social media. Particularly, each user can be represented as a graph where its posts and the words of the posts will be the nodes of the graph and they will be connected by the same words. An another approach is to represent users as nodes and they will be connected in a similar way (by common words).
- **Probing.** Probing is related with the efforts for interepretability. Specifically, we can deepen and explore in which level of BERT we can take the most useful information for our task, by creating a probe task. In this way, we can have a lot of advantages. E.g. we can try incorporating the emotional information in that exact level or try to integrate the most useful handcrafted features.

Bibliography

[1] Jurafsky Daniel και H Martin James. *Speech and Language Processing*, 2000.

[2] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever και Ruslan Salakhutdinov. *Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research*, 15(1):1929–1958, 2014.

[3] Tianyang Lin, Yuxin Wang, Xiangyang Liu και Xipeng Qiu. *A Survey of Transformers*, 2021.

[4] Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville και Yoshua Bengio. *Feature-wise transformations. Distill*, 2018. https://distill.pub/2018/feature-wise-transformations.

[5] P. Vincent Y. Bengio, R. Ducharme. *A neural probabilistic language model. Journal of Machine Learning Research.* 2003.

[6] Guozheng Rao, Chengxia Peng, Li Zhang, Xin Wang και Zhiyong Feng. *A Knowledge Enhanced Ensemble Learning Model for Mental Disorder Detection on Social Media. International Conference on Knowledge Science, Engineering and Management*, σελίδες 181–192. Springer, 2020.

[7] Lu Ren, Hongfei Lin, Bo Xu, Shaowu Zhang, Liang Yang, Shichang Sun και others. *Depression detection on reddit with an emotion-based attention network: algorithm development and validation. JMIR Medical Informatics*, 9(7):e28754, 2021.

[8] Michael M Tadesse, Hongfei Lin, Bo Xu και Liang Yang. *Detection of depression-related posts in reddit social media forum. IEEE Access*, 7:44883–44893, 2019.

[9] Inna Pirina και Çağrı Çöltekin. *Identifying depression on reddit: The effect of training data. Proceedings of the 2018 EMNLP Workshop SMM4H: The 3rd Social Media Mining for Health Applications Workshop & Shared Task*, σελίδες 9–12, 2018.

[10] Elsbeth Turcan, Smaranda Muresan και Kathleen McKeown. *Emotion-infused models for explainable psychological stress detection. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, σελίδες 2895–2909, 2021.

[11] Shaoxiong Ji, Tianlin Zhang, Luna Ansari, Jie Fu, Prayag Tiwari και Erik Cambria. *MentalBERT: Publicly Available Pretrained Language Models for Mental Healthcare. arXiv preprint arXiv:2110.15621*, 2021.

[12] Elsbeth Turcan και Kathleen McKeown. *Dreaddit: A Reddit dataset for stress analysis in social media. arXiv preprint arXiv:1911.00133*, 2019.

[13] World Health Organization και others. *Depression and other common mental disorders: global health estimates.* Τεχνική Αναφορά με αριθμό, World Health Organization, 2017.

[14] Dzmitry Bahdanau, Kyunghyun Cho και Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*, 2016.

[15] Jacob Devlin, Ming Wei Chang, Kenton Lee και Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs]*, 2019. arXiv: 1810.04805.

[16] Danai Xezonaki, Georgios Paraskevopoulos, Alexandros Potamianos και Shrikanth Narayanan. *Affective conditioning on hierarchical attention networks applied to depression detection from transcribed clinical interviews. Interspeech 2020*, σελίδες 4556–4560, 2020.

[17] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov και Quoc V Le. *Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems*, 32, 2019.

[18] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song και Qiang Yang. *Large-scale hierarchical text classification with recursively regularized deep graph-cnn. Proceedings of the 2018 world wide web conference*, σελίδες 1063–1072, 2018.

[19] Chunting Zhou, Chonglin Sun, Zhiyuan Liu και Francis Lau. *A C-LSTM neural network for text classification. arXiv preprint arXiv:1511.08630*, 2015.

[20] Diederik P Kingma και Max Welling. *Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114*, 2013.

[21] Ian J Goodfellow, Jonathon Shlens και Christian Szegedy. *Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572*, 2014.

[22] Richard S Sutton και Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[23] Tao Yang, Feifan Yang, Haolan Ouyang και Xiaojun Quan. *Psycholinguistic Tripartite Graph Network for Personality Detection. arXiv preprint arXiv:2106.04963*, 2021.

[24] Yangyang Li, Yipeng Ji, Shaoning Li, Shulong He, Yinhao Cao, Xiong Li, Jun Shi, Yangchao Yang και Yifeng Liu. *Relevance-Aware Anomalous Users Detection in Social Network via Graph Neural Network. arXiv preprint arXiv:2104.06095*, 2021.

[25] James W Pennebaker, Martha E Francis και Roger J Booth. *Linguistic inquiry and word count: LIWC 2001. Mahway: Lawrence Erlbaum Associates*, 71(2001):2001, 2001.

[26] Saif Mohammad και Peter D. Turney. *Crowdsourcing a Word-Emotion Association Lexicon. CoRR*, abs/1308.6297, 2013.

[27] Andrew Yates, Arman Cohan και Nazli Goharian. *Depression and self-harm risk assessment in online forums. arXiv preprint arXiv:1709.01848*, 2017.

[28] Guozheng Rao, Yue Zhang, Li Zhang, Qing Cong και Zhiyong Feng. *MGL-CNN: A hierarchical posts representations model for identifying depressed individuals in online forums. IEEE Access*, 8:32395–32403, 2020.

[29] Oscar Araque, Lorenzo Gatti και Kyriaki Kalimeri. *MoralStrength: Exploiting a moral lexicon and embedding similarity for moral foundations prediction. Knowledge-based systems*, 191:105184, 2020.

[30] American Psychiatric Association και others. *Diagnostic and statistical manual of mental disorders (DSM-5®).* American Psychiatric Pub, 2013.

[31] Tom Mitchell. *Machine learning.* 1997.

[32] Christopher M Bishop. *Pattern recognition. Machine learning*, 128(9), 2006.

[33] Irina Rish και others. *An empirical study of the naive Bayes classifier. IJCAI 2001 workshop on empirical methods in artificial intelligence*, τόμος 3, σελίδες 41–46, 2001.

[34] William S Noble. *What is a support vector machine? Nature biotechnology*, 24(12):1565–1567, 2006.

[35] David E Rumelhart, Geoffrey E Hinton και Ronald J Williams. *Learning internal representations by error propagation.* Τεχνική Αναφορά με αριθμό, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[36] Yann LeCun, Léon Bottou, Yoshua Bengio και Patrick Haffner. *Gradient-based learning applied to document recognition. Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[37] Sepp Hochreiter και Jürgen Schmidhuber. *Long short-term memory. Neural computation*, 9(8):1735–1780, 1997.

[38] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk και Yoshua Bengio. *Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078*, 2014.

[39] Felix A Gers, Jürgen Schmidhuber και Fred Cummins. *Learning to forget: Continual prediction with LSTM. Neural computation*, 12(10):2451–2471, 2000.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser και Illia Polosukhin. *Attention Is All You Need. arXiv:1706.03762 [cs]*, 2017. arXiv: 1706.03762.

[41] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong και Qing He. *A Comprehensive Survey on Transfer Learning*, 2020.

[42] Ronen Feldman. *Techniques and applications for sentiment analysis. Communications of the ACM*, 56(4):82–89, 2013.

[43] Hamed Yaghoobian, Hamid R Arabnia και Khaled Rasheed. *Sarcasm Detection: A Comparative Study. arXiv preprint arXiv:2107.02276*, 2021.

[44] Navonil Majumder, Soujanya Poria, Alexander Gelbukh και Erik Cambria. *Deep learning-based document modeling for personality detection from text. IEEE Intelligent Systems*, 32(2):74–79, 2017.

[45] H. P. Luhn. *A Statistical Approach to Mechanized Encoding and Searching of Literary Information.* `https://doi.org/10.1147/rd.14.0309`, 1957.

[46] Karen Spärck Jones. *A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation*, 28:11–21, 1972.

[47] U. Fano. *Effects of Configuration Interaction on Intensities and Phase Shifts. Phys. Rev.*, 124:1866–1878, 1961.

[48] Kenneth Ward Church και Patrick Hanks. *Word Association Norms, Mutual Information, and Lexicography. 27th Annual Meeting of the Association for Computational Linguistics*, σελίδες 76–83, Vancouver, British Columbia, Canada, 1989. Association for Computational Linguistics.

[49] Tomas Mikolov, Kai Chen, Greg Corrado και Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*, 2013.

[50] Armand Joulin, Edouard Grave, Piotr Bojanowski και Tomas Mikolov. *Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759*, 2016.

[51] Jeffrey Pennington, Richard Socher και Christopher Manning. *GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, σελίδες 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics.

[52] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký και Sanjeev Khudanpur. *Recurrent Neural Network Based Language Model. Proceedings of the 11th Annual Conference of the International Speech Communication Association*, INTERSPEECH 2010, σελίδες 1045–1048. ISCA, 2010.

[53] Marcel Trotzek, Sven Koitka και Christoph M Friedrich. *Utilizing neural networks and linguistic metadata for early detection of depression indications in text sequences. IEEE Transactions on Knowledge and Data Engineering*, 32(3):588–601, 2018.

[54] Zhe Ye, Fang Li και Timothy Baldwin. *Encoding sentiment information into word vectors for sentiment analysis. Proceedings of the 27th International Conference on Computational Linguistics*, σελίδες 997–1007, 2018.

[55] Liu Yi Lin, Jaime E Sidani, Ariel Shensa, Ana Radovic, Elizabeth Miller, Jason B Colditz, Beth L Hoffman, Leila M Giles και Brian A Primack. *Association between social media use and depression among US young adults. Depression and anxiety*, 33(4):323–331, 2016.

[56] Betul Keles, Niall McCrae και Annmarie Grealish. *A systematic review: the influence of social media on depression, anxiety and psychological distress in adolescents. International Journal of Adolescence and Youth*, 25(1):79–93, 2020.

[57] Munmun De Choudhury, Michael Gamon, Scott Counts και Eric Horvitz. *Predicting depression via social media. Proceedings of the International AAAI Conference on Web and Social Media*, τόμος 7, 2013.

[58] Atefeh Farzindar και Diana Inkpen. *Natural language processing for social media. Synthesis Lectures on Human Language Technologies*, 8(2):1–166, 2015.

[59] John P Pestian, Pawel Matykiewicz, Michelle Linn-Gust, Brett South, Ozlem Uzuner, Jan Wiebe, K Bretonnel Cohen, John Hurdle και Christopher Brew. *Sentiment analysis of suicide notes: A shared task. Biomedical informatics insights*, 5:BII–S9042, 2012.

[60] Sandy H Huang, Paea LePendu, Srinivasan V Iyer, Ming Tai-Seale, David Carrell και Nigam H Shah. *Toward personalizing treatment for depression: predicting diagnosis and severity. Journal of the American Medical Informatics Association*, 21(6):1069–1075, 2014.

[61] Chris Poulin, Brian Shiner, Paul Thompson, Linas Vepstas, Yinong Young-Xu, Benjamin Goertzel, Bradley Watts, Laura Flashman και Thomas McAllister. *Predicting the risk of suicide by analyzing the text of clinical notes. PloS one*, 9(1):e85733, 2014.

[62] Graeme John Hirst, Regina Jokel, Dauphin Ian Lancashire και Xuan D Le. *Method and system of longitudinal detection of dementia through lexical and syntactic changes in writing*, 2016. US Patent 9,514,281.

[63] Thin Nguyen, Dinh Phung, Bo Dao, Svetha Venkatesh και Michael Berk. *Affective and content analysis of online depression communities. IEEE Transactions on Affective Computing*, 5(3):217–226, 2014.

[64] Glen Coppersmith, Mark Dredze, Craig Harman, Kristy Hollingshead και Margaret Mitchell. *CLPsych 2015 shared task: Depression and PTSD on Twitter. Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, σελίδες 31–39, 2015.

[65] Megan A Moreno, Lauren A Jelenchick, Katie G Egan, Elizabeth Cox, Henry Young, Kerry E Gannon και Tara Becker. *Feeling bad on Facebook: Depression disclosures by college students on a social networking site. Depression and anxiety*, 28(6):447–455, 2011.

[66] Ang Li, Xiaoxiao Huang, Bibo Hao, Bridianne O'Dea, Helen Christensen και Tingshao Zhu. *Attitudes towards suicide attempts broadcast on social media: an exploratory study of Chinese microblogs. PeerJ*, 3:e1209, 2015.

[67] Megan C Lytle, Vincent MB Silenzio, Christopher M Homan, Phoenix Schneider και Eric D Caine. *Suicidal and help-seeking behaviors among youth in an online lesbian, gay, bisexual, transgender, queer, and questioning social network. Journal of homosexuality*, 65(13):1916–1933, 2018.

[68] Zhentao Xu, Verónica Pérez-Rosas και Rada Mihalcea. *Inferring Social Media Users' Mental Health Status from Multimodal Information. Proceedings of The 12th Language Resources and Evaluation Conference*, σελίδες 6292–6299, 2020.

[69] Arman Cohan, Bart Desmet, Andrew Yates, Luca Soldaini, Sean MacAvaney και Nazli Goharian. *SMHD: a large-scale resource for exploring online language usage for multiple mental health conditions. arXiv preprint arXiv:1806.05258*, 2018.

[70] Jonathan Gratch, Ron Artstein, Gale M Lucas, Giota Stratou, Stefan Scherer, Angela Nazarian, Rachel Wood, Jill Boberg, David DeVault, Stacy Marsella και others. *The distress analysis interview corpus of human and computer interviews. LREC*, σελίδες 3123–3128, 2014.

[71] Minsu Park, David W McDonald και Meeyoung Cha. *Perception differences between the depressed and non-depressed users in twitter. Seventh International AAAI Conference on Weblogs and Social Media*, 2013.

[72] Danielle Mowery, Hilary Smith, Tyler Cheney, Greg Stoddard, Glen Coppersmith, Craig Bryan και Mike Conway. *Understanding Depressive Symptoms and Psychosocial Stressors on Twitter: A Corpus-Based Study. J Med Internet Res*, 19(2):e48, 2017.

[73] Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff και Siddharth Patwardhan. *OpinionFinder: A system for subjectivity analysis. Proceedings of HLT/EMNLP 2005 Interactive Demonstrations*, σελίδες 34–35, 2005.

[74] Shweta Yadav, Joy Prakash Sain, Amit Sheth, Asif Ekbal, Sriparna Saha και Pushpak Bhattacharyya. *Assessing the severity of health states based on social media posts. arXiv preprint arXiv:2009.09600*, 2020.

[75] Shweta Yadav, Jainish Chauhan, Joy Prakash Sain, Krishnaprasad Thirunarayan, Amit Sheth και Jeremiah Schumm. *Identifying Depressive Symptoms from Tweets: Figurative Language Enabled Multitask Learning Framework. arXiv preprint arXiv:2011.06149*, 2020.

[76] Jingren Zhang, Fang'ai Liu, Weizhi Xu, Hui Yu και others. *Feature Fusion Text Classification Model Combining CNN and BiGRU with Multi-Attention Mechanism. Future Internet*, 11(11):237, 2019.

[77] Silvio Amir, Mark Dredze και John W Ayers. *Mental health surveillance over social media with digital cohorts. Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology*, σελίδες 114–120, 2019.

[78] Sida I Wang και Christopher D Manning. *Baselines and bigrams: Simple, good sentiment and topic classification. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, σελίδες 90–94, 2012.

[79] James W Pennebaker, Ryan L Boyd, Kayla Jordan και Kate Blackburn. *The development and psychometric properties of LIWC2015.* Τεχνική Αναφορά με αριθμό, 2015.

[80] Jacopo Staiano και Marco Guerini. *Depechemood: a lexicon for emotion analysis from crowd-annotated news. arXiv preprint arXiv:1405.1605*, 2014.

[81] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho και Yoshua Bengio. *Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555*, 2014.

[82] Kai Sheng Tai, Richard Socher και Christopher D Manning. *Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075*, 2015.

[83] Yahui Chen. *Convolutional neural network for sentence classification.* Μεταπτυχιακή διπλωματική εργασία, University of Waterloo, 2015.

[84] Sven Buechel και Udo Hahn. *Emobank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, σελίδες 578–585, 2017.

[85] Seyed Mahdi Rezaeinia, Rouhollah Rahmani, Ali Ghodsi και Hadi Veisi. *Sentiment analysis based on improved pre-trained word embeddings. Expert Systems with Applications*, 117:139–147, 2019.

[86] Cristina Bosco, Viviana Patti και Andrea Bolioli. *Developing corpora for sentiment analysis: The case of irony and senti-tut. IEEE intelligent systems*, 28(2):55–63, 2013.

[87] Aniruddha Ghosh και Tony Veale. *Magnets for sarcasm: Making sarcasm detection timely, contextual and very personal. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, σελίδες 482–491, 2017.

[88] Hang Jiang, Xianzhe Zhang και Jinho D Choi. *Automatic Text-Based Personality Recognition on Monologues and Multiparty Dialogues Using Attentive Networks and Contextual Embeddings (Student Abstract). Proceedings of the AAAI Conference on Artificial Intelligence*, τόμος 34, σελίδες 13821–13822, 2020.

[89] *pipelines in huggingface.* `https://huggingface.co/docs/transformers/quicktour`.

[90] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga και Adam Lerer. *Automatic differentiation in pytorch.* 2017.

[91] Rui Zhang, Honglak Lee και Dragomir Radev. *Dependency sensitive convolutional neural networks for modeling sentences and documents. arXiv preprint arXiv:1611.02361*, 2016.

[92] Xiaobin Zhang, Fucai Chen και Ruiyang Huang. *A combination of RNN and CNN for attention-based relation classification. Procedia computer science*, 131:911–917, 2018.

[93] Katerina Margatina, Christos Baziotis και Alexandros Potamianos. *Attention-based conditioning methods for external knowledge integration. arXiv preprint arXiv:1906.03674*, 2019.

[94] Efthymios Georgiou, Charilaos Papaioannou και Alexandros Potamianos. *Deep Hierarchical Fusion with Application in Sentiment Analysis. INTERSPEECH*, σελίδες 1646–1650, 2019.

[95] Efthymios Georgiou, Georgios Paraskevopoulos και Alexandros Potamianos. *M3: MultiModal Masking applied to sentiment analysis. Proc. Interspeech 2021*, σελίδες 2876–2880, 2021.

[96] Ulrike Zetsche, Catherine D'Avanzato και Jutta Joormann. *Depression and rumination: Relation to components of inhibition. Cognition & emotion*, 26(4):758–767, 2012.

[97] Maev Conneely, Paul Higgs και Joanna Moncrieff. *Medicalising the moral: the case of depression as revealed in internet blogs. Social Theory & Health*, 19(4):380–398, 2021.

[98] Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade και Sujith Ravi. *GoEmotions: A dataset of fine-grained emotions. arXiv preprint arXiv:2005.00547*, 2020.

[99] Paul Ekman. *Are there basic emotions?* 1992.

[100] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner και Sameer Singh. *Universal adversarial triggers for attacking and analyzing NLP. arXiv preprint arXiv:1908.07125*, 2019.