ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Βελτιστοποίηση απόδοσης συστήματος προτάσεων βασισμένη σε τεχνικές εξόρυξης πληροφορίας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΝΔΡΕΑ ΧΡΗΣΤΟΥ

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούνιος 2019

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Βελτιστοποίηση απόδοσης συστήματος προτάσεων βασισμένη σε τεχνικές εξόρυξης πληροφορίας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΝΔΡΕΑ ΧΡΗΣΤΟΥ

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26η Ιουνίου 2019.

.....	.....	.....
Θεοδώρα Βαρβαρίγου Καθηγήτρια Ε.Μ.Π.	Εμμανουήλ Βαρβαρίγος Καθηγητής Ε.Μ.Π.	Συμεών Παπαβασιλείου Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2019

.....
Ανδρέας Χρήστου
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

Περίληψη

Στην καθημερινή τους ζωή, οι άνθρωποι απαιτείται πολύ συχνά να παίρνουν αποφάσεις, χωρίς να έχουν πλήρη επίγνωση των εναλλακτικών τους επιλογών. Βασίζονται σε προτάσεις άλλων ανθρώπων είτε από στόμα σε στόμα, είτε διαβάζοντας κριτικές σε περιοδικά και ιστοσελίδες. Η ραγδαία αύξηση του όγκου δεδομένων έχει καταστήσει αναγκαία τη ανάπτυξη συστημάτων, τα οποία θα λαμβάνουν υπόψιν όλη τη γνώση που μπορούν να εξάγουν από τα διαθέσιμα δεδομένα. Τα συστήματα αυτά εξετάζουν όλες τις εναλλακτικές επιλογές που μπορούν να ικανοποιήσουν το χρήστη, και του προτείνουν τις καλύτερες, με σκοπό να πάρει την πιο σωστή απόφαση.

Τα τελευταία χρόνια, ο κλάδος των συστημάτων προτάσεων έχει αναπτυχθεί ραγδαία. Πολλά συστήματα έχουν αναπτυχθεί με σκοπό να προτείνουν στους χρήστες τις καλύτερες επιλογές, λαμβάνοντας υπόψιν τα διαθέσιμα δεδομένα. Κάποια από αυτά βασίζονται στις βαθμολογίες άλλων χρηστών για κάποια αντικείμενα ενώ κάποια άλλα συστήματα χρησιμοποιούν μεθόδους, ώστε να μπορούν να εξάγουν χρήσιμα συμπεράσματα αναλύοντας κείμενο.

Ένα εργαλείο με εξαιρετική ικανότητα ανάλυσης κειμένου είναι και η Elasticsearch. Η Elasticsearch είναι μια πραγματικού χρόνου (real-time) κατανεμημένη μηχανή αναζήτησης και επιτρέπει στο χρήστη να ψάχνει σε τεράστιο όγκο δεδομένων με μεγάλη ταχύτητα. Παρά την ικανότητα της να αναλύει κείμενο αποτελεσματικά, η Elasticsearch δεν παρουσιάζει την ίδια ικανότητα στην ανάλυση αριθμητικών τιμών, παρά την ανάπτυξη εργαλείων που τη βοηθούν σε αυτό τον τομέα.

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η εκμετάλλευση των εργαλείων που προσφέρει η Elasticsearch και σε συνδυασμό με κάποια άλλα εργαλεία, η ανάπτυξη ενός αξιόπιστου συστήματος προτάσεων. Η διπλωματική εργασία επικεντρώνεται στην ανάπτυξη ενός συστήματος προτάσεων και στη συνέχεια τα αποτελέσματα συγκρίνονται με αυτά της Elasticsearch. Το σύστημα αρχικά λαμβάνει υπόψιν τις παραμέτρους αναζήτησης του χρήστη και στη συνέχεια του προτείνει ταξινομημένες τις καλύτερες επιλογές. Η βάση για την εκπόνηση αυτής της διπλωματική εργασίας αποτελεί το σύστημα που δημιουργήθηκε στα πλαίσια του DITAS project.

Λέξεις κλειδιά

Συστήματα προτάσεων, εξόρυξη πληροφορίας, βελτιστοποίηση, Elasticsearch, Java, API

Abstract

In everyday life, it is often necessary to make choices without sufficient personal experience of the alternatives. People rely on recommendations from other people either by word of mouth, or reading reviews printed in journals or written on websites. The dramatic rise of data volume has made the development of systems, which consider all the knowledge they can export from the available data, necessary. These systems, take into account all the alternatives which could satisfy the user, and recommends him the best of them, helping him to make the best choice.

In the last years, recommendation systems have been developed dramatically. Many systems have been produced in order to recommend to users the best choice, considering all the available data. Some of them, rely on users' ratings for some objects/products, and some others use methods, which help them draw useful conclusions, by analyzing text.

A tool with a great ability of text analyzing is Elasticsearch. Elasticsearch is a real-time distributed search engine, allowing the user to search in a great amount of data extraordinarily fast. In spite of the fact that Elasticsearch is extremely capable in analyzing text efficiently, it does not appear to have the same ability in analyzing arithmetic values, despite that many tools have been developed to help it.

The goal of this diploma thesis is the exploitation of the tools that Elasticsearch offers and by combining them with other tools, to produce a reliable recommendation system. This diploma thesis focuses on the production of a recommendation system, and then, the results are compared to those of Elasticsearch. Initially, the recommendation system takes into account the user's search parameters and then it recommends him the best choices sorted. The system which was developed for DITAS project is considered as the basis of this diploma thesis.

Keywords

Recommender systems, Recommendation systems, data mining, information retrieval, optimization, Elasticsearch, Java, API

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε κατά τη διάρκεια του ακαδημαϊκού έτους 2018-2019 στον τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής του Εθνικού Μετσόβιου Πολυτεχνείου και σηματοδοτεί το τέλος των προπτυχιακών μου σπουδών.

Αρχικά θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια μου κα. Θεοδώρα Βαρβαρίγου για την ευκαιρία που μου έδωσε να εκπονήσω τη διπλωματική μου εργασία στο εργαστήριο της και να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα.

Ιδιαίτερες ευχαριστίες θα ήθελα να απονείμω στον Υποψήφιο Διδάκτορα Ε.Μ.Π Αλέξανδρο Ψύχα για την αμέριστη βοήθεια που μου παρείχε καθ' όλη τη διάρκεια της εκπόνησης της διπλωματικής εργασίας. Η ανά πάσα στιγμή διαθεσιμότητα του για βοήθεια και συμβουλές τους τελευταίους μήνες υπήρξε καθοριστική για την ολοκλήρωση της εργασίας, αφού χωρίς αυτά το έργο μου θα ήταν σαφώς δυσκολότερο.

Επίσης, θα ήθελα να ευχαριστήσω τους γονείς μου, Έλενα και Χρήστο, καθώς και τα αδέρφια μου, Γιάννη και Ρεβέκκα, για την αμέριστη συμπαράσταση τους κατά τη διάρκεια των σπουδών μου, αλλά και γενικότερα καθ' όλη τη διάρκεια της πορείας μου μέχρι σήμερα.

Τέλος, ένα μεγάλο ευχαριστώ στους συμφοιτητές μου, με τους οποίους μοιραστήκαμε τόσα πολλά τα τελευταία 5 χρόνια, καθώς και στους φίλους μου, οι οποίοι ήταν και είναι πάντα κοντά μου όταν τους χρειάζομαι.

Πίνακας Περιεχομένων

Πίνακας Διαγραμμάτων

Πίνακας Πινάκων

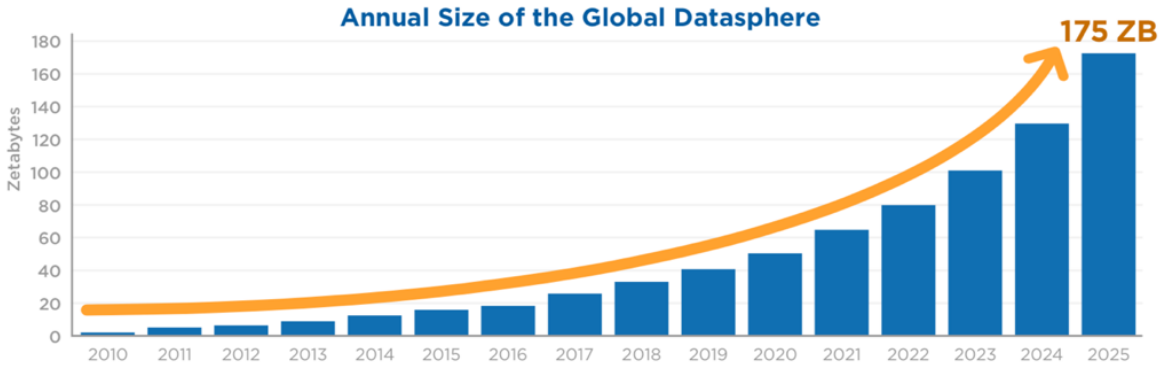
1

Εισαγωγή

1.1 Αντικείμενο της εργασίας

Στην καθημερινή τους ζωή, οι άνθρωποι απαιτείται πολύ συχνά να παίρνουν αποφάσεις, χωρίς να έχουν πλήρη επίγνωση των εναλλακτικών τους επιλογών. Βασίζονται σε προτάσεις άλλων ανθρώπων είτε από στόμα σε στόμα, είτε διαβάζοντας κριτικές σε περιοδικά και ιστοσελίδες.

Είναι γεγονός πως τα τελευταία χρόνια ο όγκος δεδομένων που παράγεται είναι θεαματικά μεγάλος. Παρόλα αυτά, η παραγωγή δεδομένων δεν σταματάει αλλά αντιθέτως, αναπτύσσεται με ακόμα μεγαλύτερους ρυθμούς. Στο πιο κάτω διάγραμμα φαίνεται ξεκάθαρα η τάση που υπάρχει στην αύξηση του όγκου των δεδομένων [1].



Διάγραμμα 1: Η πρόβλεψη της Seagate όσον αφορά την ετήσια παραγωγή δεδομένων [1]

Προβλέπεται ότι το 2025, το μέγεθος των δεδομένων που θα παραχθεί θα είναι περίπου 6 φορές μεγαλύτερο από αυτό που παράχθηκε το 2018, φτάνοντας το τρομακτικό νούμερο των 175 zettabytes (1 τρισεκατομμύριο gigabytes).

Το τεράστιο μέγεθος δεδομένων που υπάρχει διαθέσιμο και επομένως η πληροφορία και οι υπηρεσίες, οδηγεί πολλές φορές τους χρήστες σε λάθος αποφάσεις, αφού η τεράστια ποικιλία προϊόντων και υπηρεσιών που προσφέρονται αντί να λειτουργεί ως προνόμιο για το χρήστη, τον οδηγεί σε αγορές που δεν βελτιώνουν το επίπεδο ζωής του. Είναι γεγονός πως το να υπάρχουν διαθέσιμες επιλογές είναι πολύ καλό, όμως το να υπάρχουν υπερβολικά πολλές επιλογές δεν είναι κατ' ανάγκη καλύτερο. Ο τεράστιος όγκος δεδομένων κάνει δυσκολότερη την αναζήτηση και κατανόηση τους, βάζοντας ακόμα ένα εμπόδιο στο χρήστη, ώστε να λάβει τη σωστή απόφαση.

Η ραγδαία αυτή αύξηση του όγκου των δεδομένων που είναι διαθέσιμη στο διαδίκτυο, έχει καταστήσει αναγκαία τη ανάπτυξη συστημάτων, τα οποία θα λαμβάνουν υπόψιν όλη τη γνώση που μπορούν να εξάγουν από τα διαθέσιμα δεδομένα. Τα συστήματα αυτά εξετάζουν όλες τις εναλλακτικές επιλογές που μπορούν να προταθούν στο χρήστη, και του προτείνουν τις καλύτερες, με σκοπό να πάρει την πιο σωστή απόφαση.

Τα τελευταία χρόνια, ο κλάδος των συστημάτων προτάσεων έχει αναπτυχθεί με γρήγορους ρυθμούς. Πολλά συστήματα έχουν αναπτυχθεί με σκοπό να προτείνουν στους χρήστες τις καλύτερες επιλογές, λαμβάνοντας υπόψιν τα διαθέσιμα δεδομένα. Τα συστήματα προτάσεων έχει αποδειχτεί πως αποτελούν ένα χρήσιμο εργαλείο αντιμετώπισης της υπερφόρτωσης της πληροφορίας, που οδηγεί τους χρήστες σε λανθασμένες αποφάσεις.

1.2 Σκοπός Εργασίας

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η βελτιστοποίηση ενός συστήματος προτάσεων, βασισμένη σε τεχνικές εξόρυξης πληροφορίας. Για την υλοποίηση του συστήματος προτάσεων, χρησιμοποιήθηκε η κατανεμημένη μηχανή αναζήτησης Elasticsearch, η οποία είναι πολύ γρήγορη στην αναζήτηση των δεδομένων. Επιπλέον, είναι ένα εξαιρετικό εργαλείο στην ανάλυση κειμένου, παρόλα αυτά δεν παρουσιάζει την ίδια ικανότητα στην ανάλυση αριθμητικών τιμών, παρά την ανάπτυξη εργαλείων που τη βοηθούν σε αυτό τον τομέα. Στο σύστημα που υλοποιήθηκε, η Elasticsearch χρησιμοποιήθηκε στην ανάλυση κειμένου, όπου χρειάστηκε, όπως επίσης και στην εκτέλεση ερωτημάτων ώστε να δίνει γρήγορα αποτελέσματα, τα οποία στη συνέχεια με την κατάλληλη επεξεργασία δίνουν στον

χρήστη κάποιες προτάσεις ταξινομημένες με βάση την καλύτερη. Επίσης έγινε προσπάθεια ώστε να υλοποιηθεί το σύστημα προτάσεων το οποίο θα χρησιμοποιεί αποκλειστικά την Elasticsearch για να δίνει τις προτάσεις στο χρήστη. Αυτό το σύστημα υλοποιήθηκε με σκοπό να συγκριθούν τα δύο συστήματα μεταξύ τους, και να μελετηθεί αν ο αλγόριθμος που έχει υλοποιηθεί για το σύστημα προτάσεων έχει επιτύχει βελτίωση στις προτάσεις που γίνονται στους χρήστες, σε σχέση με τις προτάσεις που κάνει η αποκλειστική χρήση της Elasticsearch. Χρησιμοποιήθηκε ένα εργαλείο της Elasticsearch, τα Decay Score Functions, το οποίο υπολογίζει σκορ με βάση το πόσο μακριά βρίσκονται οι αριθμητικές τιμές μεταξύ τους. Έγινε προσπάθεια ώστε να παραμετροποιηθεί όσο το δυνατό κατάλληλα η Elasticsearch, με σκοπό να παραγάγει καλύτερα αποτελέσματα, ώστε η σύγκριση που θα γίνει να οδηγεί σε αξιόπιστα συμπεράσματα. Τέλος, τα αποτελέσματα του συστήματος προτάσεων που αναπτύχθηκε καθώς και του συστήματος που χρησιμοποιεί αποκλειστικά την Elasticsearch, συγκρίνονται μεταξύ τους και εξάγονται κάποια συμπεράσματα για την απόδοση των δύο συστημάτων.

1.3 Οργάνωση

Η διπλωματική εργασία αποτελείται από 6 κεφάλαια. Στο παρόν κεφάλαιο έχει γίνει μια εισαγωγή στους λόγους για τους οποίους είναι αναγκαία η ύπαρξη συστημάτων προτάσεων, ενώ έχει παρουσιαστεί και ο σκοπός της εργασίας. Στο επόμενο κεφάλαιο γίνεται μια ανασκόπηση κάποιων σχετικών θεμάτων, στα οποία στηρίζεται η εργασία, ενώ στο 3ο κεφάλαιο παρουσιάζονται τα εργαλεία και οι αρχιτεκτονικές που έχουν χρησιμοποιηθεί για την ανάπτυξη του συστήματος προτάσεων. Στο 4ο κεφάλαιο, περιγράφεται η υλοποίηση του συστήματος προτάσεων με τη χρήση των εργαλείων που έχουν αναλυθεί στο κεφάλαιο 3, καθώς και του θεωρητικού υπόβαθρου του 2ου κεφαλαίου. Στο 5ο κεφάλαιο γίνεται μια συνοπτική παρουσίαση και σύγκριση των αποτελεσμάτων που έχουν παραχθεί με τη χρήση του συστήματος προτάσεων και αυτών που έχουν παραχθεί με τη χρήση της Elasticsearch. Τέλος, η εργασία ολοκληρώνεται στο κεφάλαιο 6 με την εξαγωγή γενικών συμπερασμάτων και προτάσεων για μελλοντικές προεκτάσεις.

2

Θεωρητικό Υπόβαθρο

2.1 Συστήματα προτάσεων

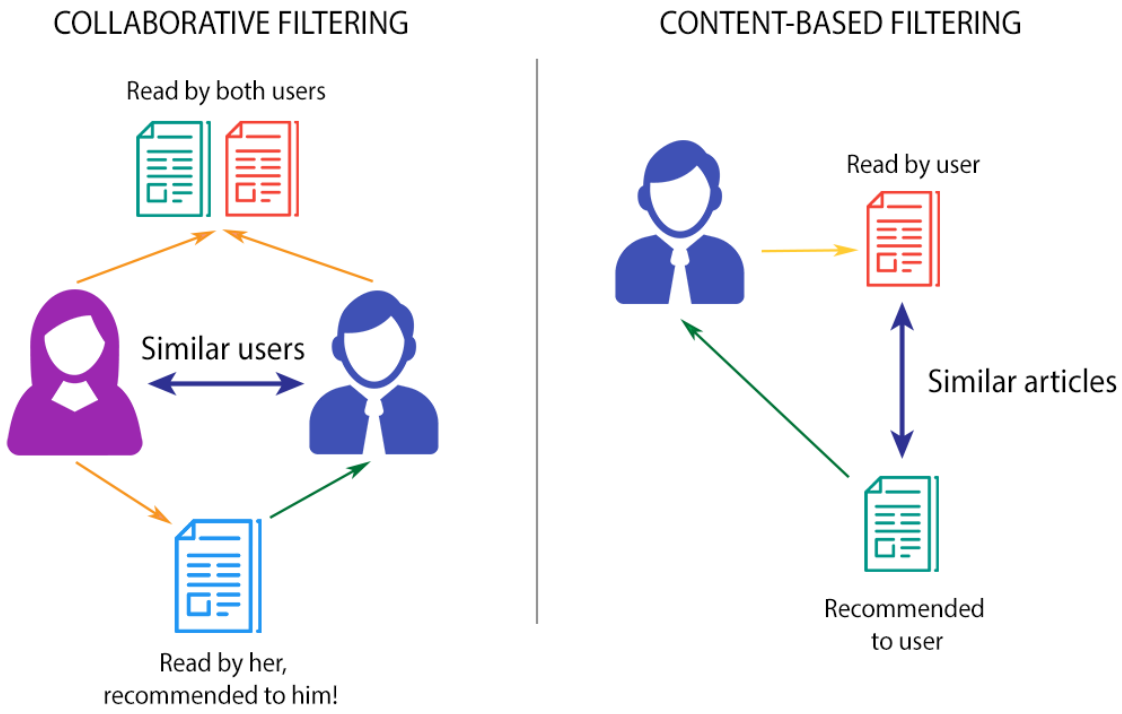
Σύστημα προτάσεων, Recommendation System ή Recommender System ονομάζεται ένα εργαλείο λογισμικού το οποίο έχει σκοπό να προσφέρει προτάσεις για κάποια αντικείμενα τα οποία θα χρησιμοποιηθούν από ένα χρήστη. Οι προτάσεις αυτές σχετίζονται άμεσα με αποφάσεις που καλείται να πάρει ο χρήστης, όπως ποιο αντικείμενο πρόκειται να αγοράσει, ποιο τραγούδι θα ακούσει, ή ποια είδηση θα διαβάσει. Τα προσωποποιημένα συστήματα προτάσεων, προσφέρουν στους χρήστες ταξινομημένες λίστες αντικειμένων, με βάση το ποιο αντικείμενο ταιριάζει περισσότερο στις προτιμήσεις του. Το σύστημα προτάσεων προσπαθεί να προβλέψει ποιο προϊόν ή ποια υπηρεσία είναι καταλληλότερη ώστε να προταθεί στον χρήστη, λαμβάνοντας υπόψιν τις προτιμήσεις και τους περιορισμούς που αυτός έχει θέσει.

Η ανάπτυξη συστημάτων προτάσεων ξεκίνησε από μια πολύ απλή καθημερινή παρατήρηση· συχνά οι άνθρωποι, βασίζονται σε προτάσεις που έχουν γίνει από άλλους, ώστε να πάρουν καθημερινές αποφάσεις [2]. Για παράδειγμα, είναι κοινό στην καθημερινή ζωή των ανθρώπων, να βασίζονται στη γνώμη κάποιων άλλων, ώστε να επιλέξουν ποιο βιβλίο θα διαβάσουν ή ποια ταινία θα δουν.

Προσπαθώντας να μιμηθούν αυτή τη συμπεριφορά, αρχικά, τα συστήματα προτάσεων υλοποίησαν αλγόριθμους με σκοπό να δημιουργήσουν προτάσεις για κάποιο χρήστη,

οι οποίες προέρχονταν από μια κοινότητα άλλων χρηστών. Οι προτάσεις αυτές αφορούσαν προτάσεις αντικείμενων τα οποία παρόμοιοι χρήστες, δηλαδή χρήστες με παρόμοιες προτιμήσεις, είχαν αγοράσει. Αυτή η προσέγγιση ονομάζεται Collaborative-Filtering. Αν ένας χρήστης έχει συμφωνήσει στο παρελθόν με κάποιους άλλους χρήστες όσον αφορά κάποια αντικείμενα, τότε τα αντικείμενα που θα προταθούν σε αυτό τον χρήστη θα προέρχονται από αυτούς τους όμοιους χρήστες με τους οποίους έχει κοινές προτιμήσεις. Η άλλη πιο διαδεδομένη προσέγγιση για τα συστήματα προτάσεων είναι το content-based filtering, όπου χρησιμοποιούνται τα χαρακτηριστικά του κάθε αντικειμένου, ώστε να υπολογιστεί η ομοιότητα και στη συνέχεια να προταθούν στο χρήστη με βάση προηγούμενες αγορές.

Οι δύο επικρατέστερες προσεγγίσεις στη σχεδίαση συστημάτων προτάσεων είναι αυτές που έχουν αναφερθεί πιο πάνω. Στο πιο κάτω διάγραμμα γίνεται ξεκάθαρο το πως λειτουργεί η κάθε προσέγγιση.



Διάγραμμα 2: Σύγκριση λειτουργίας του collaborative filtering και του content-based filtering

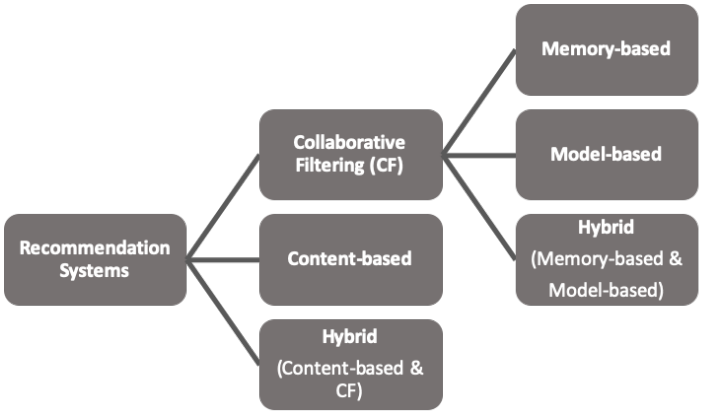
Τα συστήματα προτάσεων παίζουν σημαντικό ρόλο στις δημοφιλέστερες ιστοσελίδες όπως η Amazon, το YouTube, η Netflix. Επιπλέον, πολλές εταιρείες υλοποιούν τα δικά τους συστήματα προτάσεων τα οποία παρέχονται ως υπηρεσία στους συνδρομητές τους. Για παράδειγμα, τον Οκτώβριο του 2006, η Netflix, η διαδικτυακή υπηρεσία παροχής video streaming, είχε προκηρύξει ένα ανοιχτό διαγωνισμό για την ανάδειξη του καλύτερου συνεργατικού αλγόριθμου φιλτραρίσματος της πρόβλεψης των αξιολογήσεων των χρηστών για ταινίες [3]. Τα δεδομένα τα οποία έδωσε στη δημοσιότητα η εταιρεία, ήταν μόνο προηγούμενες αξιολογήσεις χρηστών για τις

ταινίες, χωρίς καμία άλλη πληροφορία για τους χρήστες ή τις ταινίες. Το βραβείο θα κέρδιζε η ομάδα η οποία θα επιτύγχανε 10% μεγαλύτερη ακρίβεια στις προβλέψεις από τον αλγόριθμο που χρησιμοποιούσε τότε η εταιρεία.

Η ανάπτυξη συστημάτων προτάσεων από τις μεγάλες εταιρείες είναι αναμενόμενο πως βοηθούν τους συνδρομητές τους να έχουν πιο προσωποποιημένες προτάσεις, που εν τέλει τους αφήνουν ευχαριστημένους με την υπηρεσία. Σύμφωνα με στοιχεία που παρουσιάστηκαν από την Amazon, περίπου το 35% των προϊόντων που αγοράζουν οι χρήστες, προέρχονται από τις προτάσεις που τους κάνει το σύστημα προτάσεων της εταιρείας, το οποίο χρησιμοποιεί δεδομένα όπως το ιστορικό αγορών του χρήστη, τα προϊόντα που έχει βάλει στο καλάθι του καθώς και αυτά που έχει βαθμολογήσει στο παρελθόν. Επιπλέον λαμβάνει υπόψιν του τι έχουν αγοράσει χρήστες με παρόμοιες προτιμήσεις [4].

Ακόμα μια εταιρεία με εξαιρετικά αποδοτικό σύστημα προτάσεων είναι η Netflix η οποία αναφέρθηκε και πιο πάνω. Περίπου το 80% των ταινιών, σειρών και βίντεο που παρακολουθούν οι χρήστες βασίζονται στις προτάσεις που κάνει ο αλγόριθμος και μόλις το 20% σε αυτό που έχει αναζητήσει ο χρήστης [5]. Αυτό σημαίνει πως είναι εξαιρετικά πιθανόν να εμπιστευτεί κάποιος τις προτάσεις που του γίνονται και πως οι χρήστες που το κάνουν είναι πολύ ευχαριστημένοι με αυτό. Η αποτελεσματικότητα του συστήματος προτάσεων της Netflix δεν ωφελεί μόνο τους χρήστες, αλλά και την ίδια την εταιρεία. Υπολογίζεται πως η εταιρεία γλυτώνει περίπου 1δις δολάρια το χρόνο από την ικανοποίηση των πελατών της, αφού λίγοι είναι αυτοί που δεν μένουν ευχαριστημένοι από τις υπηρεσίες που τους προσφέρονται και επιλέγουν να διακόψουν τη συνδρομή τους.

Στη συνέχεια παρουσιάζονται στο διάγραμμα και ακολούθως αναλύονται οι επικρατέστερες προσεγγίσεις στην υλοποίηση των συστημάτων προτάσεων.



Διάγραμμα 3: Ιεραρχία των συστημάτων προτάσεων

2.1.1 Collaborative Filtering

Η προσέγγιση του collaborative filtering θεωρείται ως η πιο διαδεδομένη και ευρέως χρησιμοποιούμενη στα συστήματα προτάσεων. Θεωρείται ως η απλούστερη προσέγγιση στην υλοποίηση συστημάτων προτάσεων και προτείνει στον χρήστη αντικείμενα τα οποία άλλοι χρήστες με παρόμοιες προτιμήσεις έχουν αγοράσει στο παρελθόν. Βασίζεται στη θεωρία «χρήστες που έχουν συμφωνήσει στο παρελθόν, θα συμφωνήσουν και στο μέλλον». Η ομοιότητα στις προτιμήσεις δύο χρηστών υπολογίζεται με βάση την ομοιότητα των βαθμολογιών τους σε συγκεκριμένα αντικείμενα. Η τεχνική αυτή αναφέρεται και ως «συσχέτιση χρήστη με χρήστη» [6].

Ένα από τα πιο διάσημα παραδείγματα χρήσης της συγκεκριμένης προσέγγισης θεωρείται ο Item-item collaborative filtering, ο οποίος αναπτύχθηκε από την Amazon και βασίζεται στη θεωρία ότι όποιος αγοράζει το Χ προϊόν, αγοράζει και το Υ.

Η προσέγγιση του collaborative filtering μπορεί να διακριθεί σε τρεις κατηγορίες [2]· σε memory-based και model-based και συνδυασμός των δύο [7]. Οι memory-based αλγόριθμοι πραγματοποιούν προβλέψεις οι οποίες βασίζονται στα δεδομένα που υπάρχουν αποθηκευμένα στη βάση (χρήστες, αντικείμενα, βαθμολογίες). Είναι εύκολοι στην υλοποίηση, παρόλα αυτά όταν τα δεδομένα είναι αραιά, έχουν μειωμένη απόδοση. Από την άλλη, οι model-based αλγόριθμοι εξάγουν ένα μοντέλο χρησιμοποιώντας κάποια από τα δεδομένα, το οποίο χρησιμοποιείται στη συνέχεια για την πρόβλεψη. Τα συγκεκριμένα μοντέλα αναπτύσσονται μέσω μεθόδων μηχανικής μάθησης και εξόρυξης δεδομένων, με σκοπό να εντοπιστούν κάποια πρότυπα, τα οποία θα χρησιμοποιηθούν στη συνέχεια για πρόβλεψη. Η συγκεκριμένη μέθοδος είναι ταχύτερη αφού δε χρησιμοποιεί όλα τα δεδομένα για να παραγάγει πρόβλεψη αλλά μόνο ένα μέρος τους για να δημιουργήσει το μοντέλο. Αυτό μπορεί να θεωρηθεί και ως μειονέκτημα της μεθόδου, αφού είναι δυνατόν τα αποτελέσματα να έχουν μικρότερη ακρίβεια, λόγω της χρήσης του μοντέλου και όχι όλων των δεδομένων.

2.1.2 Πλεονεκτήματα και Μειονεκτήματα του Collaborative Filtering

Το βασικότερο πλεονέκτημα του collaborative filtering έχει να κάνει με το γεγονός ότι αυτή η προσέγγιση δε βασίζεται σε ανάλυση περιεχομένου, και επομένως είναι εφικτή η ακριβέστερη πρόταση περίπλοκων αντικειμένων, όπως μια ταινία, χωρίς να χρειαστεί το σύστημα να καταλαβαίνει το αντικείμενο καθαυτό. Επιπλέον, όπως έχει προαναφερθεί, είναι η απλούστερη προσέγγιση στην υλοποίηση συστημάτων προτάσεων, και επομένως είναι η πιο διαδεδομένη.

Παρά τα πλεονεκτήματα που έχουν αναφερθεί πιο πάνω, το collaborative filtering έχει αρκετά μειονεκτήματα, με το σημαντικότερο αυτών να είναι το cold-start problem [8,9]. Αυτό συμβαίνει όταν είναι αδύνατο να παραχθούν αξιόπιστες προβλέψεις, λόγω έλλειψης δεδομένων. Το φαινόμενο αυτό μπορεί να διαχωριστεί σε 3 κατηγορίες· νέο αντικείμενο, νέα κοινότητα, νέος χρήστης.

Το λιγότερο σημαντικό πρόβλημα, είναι αυτό του νέου αντικειμένου [9], το οποίο οφείλεται στο γεγονός ότι τα νέα αντικείμενα που εισέρχονται στο σύστημα προτάσεων, δεν έχουν βαθμολογίες από τους χρήστες με αποτέλεσμα να είναι σχεδόν αδύνατον να προταθούν. Αυτό οδηγεί ένα αντικείμενο στο να περνάει απαρατήρητο από την κοινότητα.

Το πρόβλημα της νέας κοινότητας [9] αναφέρεται στη δυσκολία του συστήματος, όταν ξεκινάει τη λειτουργία του, να βρει ικανοποιητικό αριθμό δεδομένων, ώστε να παραγάγει προβλέψεις. Αυτό οφείλεται στη δυσκολία εύρεσης νέων χρηστών, αφού όταν δεν υπάρχουν χρήστες και επομένως βαθμολογίες για τα αντικείμενα, δεν παράγονται αξιόπιστες προβλέψεις και επομένως είναι εξαιρετικά δύσκολο να εμπιστευτούν οι χρήστες το νέο σύστημα.

Η τελευταία και ίσως σημαντικότερη κατηγορία cold-start, είναι αυτή του νέου χρήστη [9]. Είναι από τα μεγαλύτερα προβλήματα που έχουν να αντιμετωπίσουν τα συστήματα προτάσεων. Όταν ένας νέος χρήστης εισέρχεται στο σύστημα, το σύστημα δεν έχει δεδομένα για αυτόν. Οπότε είναι αδύνατο να παραχθούν αξιόπιστες προτάσεις που να ικανοποιούν τον χρήστη, με αποτέλεσμα ο χρήστης να θεωρεί πως το σύστημα δεν του προσφέρει τις υπηρεσίες που περίμενε, και να σταματήσει να το χρησιμοποιεί.

Τέλος αξίζει να αναφερθούν άλλα δύο προβλήματα που παρουσιάζει το collaborative filtering. Ένα από αυτά είναι η μεγάλη ποσότητα ενέργειας που χρειάζεται ένα σύστημα ώστε να προτείνει ένα αντικείμενο, λόγω του μεγάλου όγκου δεδομένων, όταν υπάρχουν στη βάση τεράστιος αριθμός χρηστών και βαθμολογιών (scalability). Το άλλο πρόβλημα είναι το sparsity, το οποίο αναφέρεται στο γεγονός πως υπάρχει τεράστιος αριθμός αντικειμένων, με αποτέλεσμα, ακόμη και οι πιο ενεργοί χρήστες θα έχουν βαθμολογήσει ένα μικρό αριθμό αυτών. Επομένως ακόμα και τα πιο δημοφιλή αντικείμενα έχουν λίγες βαθμολογίες.

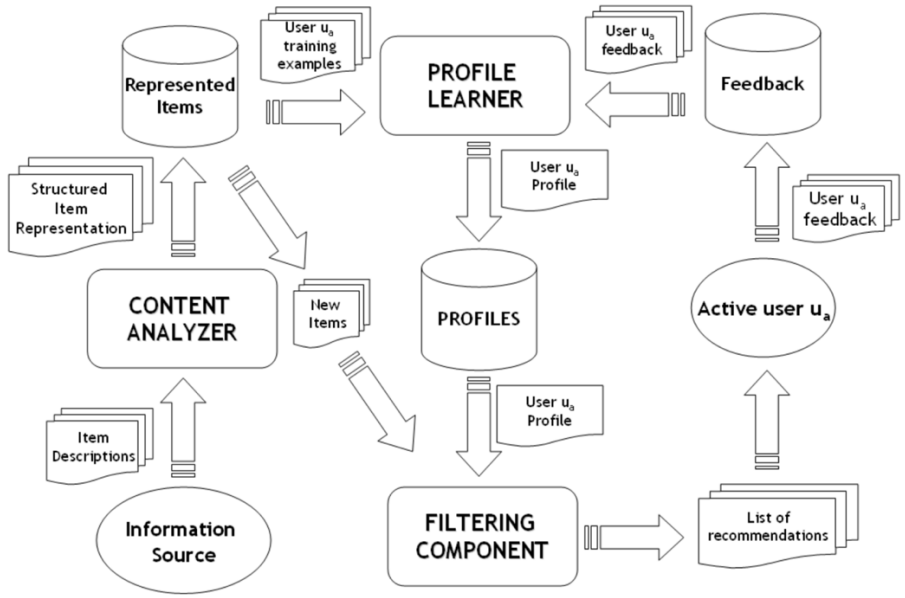
2.1.3 Content-based Filtering

Με αυτή την προσέγγιση το σύστημα μαθαίνει να προτείνει αντικείμενα τα οποία είναι όμοια με αυτά που έχουν ικανοποιήσει τον χρήστη στο παρελθόν. Η ομοιότητα των

αντικειμένων υπολογίζεται με βάση κάποια χαρακτηριστικά τα οποία σχετίζονται με αυτά των προς σύγκριση αντικειμένων.

Τα συστήματα που υλοποιούν content-based filtering αναλύουν ένα σύνολο κειμένων και τις περιγραφές των αντικειμένων που προηγουμένως έχουν βαθμολογηθεί από τον χρήστη, και δημιουργούν ένα προφίλ για τον χρήστη το οποίο περιλαμβάνει τις προτιμήσεις του με βάση τα χαρακτηριστικά των αντικειμένων που έχει βαθμολογήσει [10]. Το προφίλ αυτό είναι μια δομημένη αναπαράσταση των προτιμήσεων των χρηστών, το οποίο λαμβάνεται υπόψιν με σκοπό να προταθούν στον χρήστη νέα αντικείμενα, συγκρίνοντας τα χαρακτηριστικά του χρήστη με τα χαρακτηριστικά του κάθε αντικειμένου.

Τα συστήματα προτάσεων που υλοποιούνται με content-based filtering χρειάζονται κάποιες τεχνικές για να παρουσιάσουν τα αντικείμενα και να δημιουργήσουν το προφίλ του χρήστη και στη συνέχεια να συγκρίνουν το προφίλ με την αναπαράσταση των αντικειμένων. Αρχικά, αν η πληροφορία δεν είναι κείμενο, απαιτείται μια διαδικασία, ώστε να εξαχθεί η πληροφορία σε μορφή κατάλληλη ώστε να προχωρήσει η διαδικασία. Στη συνέχεια, το σύστημα παίρνει τα δεδομένα και οι προτιμήσεις του χρήστη και προσπαθεί να τα γενικοποιήσει ώστε να παραχθεί το προφίλ του. Συνήθως αυτή η γενικοποίηση γίνεται με μεθόδους μηχανικής μάθηση, χρησιμοποιώντας προηγούμενη αλληλεπίδραση του χρήστη με το σύστημα, πχ like/dislike, σχόλια, βαθμολόγηση. Τέλος, το προφίλ του χρήστη που έχει δημιουργηθεί, χρησιμοποιείται ώστε να συγκριθεί με την αναπαράσταση των αντικειμένων και να παραχθούν οι προτάσεις που θα παρουσιαστούν στον χρήστη. Η παρουσίαση αυτή γίνεται είτε προτείνοντας συγκεκριμένο αντικείμενο, είτε προτείνοντας μια λίστα αντικειμένων, ταξινομημένα με βάση κάποιες μετρικές ομοιότητας.



Διάγραμμα 4: Διαδικασία λειτουργίας του content-based filtering

Πιο πάνω παρουσιάζεται ένα απλό σχεδιάγραμμα που δείχνει τη διαδικασία που μόλις περιεγράφηκε.

Το αποτέλεσμα αυτής της μεθόδου δείχνει το επίπεδο ενδιαφέροντος του χρήστη για αυτό το αντικείμενο. Αν το προφίλ του χρήστη που έχει δημιουργηθεί αντικατοπτρίζει τα ενδιαφέροντα και τα χαρακτηριστικά του χρήστη, τότε το σύστημα θα είναι αποτελεσματικό, προτείνοντας στον χρήστη αντικείμενα τα οποία είναι πιθανότερο να τον ενδιαφέρουν.

2.1.4 Πλεονεκτήματα και Μειονεκτήματα του Content-based Filtering

Ένα από τα πλεονεκτήματα του content-based filtering είναι ανεξαρτησία του χρήστη. Η συγκεκριμένη μέθοδος υλοποίησης συστήματος προτάσεων, λαμβάνει υπόψιν μόνο τις βαθμολογίες για αντικείμενα που έχουν γίνει από τον συγκεκριμένο χρήστη με σκοπό τη δημιουργία του προφίλ του. Επομένως δε χρειάζεται να υπάρχει μεγάλη κοινότητα χρηστών, ώστε να συγκριθεί ο ενεργός χρήστης με τους υπόλοιπους και να βρεθούν ομοιότητες με άλλους χρήστες. Επιπλέον, τα συστήματα αυτά, έχουν τη δυνατότητα να προτείνουν νέα αντικείμενα που εισέρχονται στο σύστημα, τα οποία δεν έχουν βαθμολογηθεί από κανένα χρήστη. Αυτό, είναι συνέπεια της σύγκρισης του αντικειμένου με το προφίλ του χρήστη και όχι λαμβάνοντας υπόψιν τις βαθμολογίες των υπόλοιπων χρηστών όπως συμβαίνει στα συστήματα προτάσεων που έχουν ως βάση το collaborative filtering.

Παρά τα πλεονεκτήματα που παρουσιάζουν αυτά τα συστήματα προτάσεων, παρουσιάζουν και αρκετά μειονεκτήματα. Ένα από αυτά είναι η πιθανή περιορισμένη πληροφορία που περιέχουν κάποια αντικείμενα. Εφόσον τα αντικείμενα αυτά δεν παρέχουν αρκετές πληροφορίες, δεν είναι δυνατόν να εξαχθούν συμπεράσματα και επομένως δεν μπορούν να προταθούν στον χρήστη. Άλλο ένα μειονέκτημα που παρουσιάζεται είναι όταν ένας νέος χρήστης εισέρχεται στο σύστημα, είναι αδύνατον να υπάρχουν αξιόπιστες προτάσεις, αφού για να δημιουργηθεί το προφίλ του χρήστη απαιτείται πρώτα η αλληλεπίδραση του (likes/dislikes, comments, ratings) με τα αντικείμενα. Από τη στιγμή που δεν έχει βαθμολογήσει κάποιο αντικείμενο, ή έχει βαθμολογήσει λίγα, είναι δύσκολο να κατανοήσει το σύστημα τις προτιμήσεις του χρήστη με αποτέλεσμα να μην μπορεί να κάνει αξιόπιστες προτάσεις.

2.1.5 Υβριδικά Συστήματα Προτάσεων με χρήση Collaborative Filtering και Content-based Filtering

Τα υβριδικά συστήματα προτάσεων συνδυάζουν τις προαναφερθείσες τεχνικές υλοποίησης, με σκοπό να επιτύχουν καλύτερα αποτελέσματα [11]. Αυτή η μέθοδος χρησιμοποιείται ώστε να προσπελάσει τα κοινά προβλήματα που παρουσιάζει η κάθε μέθοδος ξεχωριστά, όπως το cold-start problem που παρουσιάζεται με διαφορετικούς τρόπους στα συστήματα προτάσεων με collaborative filtering και content-based filtering. Υπάρχουν διάφορες τεχνικές που χρησιμοποιούνται στα υβριδικά συστήματα και οι πιο σημαντικές από αυτές παρουσιάζονται πιο κάτω.

- *Weighted:* Το σκορ του προτεινόμενου αντικειμένου υπολογίζεται συνδυάζοντας τα αποτελέσματα των επί μέρους μεθόδων. Αρχικά δίνεται ίσο βάρος στις τεχνικές και στη συνέχεια προσαρμόζεται το βάρος ώστε να επιτυγχάνονται καλύτερα αποτελέσματα.

- *Switching:* Η συγκεκριμένη μέθοδος έχει τη δυνατότητα να καταλαβαίνει κάθε φορά ποια από τις επί μέρους τεχνικές οφείλει να χρησιμοποιήσει. Αρχικά υλοποιείται content-based filtering και στη συνέχεια, αν δεν μπορεί να δώσει αξιόπιστα αποτελέσματα, τότε γίνεται προσπάθεια να γίνουν προτάσεις με την collaborative filtering μέθοδο.

- *Mixed:* Χρησιμοποιούνται και παρουσιάζονται ταυτόχρονα οι προτάσεις από τις διαφορετικές τεχνικές πρόβλεψης.

- *Feature Combination:* Αυτή η μέθοδος χρησιμοποιεί την πληροφορία που παράγεται από το collaborative filtering, ως επιπρόσθετη πληροφορία ώστε να χρησιμοποιηθεί στην content-based filtering τεχνική. Δεν εξαρτάται αποκλειστικά από τα αποτελέσματα που παράγει η τεχνική collaborative filtering, αλλά τα χρησιμοποιεί ώστε να μειώσει την ευαισθησία του συστήματος στον αριθμό τον χρηστών που έχουν βαθμολογήσει ένα αντικείμενο.

- *Cascade:* Αυτή η τεχνική χρησιμοποιεί ένα διαφορετικό τρόπο ώστε να παραγάγει τις προτάσεις [12]. Αρχικά υλοποιείται η μία μέθοδος (collaborative ή content-based) και παράγονται κάποια υποψήφια προς πρόταση αντικείμενα. Στη συνέχεια η άλλη μέθοδος υλοποιείται ώστε να επιλεγεί ποιο από τα αντικείμενα που έχουν προκριθεί από την προηγούμενη μέθοδο θα προταθεί.

2.2 Εξόρυξη πληροφορίας (Data Mining ή KDD)

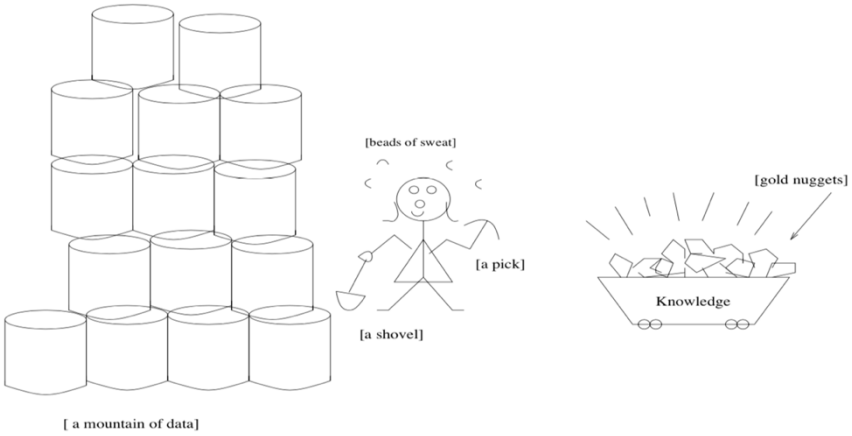
Ένα διάσημο ρητό αναφέρει το εξής: «*Ζούμε στην εποχή της πληροφορίας*». Αυτό κατά κάποιο τρόπο είναι λάθος. Θα ήταν σωστότερο το εξής: «*Ζούμε στην εποχή των δεδομένων*». Τεράστιες ποσότητες δεδομένων συλλέγονται καθημερινά. Το έναυσμα για αυτή την τρομακτική αύξηση στο μέγεθος των δεδομένων, το έδωσε η απίστευτη πρόοδος που παρουσίασε η εξέλιξη του hardware τις τελευταίες δεκαετίες, αφού το υλικό αποθήκευσης πληροφορίας έχει γίνει πολύ φθηνότερο. Η δημοτικότητας της εξόρυξης της πληροφορίας πηγάζει από το γεγονός ότι στην εποχή της κοινωνίας της πληροφορίας και της γνώσης, υπάρχει μια διαρκώς αυξανόμενη ανάγκη για ανάκτηση της πληροφορίας και μετατροπή της σε γνώση.

Η δραματική αύξηση στην ποσότητα των δεδομένων, έχει καταστίσει την ανάλυση και κατανόηση τους απαραίτητες. Είναι γεγονός πως υπάρχει μια κατάσταση όπου παρόλο που υπάρχουν απίστευτες ποσότητες δεδομένων, η πληροφορία που δίνεται από αυτά είναι εξαιρετικά μικρή. Το μέγεθος της πληροφορίας που έχει συλλεχθεί και αποθηκευτεί σε αμέτρητες βάσεις δεδομένων, έχει ξεπεράσει κατά πολύ την ικανότητα του ανθρώπου να την κατανοήσει χωρίς τη βοήθεια κάποιων εργαλείων, τα οποία θα αποκαλύπτουν αυτόματα τη χρήσιμη πληροφορία που κρύβεται πίσω από τους τεράστιους όγκους δεδομένων.

Πολλές φορές η δυσκολία που υπάρχει στην επεξεργασία και κατανόηση των δεδομένων χωρίς τα εργαλεία, οδηγεί στην έλλειψη της πληροφορίας. Η έλλειψη αυτής αναγκάζει αυτόν που λαμβάνει τις αποφάσεις, να το κάνει λαμβάνοντας υπόψιν μόνο τη διαίσθηση του, οδηγώντας τον πολλές φορές σε λανθασμένες αποφάσεις.

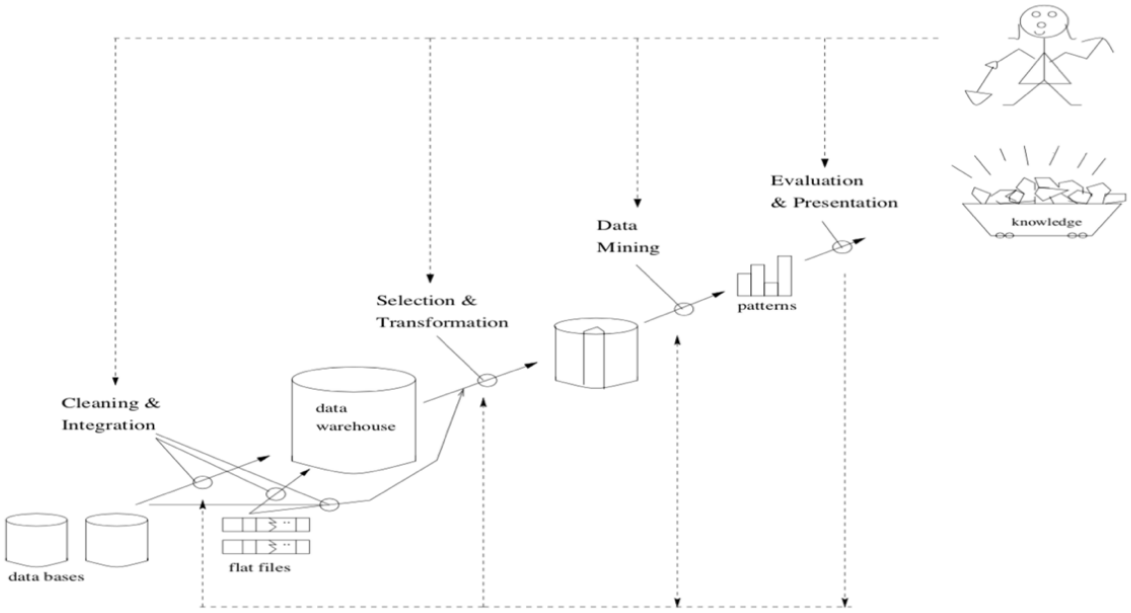
Ο όρος εξόρυξη της πληροφορίας [2,13,14] παραπέμπει σε ένα μεγάλο βουνό από δεδομένα, μέσα από το οποίο γίνεται προσπάθεια ώστε να παραχθεί όσο το δυνατόν περισσότερη και χρησιμότερη γνώση. Ο αγγλικός όρος μπορεί να θεωρηθεί λάθος αφού παραπέμπει σε εξόρυξη δεδομένων και όχι σε εξόρυξη πληροφορίας χρησιμοποιώντας τα δεδομένα. Πολλοί χρησιμοποιούν τον όρο «*ανακάλυψη της γνώσης με χρήση των δεδομένων*» (*knowledge discovery from data ή KDD*) [15] ο οποίος είναι πιο αντιπροσωπευτικός.

Η δυσκολία που παρουσιάζει η ανάλυση και κατανόηση των δεδομένων φαίνεται στην πιο κάτω εικόνα.



Διάγραμμα 5: Η δυσκολία που παρουσιάζει η ανάλυση και κατανόηση των δεδομένων [14]

Η διαδικασία εξόρυξης της πληροφορίας χρησιμοποιώντας τα δεδομένα είναι μια λεπτομερής και χρονοβόρα διαδικασία, που όμως είναι απαραίτητη [14,15,16]. Αρχικά γίνεται ο καθαρισμός των δεδομένων ώστε να απομακρυνθούν τα ασυνεπή δεδομένα και ο θόρυβος και στη συνέχεια όλες οι βάσεις δεδομένων ενώνονται ώστε όλα τα δεδομένα να είναι μαζεμένα σε μία βάση. Στη συνέχεια, γίνεται επιλογή των δεδομένων που θα χρειαστούν στην ανάλυση. Έπειτα, τα δεδομένα δέχονται την κατάλληλη επεξεργασία με σκοπό να έχουν κατάλληλη μορφή για να υποστούν την εξόρυξη, η οποία είναι μια σημαντική διαδικασία, όπου εφαρμόζονται έξυπνες μέθοδοι. Με την εξόρυξη δεδομένων εξάγονται κάποια πρότυπα δεδομένων, τα οποία στη συνέχεια αξιολογούνται ώστε να διερευνηθεί ποια πρότυπα αναπαριστούν χρήσιμη γνώση. Τέλος γίνεται η παρουσίαση των αποτελεσμάτων της διαδικασίας στους χρήστες. Στο πιο κάτω διάγραμμα παρουσιάζεται η διαδικασία της εξόρυξης πληροφορίας.



Διάγραμμα 6: Διαδικασία εξόρυξης της πληροφορίας [14]

Η εξόρυξη πληροφορίας είναι ένα πεδίο το οποίο έχει άμεση σχέση και συνεργασία με πολλά άλλα πεδία, μερικά από τα οποία είναι οι βάσεις δεδομένων, η στατιστική, η μηχανική μάθηση, η αναγνώριση προτύπων, η άντληση της πληροφορίας και οι αλγόριθμοι. Όλα αυτά τα επιστημονικά πεδία συνεργάζονται με σκοπό την όσο το δυνατό αποτελεσματικότερη ανάκτηση της πληροφορίας και μετατροπής της σε γνώση.

2.3 Υπολογισμός Ομοιότητας

Μέτρο ομοιότητας ή συνάρτηση ομοιότητας, ορίζεται μια πραγματική συνάρτηση, η οποία ποσοτικοποιεί την ομοιότητα μεταξύ δύο αντικειμένων. Η ομοιότητα είναι αντίστροφη της απόστασης που υπάρχει μεταξύ των δύο αυτών αντικειμένων. Μικρή απόσταση μεταξύ τους υποδεικνύει μεγάλη ομοιότητα (~1) ενώ μεγάλη απόσταση υποδεικνύει μικρή ομοιότητα (~0).
Παρακάτω παρουσιάζονται μερικές μετρικές υπολογισμού ομοιότητας δύο αντικειμένων.

2.3.1 Απόσταση Manhattan ή Taxicab Metric

Η απόσταση Manhattan ορίζεται ως το άθροισμα των μηκών των προβολών δύο σημείων σε ένα ευθύγραμμο τμήμα. Πιο συγκεκριμένα η απόσταση Manhattan μεταξύ δύο σημείων ορίζεται ως εξής:

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{n} |p_i - q_i|$$

όπου **p** = (p_1, p_2, … , p_n) και **q** = (q_1, q_2, … , q_n).

Στη συνέχεια, η απόσταση Manhattan μπορεί να προσαρμοστεί ώστε να δώσει μέτρο ομοιότητας με μέγιστο τον αριθμό 1, με την πιο κάτω εξίσωση:

$$similarity = 1 - \frac{d(\boldsymbol{p}, \boldsymbol{q})}{n}$$

όπου **n** το μέγεθος των διανυσμάτων **p** και **q**.

Διαφορετικά, μπορεί να χρησιμοποιηθεί ως συνάρτηση μετατροπής της απόστασης σε ομοιότητα, η πιο κάτω συνάρτηση [17]:

$$similarity = e^{-d(\boldsymbol{p},\boldsymbol{q})}$$

2.3.2 Ευκλείδεια Απόσταση

Η ευκλείδεια απόσταση μεταξύ δύο σημείων ορίζεται το μήκος της γραμμής που τα συνδέει. Με άλλα λόγια, η ευκλείδεια απόσταση είναι η τετραγωνική ρίζα του αθροίσματος των διαφορών μεταξύ αντίστοιχων στοιχείων των δύο διανυσμάτων [18]. Πιο συγκεκριμένα η ευκλείδεια απόσταση μεταξύ δύο σημείων ορίζεται ως εξής:

$$d(\boldsymbol{p},\boldsymbol{q}) = d(\boldsymbol{q},\boldsymbol{p}) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

όπου **p** = (p_1, p_2, … , p_n) και **q** = (q_1, q_2, … , q_n).

Στη συνέχεια, η ευκλείδεια απόσταση μπορεί να προσαρμοστεί ώστε να δώσει μέτρο ομοιότητας με μέγιστο τον αριθμό 1, με την πιο κάτω εξίσωση:

$$similarity = \frac{1}{1 + d(\boldsymbol{p},\boldsymbol{q})}$$

Ομοίως με την απόσταση Manhattan, μπορεί να χρησιμοποιηθεί ως συνάρτηση μετατροπής της απόστασης σε ομοιότητα, η πιο κάτω συνάρτηση [17]:

$$similarity = e^{-d(\boldsymbol{p},\boldsymbol{q})}$$

2.3.3 Cosine Similarity

Το Cosine Similarity [19] είναι μια μετρική ομοιότητας μεταξύ δύο μη μηδενικών διανυσμάτων η οποία υπολογίζει το συνημίτονο της γωνίας μεταξύ των δύο αυτών διανυσμάτων. Η ομοιότητα συνημίτονου έχει να κάνει με προσανατολισμό και όχι με το μέτρο. Παίρνει τιμές μεταξύ -1 και +1. Δύο διανύσματα με cosine similarity ίσο με 1, δείχνει δύο διανύσματα ίδιας διεύθυνσης και κατεύθυνσης. Αν το cosine similarity είναι ίσο με 0, τότε τα δύο διανύσματα είναι κάθετα μεταξύ τους, και επομένως δεν υπάρχει συσχέτιση. Σε περίπτωση που το similarity είναι -1, τότε ειναι διαμετρικά αντίθετα. Η εξίσωση για το Cosine Similarity είναι η εξής.

$$Cosine\ Similarity = \frac{\sum_{i=1}^{n} p_i * q_i}{\sqrt{\sum_{i=1}^{n} p_i^2} * \sqrt{\sum_{i=1}^{n} q_i^2}}$$

όπου **p** = (p_1, p_2, … , p_n) και **q** = (q_1, q_2, … , q_n).

2.3.4 Pearson Correlation Coefficient

Ο συντελεστής συσχέτισης του Pearson ή αλλιώς Pearson product-moment correlation coefficient (PPMCC) [19], είναι μια μετρική γραμμικής συσχέτισης δύο μεταβλητών **p** και **q**. Ο συντελεστής παίρνει τιμές από -1 μέχρι +1, όπου η τιμή +1 υποδηλώνει πλήρη συσχέτιση, η τιμή 0 δείχνει πως δεν υπάρχει συσχέτιση, ενώ η τιμή -1 δείχνει πλήρη αρνητική συσχέτιση μεταξύ των δύο μεταβλητών. Ο συντελεστής συσχέτισης Pearson, όταν εφαρμόζεται σε δείγμα πληθυσμού μπορεί να οριστεί ως εξής.

$$Pearson\ Correlation\ Coefficient\ (r_{x,y}) = \frac{\sum_{i=1}^{n}(p_i - \overline{p}) * (q_i - \overline{q})}{\sqrt{\sum_{i=1}^{n}(p_i - \overline{p})^2} * \sqrt{\sum_{i=1}^{n}(q_i - \overline{q})^2}}$$

όπου **p** = (p_1, p_2, … , p_n), **q** = (q_1, q_2, … , q_n), $\overline{p} = \frac{1}{n}\sum_{i=1}^{n} p_i$ και $\overline{q} = \frac{1}{n}\sum_{i=1}^{n} q_i$.

2.4 Ακρίβεια Προβλέψεων

Ο έλεγχος ακρίβειας ενός συστήματος που παράγει προβλέψεις είναι εξαιρετικά σημαντικός ώστε να παραχθούν χρήσιμα συμπεράσματα για την εγκυρότητα του αλγορίθμου. Υπάρχουν πολλές μέθοδοι υπολογισμού του σφάλματος, κάποιες από τις οποίες αναλύονται παρακάτω [20,21]. Ο δείκτης για κάποιες από αυτές τις μεθόδους διατηρεί τις μονάδες μέτρησης των δεδομένων, ενώ κάποιες άλλες ο δείκτης έχει καθαρά ποσοστιαία μορφή. Η διατήρηση των μονάδων μέτρησης των δεδομένων είναι χρήσιμο ώστε να καταλάβει κάποιος το πραγματικό σφάλμα του αλγόριθμου, παρόλα αυτά καθιστά εξαιρετικά δύσκολη τη σύγκριση του σφάλματος όταν η κλίμακα των δεδομένων είναι διαφορετική.

2.4.1 Μέσο Απόλυτο Σφάλμα (Mean Absolute Error – MAE)

Το μέσο απόλυτο σφάλμα εκφράζει ένα μέτρο της ακρίβειας πρόβλεψης έναντι των πραγματικών τιμών διατηρώντας τις μονάδες μέτρησης των δεδομένων. Δηλώνει ένα μέσο μέτρο της αστοχίας της πρόβλεψης, δίχως όμως να δίνει έμφαση στην

κατεύθυνση της πρόβλεψης. Όσο μεγαλύτερη η τιμή του δείκτη, τόσο μικρότερη η ακρίβεια της μεθόδου που εφαρμόστηκε. Υπολογίζεται ως εξής.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - F_i|$$

όπου **n** είναι ο αριθμός των προβλέψεων, $\mathbf{Y_i}$ η πραγματική τιμή και $\mathbf{F_i}$ η τιμή που έχει προβλέψει ο αλγόριθμος.

2.4.2 Κανονικοποιημένο Μέσο Απόλυτο Σφάλμα (Normalized Mean Absolute Error – NMAE)

Το κανονικοποιημένο μέσο απόλυτο σφάλμα χρησιμοποιείται κυρίως στα συστήματα προτάσεων, με σκοπό να δημιουργήσει συγκρίσιμους δείκτες, χωρίς να εξαρτώνται από την κλίμακα των βαθμολογιών [21]. Ο δείκτης NMAE παίρνει τιμές από 0 μέχρι 1 και υπολογίζεται ως εξής.

$$NMAE = \frac{MAE}{(maxR - minR)} = \frac{1}{n * (maxR - minR)}\sum_{i=1}^{n}|Y_i - F_i|$$

όπου **n** είναι ο αριθμός των προβλέψεων, **maxR** και **minR** η μέγιστη και ελάχιστη τιμή αντίστοιχα που μπορεί να πάρει η βαθμολογία ενός αντικειμένου στο σύστημα προτάσεων, $\mathbf{Y_i}$ η πραγματική τιμή και $\mathbf{F_i}$ η τιμή που έχει προβλέψει ο αλγόριθμος.

Χρησιμοποιώντας το δείκτη NMAE είναι δύσκολο κάποιος να αντιληφθεί το πραγματικό σφάλμα του αλγορίθμου, παρόλα αυτά είναι εξαιρετικά χρήσιμος στη σύγκριση σφαλμάτων για αλγόριθμους που χρησιμοποιούνται στα συστήματα προτάσεων.

2.4.3 Μέσο Τετραγωνικό σφάλμα (Mean Squared Error – MSE)

Όπως και το μέσο απόλυτο σφάλμα, έτσι και το μέσο τετραγωνικό σφάλμα, είναι μέσο ακρίβειας της πρόβλεψης, το οποίο όμως δίνει πολύ μεγαλύτερο βάρος στα μεγάλα σφάλματα, αφού τετραγωνίζονται, και πολύ μικρότερο στα μικρά σφάλματα. Υπολογίζεται από τον παρακάτω τύπο.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - F_i)^2$$

όπου **n** είναι ο αριθμός των προβλέψεων, $\mathbf{Y_i}$ η πραγματική τιμή και $\mathbf{F_i}$ η τιμή που έχει προβλέψει ο αλγόριθμος.

2.4.4 Ρίζα Μέσου Τετραγωνικού Σφάλματος (Root Mean Squared Error – RMSE)

Υπολογίζεται άμεσα από την τετραγωνική ρίζα του μέσου τετραγωνικού σφάλματος. Έχει τις ίδιες ιδιότητες με αυτό, αλλά είναι εκφρασμένος στις μονάδες μέτρησης των δεδομένων.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - F_i)^2}$$

όπου **n** είναι ο αριθμός των προβλέψεων, $\mathbf{Y_i}$ η πραγματική τιμή και $\mathbf{F_i}$ η τιμή που έχει προβλέψει ο αλγόριθμος.

2.4.5 Μέσο Απόλυτο Ποσοστιαίο Σφάλμα (Mean Absolute Percentage Error – MAPE)

Το μέσο απόλυτο ποσοστιαίο σφάλμα είναι εκφρασμένο επί τις εκατό και λαμβάνει τιμές μεγαλύτερες ή ίσες το μηδενός, με τις μικρότερες τιμές να υποδηλώνουν και καλύτερη απόδοση του αλγορίθμου πρόβλεψης. Υπολογίζεται με τον πιο κάτω τύπο.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - F_i}{Y_i}\right| * 100\ (\%)$$

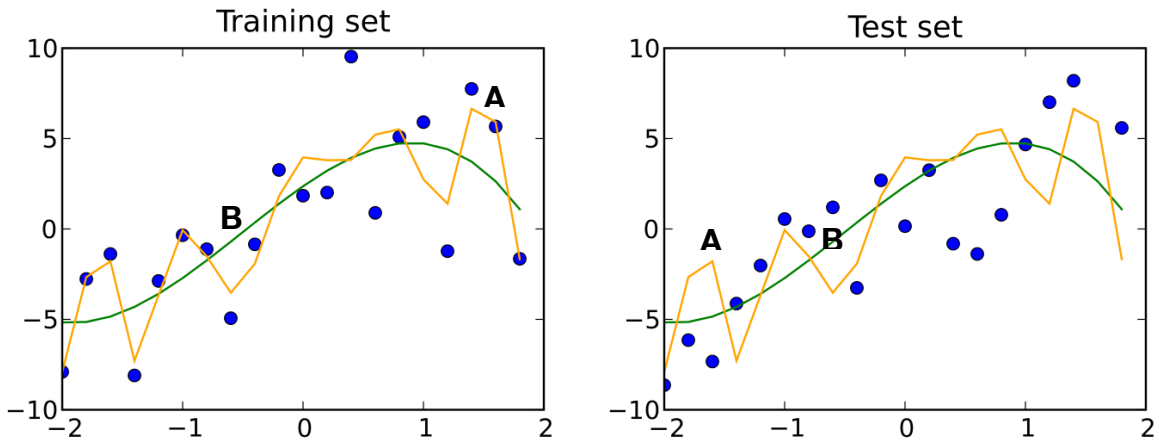
όπου **n** είναι ο αριθμός των προβλέψεων, $\mathbf{Y_i}$ η πραγματική τιμή και $\mathbf{F_i}$ η τιμή που έχει προβλέψει ο αλγόριθμος.

2.5 Επαλήθευση Μετρήσεων (Validation)

Σε ένα πρόβλημα πρόβλεψης, δίνονται συνήθως κάποια γνωστά δεδομένα, στα οποία γίνεται η εκπαίδευση (training dataset), και κάποια άγνωστα δεδομένα στα οποία καλείται το μοντέλο να κάνει την πρόβλεψη (testing dataset). Χρησιμοποιείται συνήθως στις περιπτώσεις όπου ο σκοπός είναι η πρόβλεψη, ώστε να υπολογιστεί πόσο ακριβής είναι η συμπεριφορά του μοντέλου. Ο σκοπός του Validation είναι να ελέγξει την ικανότητα του μοντέλου να προβλέπει το αποτέλεσμα των δεδομένων που δεν χρησιμοποιήθηκαν για την εκπαίδευση του και να αναδείξει τυχόν προβλήματα που μπορεί να παρουσιαστούν (overfitting ή επιλεκτική προκατάληψη), ώστε να εξακριβωθεί αν μπορεί το μοντέλο να χρησιμοποιηθεί σε πραγματικό πρόβλημα με άγνωστο dataset.

Αν ένα μοντέλο έχει καλά αποτελέσματα τόσο για το training dataset όσο και για το testing dataset, υποδεικνύει πως το μοντέλο δεν παρουσιάζει overfitting. Σε περίπτωση όμως που το μοντέλο παρουσιάζει καλά αποτελέσματα για το training dataset ενώ για το testing dataset όχι, είναι εξαιρετικά πιθανή η ύπαρξη overfitting στο training dataset.

Πιο κάτω παρουσιάζονται τα διαγράμματα των training και testing datasets με τον ίδιο πληθυσμό για δυο μοντέλα (A και B).



Διάγραμμα 7: Το φαινόμενο του overfitting στο training set

Το dataset παρουσιάζεται με κουκίδες. Τα δύο μοντέλα παρουσιάζονται και στο training set και στο test set. Όσον αφορά το μοντέλο A, στο training set παρουσιάζει μέσο τετραγωνικό σφάλμα (MSE) 4, ενώ στο testing set παρουσιάζει μέσο τετραγωνικό σφάλμα 15. Αντιθέτως, το μοντέλο B παρουσιάζει στο training set μέσο τετραγωνικό σφάλμα 9, ενώ στο testing set, 13. Από τις πιο πάνω παρατηρήσεις είναι ξεκάθαρο το overfitting που παρουσιάζει το μοντέλο A για το dataset, αφού το μέσο

τετραγωνικό σφάλμα είναι σχεδόν 4 φορές μεγαλύτερο στο testing set. Όσον αφορά το μοντέλο B, το μέσο τετραγωνικό σφάλμα για το training set είναι λιγότερο από 2 φορές μεγαλύτερο, γεγονός που δείχνει πως δεν υπάρχει overfitting στο dataset.

Όσον αφορά τον διαχωρισμό του dataset σε training dataset και test dataset, υπάρχουν διάφορες τακτικές, οι οποίες παρουσιάζονται πιο κάτω.

2.5.1 Re-substitution Validation

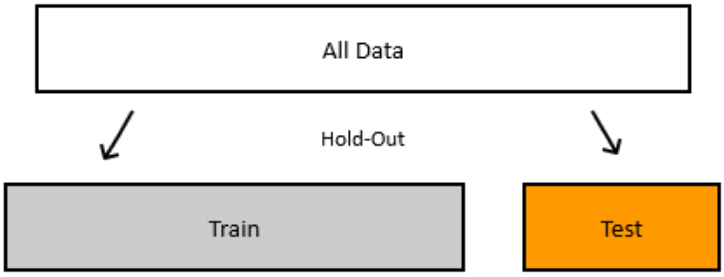
Σε αυτή την περίπτωση, το μοντέλο εκπαιδεύεται σε όλα τα διαθέσιμα δεδομένα, και στη συνέχεια το ίδιο dataset χρησιμοποιείται για να γίνει το testing [22]. Στην προκειμένη περίπτωση, υπάρχει το πρόβλημα του overfitting, αφού το testing γίνεται στα ίδια δεδομένα στα οποία έχει εκπαιδευτεί το μοντέλο.

2.5.2 Hold-out Validation

Η τεχνική hold-out validation [22] χωρίζει το dataset σε δύο μέρη, εκ των οποίων το ένα χρησιμοποιείται για εκπαίδευση και το άλλο για έλεγχο. Το σύνολο του dataset μπορεί να κατανεμηθεί σε training dataset και testing dataset με διάφορες αναλογίες με τις πιο συνηθισμένες να κυμαίνονται γύρω στα 2/3 ολόκληρου του dataset για το training και 1/3 για το testing [23]. Η πιο διαδεδομένη αναλογία είναι η 70-30, στην οποία το 70% των δεδομένων χρησιμοποιούνται για training, ενώ το υπόλοιπο 30% για testing. Άλλες διαδεδομένες αναλογίες είναι η 75%-25% και η 80%-20%.

Ένα σημαντικό μειονέκτημα του hold-out validation είναι το γεγονός ότι η διαδικασία αυτή δε χρησιμοποιεί όλα τα διαθέσιμα δεδομένα και επομένως τα αποτελέσματα είναι εξαρτώμενα από την επιλογή των training και testing sets.

Στο πιο κάτω διάγραμμα παρουσιάζεται η διαδικασία του hold-out validation.



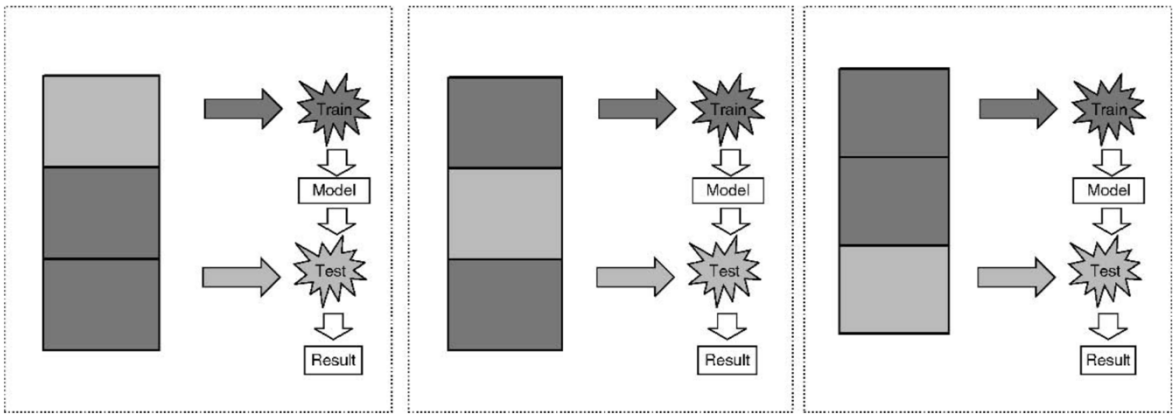
Διάγραμμα 8: Validation με τη μέθοδο hold-out

2.5.3 k-fold Cross-Validation

Η τεχνική Cross-Validation [22,24,25] είναι μια από τις διάφορες τεχνικές επαλήθευσης μοντέλων που χρησιμοποιούνται για να προσδιοριστεί πως τα αποτελέσματα μιας στατιστικής ανάλυσης θα γενικευθούν σε ένα ανεξάρτητο dataset. Αρχικά το dataset χωρίζεται σε k ίσα ή σχεδόν ίσα μέρη. Το μοντέλο ακολουθεί τη διαδικασία του training και του testing k φορές. Κάθε φορά, επιλέγονται k-1 υποσύνολα για το training και το υπολειπόμενο υποσύνολο θεωρείται το testing set. Το μοντέλο κάθε φορά υπολογίζει την ακρίβειά του για το testing set. Η διαδικασία ολοκληρώνεται μετά από k επαναλήψεις, όταν όλα τα υποσύνολα θα έχουν θεωρηθεί ως το testing set. Η ακρίβεια του μοντέλου προκύπτει από τον μέσο όρο της ακρίβειας για την κάθε επανάληψη [25].

Είναι σημαντικό να αναφέρουμε ότι πριν γίνει ο διαχωρισμός του dataset σε k υποσύνολα, το dataset αναδιατάσσεται με τέτοιο τρόπο, ώστε το κάθε μέρος να είναι όσο το δυνατόν περισσότερο μια αναπαράσταση ολόκληρου του dataset.

Στα πεδία της εξόρυξης πληροφορίας και της μηχανικής μάθησης, η πιο συνήθης επιλογή για το k είναι το 10 [24,25]. Στο πιο κάτω διάγραμμα παρουσιάζεται η διαδικασία για k=3.



Διάγραμμα 9: Validation με τη μέθοδο k-fold cross-validation με k = 3 [25]

Διαφορά μεταξύ 2-Fold Cross-Validation και 50% Hold-out Validation

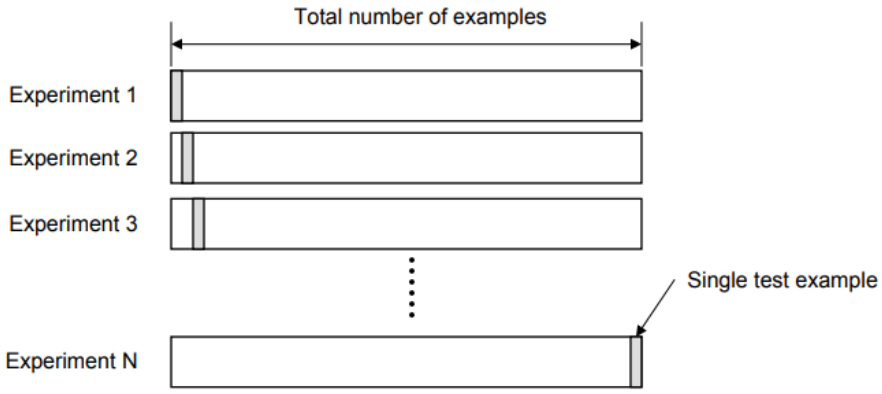
Υπάρχουν πολλά άρθρα στο διαδίκτυο στα οποία αναφέρεται πως οι δύο αυτές διαδικασίες είναι οι ίδιες. Αυτό πιστεύεται ακόμα και από κάποιους ειδικούς στη μηχανική μάθηση. Υπάρχει όμως διαφορά και είναι πολύ απλή. Στο 50% Hold-out Validation, τα δεδομένα χωρίζονται σε 2 ίσα υποσύνολα. Το μοντέλο εκπαιδεύεται στο ένα υποσύνολο, το οποίο θεωρείται το training set, ενώ γίνεται η επαλήθευση στο άλλο υποσύνολο, το testing set. Αυτή η διαδικασία γίνεται μόνο μια φορά. Αντιθέτως,

στο 2-Fold Cross-Validation, ακολουθείται η ίδια διαδικασία, η οποία όμως γίνεται δύο φορές. Στην πρώτη επανάληψη, το πρώτο μισό υποσύνολο έχει τον ρόλο του training set, ενώ στην δεύτερη επανάληψη, έχει τον ρόλο του testing set. Η ακρίβεια του μοντέλου προκύπτει από τον μέσο όρο των δύο επαναλήψεων.

2.5.4 Leave-One-Out Cross-Validation (LOOCV)

Η συγκεκριμένη διαδικασία είναι μια ειδική περίπτωση του k-Fold Cross Validation, όπου το k ισούται με τον αριθμό των παρατηρήσεων του dataset (n). Με πιο απλά λόγια, κάθε φορά που γίνεται μια επανάληψη, χρησιμοποιείται ως training set το σύνολο του dataset εκτός από μια παρατήρηση, η οποία χρησιμοποιείται ως testing set. Με αυτό τον τρόπο, γίνονται n επαναλήψεις και επομένως τα αποτελέσματα να είναι αμερόληπτα, όμως λόγω της μεγάλης διακύμανσης που παρουσιάζεται, να μην θεωρούνται αξιόπιστα [26]. Η συγκεκριμένη διαδικασία ακολουθείται για πολύ μικρά datasets.

Στο παρακάτω διάγραμμα παρουσιάζεται η διαδικασία Leave-One-Out Cross-Validation.



Διάγραμμα 10: Validation με τη μέθοδο LOOCV

2.6 Αρχιτεκτονική Representational State Transfer (REST) και RESTful API

Το REST [27,28] είναι ένα αρχιτεκτονικό στυλ για τη λειτουργία του παγκόσμιου ιστού ή γενικότερα των κατανεμημένων συστημάτων. Σκοπός του REST είναι να οριοθετεί αρχές, περιορισμούς και βασικές λειτουργίες, ανεξάρτητα από τη γλώσσα

προγραμματισμού, το πρωτόκολλο επικοινωνίας ή το είδος των δεδομένων. Η σωστή χρήση των RESTful αρχιτεκτονικών βελτιώνει όλα τα βασικά χαρακτηριστικά μιας αρχιτεκτονικής (απόδοση, κλιμάκωση, απλότητα, επεκτασιμότητα, αξιοπιστία).

Η αρχιτεκτονική REST διέπεται από κάποιες βασικές αρχές. Η πρώτη δηλώνει τη χρήση ομοιόμορφης διεπαφής για την επικοινωνία client και server, παράλληλα όμως δηλώνει και τον πλήρη διαχωρισμό των ενδιαφερόντων μεταξύ τους. Επίσης ο server δεν αποθηκεύει καμία πληροφορία για την κατάσταση της εφαρμογής κατά την επικοινωνία με τον client. Επιπλέον, ο client μπορεί να αποθηκεύσει προσωρινά επιλεγμένες απαντήσεις από τον server, αλλά και να μπορεί να συνδεθεί είτε με τον end server είτε με κάποιον ενδιάμεσο server, χωρίς να μπορεί να διακρίνει τη διαφορά. Τέλος, η διαχείριση των πόρων γίνεται μέσω των αναπαραστάσεων τους.

Υπάρχουν τέσσερις βασικές HTTP μέθοδοι οι οποίες χρησιμοποιούνται στη RESTful αρχιτεκτονική και εκτελούν διαφορετική λειτουργία με βάση το HTTP base URL ή αλλιώς το REST endpoint.

Οι τέσσερις αυτές μέθοδοι παρουσιάζονται στον πιο κάτω πίνακα με τη λειτουργία που εκτελούν όταν το REST endpoint αποτελεί συλλογή ή στοιχείο.

Μέθοδος HTTP	*Συλλογή (δηλ. /collection)*	*Στοιχείο (δηλ. /collection/element)*
GET	Εμφάνιση της λίστας με τα στοιχεία της συλλογής	Ανάκτηση της αναπαράστασης του συγκεκριμένου στοιχείου
PUT	Αντικατάσταση της υπάρχουσας συλλογής με νέα	Τροποποίηση του συγκεκριμένου στοιχείου
POST	Προσθήκη νέου στοιχείου στη συλλογή με απροσδιόριστο id	-
DELETE	Διαγραφή της υπάρχουσας συλλογής	Διαγραφή του συγκεκριμένου στοιχείου

Πίνακας 1: Οι λειτουργίες των HTTP μεθόδων σε συλλογές και στοιχεία

Τα RESTful HTTP endpoints (*δηλ. /endpoint*) ανταλλάζουν δεδομένα σε μορφή JSON, η μορφή η οποία αποτελεί την αναπαράσταση των δεδομένων. Η αναπαράσταση αυτή αποτελεί την κατάσταση των δεδομένων την ώρα που ο client καταχωρεί αίτημα στον server.

3

Τεχνολογίες

3.1 Java

Η γλώσσα προγραμματισμού Java [29,30], είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, δηλαδή έχει σαν βάση της κλάσεις οι οποίες αποτελούν αντικείμενα. Η κεντρική ιδέα του αντικειμενοστραφούς προγραμματισμού είναι πως το αντικείμενο ξέρει να κάνει μια ενέργεια για τον εαυτό του και όχι πως το πρόγραμμα ξέρει πως να κάνει μια εργασία για το αντικείμενο. Έχει σχεδιαστεί με σκοπό να υπάρχουν όσο το δυνατό λιγότερες υλοποιήσεις εξαρτήσεων και να μπορεί να τρέχει σε όλες τις πλατφόρμες χωρίς να χρειάζεται η επαναμεταγλώττιση του κώδικα. Οι εφαρμογές που είναι γραμμένες στην Java, μπορούν να τρέχουν σε οποιαδήποτε εικονική μηχανή Java (Java Virtual Machine – JVM), χωρίς να εξαρτάται από το λειτουργικό σύστημα ή την αρχιτεκτονική του εκάστοτε συστήματος.

Η Java μοιάζει αρκετά με τη γλώσσα προγραμματισμού C++, έχοντας όμως λιγότερες low-level δυνατότητες. Αυτός είναι και ο λόγος που η Java είναι αρκετά πιο αργή και απαιτεί περισσότερη μνήμη από τη C++. Παρόλα αυτά γίνονται προσπάθειες ώστε να γίνουν βελτιώσεις, με σκοπό να μειωθούν ο χρόνος εκτέλεσης και οι απαιτήσεις μνήμης.

Τα αντικείμενα στην Java δημιουργούνται με την κλήση της *new.* Όμως δεν υπάρχει άμεσος τρόπος να καταστραφούν αυτά τα αντικείμενα ή να αποδεσμεύσουν τη μνήμη που καταλαμβάνουν. Αυτό γίνεται αυτόματα μέσω της συλλογής σκουπιδιών (garbage

collection). Η Java χρησιμοποιεί αυτή τη λειτουργία ώστε να διαχειρίζεται τη μνήμη καθ' όλη τη διάρκεια ζωής ενός αντικειμένου. Στη C++ όπου δεν υπάρχει συλλέκτης σκουπιδιών, είναι υποχρέωση του προγραμματιστή να καταστρέψει τα αντικείμενα και να αποδεσμεύσει τη μνήμη που αυτά κατέχουν. Αν δε γίνει αυτό, είναι εξαιρετικά πιθανή η υπερχείλιση μνήμης. Ο προγραμματιστής καθορίζει ποια αντικείμενα δημιουργούνται, και η Java, μέσω του συλλέκτη σκουπιδιών, είναι υπεύθυνη να καταστρέψει τα αντικείμενα όταν δε χρησιμοποιούνται ελευθερώνοντας μνήμη.

Ένα από τα σημαντικότερα πλεονεκτήματα της Java είναι οι βιβλιοθήκες που υπάρχουν, παρέχοντας στους προγραμματιστές πολλές από τις κοινές λειτουργίες που χρησιμοποιούνται συχνά, έτοιμες για χρήση. Επιπλέον, προσφέρονται βιβλιοθήκες για κάποιες λειτουργίες οι οποίες είναι άκρως εξαρτώμενες από το λειτουργικό σύστημα ή την αρχιτεκτονική του συστήματος, όπως πρόσβαση στο δίκτυο ή σε αρχεία. Αυτός είναι ο λόγος, όπως αναφέρθηκε και προηγουμένως, για τον οποίο η Java μπορεί να τρέχει με τον ίδιο τρόπο σε οποιοδήποτε σύστημα χωρίς να χρειάζεται αλλαγή στον κώδικα και επαναμεταγλώττιση.

Παρά το γεγονός ότι δεν είναι δυνατό να μετρηθούν οι προγραμματιστές για κάθε γλώσσα προγραμματισμού, η Java θεωρείται, αν όχι η πιο διαδεδομένη, μια από τις πιο διαδεδομένες και ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού.

3.2 Gradle

Το Gradle [31,32] είναι ένα ανοικτό λογισμικό αυτοματοποίησης του build, το οποίο σχεδιάστηκε ώστε να είναι αρκετά ευέλικτο με σκοπό να μπορεί να υποστηρίζει την ανάπτυξη κώδικα σε πολλές γλώσσες προγραμματισμού και να κάνει build σχεδόν οποιοδήποτε τύπο λογισμικού, υποστηρίζοντας το για όλο τον κύκλο ζωής του. Επιπρόσθετα, το Gradle κάνει τη διαδικασία ανάπτυξης κώδικα ευκολότερη, βοηθώντας επίσης και στην εύκολη επέκταση και συντήρηση του λογισμικού.

Ένα από τα πλεονεκτήματα που προσφέρει η ανάπτυξη λογισμικού με τη βοήθεια του Gradle, είναι η ταχύτητα με την οποία ολοκληρώνει κάποιες εργασίες, αφού επαναχρησιμοποιεί τα δεδομένα εξόδου από τις προηγούμενες εκτελέσεις του κώδικα και εκτελεί το πρόγραμμα μόνο τα δεδομένα εισόδου που έχουν αλλάξει. Επιπλέον, εκτελεί πολλές εργασίες παράλληλα, αυξάνοντας την ταχύτητα εκτέλεσης των εργασιών. Ένας ακόμη λόγος για τον οποίο η ανάπτυξη λογισμικού με το Gradle είναι γρηγορότερη, είναι το γεγονός ότι χρησιμοποιεί κατευθυνόμενο ακυκλικό γράφο (DAG) [31] ώστε να αποφασίζει ποιες εργασίες χρειάζεται να εκτελεστούν και στη συνέχεια τη σειρά με την οποία αυτές θα εκτελεστούν.

Ακόμα ένα πλεονέκτημα του Gradle, οφείλεται στον σχεδιασμό του, ο οποίος εστιάζει στην ελαστικότητα και στην βελτίωση της απόδοσης του λογισμικού. Το Gradle περιέχει ενσωματωμένα εργαλεία ανάπτυξης και servers που επιτρέπουν τη συνεχή ενσωμάτωση (continuous integration), και έτσι διατηρούνται ενημερωμένες οι βιβλιοθήκες και οι πηγές που δημιουργούν εξαρτήσεις, με αποτέλεσμα να είναι πάντα ενημερωμένος ο κώδικας.

3.3 Spring Framework

Το Spring Framework [33,34] είναι ένα από τα πιο διαδεδομένα frameworks ανάπτυξης λογισμικού στη γλώσσα προγραμματισμού Java. Χρησιμοποιείται για την ανάπτυξη λογισμικού με μεγάλη έκταση.

Ένα από τα μεγαλύτερα πλεονεκτήματα του Spring Framework είναι το Inversion of Control, το οποίο αποτελεί μια από τις βασικές αρχές του software engineering. Το Inversion of Control δίνει τον έλεγχο των αντικειμένων ή κομματιών του κώδικα σε ένα framework ή container. Βασικό πλεονέκτημα της συγκεκριμένης τεχνικής, είναι η αποσύνδεση της εκτέλεσης μιας εργασίας από την υλοποίησή της, κάνοντας ευκολότερη την εναλλαγή μεταξύ διαφορετικών υλοποιήσεων της εργασίας αυτής.

Το Inversion of Control μπορεί να επιτευχθεί με διάφορους μηχανισμούς με τον σημαντικότερο από αυτούς να είναι το Dependency Injection το οποίο επιτρέπει στον προγραμματιστή να αναπτύσσει χαλαρά συζευγμένες εφαρμογές, οι οποίες είναι εύκολες στο unit-testing. Η διασύνδεση των αντικειμένων με άλλα αντικείμενα γίνεται από ένα συμβολομεταφραστή (assembler), και όχι μεταξύ των αντικειμένων. Το Dependency Injection μπορεί να υλοποιηθεί στο Spring Framework μέσω των πεδίων (fields), των κατασκευαστών (constructors), ή των setters.

Στο Spring Framework το Inversion of Control container είναι υπεύθυνο για την αρχικοποίηση, διαμόρφωση και διαχείριση του κύκλου ζωής των αντικειμένων (beans). Η συμβολομετάφραση των αντικειμένων (beans) γίνεται τη στιγμή που τρέχει το πρόγραμμα, αφού έχουν χρησιμοποιηθεί τα μεταδεδομένα (metadata), τα οποία διαμορφώνονται είτε σε μορφή XML είτε σε σχολιασμούς (annotations).

Η διασύνδεση (wiring) επιτρέπει στο container να υλοποιεί τις εξαρτήσεις μεταξύ των αντικειμένων (beans), αναγνωρίζοντας τα αντικείμενα που έχουν οριστεί. Υπάρχουν τέσσερις τρόποι διασύνδεσης (wiring) των αντικειμένων. Ο πρώτος τρόπος είναι

καθορίζοντας κάθε φορά τις εξαρτήσεις, οπότε δε χρησιμοποιείται το autowiring. Οι άλλοι τρεις χρησιμοποιούν την αυτόματη διασύνδεση (autowiring)· με τη χρήση του *constructor*, το οποίο σημαίνει πως το Spring Framework θα αναζητήσει αντικείμενα με τα δεδομένα του constructor, *byName* όπου η διασύνδεση γίνεται με βάση το όνομα, οπότε το Spring αναζητεί αντικείμενα με το ίδιο όνομα, και τέλος *byType*, όπου η διασύνδεση γίνεται με βάση τον τύπο (δεν μπορούν να υπάρχουν περισσότερα από ένα αντικείμενα με τον ίδιο τύπο).

Ακόμα μια λειτουργία που καθιστά το Spring Framework ένα σημαντικό εργαλείο ανάπτυξης λογισμικού, είναι το γεγονός ότι είναι Aspect-Oriented (AOP), δηλαδή δε στηρίζεται στις κλάσεις όπως συμβαίνει στον αντικειμενοστραφή προγραμματισμό (OOP), αλλά στα χαρακτηριστικά. Με τη χρήση αυτής της ιδιότητας, το Spring Framework επιτρέπει τη σωστή διαχείριση τμημάτων ενός προγράμματος που επηρεάζουν ή βασίζονται σε άλλα τμήματα του συστήματος και είναι δύσκολο να απομονωθούν από το υπόλοιπο σύστημα, τόσο στο σχεδιασμό, όσο και στην υλοποίηση.

Τέλος, δύο άλλα σημαντικά εργαλεία που περιέχει το Spring Framework, είναι η πρόσβαση σε δεδομένα (θα εξηγηθεί περαιτέρω πιο κάτω) και το Model-View-Controller (MVC). Το MVC βασίζεται σε HTTP και servlets και παρέχει τη δυνατότητα για επεκτάσεις και προσαρμογές σε διαδικτυακές εφαρμογές και διαδικτυακές υπηρεσίες RESTful.

3.3.1 Spring Boot

Παρά τα μεγάλα πλεονεκτήματα που προσφέρει το Spring Framework, είναι εξαιρετικά δύσχρηστο στην εγκατάσταση ενός νέου πρότζεκτ, ακόμα και με τις λιγότερες προδιαγραφές. Εδώ είναι που έρχεται το Spring Boot [35], φροντίζοντας από μόνο του όλες τις εξαρτήσεις του λογισμικού και το μόνο που αφήνει στον προγραμματιστή να κάνει είναι να τρέξει τον κώδικα (“just run”).

Επίσης μια διαδικτυακή εφαρμογή έχει κάποιες συγκεκριμένες ανάγκες (πχ Spring MVC, Jackson Databind κτλ) οι οποίες είναι σχεδόν ίδιες σε κάθε διαδικτυακή εφαρμογή. Το Spring Boot με σκοπό να μειώσει την πολυπλοκότητα και να βοηθήσει τον προγραμματιστή, του προσφέρει κάποια starters. Τα starters είναι ένα σύνολο εξαρτήσεων που περιλαμβάνει μια εφαρμογή που χρησιμοποιεί το Spring Framework. Αυτό προσφέρει στον προγραμματιστή τη δυνατότητα να μην ανησυχεί όσον αφορά τις εξαρτήσεις και τις εκδόσεις τους, αλλά να επικεντρωθεί στην ανάπτυξη του λογισμικό γλυτώνοντας του κόπο και χρόνο.

Επιπλέον προσφέρει κάποιες δυνατότητες οι οποίες είναι χρήσιμες στην παρατήρηση του λογισμικού, όπως κάποιες μετρικές, και ελέγχους υγείας του λογισμικού. Τέλος, περιέχει ενσωματωμένο το Tomcat, προσφέροντας ένα «καθαρό» HTTP Web Server περιβάλλον, όπου μπορεί να τρέξει ο κώδικας σε Java.

3.3.2 Spring Boot και Gradle

Κατά τη διάρκεια της αρχικοποίησης μιας νέας εφαρμογής με τη βοήθεια του Spring Boot, δίνεται η δυνατότητα να περιληφθεί στην εφαρμογή αυτή και το Spring Boot Gradle Plugin [36]. Το εργαλείο αυτό προσφέρει στην εφαρμογή υποστήριξη από το Gradle, δίνοντας τη δυνατότητα στον προγραμματιστή να τρέχει τις Spring Boot εφαρμογές και να χρησιμοποιεί τη διαχείριση εξαρτήσεων που προσφέρει ο συνδυασμός των δύο.

3.3.3 Spring Data

Ο σκοπός του Spring Data [37] είναι να παρέχει ένα οικείο και συνεπές μοντέλο προγραμματισμού για σχεσιακές και μη σχεσιακές βάσεις δεδομένων, βασισμένο στο Spring Framework, ώστε να παρέχει πρόσβαση στα δεδομένα διατηρώντας παράλληλα τα βασικά χαρακτηριστικά των βάσεων δεδομένων. Το Spring Data περιλαμβάνει μια ισχυρή αποθήκη δεδομένων, στην οποία έχουν υλοποιηθεί οι βασικές CRUD (Create – Read – Update – Delete) μέθοδοι και κάποια ερωτήματα βασισμένα στα ονόματα των πεδίων και των μεθόδων. Οι αποθήκες δεδομένων έχουν δημιουργηθεί για να μειώσουν σημαντικά το μέγεθος του κοινού κώδικα που χρειάζεται να γραφτεί, ώστε να διασφαλιστεί η πρόσβαση στα διάφορα επίπεδα δεδομένων.

Αποτελεί ένα μεγάλο πρότζεκτ του Spring Framework, καθώς έχουν υλοποιηθεί πολλές σχεσιακές και μη σχεσιακές βάσεις δεδομένων, παρέχοντας στους προγραμματιστές πληθώρα επιλογών. Κάποιες από τις πιο ευρέως χρησιμοποιούμενες βάσεις δεδομένων που υποστηρίζει το Spring Data είναι οι εξής: JDBC, JPA, MongoDB, REST, Cassandra, Apache Solr και φυσικά η Elasticsearch (Spring-Data Elasticsearch) [38,39,40] η οποία χρησιμοποιήθηκε στα πλαίσια της υλοποίησης της παρούσας διπλωματικής εργασίας.

3.4 Elasticsearch

Η Elasticsearch [41] είναι μια ανοικτού λογισμικού, real-time κατανεμημένη μηχανή αναζήτησης κτισμένη πάνω στην Apache Lucene [42], την πιο ολοκληρωμένη και αποτελεσματική μηχανή αναζήτησης κειμένου. Η Apache Lucene είναι μια βιβλιοθήκη, η οποία είναι πολύ δύσκολη στη χρήση και απαιτεί πρόσβαση από την Java. Η Elasticsearch είναι γραμμένη σε Java και χρησιμοποιεί την Apache Lucene, με σκοπό να κρύβει τα μειονεκτήματα της πίσω από ένα απλό RESTful API. Μέσα από ένα σύνολο από APIs και DSL ερωτήματα (Domain Specific Language), τα οποία αποτελούν μια ευέλικτη και πλήρως εκφραστική γλώσσα αναζήτησης που χρησιμοποιεί η Elasticsearch για να προβάλει τη δύναμη της Lucene χρησιμοποιώντας μια απλή JSON διεπαφή, καθώς και με clients για τις πιο γνωστές γλώσσες προγραμματισμού, προσφέρει απεριόριστες δυνατότητες στο πεδίο της τεχνολογίας αναζήτησης.

Η Elasticsearch επιτρέπει στο χρήστη να ψάχνει σε τεράστιο όγκο δεδομένων με μεγάλη ταχύτητα [41,43,44]. Χρησιμοποιείται κυρίως για αναζήτηση κειμένου και για ανάλυση. Κανένα από τα χαρακτηριστικά της Elasticsearch δεν είναι καινούργιο. Οι κατανεμημένες βάσεις δεδομένων, τα συστήματα ανάλυσης όπως επίσης και η αναζήτηση κειμένου έχουν ξαναχρησιμοποιηθεί στο παρελθόν. Αυτό που κάνει την Elasticsearch να ξεχωρίζει είναι ο συνδυασμός αυτών των επί μέρους τεχνολογιών, ο οποίος επιτρέπει στο χρήστη να μπορεί να χρησιμοποιήσει τα δεδομένα του με τρόπο που να τα καθιστούν χρήσιμα, εξάγοντας χρήσιμα συμπεράσματα.

Η Elasticsearch είναι μια schema-less αποθήκη δεδομένων. Επομένως, ο χρήστης δεν εισάγει ένα αντικείμενο σε σειρές και στήλες όπως γίνεται στην SQL, αφού αυτό θα άφηνε ανεκμετάλλευτο αυτό το πλεονέκτημα της Elasticsearch. Κάθε φορά που εισάγεται νέα εγγραφή, χωρίζεται σε πεδία και ανανεώνεται το σχέδιο της βάσης λαμβάνοντας υπόψιν τα δεδομένα που έχουν εισαχθεί, χωρίς να χρειάζεται να ανακατασκευάζει τη βάση κάθε φορά.

Κάθε εγγραφή που εισάγεται στην Elasticsearch πρέπει να είναι σε μορφή JSON. Η Elasticsearch είναι document-oriented, δηλαδή αποθηκεύει ολόκληρα αντικείμενα-documents. Παρόλα αυτά δεν κάνει μόνο αυτό, αφού χωρίζει το περιεχόμενο του κάθε κειμένου, κάνοντάς εύκολη την αναζήτησή του. Στην Elasticsearch ο χρήστης εισάγει, αναζητεί και φιλτράρει αντικείμενα-documents και όχι εγγραφές οι οποίες χωρίζονται σε στήλες.

Θα μπορούσε να γίνει μια σύγκριση της Elasticsearch με μια παραδοσιακή SQL βάση δεδομένων ώστε να γίνουν πιο ξεκάθαροι οι ορισμοί της Elasticsearch. Θα μπορούσε κανείς να θεωρήσει πως το *Index* στην Elasticsearch, το οποίο είναι ο χώρος

αποθήκευσης των *Types* και των Documents, είναι το αντίστοιχο της βάσης δεδομένων στην SQL, ενώ ο τύπος (*Type*) του κειμένου-αντικειμένου θα μπορούσε να θεωρηθεί ότι είναι ο κάθε πίνακας που υπάρχει στη βάση δεδομένων και περιγράφει το σχέδιο (*schema*) της βάσης. Επομένως οι γραμμές που υπάρχουν στον κάθε πίνακα μιας SQL βάσης δεδομένων, αντιστοιχούν στα κείμενα-αντικείμενα (*Documents*) της Elasticsearch, ενώ οι στήλες στα πεδία που περιλαμβάνει το κάθε document. Στον παρακάτω πίνακα φαίνεται η σύγκριση αυτή.

SQL	*Elasticsearch*
Database	Index
Tables	Document Type
Rows	Documents
Columns	Fields

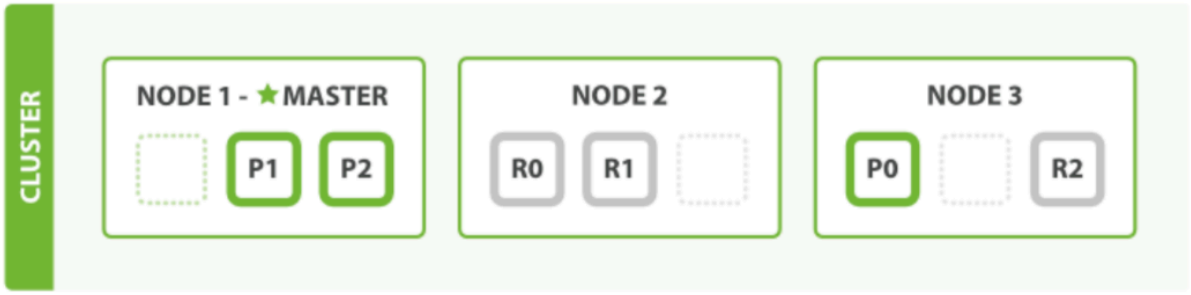
Πίνακας 2: Αντιστοίχιση ορολογίας SQL βάσης δεδομένων και της Elasticsearch

Αν κοιτάξει κάποιος την αρχιτεκτονική της Elasticsearch [41,43,44], θα παρατηρήσει πως το cluster, το οποίο καθορίζεται μοναδικά από το όνομά του, αποτελεί ένα σύνολο από κόμβους (servers). Σε ένα σύστημα μπορούν να υπάρχουν πολλά ανεξάρτητα clusters. Σε κάθε cluster είναι δυνατό να υπάρχουν πολλοί servers, οι οποίοι στην Elasticsearch έχουν την ονομασία nodes. Τα nodes συνεισφέρουν στην αποθήκευση, στο indexing και στην αναζήτηση δεδομένων.

Ένα από τα μεγαλύτερα πλεονεκτήματα της Elasticsearch είναι η διαχείριση τεράστιου όγκου δεδομένων, τα οποία μπορεί να μη χωράνε σε ένα κόμβο, ή ακόμα κι αν χωράνε να μην υπάρχει η υπολογιστική ισχύς ώστε να προσφέρει τις υπηρεσίες αναζήτησης αποδοτικά. Για την επίλυση του συγκεκριμένου προβλήματος, η Elasticsearch δίνει τη δυνατότητα στο index να διαιρεθεί σε πολλά κομμάτια, τα οποία ονομάζονται shards [43]. Κάθε shard μπορεί να αποθηκευτεί σε ένα κόμβο και αποτελεί ένα πλήρως λειτουργικό και ανεξάρτητο ευρετήριο, καθιστώντας ταυτόχρονα όλα τα δεδομένα να είναι διαθέσιμα στο cluster. Όταν εκτελούνται τα ερωτήματα (queries) σε ένα index, τα αποτελέσματα επιστρέφονται από όλα τα shards. Η Elasticsearch στέλνει το κάθε ερώτημα στα σχετικά shards και στη συνέχεια όλα τα αποτελέσματα ενώνονται μεταξύ τους για να παρουσιαστούν στο χρήστη. Όλες οι λειτουργίες (εισαγωγή, ενημέρωση και διαγραφή) γίνονται μέσα στο κομμάτι (*shard*) που περιέχει την κάθε εγγραφή. Αυτή η διαίρεση του index σε πολλά κομμάτια είναι πολύ σημαντική, αφού μέσω των shards εξασφαλίζεται η κατανεμημένη φύση του συστήματος, γεγονός που επιτρέπει την κατακόρυφη κλιμάκωση του περιεχομένου ενός index, αυξάνοντας ταυτόχρονα και την απόδοση του συστήματος.

Επιπλέον, η Elasticsearch προσφέρει τη δυνατότητα της ύπαρξης αντιγράφων ασφαλείας για τα shards. Αυτό προσφέρει στην εφαρμογή μεγάλη διαθεσιμότητα των δεδομένων της, αφού όταν το κύριο shard δεν είναι διαθέσιμο λόγω αδυναμίας του υλικού, τότε η Elasticsearch μπορεί να χρησιμοποιήσει το αντίγραφο του συγκεκριμένου shard το οποίο υπάρχει στο ίδιο cluster, αλλά σε διαφορετικό κόμβο (node). Είναι δυνατόν να υπάρχουν περισσότερα από ένα αντίγραφα για το κάθε shard. Στο κύριο shard εκτελούνται όλες οι λειτουργίες, και όταν δεν είναι διαθέσιμο, τότε οι λειτουργίες μεταφέρονται στο αντίγραφό του.

Στο πιο κάτω διάγραμμα φαίνεται ένα cluster με τρεις κόμβους (nodes). Στον κόμβο 1 είναι εγκατεστημένα τα κύρια shards P1 και P2 ενώ το άλλο κύριο shard (P0) βρίσκεται στον κόμβο 3. Όπως φαίνεται στον κόμβο 2 δεν υπάρχει κάποιο κύριο shard, παρά μόνο αντίγραφα των P0 και P1 (R0 και R1 αντίστοιχα), ενώ στον κόμβο 3 υπάρχει αντίγραφου του P2 (R2). Ο διαχωρισμός των shards σε όσο το δυνατόν περισσότερους κόμβους επιτρέπει τη χρήση του hardware του κάθε κόμβου από λιγότερα shards, επιτρέποντας την βελτίωση της λειτουργίας του κάθε shard.



Διάγραμμα 11: Cluster, Node, Shard [41,43]

3.4.1 Υπολογισμός Ομοιότητας Κειμένων – TF-IDF και Vector Space Model

Οι βάσεις δεδομένων οι οποίες στηρίζονται αποκλειστικά σε δομημένα δεδομένα όπως ημερομηνίες, αριθμούς και ομάδες λέξεων, είναι εύκολο να ελεγχθεί αν ικανοποιούν κάποιο ερώτημα απαντώντας στην ερώτηση *ναι/όχι*. Αυτός όμως ο τρόπος απάντησης δεν είναι ικανοποιητικός όταν γίνεται για αναζήτηση σε κείμενο. Οι μηχανές αναζήτησης κειμένου προσπαθούν όχι μόνο να βρουν τα κείμενα που ικανοποιούν τα ερωτήματα, αλλά ταυτόχρονα να τα ταξινομήσουν με βάση τη σχετικότητα που έχουν, δίνοντας ένα σκορ συσχετισμού με το κάθε κείμενο. Παρόλα αυτά το σκορ συσχετισμού δεν παράγεται μόνο για ερωτήματα που έχουν να κάνουν με κείμενο, αλλά και για δομημένα δεδομένα όπως αναλύεται στο επόμενο υποκεφάλαιο.

Η Apache Lucene και επομένως η Elasticsearch χρησιμοποιούν το Boolean μοντέλο με σκοπό να βρουν κείμενα τα οποία θα ταιριάζουν με το ζητούμενο, και στη συνέχεια να υπολογίσουν το πόσο σχετικό είναι το κείμενο που αναζητεί ο χρήστης, σε σχέση με τα άλλα με τη βοήθεια του TF-IDF [43,45] , το οποίο χρησιμοποιείται στην εξόρυξη κειμένων. Το πόσο σχετικό είναι ένα κείμενο σε σχέση με τα άλλα εξαρτάται από τρεις παραμέτρους, οι οποίες αναλύονται πιο κάτω.

- *Term Frequency:* Η παράμετρος αυτή αντιστοιχεί στον αριθμό εμφανίσεων μιας λέξης ή ενός όρου μέσα στο κείμενο. Όσο περισσότερες φορές εμφανίζεται, τόσο πιο σχετικό θα είναι και το κείμενο. Η συχνότητα με την οποία εμφανίζεται ο όρος t στο κείμενο d δίνεται από την πιο κάτω εξίσωση.

$$TF(t,d) = \sqrt{frequency}$$

  όπου *frequency* η συχνότητα εμφάνισης του όρου *t* στο κείμενο *d*.

- *Inverse Document Frequency:* Η παράμετρος αυτή δείχνει το πόσο συχνά ένας όρος εμφανίζεται σε όλα τα κείμενα. Όσο μεγαλύτερο το IDF τόσο μικρότερο είναι το βάρος που δίνεται στο συγκεκριμένο όρο (πχ οι όροι “το”, “ο”, “και” κτλ). Το IDF υπολογίζεται με την πιο κάτω εξίσωση.

$$IDF(t) = 1 + \ln\left(\frac{docsNum}{1 + docsCont}\right)$$

  όπου *docsNum* ο συνολικός αριθμός των κειμένων και *docsCont* ο συνολικός αριθμός των κειμένων που περιέχουν τον όρο t.

- *Field-length norm:* Η παράμετρος αυτή δείχνει το πόσο μεγάλο είναι το κείμενο που γίνεται η αναζήτηση. Όσο μικρότερο το κείμενο τόσο μεγαλύτερο θα είναι και το βάρος που θα δοθεί. Αν ένας όρος εμφανίζεται στον τίτλο ενός άρθρου, τότε το βάρος που θα έχει θα είναι μεγαλύτερο σε σχέση με το αν βρεθεί στο άρθρο. Υπολογίζεται από την πιο κάτω εξίσωση.

$$norm(d) = \frac{1}{\sqrt{docSize}}$$

  όπου *docSize* το μέγεθος του κειμένου *d.*

Στη συνέχεια οι τρεις αυτές παράμετροι πολλαπλασιάζονται ώστε να παραχθεί το βάρος που θα δοθεί στον κάθε όρο. Η εξίσωση είναι η ακόλουθη.

$$weight\,(t,d) = tf * idf * norm$$

όπου *weight* είναι το βάρος που δίνεται για τον όρο *t* στο κείμενο *d*.

Στη συνέχεια τα σκορ για τον κάθε όρο του ερωτήματος του χρήστη συνθέτουν ένα διάνυσμα (*Vector Space Model*) [46]. Κάθε αριθμός στο διάνυσμα αυτό αντιπροσωπεύει το βάρος του κάθε όρου του ερωτήματος, όπως υπολογίστηκε με την παραπάνω μέθοδο *TF-IDF,* η οποία είναι η προκαθορισμένη μέθοδος υπολογισμού των βαρών των όρων των κειμένων. Η Elasticsearch προσφέρει κι άλλες μεθόδους υπολογισμού των βαρών, όπως η Okapi-BM25, αλλά χρησιμοποιείται η TF-IDF λόγω της απλότητας της και των αξιόπιστων αποτελεσμάτων που παράγει.

Το σημαντικό πλεονέκτημα της πιο πάνω μεθόδου και της σύνθεσης του διανύσματος με τα βάρη των όρων, έχει να κάνει με το γεγονός ότι τα διανύσματα μπορούν να συγκριθούν μεταξύ τους. Μετρώντας τη γωνία μεταξύ του διανύσματος του ερωτήματος με το διάνυσμα του κειμένου, μπορεί να υπολογιστεί το πόσο σχετικά είναι τα δύο αυτά κείμενα [46]. Όσο μικρότερη είναι η γωνία μεταξύ των διανυσμάτων, τόσο μεγαλύτερη είναι η συσχέτιση τους, και επομένως τα κείμενα είναι πιο όμοια μεταξύ τους.

3.4.2 Function Score – Decay Functions

Η Elasticsearch εκτός από τον υπολογισμό του σκορ για το πόσο σχετικό είναι ένα κείμενο με κάποιο άλλο, περιλαμβάνει επίσης συναρτήσεις για να υπολογίζει ένα σκορ με βάση το πόσο κοντά βρίσκονται δύο αριθμητικές τιμές. Αυτή η λειτουργία της Elasticsearch είναι πολύ σημαντική, αφού δίνει στον χρήστη τη δυνατότητα να μην αποκλείει κάποιες επιλογές όταν δεν είναι στο εύρος τιμών που ψάχνει, αλλά να τις εμφανίζει δίνοντας τους μικρότερο σκορ, προσφέροντας του περισσότερες επιλογές.

Το πόσο σημαντική είναι αυτή η λειτουργία, μπορεί να την αντιληφθεί κάποιος, αν σκεφτεί ένα απλό παράδειγμα. Έστω ότι ένας χρήστης ψάχνει ξενοδοχεία σε απόσταση μικρότερη του ενός χιλιομέτρου από το κέντρο μιας πόλης με τιμή ανά βράδυ μικρότερη από 100 ευρώ. Το απόλυτο φιλτράρισμα με βάση τις προτιμήσεις του χρήστη, αποκλείει αποτελέσματα με τα οποία θα μπορούσε να συμβιβαστεί ο χρήστης, όπως για παράδειγμα 1.2 χιλιόμετρα και 85 ευρώ ανά βράδυ.

Το *function_score* [43,44,47] της Elasticsearch δίνει στο χρήστη τη δυνατότητα να μπορεί να συμβιβαστεί με κάποιες παραμέτρους, εμφανίζοντας ταξινομημένα τα αποτελέσματα, δίνοντας του την ευκαιρία να επιλέξει αυτό που ταιριάζει καλύτερα στις προτιμήσεις του.

Οι συναρτήσεις που υπολογίζουν το σκορ με βάση αριθμητικές τιμές ονομάζονται *decay functions* [43,44,48] και είναι τρεις· η γραμμική, όπου η καμπύλη είναι μια

ευθεία γραμμή και όλες οι τιμές οι οποίες είναι εκτός αυτής δίνουν σκορ ίσο με 0 και άρα δε θα εμφανίζονται στα αποτελέσματα, η εκθετική η οποία μειώνεται γρήγορα και στη συνέχεια ελαττώνεται και η gauss, η οποία έχει σχήμα καμπάνας και αρχικά μειώνεται αργά αλλά στη συνέχεια κάπως γρηγορότερα. Και οι τρεις συναρτήσεις έχουν τις ίδιες παραμέτρους, όπως αναλύονται πιο κάτω.

- *origin:* Το κεντρικό σημείο, ή αλλιώς η καλύτερη δυνατή τιμή για την παράμετρο. Οι τιμές που βρίσκονται όσο το δυνατό πιο κοντά στο origin, παίρνουν τιμές όσο το δυνατό πιο κοντά στο 1.

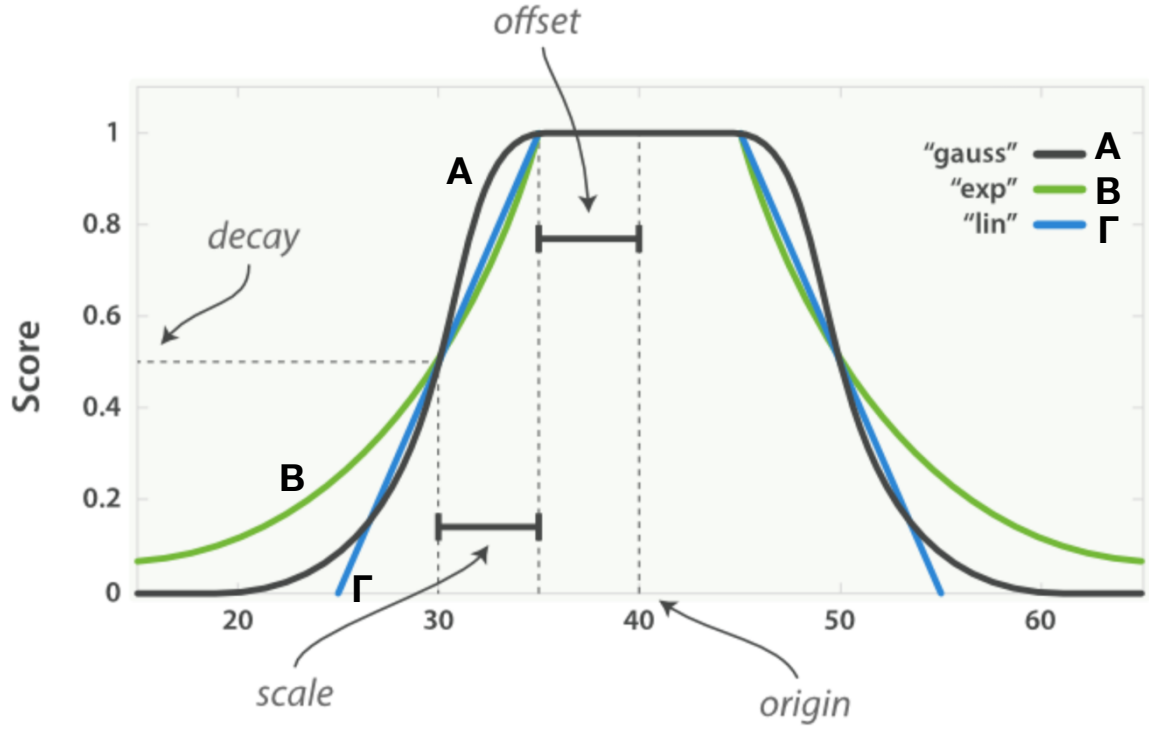
- *scale:* Ο ρυθμός με τον οποίο μειώνεται το σκορ.

- *decay (προαιρετική):* Το σκορ θα είναι 0.5 (default) όταν η παράμετρος έχει τιμή ίση με το scale.

- *offset (προαιρετική):* Ορίζοντας τιμή στο offset, επεκτείνεται το κεντρικό σημείο (origin) ώστε να λαμβάνει ένα εύρος τιμής. Επομένως, αν η παράμετρος ικανοποιεί την παρακάτω συνθήκη, το σκορ παίρνει τιμή 1.

$$origin - offset \leq παράμετρος \leq origin + offset$$

Πιο κάτω φαίνεται το διάγραμμα των τριών συναρτήσεων, μαζί με τις τέσσερις μεταβλητές που μόλις αναλύθηκαν.



Διάγραμμα 12: Εκθετική, γραμμική και συνάρτηση gauss στον υπολογισμό ομοιότητας με τη χρήση της Elasticsearch και οι τέσσερις παράμετροι [48]

Στο διάγραμμα φαίνεται πιο ξεκάθαρα πως επηρεάζει την κάθε συνάρτηση η κάθε μεταβλητή. Η τιμή του offset, η οποία είναι 5 στο συγκεκριμένο παράδειγμα, δείχνει ότι αν η παράμετρος που ψάχνει ο χρήστης είναι μεταξύ 35 και 45 (δηλ. 40, που είναι το origin ± 5), τότε το σκορ που θα επιστραφεί θα είναι 1. Επίσης φαίνεται η ταχύτητα με την οποία μειώνεται η κάθε συνάρτηση. Η τιμή του scale δείχνει πότε το σκορ θα είναι ίσο με το decay. Στο συγκεκριμένο παράδειγμα, το scale έχει τιμή 30 και το decay τιμή 0.5, η οποία είναι και η προκαθορισμένη. Αυτό σημαίνει πως όταν η τιμή που αναζητεί ο χρήστης έχει τιμή 30 τότε το σκορ που θα επιστρέψει η Elasticsearch είναι 0.5.

Η επιλογή της συνάρτησης, αλλά και των παραμέτρων που θα χρησιμοποιηθούν, σχετίζεται με το πόσο γρήγορα θέλει ο χρήστης να μειώνεται το σκορ, ανάλογα με το πόσο απομακρύνεται από την κεντρική τιμή (origin).

4

Ανάπτυξη του συστήματος

Στο παρόν κεφάλαιο αρχικά εξετάζεται ο σκοπός για τον οποίο υλοποιείται το σύστημα και έπειτα γίνεται παρουσίαση της βάσης δεδομένων του συστήματος ενώ στη συνέχεια αναλύεται η δημιουργία των datasets που χρησιμοποιήθηκαν για το validation. Ακολούθως, αναλύεται το API του συστήματος με τις λειτουργίες που εκτελεί η κάθε μέθοδος. Έπειτα, γίνεται μια παρουσίαση του τρόπου με τον οποίο το σύστημα χρησιμοποιεί τα δεδομένα εισόδου, ώστε να παραγάγει τις προβλέψεις και τις προτάσεις που θα γίνουν στο χρήστη. Τέλος, αναλύεται ο τρόπος με τον οποίο θα γίνει η επιβεβαίωση ότι τα αποτελέσματα που παράγει το σύστημα που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας είναι καλύτερα από αυτά που παράγει η Elasticsearch.

4.1 Σκοπός του συστήματος

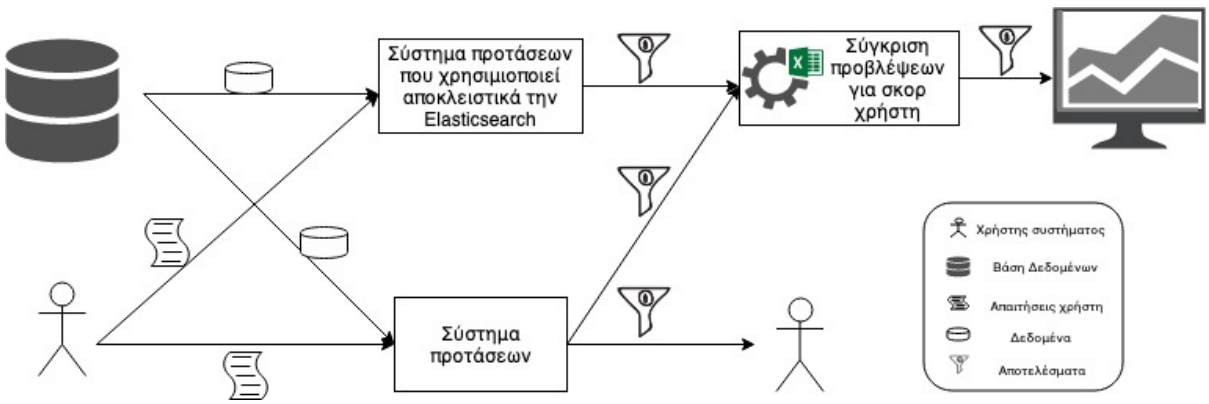
Είναι γεγονός ότι σήμερα, οι εφαρμογές είναι πρόθυμες να ανακτούν και να χρησιμοποιούν όλο και μεγαλύτερο όγκο δεδομένων. Αυτά τα δεδομένα είναι πολύ πιθανό να προέρχονται από κατανεμημένες ετερογενείς συσκευές, όπως IoT και εφαρμογές κινητών τηλεφώνων. Επομένως, υπάρχει η ανάγκη μια εφαρμογή η οποία χρησιμοποιεί αυτά τα δεδομένα, να το κάνει με γρήγορο και αποτελεσματικό τρόπο. Σε αυτό το σημαντικό πρόβλημα της διαφορετικής προέλευσης των δεδομένων, δίνει

τη λύση το Virtual Data Container (VDC). Ένα VDC παρέχει ένα αφαιρετικό επίπεδο για τους developers, το οποίο τους επιτρέπει να ασχολούνται μόνο με τα δεδομένα και στο τι θέλουν να χρησιμοποιήσουν και τι θέλουν να κάνουν. Με τη βοήθεια των VDCs, οι εφαρμογές μπορούν να έχουν άμεση πρόσβαση στα δεδομένα, τα οποία βρίσκονται στην κατάλληλη μορφή, χωρίς να υπάρχει η ανάγκη να γίνεται αναζήτηση σε διαφορετικούς παρόχους δεδομένων. Τα VDCs επιτρέπουν στους developers να υλοποιούν ευκολότερα εφαρμογές, οι οποίες διαχειρίζονται μεγάλο όγκο δεδομένων, αφού πλέον τα δεδομένα υπάρχουν σε κατάλληλη μορφή, έτοιμα για εκμετάλλευση.

Ο σκοπός για τον οποίο αναπτύχθηκε το παρόν σύστημα είναι να βοηθήσει το χρήστη, χρησιμοποιώντας τις λιγότερες δυνατές πληροφορίες, να επιλέξει το Virtual Data Container που του ταιριάζει. Ο χρήστης ο οποίος ζητάει προτάσεις από το σύστημα, καταχωρεί τις απαιτήσεις που έχει από το VDC, με σκοπό να του προταθούν κάποια blueprints, τα οποία αποτελούν την αναπαράσταση των VDCs.

Μετά την καταχώρηση των απαιτήσεων του χρήστη, γίνεται μια ανάλυση αυτών των απαιτήσεων και συγκρίνονται με τις απαιτήσεις που είχαν άλλοι χρήστες. Ακολούθως με βάση την ομοιότητα μεταξύ των χρηστών αλλά και τις βαθμολογίες που έχουν καταχωρήσει για τα blueprints που έχουν αγοράσει, παράγεται ένα σκορ για κάθε blueprint. Τέλος παρουσιάζονται όλα τα blueprints που πιθανόν να ενδιαφέρουν το χρήστη ταξινομημένα με βάση αυτό που έχει το μεγαλύτερο σκορ, και επομένως είναι πιθανότερο να τον ικανοποιήσει. Η υλοποίηση του συστήματος έχει γίνει με 2 τρόπους. Η μια υλοποίηση χρησιμοποιεί αποκλειστικά την Elasticsearch για να υπολογίσει το σκορ του κάθε blueprint, ενώ η άλλη υλοποίηση προσπαθεί με ένα απλό αλγόριθμο να βελτιώσει την απόδοση του πρώτου συστήματος.

Πιο κάτω φαίνεται διαγραμματικά το τι έχει υλοποιηθεί στην παρούσα διπλωματική εργασία.



Διάγραμμα 13: Συστήματα προτάσεων και αποτελέσματα

4.2 Βάση Δεδομένων

Στην παρούσα υποενότητα γίνεται παρουσίαση των κυριότερων στοιχείων της βάσης δεδομένων του συστήματος και στη συνέχεια παρουσιάζεται το διάγραμμα κλάσεων της βάσης δεδομένων. Τα δύο κυριότερα στοιχεία της βάσης είναι τα blueprints και οι βαθμολογίες των χρηστών για αυτά, στις οποίες περιέχονται και οι απαιτήσεις του κάθε χρήστη για το blueprint που αγόρασε και βαθμολόγησε.

4.2.1 Blueprints

Το κάθε blueprint όπως παρουσιάζεται πιο κάτω έχει ένα αναγνωριστικό (id), μερικές μετρικές ποιότητας του συστήματος (Quality of Service – QoS), όπως επίσης και κάποια πεδία που περιγράφουν το περιεχόμενο του κάθε blueprint.

Πιο κάτω παρουσιάζεται η λίστα μετρικών που χρησιμοποιήθηκαν για τη βάση δεδομένων των blueprints.

- *Accuracy*
- *Availability*
- *Process Completeness*
- *Volume*
- *Ram Gain*
- *Space Gain*
- *Average Response Time*

Οι μετρικές αυτές χρησιμοποιούνται για να παραχθούν κανονικοποιημένες τιμές, από το 0 μέχρι το 1, ώστε να μπορούν να χρησιμοποιηθούν στη συνέχεια για να συγκριθούν με τις απαιτήσεις του χρήστη. Σημειώνεται πως το κάθε blueprint περιλαμβάνει τιμές για όλα τα πιο πάνω πεδία.

Όπως αναφέρθηκε και πιο πάνω, το κάθε blueprint χρησιμοποιείται κυρίως σε κάποιες περιπτώσεις, τα οποία έχουν τη μορφή πίνακα με τιμές κειμένου. Οι δύο πίνακες που περιλαμβάνουν το περιεχόμενο του blueprint είναι οι εξής.

- *methodTags*
- *vdcTags*

Οι τιμές των δύο αυτών πεδίων χρησιμοποιούνται ώστε να παραχθεί ένα string, το οποίο στη συνέχεια θα χρησιμοποιηθεί για να εξαχθεί μια τιμή μεταξύ 0 και 1. Η

μετατροπή των δύο πινάκων σε ένα string, γίνεται με σκοπό να χρησιμοποιηθεί η δυνατότητα της Elasticsearch να αναλύει κείμενο και να παράγει ένα score με βάση την ομοιότητα.

4.2.2 Ratings

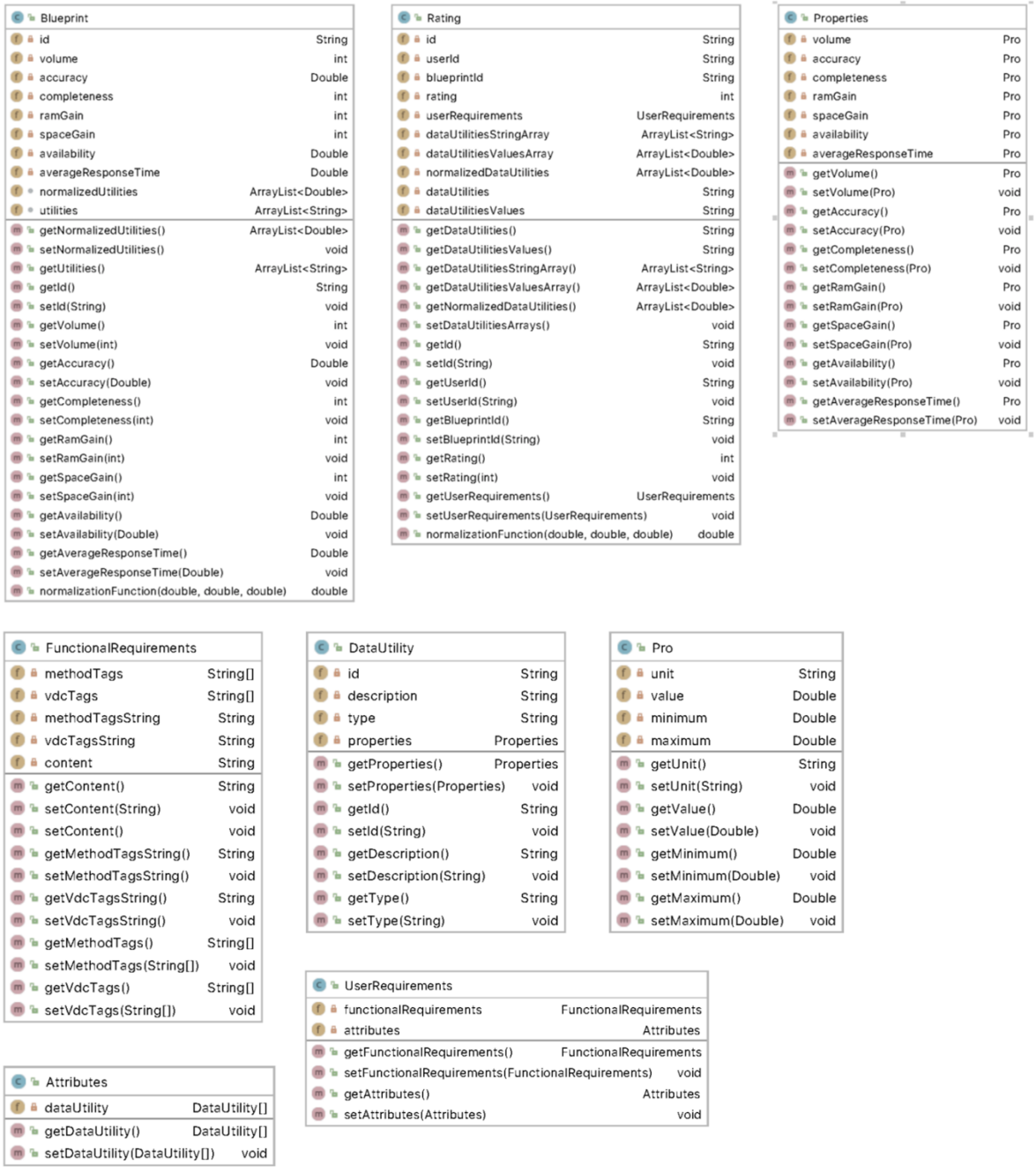
Τα συστήματα προτάσεων με collaborative filtering στηρίζονται στις βαθμολογίες που δίνουν οι χρήστες για τα προϊόντα/υπηρεσίες που έχουν αγοράσει. Όταν κάποιος χρήστης αγοράζει ένα blueprint, στη συνέχεια το βαθμολογεί. Οι βαθμολογίες που επιτρέπεται να καταχωρούνται στο σύστημα που αναπτύχθηκε είναι μεταξύ 1 και 5. Στη βάση επομένως καταχωρούνται τα εξής:

- ratingId
- userId
- blueprintId
- rating
- userRequirements

Κάθε βαθμολογία που καταχωρείται στο σύστημα λαμβάνει ένα μοναδικό αναγνωριστικό αριθμό (*ratingId*). Ο χρήστης με αναγνωριστικό χρήστη *userId* βαθμολογεί το blueprint με αναγνωριστικό αριθμό *blueprintId*, με βαθμολογία rating. Στη βάση είναι επίσης σημαντικό να καταχωρούνται κάθε φορά και οι απαιτήσεις του χρήστη που έχει βαθμολογήσει το συγκεκριμένο blueprint, αφού με βάση τις απαιτήσεις των χρηστών γίνεται η ανάλυση ώστε να παραχθούν οι προτάσεις που θα γίνουν στη συνέχεια στο χρήστη.

Τα *userRequirements* περιλαμβάνουν όλες ή μερικές από τις μετρικές που χαρακτηρίζουν τα blueprints (*attributes*). Επίσης περιλαμβάνουν και πίνακες που περιγράφουν το περιεχόμενο του blueprint που αναζητούν, το οποίο όπως και στα blueprints είναι σε μορφή πινάκων με τιμές κειμένου. Η ανάλυση που γίνεται για τα blueprints, γίνεται και για τις απαιτήσεις του χρήστη ώστε οι πίνακες να πάρουν τη μορφή κειμένου, με σκοπό να χρησιμοποιηθούν στη συνέχεια από την Elasticsearch για να γίνει η σύγκριση.

Στην επόμενη σελίδα παρουσιάζεται το διάγραμμα κλάσεων της βάσης δεδομένων του συστήματος που αναπτύχθηκε.



Διάγραμμα 14: Βάση Δεδομένων – Διάγραμμα κλάσεων

4.2.3 Elasticsearch Repository

Όλα τα στοιχεία της βάσης που περιεγράφηκαν πιο πάνω καταχωρούνται στο Elasticsearch Repository που προσφέρει το Spring Data Elasticsearch. Το Elasticsearch Repository επεκτείνει το CRUD Repository (*Create-Read-Update-Delete*) του Spring Framework δίνοντας τη δυνατότητα να υπάρχει πλήρης λειτουργικότητα των κλάσεων που καλείται να αποθηκεύσει.

Επιπλέον έχει τη δυνατότητα χρησιμοποιώντας τα ονόματα των πεδίων των αντικειμένων, να δημιουργεί αυτόματα κάποια ερωτήματα (queries) της Elasticsearch, τα οποία χρησιμοποιούνται στη συνέχεια. Ακόμη επιτρέπει τη δημιουργία ερωτημάτων στο repository, ώστε να γίνεται γρηγορότερα η εκτέλεσή τους.

4.3 Datasets

Ένα από τα μεγαλύτερα προβλήματα που παρουσιάστηκαν κατά την ανάπτυξη του συστήματος προτάσεων ήταν η έλλειψη πραγματικών δεδομένων. Για αυτό τον λόγο έγινε προσπάθεια ώστε να παραχθούν όσο το δυνατό πιο πραγματικά δεδομένα ώστε τα αποτελέσματα και τα συμπεράσματα που θα εξαχθούν να είναι έμπιστα.

Αρχικά δημιουργήθηκαν 20 τυχαία blueprints τα οποία περιλαμβάνουν όλα τα πεδία που αναφέρθηκαν πιο πάνω. Τα blueprints αυτά πήραν τυχαίες αριθμητικές τιμές για τα πεδία, καθώς και μερικά τυχαία strings όσον αφορά το περιεχόμενο του κάθε blueprint. Οι τιμές των μεταβλητών αυτών, έχουν κανονικοποιηθεί ώστε να παίρνουν τιμές μεταξύ 0 και 1.

Στη συνέχεια έχει δημιουργηθεί ένας μεγάλος αριθμός χρηστών. Οι χρήστες λαμβάνουν μόνο ένα τυχαίο αναγνωριστικό (userId) και έπειτα δημιουργούνται τυχαία πάλι οι απαιτήσεις που έχουν από το σύστημα (userRequirements), δηλαδή οι τιμές για όλες ή κάποιες από τις μεταβλητές που ψάχνουν, καθώς και κάποια strings για το περιεχόμενο του blueprint. Αυτές οι απαιτήσεις δημιουργούνται σε μορφή JSON, αφού το API που χρησιμοποιείται, απαιτεί τα δεδομένα εισόδου και εξόδου να είναι σε τέτοια μορφή. Αφού δημιουργηθεί ένας χρήστης, με την κατάλληλη επεξεργασία του αρχείου JSON, δημιουργείται το αντικείμενο του χρήστη στην Java.

Αφού δημιουργηθούν τα blueprints και οι χρήστες, στη συνέχεια πρέπει να καταχωρηθούν οι βαθμολογίες των χρηστών που έχουν αγοράσει το κάθε blueprint. Επομένως, με βάση τις απαιτήσεις του κάθε χρήστη, αποκλείονται τα blueprints που δεν τις ικανοποιούν. Αυτό γίνεται ώστε να υπάρχει όσο το δυνατό καλύτερη

προσέγγιση σε πραγματικό σύστημα όπου ο χρήστης αγοράζει το blueprint που ικανοποιεί τις απαιτήσεις του. Δεν είναι δυνατό να δημιουργηθούν δεδομένα που να μην ικανοποιούν αυτή τη συνθήκη, αφού έχει γίνει μεγάλη προσπάθεια ώστε το σύστημα να προσομοιώσει όσο το δυνατό καλύτερα τη συμπεριφορά των πραγματικών χρηστών. Αφού έχουν αποκλειστεί τα blueprints που δεν ικανοποιούν τις απαιτήσεις του χρήστη, δίνεται σε αυτά που τις ικανοποιούν μια τυχαία βαθμολογία. Η βαθμολογία αυτή δεν είναι απολύτως τυχαία, αφού υπολογίζεται μια απόσταση βάσει των χαρακτηριστικών του blueprint και των απαιτήσεων του χρήστη, και όσο πιο κοντινή είναι, τόσο καλύτερη βαθμολογία δίνεται με την απαραίτητη τυχαιότητα. Τέλος, αφού έχουν δημιουργηθεί οι χρήστες και έχουν δώσει μια βαθμολογία από το 1 μέχρι το 5 στα blueprints που ικανοποιούν τις απαιτήσεις τους, μετά την κατάλληλη επεξεργασία καταχωρούνται στη βάση δεδομένων σε μορφή αντικειμένου.

Με την πιο πάνω διαδικασία που μόλις περιεγράφηκε, γίνεται προσπάθεια να γίνει όσο το δυνατό καλύτερη προσέγγιση σε πραγματικά δεδομένα, ώστε να μπορεί να εξεταστεί η αποτελεσματικότητα του συστήματος που αναπτύχθηκε.

4.4 API

Οι ενέργειες των χρηστών κατά τη διάρκεια της αλληλεπίδρασής τους με το σύστημα αντλούνται από το API που δημιουργήθηκε μέσω των διαφόρων HTTP μεθόδων. Οι μέθοδοι αυτές χρησιμοποιούνται κάθε φορά που ο χρήστης θέλει να εκτελέσει κάποια λειτουργία. Πιο κάτω αναλύεται η λειτουργία όλων των HTTP μεθόδων που έχουν υλοποιηθεί στην εφαρμογή.

POST /ratings
Χρησιμοποιώντας την εντολή αυτή ο χρήστης μπορεί να καταχωρήσει μια λίστα βαθμολογιών κάποιων χρηστών με τις απαιτήσεις τους. Τα δεδομένα αυτά δεν παράγονται τυχαία αλλά τα καταχωρεί ο χρήστης χειροκίνητα. Τα στοιχεία αυτά τα καταχωρεί ο χρήστης σε μορφή JSON, κατά την εκτέλεση της εντολής. Αυτή η μέθοδος χρησιμοποιείται κυρίως όταν υπάρχουν πραγματικά δεδομένα, ώστε να καταχωρούνται στη βάση δεδομένων.

POST /createBlueprints_Requirements_Ratings
Αυτή η μέθοδος χρησιμοποιείται για να παραχθούν ένα τυχαίο datasets με τη μέθοδο που περιεγράφηκε στην ενότητα 4.3. Η μέθοδος αυτή δεν παίρνει κάποια δεδομένα εισόδου.

GET /ratings
Ο χρήστης εκτελεί αυτή την εντολή όταν θέλει να δει όλες τις βαθμολογίες που έχουν καταχωρηθεί στη βάση δεδομένων. Τα δεδομένα αυτά επιστρέφονται σε μορφή JSON.

GET /ratings/{id}
Αν ο χρήστης γνωρίζει το αναγνωριστικό μιας βαθμολογίας, μπορεί να ζητήσει από το API να επιστραφεί μόνο η συγκεκριμένη βαθμολογία.

DELETE /ratings
Η συγκεκριμένη μέθοδος χρησιμοποιείται για να διαγραφούν όλες οι βαθμολογίες από τη βάση δεδομένων

DELETE /ratings/{id}
Αν ο χρήστης γνωρίζει το αναγνωριστικό μιας βαθμολογίας που θέλει να διαγράψει, μπορεί να ζητήσει από το API να διαγραφεί μόνο η συγκεκριμένη βαθμολογία.

GET /elasticsearchScores
Καλώντας αυτή τη μέθοδο ο χρήστης, καταχωρεί στο σύστημα τις απαιτήσεις του, και το σύστημα προτάσεων του επιστρέφει ταξινομημένα όλα τα blueprints, με βάση το ποιο θεωρεί ότι θα αφήσει περισσότερο ικανοποιημένο το χρήστη. Επιπλέον επιστρέφεται μια τιμή, η οποία είναι η βαθμολογία που υπολογίζεται ότι θα δώσει ο χρήστης αν αγοράσει το συγκεκριμένο blueprint. Αυτή η μέθοδος έχει υλοποιηθεί με σκοπό να συγκριθεί η αξιοπιστία των αποτελεσμάτων της Elasticsearch με αυτή των αποτελεσμάτων του συστήματος που αναπτύχθηκε και θα αναλυθεί εκτενέστερα στη συνέχεια.

GET /scoreByUR
Αυτή η μέθοδος αποτελεί ουσιαστικά το σύστημα προτάσεων που έχει αναπτυχθεί στα πλαίσια της διπλωματικής εργασίας. Ο χρήστης καταχωρεί τις απαιτήσεις του σε μορφή JSON, και καλεί τη μέθοδο αυτή. Οι απαιτήσεις αυτές συγκρίνονται με τις απαιτήσεις των άλλων χρηστών, ώστε να παραχθεί η ομοιότητα μεταξύ τους, και στη συνέχεια να υπολογιστεί η βαθμολογία που πιθανόν να δώσει ο χρήστης στο κάθε blueprint. Η μέθοδος επιστρέφει ταξινομημένα τα αποτελέσματα με βάση αυτό που θεωρεί το σύστημα ότι θα ικανοποιήσει περισσότερο τις απαιτήσεις του χρήστη. Η μέθοδος αυτή αναλύεται εκτενώς σε επόμενη ενότητα.

GET /validation
Η συγκεκριμένη μέθοδος χρησιμοποιήθηκε ώστε να παραχθούν κάποια αποτελέσματα τα οποία θα βοηθούσαν στη συνέχεια στην εξαγωγή συμπερασμάτων, όσον αφορά την εγκυρότητα του συστήματος που αναπτύχθηκε, καθώς και τη βελτίωση που πιθανό να επιτεύχθηκε σε σύγκριση με τα αποτελέσματα που παρήγαγε η Elasticsearch. Αυτή

η μέθοδος έχει χρησιμοποιηθεί ώστε να γίνουν οι απαραίτητες συγκρίσεις, με σκοπό να βρεθεί η καλύτερη παραμετροποίηση της Elasticsearch, με σκοπό να παραγάγει όσο το δυνατό καλύτερα αποτελέσματα, και έτσι να συμπεράσματα να είναι αξιόπιστα. Τα αποτελέσματα που βγάζει η συγκεκριμένη μέθοδος, είναι σε μορφή csv, μορφή η οποία επιτρέπει την επεξεργασία τους και την εξαγωγή συμπερασμάτων με τη χρήση κάποιου υπολογιστικού φύλλου.

GET /kfold_validation
Χρησιμοποιώντας τη συγκεκριμένη μέθοδο, επιτυγχάνεται η εξαγωγή των αποτελεσμάτων όπως και στη μέθοδο *validation.* Η διαφορά αυτής της μεθόδου έχει να κάνει με το γεγονός ότι σε αυτή τη μέθοδο υλοποιείται το 10fold cross-validation όπως αυτό περιεγράφηκε στην ενότητα 2.5.3. Τα δεδομένα εξόδου είναι όπως και στη μέθοδο *validation* σε μορφή csv με σκοπό την ευκολότερη επεξεργασία. Η υλοποίηση της μεθόδου αυτής θα αναλυθεί εκτενώς στη συνέχεια.

4.5 Υπολογισμός βαθμολογίας με αποκλειστική χρήση της Elasticsearch

Όπως έχει αναφερθεί στην προηγούμενη ενότητα, ο χρήστης μπορεί να καλέσει τη μέθοδο *elasticsearchScores* με σκοπό να ζητήσει από το σύστημα να υπολογιστούν τα σκορ για κάθε blueprint. Το ερώτημα που αναπτύχθηκε ώστε να δίνει αποτελέσματα στο χρήστη έχει παραμετροποιηθεί με σκοπό να βρεθεί ο καλύτερος συνδυασμός συνάρτησης και ταχύτητας μείωσης του σκορ που επιστρέφει η Elasticsearch, όπως περιεγράφηκε στην ενότητα 3.4 και αναλύονται τα αποτελέσματα στο κεφάλαιο 5.

Αρχικά ο χρήστης καταχωρεί τις απαιτήσεις του σε μορφή JSON. Στη συνέχεια, για κάθε blueprint, εκτελείται το ερώτημα (query) όσες φορές έχει βαθμολογηθεί από τους χρήστες, χρησιμοποιώντας τις απαιτήσεις τους, ως παραμέτρους. Το σκορ που δίνει το ερώτημα αυτό αποτελεί την ομοιότητα μεταξύ του χρήστης που εκτελεί την εντολή και του χρήστη που έχει βαθμολογήσει το συγκεκριμένο blueprint. Αφού υπολογιστούν οι ομοιότητες του χρήστη με τους υπόλοιπους, υπολογίζεται για κάθε blueprint, το σκορ που πιθανόν να δώσει ο χρήστης στο σε αυτό. Ο υπολογισμός αυτός γίνεται με τον τύπο του σταθμισμένου μέσου όρου, όπου το βάρος για κάθε βαθμολογία είναι το σκορ που δίνει η Elasticsearch κάθε φορά ή αλλιώς ο συντελεστής ομοιότητας μεταξύ των χρηστών, και δίνεται από τον πιο κάτω τύπο.

$$blueprintScore\ (id)\ =\ \frac{\sum_{i=1}^{n}(elasticsearchScore * userRating)}{\sum_{i\ =\ 1}^{n} elasticsearchScore}$$

Μετά τον υπολογισμό του σκορ για το κάθε blueprint, επιστρέφονται τα αποτελέσματα ταξινομημένα με βάση το καλύτερο σκορ.

4.6 Υλοποίηση συστήματος προτάσεων

Στο παρόν υποκεφάλαιο παρουσιάζεται συνοπτικά η υλοποίηση του συστήματος προτάσεων. Το σύστημα που αναπτύχθηκε, προσπαθεί να εκμεταλλευτεί πολλές από τις χρήσιμες λειτουργίες που προσφέρει η Elasticsearch, και με την κατάλληλη επεξεργασία των δεδομένων, παράγει ένα σκορ για κάθε blueprint. Τα blueprints προτείνονται στο χρήστη ταξινομημένα με βάση αυτό που είναι πιθανότερο να τον αφήσει περισσότερο ικανοποιημένο.

Αρχικά παρουσιάζεται ο τρόπος με τον οποίο διαχειρίζεται το σύστημα τις απαιτήσεις του κάθε χρήστη. Στη συνέχεια αναλύεται η προσέγγιση που χρησιμοποιήθηκε στην σύγκριση των απαιτήσεων του περιεχομένου του blueprint από τους χρήστες. Ακολούθως, παρουσιάζεται η παραγωγή του συντελεστή ομοιότητας των χρηστών και τέλος αναλύεται ο τρόπος με τον οποίο υπολογίζεται η βαθμολογία την οποία προβλέπεται ότι θα δώσει ο χρήστης στο κάθε blueprint, ενώ στη συνέχεια παρουσιάζεται και το διάγραμμα δραστηριοτήτων (activity diagram) του συστήματος προτάσεων.

4.6.1 Διανυσματοποίηση των απαιτήσεων

Ο χρήστης καλώντας τη μέθοδο *scoreByUR,* θέτει σαν δεδομένα εισόδου τις απαιτήσεις που έχει από το blueprint που προτίθεται να αγοράσει. Όπως έχει αναφερθεί και πιο πάνω, ο χρήστης μπορεί να θέσει απαιτήσεις για όλα ή και για μερικά πεδία αναζήτησης. Αφού θέσει ο χρήστης τις απαιτήσεις του, στη συνέχεια ακολουθείται μια διαδικασία ώστε οι απαιτήσεις αυτές να πάρουν τη μορφή διανύσματος, το οποίο στη συνέχεια θα χρησιμοποιηθεί για να βρεθεί η ομοιότητα του με τους υπόλοιπους χρήστες.

Αρχικά δημιουργούνται δύο κενοί πίνακες. Στους δύο αυτούς πίνακες, αποθηκεύονται οι τιμές και τα ονόματα των παραμέτρων που περιλαμβάνονται στις απαιτήσεις του χρήστη σε αντίστοιχες θέσεις, ώστε να είναι ξεκάθαρο το τι αντιπροσωπεύει η κάθε τιμή. Στη συνέχεια, για κάθε παράμετρο, θεωρείται μια ελάχιστη και μια μέγιστη τιμή που μπορεί να λάβει η συγκεκριμένη παράμετρος, και με βάση αυτές και την τιμή που έχει καθορίσει ο χρήστης, υπολογίζεται μια κανονικοποιημένη τιμή η οποία παίρνει

τιμές μεταξύ του 0 και του 1. Ο τύπος που χρησιμοποιείται για την κανονικοποίηση της κάθε παραμέτρου είναι ο πιο κάτω.

$$normalizedValue = \frac{parameterValue - minValue}{maxValue - minValue}$$

Οι κανονικοποιημένες τιμές για τις παραμέτρους εισάγονται σε ένα καινούργιο πίνακα, ο οποίος περιλαμβάνει όλες τις κανονικοποιημένες τιμές. Η κανονικοποίηση αυτή των παραμέτρων γίνεται με σκοπό να δίνεται ίσο βάρος σε όλες τις παραμέτρους, χωρίς να παίζει ρόλο το πόσο μεγάλο ή μικρό είναι το εύρος τιμών της κάθε μιας. Στο τέλος της διαδικασίας που μόλις περιεγράφηκε τα διανύσματα για το χρήστη που χρησιμοποιεί το σύστημα, έχουν μια μορφή παρόμοια με την πιο κάτω.

$$stringsVector = [accuracy, availability, volume]$$
$$valuesVector = [0.80, 85, 7000]$$
$$normalizedValuesVector = [0.80, 0.85, 0.7]$$

Επιπλέον στη βάση δεδομένων του συστήματος, για όλες τις βαθμολογίες που έχουν καταχωρηθεί από τους χρήστες για τα blueprints, κανονικοποιούνται με τον ίδιο τρόπο τα διανύσματα των απαιτήσεων των χρηστών, ώστε να μπορεί εύκολα να γίνει η σύγκριση μεταξύ τους.

4.6.2 Περιεχόμενο

Όπως έχει αναφερθεί και κατά την περιγραφή της βάσης δεδομένων του συστήματος, το κάθε blueprint περιλαμβάνει δύο πίνακες οι οποίοι περιγράφουν το περιεχόμενο του. Ο χρήστης όταν καταχωρεί τις απαιτήσεις του στο σύστημα, συμπληρώνει και αυτός τους δύο πίνακες, ώστε τα blueprints που θα του προταθούν να έχουν όσο το δυνατό πιο κοντινό περιεχόμενο.

Η Elasticsearch όπως έχει αναλυθεί και στο υποκεφάλαιο 3.4 χρησιμοποιεί την Apache Lucene ώστε να μπορεί να αναλύει κείμενο. Η εξαιρετική ικανότητα αυτή της Elasticsearch, δε θα μπορούσε να μη χρησιμοποιηθεί και στο σύστημα που έχει αναπτυχθεί. Για το σκοπό αυτό οι τιμές που περιλαμβάνουν οι δύο πίνακες που περιγράφουν το περιεχόμενο, methodTags και vdcTags, συγχωνεύονται με σκοπό να παραχθεί ένα string το οποίο περιλαμβάνει όλες τις επιμέρους τιμές των πινάκων περιεχομένου σε μορφή ενιαίου κειμένου.

Κατά τη σύγκριση του string περιεχομένου του χρήστη με το string περιεχομένου των χρηστών με τους οποίους γίνεται η σύγκριση, παράγεται ένα σκορ από το ερώτημα της Elasticsearch, το οποίο παίρνει τιμές μεταξύ του 0 και του 1. Παρόλα αυτά η

ταχύτητα με την οποία μειώνεται το σκορ του ερωτήματος είναι μεγάλη, με αποτέλεσμα όταν δεν υπάρχει μεγάλη ομοιότητα μεταξύ των περιεχομένων που αναζητούν οι χρήστες, το σκορ που παράγεται είναι σχεδόν 0. Για αυτό το λόγο, χρησιμοποιήθηκε άλλη μια λειτουργία της Elasticsearch, η οποία επιτρέπει την αλλαγή του score_function, με τη συγγραφή scripts. Η συνάρτηση που χρησιμοποιηθεί με σκοπό να μειώσει την ταχύτητα μείωσης του σκορ που παράγει η Elasticsearch είναι η εξής.

$$contentScore = maxScore * \left(1 - \frac{1}{25 * \_score}\right)$$

όπου *maxScore* είναι η μέγιστη τιμή που παίρνει το σκορ, στην περίπτωση του συστήματος που υλοποιήθηκε η τιμή αυτή είναι 1, και *_score* είναι το σκορ που παράγεται χωρίς την αλλαγή στη συνάρτηση. Με αυτό τον τρόπο επιτυγχάνεται η μείωση της επίδρασης του σκορ που σχετίζεται με το περιεχόμενο, κάνοντας το λιγότερο ευαίσθητο σε αυτό. Θα μπορούσε βέβαια η συνάρτηση να μειώνει περισσότερο ή λιγότερο την ταχύτητα μείωσης του σκορ της Elasticsearch, όμως έχει επιλεχθεί η συγκεκριμένη συνάρτηση, η οποία έχει θεωρηθεί αρκετά καλή προσέγγιση στο βάρος που χρειάζεται να δοθεί στο περιεχόμενο. Η πιο πάνω συνάρτηση έχει χρησιμοποιηθεί και στην υλοποίηση της Elasticsearch, ώστε να μην δίνεται τεράστιο βάρος στο περιεχόμενο.

Μετά την παραγωγή του σκορ όπως περιεγράφηκε πιο πάνω, μπορεί να θεωρηθεί και το περιεχόμενο ως ακόμα μια παράμετρος που επηρεάζει την ομοιότητα μεταξύ των χρηστών. Επομένως στο διάνυσμα του χρήστη που χρησιμοποιεί το σύστημα για να του προταθούν τα blueprints, καταχωρείται ακόμα μια παράμετρος με το όνομα content και τιμή 1, ενώ στο διάνυσμα του χρήστη με τον οποίο γίνεται η σύγκριση καταχωρείται το σκορ που παράγεται με την πιο πάνω συνάρτηση. Με αυτό τον τρόπο, το περιεχόμενο θεωρείται ως ακόμα μια παράμετρος που έχει ίσο βάρος με τις υπόλοιπες κατά τη διαδικασία του υπολογισμού της ομοιότητας. Σε συνέχεια του παραδείγματος που δόθηκε στην ενότητα 4.6.2, τα διανύσματα μετά την προσθήκη του περιεχομένου έχουν την εξής μορφή.

$$stringsVector = [accuracy, availability, volume, content]$$
$$valuesVector = [0.80, 85, 7000, 1]$$
$$normalizedValuesVector = [0.80, 0.85, 0.7, 1]$$

4.6.3 Υπολογισμός ομοιότητας χρηστών

Όπως έχει αναφερθεί και στο υποκεφάλαιο 2.3, για τον υπολογισμό της ομοιότητας μεταξύ των χρηστών μπορούν να χρησιμοποιηθούν διάφορες μέθοδοι. Στη σύστημα

προτάσεων που έχει υλοποιηθεί, σημαντικότερο ρόλο στον υπολογισμό της ομοιότητας μεταξύ των χρηστών έχει η διαφορά στις απαιτήσεις που έχουν. Για αυτό τον λόγο καλύτερη επιλογή θα ήταν η χρήση μιας μεθόδου η οποία θα χρησιμοποιεί την απόσταση. Μία από αυτές είναι η ευκλείδεια απόσταση, η οποία χρησιμοποιήθηκε στην υλοποίηση του συστήματος, και δίνεται από τον πιο κάτω τύπο.

$$d(\boldsymbol{p}, \boldsymbol{q}) = d(\boldsymbol{q}, \boldsymbol{p}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

όπου **p** = (p_1, p_2, … , p_n) και **q** = (q_1, q_2, … , q_n). Το μέγεθος των διανυσμάτων **p** και **q** είναι ίσο με τον αριθμό των κοινών απαιτήσεων που έχουν ο χρήστης που χρησιμοποιεί το σύστημα και ο χρήστης με τον οποίο γίνεται ο υπολογισμός της ομοιότητας. Το διάνυσμα **p** αντιπροσωπεύει το διάνυσμα με τις κανονικοποιημένες τιμές των απαιτήσεων του χρήστη που χρησιμοποιεί το σύστημα ενώ το διάνυσμα **q** αντιπροσωπεύει τα διανύσματα των κανονικοποιημένων τιμών των απαιτήσεων των υπολοίπων χρηστών. Πιο κάτω δίνεται ένα παράδειγμα ώστε να γίνει ξεκάθαρος ο υπολογισμός των διανυσμάτων **p** και **q**, όπου userA είναι ο χρήστης που χρησιμοποιεί το σύστημα, και userB ο οποιοσδήποτε άλλος χρήστης με τον οποίο θα υπολογιστεί η ομοιότητα.

$$stringsVector\ (userA) = [accuracy, availability, volume, content]$$
$$valuesVector\ (userA) = [0.80, 85, 7000, 1]$$
$$normalizedValuesVector\ (userA) = [0.80, 0.85, 0.70, 1]$$

$$stringsVector\ (userB) = [accuracy, ramGain, volume, content]$$
$$valuesVector\ (userB) = [0.87, 150, 9000, 0.83]$$
$$normalizedValuesVector\ (userB) = [0.87, 0.75, 0.90, 0.83]$$

$$common = [accuracy, volume, content]$$
$$\boldsymbol{p} = [0.80, 0.70, 1]$$
$$\boldsymbol{q} = [0.87, 0.90, 0.83]$$

Στη συνέχεια, αφού υπολογιστεί η ευκλείδεια απόσταση μεταξύ των διανυσμάτων **p** και **q** χρησιμοποιώντας τον πιο πάνω τύπο, μπορεί να προσαρμοστεί ώστε να δώσει μέτρο ομοιότητας με μέγιστο τον αριθμό 1, με την πιο κάτω εξίσωση:

$$similarity = \frac{1}{1 + d(\boldsymbol{p}, \boldsymbol{q})}$$

όπου *d(**p**,**q**)* είναι η ευκλείδεια απόσταση όπως έχει υπολογιστεί πιο πάνω.

4.6.4 Υπολογισμός σκορ για κάθε blueprint

Για κάθε blueprint που υπάρχει στη βάση δεδομένων έχουν καταχωρηθεί βαθμολογίες από τους χρήστες οι οποίοι το έχουν αγοράσει στο παρελθόν. Επομένως, για να υπολογιστεί η πιθανή βαθμολογία που θα δώσει ο χρήστης στο κάθε blueprint, θα ληφθούν υπόψιν, όλες οι βαθμολογίες που έχουν καταχωρηθεί για αυτό. Ο υπολογισμός αυτός θα μπορούσε εύκολα να γίνει με τη χρήση του σταθμισμένου μέσου όρου, όπως φαίνεται από τον πιο κάτω τύπο.

$$blueprintScore\ (id)\ =\ \frac{\sum_{i=1}^{n}(similarity * userRating)}{\sum_{i\ =\ 1}^{n} similarity}$$

Παρόλα αυτά, τα αποτελέσματα που θα παραχθούν από τον πιο πάνω τύπο δεν μπορούν να θεωρηθούν αξιόπιστα. Αυτό συμβαίνει λόγω της ομοιότητας που υπολογίζεται με την Ευκλείδεια απόσταση και η οποία υπολογίζει την απόσταση μεταξύ διαφορετικού αριθμού παραμέτρων κάθε φορά. Επομένως, όπως είναι λογικό, είναι πιθανότερο όταν οι χρήστες έχουν μόνο μια κοινή παράμετρο, η ομοιότητα που έχουν αυτοί οι δύο χρήστες μεταξύ τους, να είναι αρκετά μεγάλη. Το αποτέλεσμα αυτό δεν είναι λογικό, αφού όσο πιο πολλές παραμέτρους θέτει ο χρήστης κατά την αναζήτηση, τόσο πιο στοχευμένη είναι η αναζήτησή του, και επομένως, είναι πιο πιθανό οι προτάσεις που θα του γίνουν να τον ικανοποιούν.

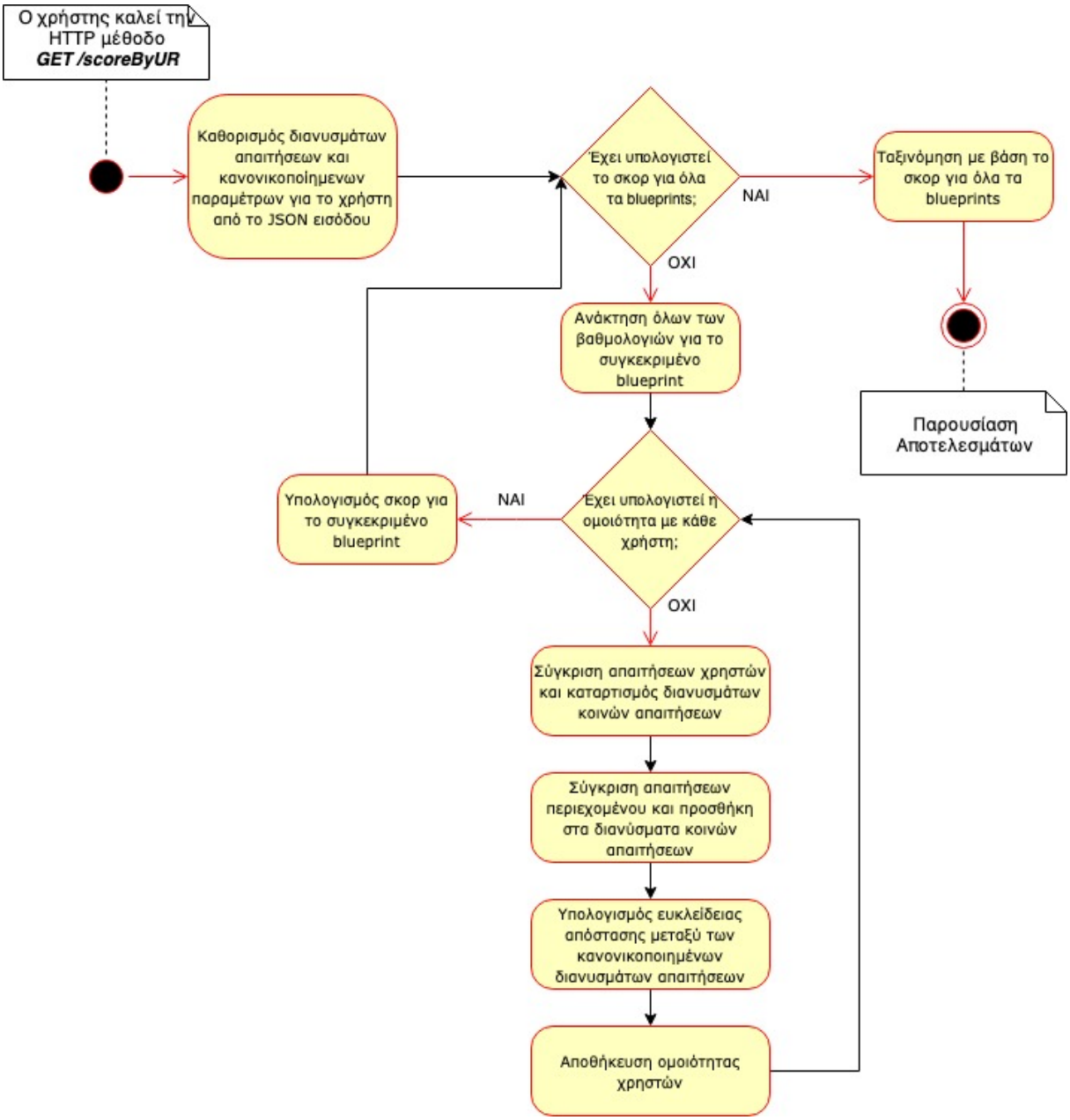
Για αυτό τον λόγο, δε θα ήταν σωστό να μην λαμβάνεται υπόψιν ο αριθμός των κοινών παραμέτρων που θέτουν οι χρήστες στον υπολογισμό του σκορ του κάθε blueprint. Με τις κατάλληλες προσαρμογές ο υπολογισμός του σκορ για το κάθε blueprint γίνεται με τον πιο κάτω τύπο.

$$blueprintScore\ (id)\ =\ \frac{\sum_{i=1}^{n}(similarity * userRating * k\ )}{\sum_{i\ =\ 1}^{n}(similarity * k)}$$

όπου *k* το μέγεθος των διανυσμάτων **p** και **q**, ή αλλιώς ο αριθμός των κοινών απαιτήσεων των χρηστών για τους οποίους γίνεται η σύγκριση.

Τέλος, μετά τον υπολογισμό των σκορ για το κάθε blueprint, παρουσιάζονται στο χρήστη ταξινομημένα με βάση το σκορ, δηλαδή αυτό που θεωρεί το σύστημα ότι θα αφήσει περισσότερο ικανοποιημένο το χρήστη.

Στη συνέχεια παρουσιάζεται το διάγραμμα δραστηριοτήτων (activity diagram) για τον υπολογισμό των σκορ για όλα τα blueprints.



Διάγραμμα 15: Activity Diagram για το σύστημα προτάσεων (GET /scoreByUR)

4.7 Παραγωγή αποτελεσμάτων για αξιολόγηση

Με την ολοκλήρωση της υλοποίησης του συστήματος προτάσεων καθώς και της διαμόρφωσης του συστήματος το οποίο παράγει σκορ χρησιμοποιώντας την Elasticsearch, μπορούν πλέον τα δύο αυτά συστήματα να χρησιμοποιηθούν ταυτόχρονα ώστε να παραγάγουν αποτελέσματα. Τα αποτελέσματα αυτά στη συνέχεια θα συγκριθούν μεταξύ τους ώστε να εξαχθούν χρήσιμα συμπεράσματα για τη συμπεριφορά τους καθώς και για την ακρίβεια που έχουν στις προβλέψεις τους.

Για το λόγο αυτό, όπως έχει αναφερθεί και στο υποκεφάλαιο 4.4, έχει υλοποιηθεί μια μέθοδος η οποία χρησιμοποιεί το kfold cross-validation για το σκοπό αυτό. Η συγκεκριμένη μέθοδος εκτελείται με τη κλήση του *GET /kfold_validation,* και έχει σαν σκοπό να παράγει μια πρόβλεψη χρησιμοποιώντας την Elasticsearch, και στη συνέχεια να παράγει άλλη μια πρόβλεψη χρησιμοποιώντας το σύστημα προτάσεων. Ακολούθως, οι δύο αυτές προβλέψεις συγκρίνονται με τη βαθμολογία που έχει δώσει ήδη ο χρήστης, ώστε να συγκριθούν και να εξαχθούν χρήσιμα συμπεράσματα.

Η παραγωγή των αποτελεσμάτων έχει γίνει χρησιμοποιώντας το 10fold cross-validation αφού όπως έχει αναφερθεί και στην ενότητα 2.5.3, στα πεδία της εξόρυξης πληροφορίας και της μηχανικής μάθησης, η πιο συνήθης επιλογή για το k στο kfold cross-validation είναι το 10 [24,25]. Το μοντέλο σε αυτή την περίπτωση ακολουθεί τη διαδικασία του training και του testing 10 φορές. Η διαδικασία που έχει ακολουθηθεί για την παραγωγή των αποτελεσμάτων παρουσιάζεται παρακάτω.

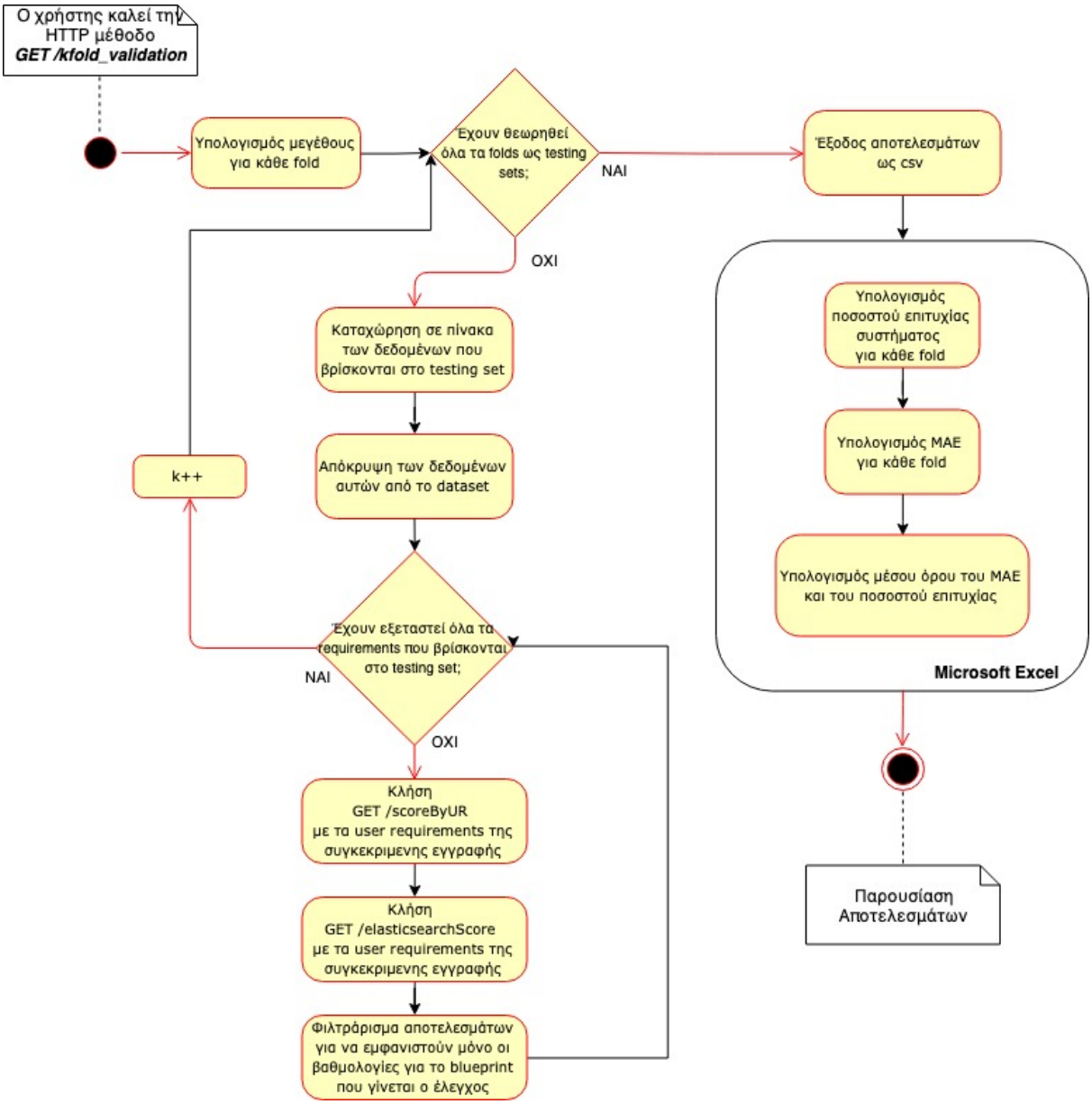
Αρχικά, χωρίζεται το dataset σε 10 ίσα ή σχεδόν ίσα μέρη, τα οποία είναι όσο το δυνατό πιο αντιπροσωπευτικά για ολόκληρο το dataset. Στη συνέχεια αποκρύπτεται το ένα από τα 10 μέρη, ώστε να μην είναι γνωστές οι βαθμολογίες που έχει δώσει ο χρήστης στα blueprints. Το υποσύνολο αυτό που δεν είναι εμφανές κατά τη διαδικασία θεωρείται το testing set, ενώ τα υπόλοιπα 9 υποσύνολα θεωρούνται το training set. Για το κάθε rating το οποίο ανήκει στο testing set, αρχικά υπολογίζεται μια πρόβλεψη για τη βαθμολογία που θα έδινε ο χρήστης με τη χρήση του συστήματος προτάσεων που έχει αναπτυχθεί. Ακολούθως, για το ίδιο rating, υπολογίζεται η βαθμολογία που θα έδινε ο χρήστης, χρησιμοποιώντας αυτή τη φορά, το σύστημα που έχει υλοποιηθεί με χρήση μόνο της Elasticsearch. Όταν θα έχουν υπολογιστεί οι βαθμολογίες για όλα τα ratings που βρίσκονται στο testing set, ένα άλλο υποσύνολο θα θεωρηθεί πλέον ως testing set, και θα παραχθούν οι βαθμολογίες για τα ratings αυτά, χρησιμοποιώντας ως training set τα υπόλοιπα 9 υποσύνολα του dataset. Η διαδικασία ολοκληρώνεται μετά από 10 επαναλήψεις, όταν όλα τα υποσύνολα θα έχουν θεωρηθεί ως το testing set.

Οι προβλέψεις που παράγονται με την πιο πάνω διαδικασία, έχουν ως στόχο την άμεση σύγκριση της απόδοσης των δύο συστημάτων. Το ένα σύστημα θεωρείται πως έχει καλύτερη απόδοση σε μια πρόβλεψη, όταν η πρόβλεψη αυτή είναι πιο κοντά στη βαθμολογία που έχει δώσει ο χρήστης στο blueprint, σε σχέση με την πρόβλεψη που έχει δώσει το άλλο σύστημα. Αυτή η μέθοδος σύγκρισης των δύο συστημάτων, μπορεί να δώσει πιο ξεκάθαρα συμπεράσματα όσον αφορά την αποδοτικότητα του συστήματος προτάσεων που υλοποιήθηκε, σε σύγκριση με την Elasticsearch.

Τα δύο συστήματα κάθε φορά υπολογίζουν την ακρίβειά τους για το testing set. Εφόσον έχουν παραχθεί οι προβλέψεις για κάθε rating με τη χρήση της πιο πάνω μεθόδου, υπολογίζεται για κάθε υποσύνολο το μέσο απόλυτο σφάλμα (MAE), καθώς και το ποσοστό των φορών που το σύστημα προτάσεων που έχει υλοποιηθεί, έχει επιτύχει πιο κοντινή πρόβλεψη, από αυτή που έχει υπολογιστεί με τη χρήση η Elasticsearch. Μετά τον υπολογισμό του μέσου απόλυτου σφάλματος (MAE) στον υπολογισμό της βαθμολογίας με τη χρήση των δύο μεθόδων για το κάθε υποσύνολο, αλλά και του ποσοστού καλύτερης απόδοσης τους, υπολογίζεται η ακρίβεια του συνολικού μοντέλου για την κάθε μέθοδο ξεχωριστά, αλλά και για το ποσοστό επιτυχίας. Η ακρίβεια του μοντέλου προκύπτει από τον μέσο όρο της ακρίβειας για την κάθε επανάληψη στα 10 υποσύνολα [25].

Η χρήση της μεθόδου 10fold cross-validation έχει γίνει με σκοπό την όσο το δυνατό περισσότερη μείωση του overfitting στα διαθέσιμα δεδομένα. Η χρήση άλλων μεθόδων validation, όπως το hold-out validation (70%-30%), είναι πολύ πιθανό να παρουσιάσει overfitting στο training set, αφού με αυτό τον τρόπο αγνοείται μεγάλο μέρος ολόκληρου του dataset το οποίο μπορεί να χρησιμοποιηθεί για training. Χρησιμοποιώντας όμως το 10fold cross-validation, επιτυγχάνεται η χρήση όλου του dataset και για training αλλά και για testing. Επομένως μειώνονται αρκετά οι πιθανότητες να παρουσιάζει overfitting το μοντέλο, καθιστώντας τα αποτελέσματα και τα συμπεράσματα αρκετά πιο αξιόπιστα.

Στην επόμενη σελίδα παρουσιάζεται το διάγραμμα δραστηριοτήτων (activity diagram) για την παραγωγή των αποτελεσμάτων για σύγκριση με τη μέθοδο 10fold cross-validation.



Διάγραμμα 16: Activity Diagram για το 10fold cross-validation (GET /kfold_validation)

Τα αποτελέσματα και τα συμπεράσματα που έχουν εξαχθεί χρησιμοποιώντας τα δύο συστήματα και το 10fold cross-validation παρουσιάζονται αναλυτικά στο επόμενο κεφάλαιο.

5

Αξιολόγηση του συστήματος

Στο παρόν κεφάλαιο γίνεται αξιολόγηση του συστήματος που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας και το οποίο περιεγράφηκε στα προηγούμενα κεφάλαια. Αρχικά, αναλύεται η παραμετροποίηση της Elasticsearch ώστε να παραγάγει όσο το δυνατό καλύτερα αποτελέσματα και στη συνέχεια τα καλύτερα αποτελέσματα που παράγονται από την Elasticsearch συγκρίνονται με τα αποτελέσματα που δίνει το σύστημα που αναπτύχθηκε. Στο κεφάλαιο αυτό παρουσιάζεται η διαδικασία που ακολουθήθηκε καθώς και τα αποτελέσματα και τα συμπεράσματα που εξήχθησαν από αυτά.

5.1 Παραμετροποίηση της Elasticsearch

Όπως έχει αναφερθεί και στο κεφάλαιο 3, στα πλαίσια της ανάπτυξης της Elasticsearch, υλοποιήθηκε ένα εργαλείο που έχει σκοπό την καλύτερη ανάλυση αριθμητικών τιμών και παραγωγή ενός σκορ με βάση αυτές. Το εργαλείο αυτό περιλαμβάνει 3 συναρτήσεις (γραμμική, εκθετική, gauss) οι οποίες παραμετροποιούνται κατάλληλα, ανάλογα με το πόσο γρήγορα θα πρέπει να μειώνεται το σκορ όσο απομακρύνεται η αριθμητική τιμή από αυτή που ψάχνει η χρήστης.

Κατά τη διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας, έγινε προσπάθεια ώστε να βρεθεί ο καλύτερος συνδυασμός των παραμέτρων αυτών, ώστε η Elasticsearch να δώσει το καλύτερο αποτέλεσμα. Για κάθε μία από τις τρεις συναρτήσεις της Elasticsearch, έχει εξεταστεί η ακρίβεια των αποτελεσμάτων, αλλάζοντας κάθε φορά το πόσο γρήγορα θα μειώνεται το σκορ. Για τέσσερα τυχαία datasets τα αποτελέσματα παρουσιάζονται πιο κάτω.

MAE	***Exp10***	***Exp25***	***Exp50***	***Ga10***	***Ga25***
Dataset1	0.7516	0.7567	0.7591	0.7511	0.7583
Dataset2	0.7726	0.7771	0.7800	0.7678	0.7766
Dataset3	0.6633	0.6700	0.6735	0.6641	0.6708
Dataset4	0.6951	0.7031	0.7075	0.6947	0.7043

MAE	***Ga50***	***Lin10***	***Lin25***	***Lin50***
Dataset1	0.7610	0.7502	0.7558	0.7597
Dataset2	0.7812	0.7502	0.7558	0.7597
Dataset3	0.6755	0.6624	0.6695	0.6741
Dataset4	0.7098	0.6932	0.7030	0.7082

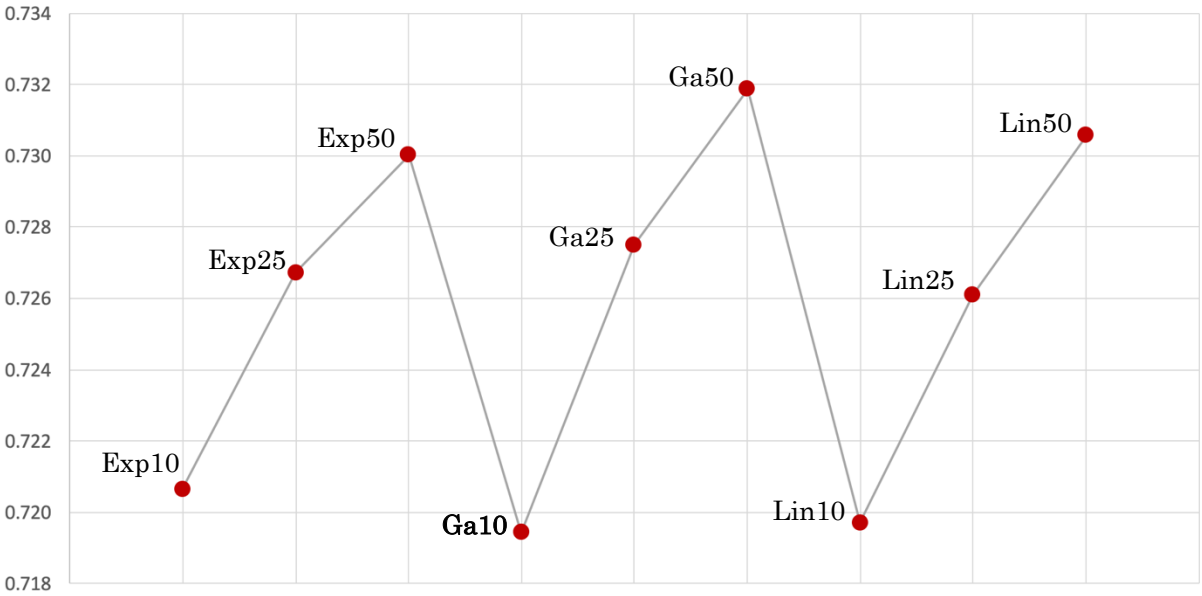
Πίνακας 3: Μέσο απόλυτο σφάλμα (MAE) για διάφορες παραμετροποιήσεις

Με βάση τα πιο πάνω αποτελέσματα υπολογίζονται οι μέσοι όροι για κάθε παραμετροποίηση, ώστε να γίνει πιο ξεκάθαρο ποια παραμετροποίησης έδωσε καλύτερα αποτελέσματα.

Παραμετροποίηση	***Μέσος όρος σφάλματος***
Exp10	0.721
Exp25	0.727
Exp50	0.730
Ga10	**0.719**
Ga25	0.728
Ga50	0.732
Lin10	0.720
Lin25	0.726
Lin50	0.731

Πίνακας 4: Μέσοι όροι απόλυτου σφάλματος (MAE)

Παρατηρώντας τον πιο πάνω πίνακα φαίνεται πως η παραμετροποίηση της Elasticsearch δεν έχει μεταβάλει δραματικά τα αποτελέσματα σε κάθε περίπτωση, παρόλα αυτά φαίνεται μια μικρή διαφορά στο σφάλμα που παράγεται σε κάθε περίπτωση. Αυτό συμβαίνει επειδή στο σύστημα υπάρχουν 8 παράμετροι αναζήτησης οι οποίες λαμβάνονται υπόψιν, και επομένως η κάθε συνάρτηση είναι ο μέσος όρος των 8 αυτών συναρτήσεων. Πιο κάτω φαίνεται διαγραμματικά το σφάλμα σε κάθε περίπτωση.



Διάγραμμα 17: Μέσοι όροι σφάλματος για κάθε παραμετροποίηση της Elasticsearch

Όπως φαίνεται στο διάγραμμα, το μικρότερο σφάλμα που παράγεται από την Elasticsearch, δίνεται όταν χρησιμοποιείται η συνάρτηση Gauss και το σκορ πέφτει στο 0.5 όταν η αριθμητική τιμή βρίσκεται 10% μακριά από την τιμή που αναζητάει ο χρήστης.

5.2 Επαλήθευση επιλογής καλύτερης παραμετροποίησης της Elasticsearch

Κατά τη διάρκεια της παραμετροποίησης της Elasticsearch με σκοπό να βρεθεί ο συνδυασμός που δίνει τα καλύτερα αποτελέσματα, παρατηρήθηκαν μικρές διαφορές μεταξύ των διαφόρων περιπτώσεων. Τελικά επιλέχθηκε η παραμετροποίηση με το μικρότερο σφάλμα, όμως λόγω των μικρών διαφορών, είναι πιθανό αυτές οι παράμετροι να μην είναι οι βέλτιστες.

Για να επαληθευτεί το γεγονός ότι επιλέχθηκε ο καλύτερος συνδυασμός των παραμέτρων της Elasticsearch ώστε να συγκριθεί με το σύστημα που αναπτύχθηκε, η σύγκριση που έγινε με όλους τους συνδυασμούς που αναφέρθηκαν πιο πάνω. Στον πιο κάτω πίνακα φαίνονται τα αποτελέσματα που παρήχθησαν κατά τη διάρκεια της σύγκρισης του συστήματος που αναπτύχθηκε με τα αποτελέσματα για κάθε παραμετροποίηση της Elasticsearch. Όπως και πριν, έτσι και τώρα, το σύστημα θεωρείται καλύτερο από την Elasticsearch όταν το σφάλμα πρόβλεψης του συστήματος είναι μικρότερο από αυτό που υπολογίζει η Elasticsearch.

Παραμετροποίηση	***Dataset1***	***Dataset2***	***Dataset3***	***Dataset4***
Exp10	61.90%	62.47%	65.24%	63.51%
Exp25	61.51%	62.19%	66.95%	65.54%
Exp50	61.90%	63.01%	68.24%	66.55%
Ga10	59.52%	61.37%	64.82%	63.51%
Ga25	61.51%	61.64%	65.67%	64.86%
Ga50	63.49%	63.01%	66.95%	65.88%
Lin10	59.52%	61.92%	64.81%	63.85%
Lin25	61.11%	61.92%	66.09%	65.20%
Lin50	63.10%	63.29%	68.24%	65.20%

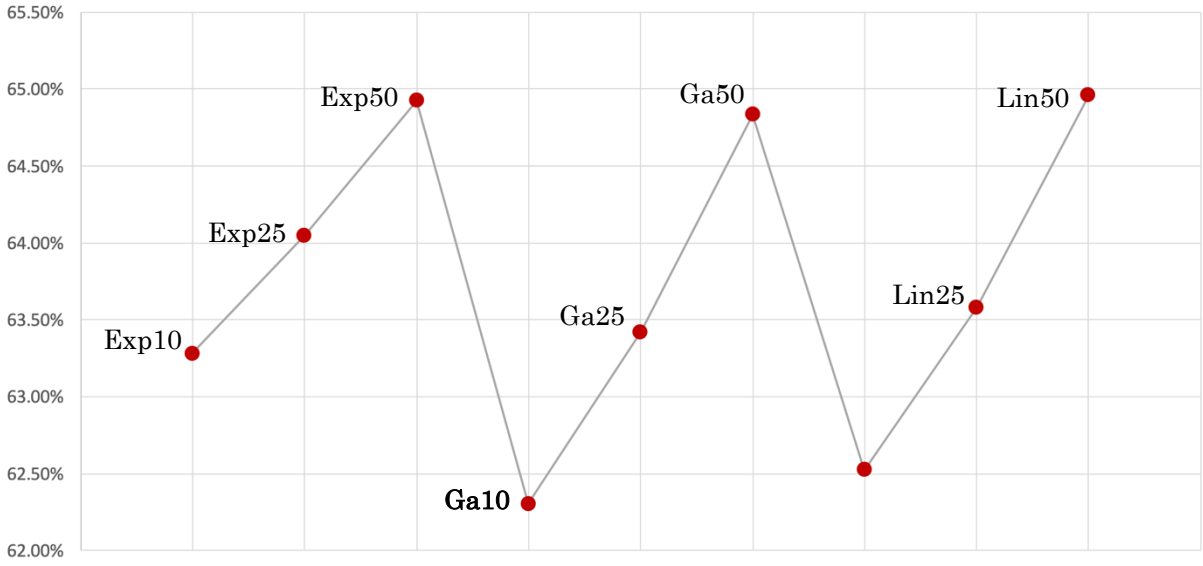
Πίνακας 5: Σύγκριση του συστήματος προτάσεων με κάθε παραμετροποίηση της Elasticsearch για την απόδοση των τεσσάρων datasets για επαλήθευση

Στον πιο κάτω πίνακα φαίνεται ο μέσος όρος των φορών που το σύστημα που αναπτύχθηκε παρήγαγε καλύτερα αποτελέσματα από την Elasticsearch και στη συνέχεια φαίνεται στα διαγράμματα ο μέσος όρος για όλα τα datasets για κάθε παραμετροποίηση.

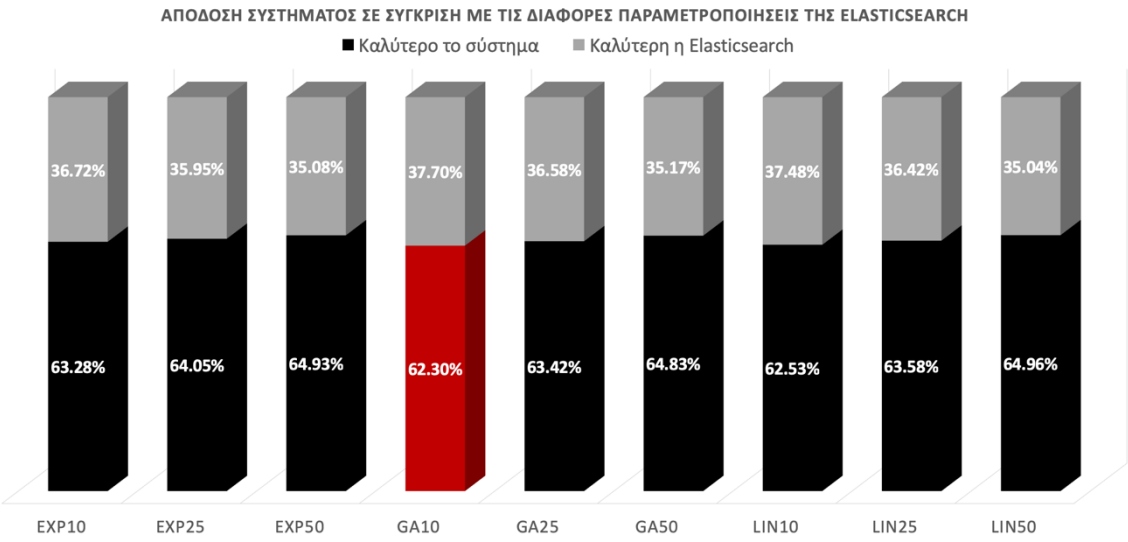
Παραμετροποίηση	***Μέσος Όρος***
Exp10	63.28%
Exp25	64.05%
Exp50	64.93%
Ga10	**62.30%**
Ga25	63.42%

Παραμετροποίηση	***Μέσος Όρος***
Ga50	64.82%
Lin10	62.53%
Lin25	63.58%
Lin50	64.96%

Πίνακας 6: Σύγκριση συστήματος προτάσεων με κάθε παραμετροποίηση της Elasticsearch για τους μέσους όρους της απόδοσης των τεσσάρων datasets για επαλήθευση



Διάγραμμα 18: Μέσοι όροι απόδοσης του συστήματος σε σύγκριση



Διάγραμμα 19: Αναλυτική σύγκριση της απόδοσης του συστήματος

Όπως φαίνεται και από τον πίνακα, αλλά και από το διάγραμμα, ο συνδυασμός της συνάρτησης Gauss με το 10% έχει δώσει τα καλύτερα αποτελέσματα για την Elasticsearch. Αυτό έχει σαν αποτέλεσμα να είναι ο πιο κατάλληλος συνδυασμός ώστε να συγκριθεί με το σύστημα που υλοποιήθηκε. Επομένως, φαίνεται πως η επιλογή που είχε γίνει πιο πάνω είναι σωστή και έτσι μπορεί να γίνει αντικειμενική σύγκριση των δύο συστημάτων.

5.3 Σύγκριση των δύο συστημάτων και επαλήθευση με 10fold cross-validation.

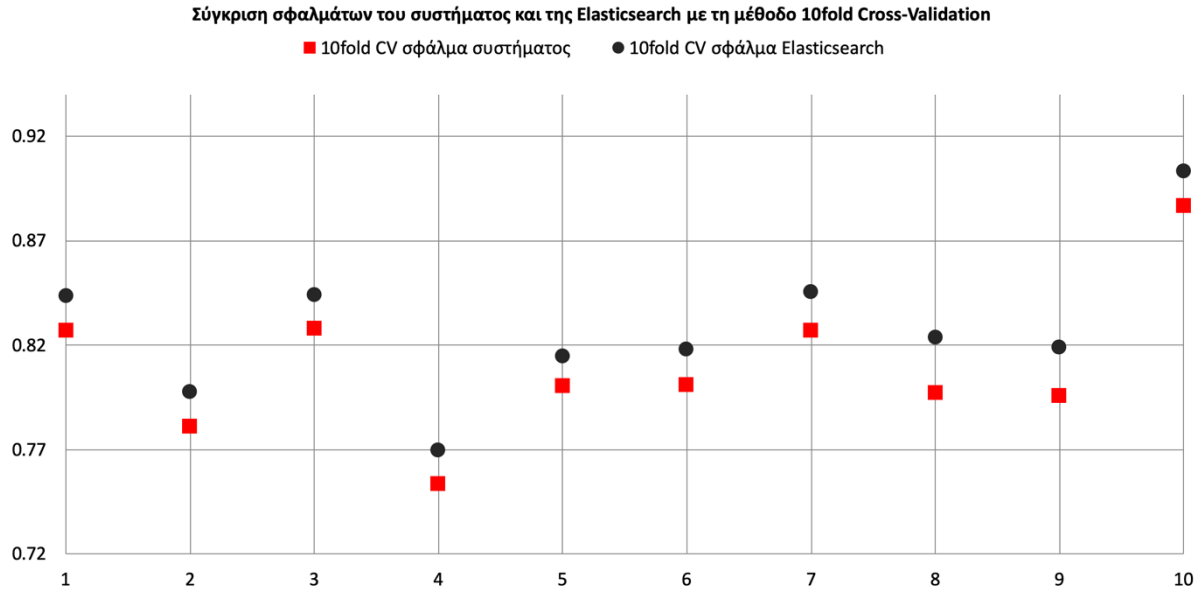
Εφόσον έχει βρεθεί και επαληθευτεί η κατάλληλη παραμετροποίηση της Elasticsearch, μπορεί να γίνει μια αξιόπιστη σύγκριση των αποτελεσμάτων ώστε να αποδειχθεί η βελτίωση της απόδοσης του συστήματος που υλοποιήθηκε σε σύγκριση με την Elasticsearch.

Για τον σκοπό αυτό έχουν δημιουργηθεί 10 αρκετά μεγάλα τυχαία datasets ώστε τα αποτελέσματα να μπορούν να θεωρηθούν πιο αξιόπιστα, και επίσης, το validation έχει γίνει με τη μέθοδο του 10fold cross validation, όπως περιεγράφηκε στο κεφάλαιο 2. Με αυτό τον τρόπο τα αποτελέσματα και τα συμπεράσματα που θα εξαχθούν θα είναι αρκετά αξιόπιστα αφού με αυτό τον τρόπο σχεδόν εξαλείφεται το overfitting και τα αποτελέσματα που δίνει το σύστημα δεν εξαρτώνται από τα δεδομένα. Στον πιο κάτω πίνακα φαίνεται το ποσοστό των φορών που το σύστημα που αναπτύχθηκε, προέβλεψε καλύτερα αποτελέσματα σε σχέση με την Elasticsearch, όπως επίσης και τα σφάλματα για την κάθε μέθοδο.

Datasets	***Καλύτερο το σύστημα***	***10fold CV σφάλμα συστήματος***	***10fold CV σφάλμα Elasticsearch***
Dataset1	59.41%	0.8270	0.8434
Dataset2	56.81%	0.7811	0.7975
Dataset3	57.58%	0.8277	0.8442
Dataset4	57.70%	0.7535	0.7697
Dataset5	58.56%	0.8004	0.8146
Dataset6	59.77%	0.8009	0.8181
Dataset7	57.55%	0.8269	0.8453
Dataset8	64.06%	0.7972	0.8235
Dataset9	59.84%	0.7955	0.8191
Dataset10	57.96%	0.8865	0.9035
Μέσος όρος	***58.87%***	***0.8097***	***0.8284***

Πίνακας 7: Σύγκριση του συστήματος προτάσεων για 10 μεγάλα datasets και παρουσίαση των σφαλμάτων με τη μέθοδο του 10fold cross-validation

Στη συνέχεια παρουσιάζεται το διάγραμμα με τη σύγκριση των σφαλμάτων του αλγορίθμου που υλοποιήθηκε και της Elasticsearch. Είναι ξεκάθαρο πως ο αλγόριθμος που υλοποιήθηκε έχει καλύτερη συμπεριφορά από την Elasticsearch αφού σε όλα τα datasets έχει επιτύχει μικρότερα σφάλματα. Η χρήση της μεθόδου 10fold cross-validation, μειώνει την πιθανότητα του overfitting του training dataset, και ως εκ τούτου μπορούν να θεωρηθούν τα αποτελέσματα εξαιρετικά αξιόπιστα.



Διάγραμμα 20: Σύγκριση των σφαλμάτων του συστήματος προτάσεων και της Elasticsearch

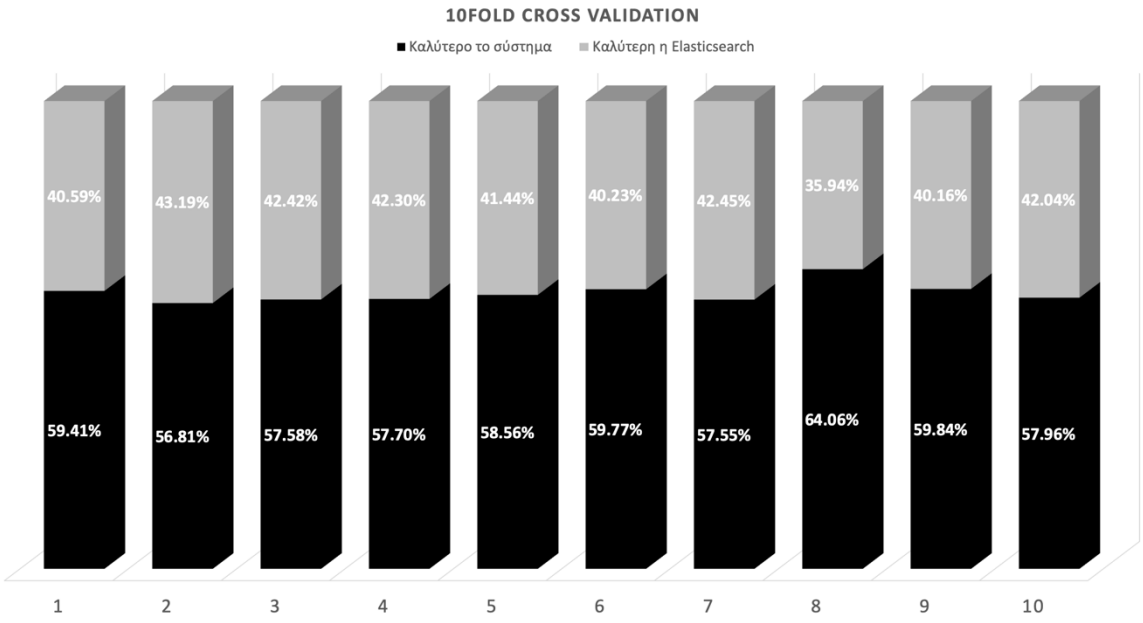
Είναι σημαντικό να μπορεί ένα σύστημα προτάσεων να μπορεί να συγκριθεί με άλλα συστήματα. Η διαφορά όμως στις τιμές που παίρνουν οι βαθμολογίες του κάθε συστήματος προτάσεων, καθιστά αυτή τη σύγκριση εξαιρετικά δύσκολη. Οπότε καλό θα ήταν να εξαχθεί ένα κανονικοποιημένο μέσο απόλυτο σφάλμα ώστε να μπορεί να γίνει σύγκριση με άλλα συστήματα προτάσεων. Όπως αναφέρθηκε και στο κεφάλαιο 2 το κανονικοποιημένο μέσο απόλυτο σφάλμα υπολογίζεται από τον πιο κάτω τύπο και παίρνει τιμές από 0 μέχρι 1.

$$NMAE = \frac{1}{n * (maxR - minR)} \sum_{i=1}^{n} |Y_i - F_i| = \frac{MAE}{(maxR - minR)} = \frac{0.8097}{5 - 1} = 0.2024$$

Η τιμή *0.2024* μπορεί να χρησιμοποιηθεί ώστε να είναι δυνατό να εξαχθεί κάποιο συμπέρασμα όσον αφορά την ικανότητα του συγκεκριμένου συστήματος προτάσεων όταν συγκριθεί με άλλα τα οποία έχουν διαφορετική κλίμακα βαθμολόγησης.

Από τον πίνακα φαίνεται πως περίπου το 59% των περιπτώσεων, προβλέπει τιμή η οποία είναι πιο κοντά στη βαθμολογία που έδωσε εν τέλει ο χρήστης. Στο πιο κάτω

διάγραμμα φαίνεται ακόμα πιο ξεκάθαρα ότι σε όλα τα datasets το σύστημα που υλοποιήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας έχει επιτύχει καλύτερες προβλέψεις σε σχέση με τις προβλέψεις που έχει παράγει η Elasticsearch.



Διάγραμμα 21: Αναλυτική σύγκριση της απόδοσης του συστήματος προτάσεων με την Elasticsearch

Λαμβάνοντας υπόψιν τους πίνακες και τα διαγράμματα που έχουν αναλυθεί πιο πάνω, φαίνεται ότι το σύστημα που υλοποιήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας, έχει βελτιώσει σε σημαντικό βαθμό την αξιοπιστία του συστήματος προτάσεων.

6

Σύνοψη

6.1 Συμπεράσματα

Όπως έχει αναφερθεί και στα προηγούμενα κεφάλαια, η Elasticsearch αποτελεί ένα εξαιρετικό εργαλείο αναζήτησης και εξόρυξης πληροφορίας. Τα εργαλεία που περιέχει τη βοηθούν ώστε να αποτελεί ένα αξιόπιστο και εξαιρετικά χρήσιμο εργαλείο. Παρά το γεγονός ότι η Elasticsearch χειρίζεται με αξιοσημείωτο τρόπο την ανάλυση και σύγκριση κειμένων, όταν φτάσει στο σημείο να συγκρίνει αριθμητικές τιμές, δεν παρουσιάζει την ίδια ικανότητα. Αυτή ήταν η υπόθεση που είχε δώσει το έναυσμα ώστε να ξεκινήσει η παρούσα διπλωματική εργασία, με σκοπό να αποδειχθεί ότι η υπόθεση αυτή είναι σωστή.

Παρατηρώντας τα αποτελέσματα που έχουν παραχθεί και αναλυθεί στο κεφάλαιο 5, φαίνεται πως το σύστημα που υλοποιήθηκε έχει καλύτερα αποτελέσματα από την Elasticsearch. Υλοποιώντας ένα σχετικά απλό αλγόριθμο για τη δημιουργία του συστήματος προτάσεων, έχει αποδειχθεί ότι το σύστημα αυτό συμπεριφέρεται αρκετά καλύτερα από ότι συμπεριφέρεται η Elasticsearch όταν καλείται να προτείνει αντικείμενα τα οποία περιέχουν και αριθμητικές τιμές. Η σύγκριση που έχει γίνει μεταξύ των δύο συστημάτων, έχει δείξει ότι περίπου το 60% των περιπτώσεων, το σύστημα που έχει υλοποιηθεί είναι καλύτερο από το σύστημα που χρησιμοποιεί αποκλειστικά την Elasticsearch.

6.2 Μελλοντικές προεκτάσεις

Το σύστημα που έχει υλοποιηθεί στα πλαίσια της παρούσας διπλωματικής εργασίας είναι γεγονός πως έχει παρουσιάσει μια σημαντική βελτίωση στην ακρίβεια των προβλέψεων των βαθμολογιών των χρηστών για τα blueprints, σε σχέση με την αποκλειστική χρήση της Elasticsearch για την υλοποίηση του συστήματος προτάσεων. Παρά το γεγονός ότι παρουσιάστηκε μια αξιοσημείωτη βελτίωση στην ακρίβεια του συστήματος, δεν παύουν να υπάρχουν περιθώρια βελτίωσης για το σύστημα προτάσεων.

Μια βελτίωση που όμως δεν εξαρτάται από το σύστημα είναι η ύπαρξη πραγματικών datasets. Η ύπαρξη πραγματικών blueprints, χρηστών, καθώς και βαθμολογιών από τους χρήστες για τα blueprints, θα έδειχνε την πραγματική βελτίωση που θα είχε το σύστημα, η οποία θα ήταν μεγαλύτερη.

Όσον αφορά τώρα τις βελτιώσεις που αφορούν το σύστημα προτάσεων, θα μπορούσαν να χρησιμοποιηθούν πιο προηγμένοι αλγόριθμοι για να παράγουν προβλέψεις για τις βαθμολογίες των χρηστών. Κάποιοι γνωστοί αλγόριθμοι που χρησιμοποιούνται στα υπάρχοντα συστήματα προτάσεων είναι ο k-nearest neighbors, το clustering καθώς και το matrix factorization. Επιπλέον, θα μπορούσαν τα δεδομένα να χρησιμοποιηθούν ως training dataset, ώστε να παραχθούν πιθανά μοτίβα που υπάρχουν με σκοπό την αποτελεσματικότερη και ακριβέστερη πρόβλεψη. Αυτό μπορεί να γίνει με χρήση αλγορίθμων μηχανικής μάθησης και νευρωνικών δικτύων, οι οποίοι έχουν αναπτυχθεί πολύ τα τελευταία χρόνια.

Επιπλέον, στο μέλλον, με την ύπαρξη πραγματικών και μεγάλου μεγέθους datasets, θα μπορεί να χρησιμοποιηθεί σε μεγαλύτερο βαθμό η κατανεμημένη μορφή της Elasticsearch, η οποία θα μπορεί να προσφέρει σημαντική βελτίωση στην ταχύτητα ανάκτησης των δεδομένων από τη βάση δεδομένων.

6.3 Επίλογος

Κατά τη διάρκεια υλοποίησης της παρούσας διπλωματικής εργασίας έχουν μελετηθεί κάποιες πολύ χρήσιμες έννοιες, όπως τα συστήματα προτάσεων και η εξόρυξη πληροφορίας. Επίσης έχουν χρησιμοποιηθεί έννοιες όπως είναι ο υπολογισμός ομοιότητας χρηστών και η ακρίβεια των προβλέψεων. Επιπλέον, κατά τη διάρκεια ανάπτυξης του συστήματος προτάσεων έχουν χρησιμοποιηθεί κάποια εξαιρετικά χρήσιμα εργαλεία που χρησιμοποιούνται ευρέως στη βιομηχανία, όπως είναι η Java και το Spring Framework. Επίσης έχει μελετηθεί σε βάθος και έχει χρησιμοποιηθεί μια

κατανεμημένη μηχανή αναζήτησης, όπως είναι η Elasticsearch. Η τεράστια αύξηση που παρατηρείται στην παραγωγή των δεδομένων κάθε χρόνο, καθιστά τη χρήση κατανεμημένων βάσεων δεδομένων αδήριτη ανάγκη. Τέλος, έχουν μελετηθεί διάφορες τεχνικές επαλήθευσης των αποτελεσμάτων, όπως είναι το hold-out validation (70%-30%), καθώς και το k-fold cross-validation, η οποία αποτελεί μια από τις πιο αξιόπιστες μεθόδους validation.

Η εμπειρίες που έχουν αποκτηθεί στα συγκεκριμένα θέματα και εργαλεία κατά τη διάρκεια της υλοποίησης της διπλωματική εργασίας, αποτελούν σημαντικά εφόδια για το μέλλον.

Βιβλιογραφία

[1]D. Reinsel, J. Gantz and J. Rydning, "Seagate: The Digitization of the World: From Edge to Core", 2018.

[2]F. Ricci, P. Kantor, L. Rokach and B. Shapira, *Recommender Systems Handbook*. Boston, MA: Springer Science+Business Media, LLC, 2011.

[3]J. Bennett and S. Lanning, "The Netflix Prize", 2007.

[4]G. Linden, B. Smith and J. York, "Amazon.com recommendations: item-to-item collaborative filtering", *IEEE Internet Computing*, vol. 7, no. 1, pp. 76-80, 2003. Available: 10.1109/mic.2003.1167344.

[5]C. Gomez-Uribe and N. Hunt, "The Netflix Recommender System", *ACM Transactions on Management Information Systems*, vol. 6, no. 4, pp. 1-19, 2015. Available: 10.1145/2843948.

[6]J. Schafer, J. Konstan and J. Riedl, *Data Mining and Knowledge Discovery*, vol. 5, no. 12, pp. 115-153, 2001. Available: 10.1023/a.

[7]D. Pennock, E. Horvitz, S. Lawrence and L. Giles, "Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-Based Approach", *Uncertainty in Artificial Intelligence Proceedings*, 2000.

[8]J. Schafer, D. Frankowski, J. Herlocker and S. Sen, "Collaborative Filtering Recommender Systems", *The Adaptive Web*, pp. 291-324. Available: 10.1007/978-3-540-72079-9_9.

[9]J. Bobadilla, F. Ortega, A. Hernando and J. Bernal, "A collaborative filtering approach to mitigate the new user cold start problem", *Knowledge-Based Systems*, vol. 26, pp. 225-238, 2012. Available: 10.1016/j.knosys.2011.07.021.

[10]D. Mladenic, "Text-learning and related intelligent agents: a survey", *IEEE Intelligent Systems*, vol. 14, no. 4, pp. 44-54, 1999. Available: 10.1109/5254.784084.

[11]R. Burke, *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331-370, 2002. Available: 10.1023/a:1021240730564.

[12]A. Lampropoulos, D. Sotiropoulos and G. Tsihrintzis, "Evaluation of a Cascade Hybrid Recommendation as a Combination of One-Class Classification and Collaborative Filtering", *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, 2012. Available: 10.1109/ictai.2012.96.

[13]S. Theodoridis and K. Koutroumbas, Pattern Recognition, 4th ed. Burlington: Elsevier Science, 2012, pp. 1-11.

[14]J. Han, M. Kamber and J. Pei, Data mining, 1st ed. Amsterdam: Elsevier/Morgan Kaufmann, 2011.

[15]U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, "From Data Mining to Knowledge Discovery in Databases", AI Magazine, vol. 17, no. 3, 1996.

[16]D. Hand, "Data MiningBased in part on the article “Data mining” by David Hand, which appeared in theEncyclopedia of Environmetrics.", Encyclopedia of Environmetrics, 2013. Available: 10.1002/9780470057339.vad002.pub2.

[17]C. Rasmussen, H. Bülthoff, B. Schölkopf and M. Giese, Pattern Recognition. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 220-227.

[18]T. Arsan, E. Koksal and Z. Bozkus, "Comparison of Collaborative Filtering Algorithms with Various Similarity Measures for Movie Recommendation", International Journal of Computer Science, Engineering and Applications, vol. 6, no. 3, pp. 1-20, 2016. Available: 10.5121/ijcsea.2016.6301.

[19]M, Senthilkumar. Calculating the User-item Similarity using Pearson's and Cosine Correlation Senthilkumar, M. 2017

[20]M. Ekstrand, "Collaborative Filtering Recommender Systems", Foundations and Trends® in Human–Computer Interaction, vol. 4, no. 2, pp. 81-173, 2011. Available: 10.1561/1100000009.

[21]K. Goldberg, T. Roeder, D. Gupta and C. Perkins, Information Retrieval, vol. 4, no. 2, pp. 133-151, 2001. Available: 10.1023/a:1011419012209.

[22]R. Kohavi, A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, 2001

[23]D. Anguita, S. Ridella and F. Rivieccio, "K-fold generalization capability assessment for support vector classifiers", Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.. Available: 10.1109/ijcnn.2005.1555964.

[24]S. Yadav and S. Shukla, "Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification", 2016 IEEE 6th International Conference on Advanced Computing (IACC), 2016. Available: 10.1109/iacc.2016.25.

[25]P. Refaeilzadeh, L. Tang and H. Liu, "Cross-Validation", Encyclopedia of Database Systems, pp. 532-538, 2009. Available: 10.1007/978-0-387-39940-9_565.

[26]B. Efron, "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation", Journal of the American Statistical Association, vol. 78, no. 382, pp. 316-331, 1983. Available: 10.1080/01621459.1983.10477973.

[27]L. Richardson and S. Ruby, RESTful web services. Beijing: O'Reilly, 2007.

[28]A. Rodriguez, "RESTful Web services", IBM Developer, 2019. [Online]. Available: https://developer.ibm.com/articles/ws-restful/.

[29]"Java Resources for Students, Hobbyists and More | Java | Oracle", Go.java. [Online]. Available: https://go.java/index.html?intcmp=gojava-banner-java-com.

[30]"Java Platform Standard Edition 8 Documentation", Docs.oracle.com. [Online]. Available: https://docs.oracle.com/javase/8/docs/.

[31]"What is Gradle?", Docs.gradle.org. [Online]. Available: https://docs.gradle.org/current/userguide/what_is_gradle.html.

[32]"Gradle User Manual", Docs.gradle.org. [Online]. Available: https://docs.gradle.org/current/userguide/userguide.html.

[33]"Spring", Spring.io. [Online]. Available: https://spring.io/.

[34]"Spring Framework", Spring.io. [Online]. Available: https://spring.io/projects/spring-framework.

[35]"Spring Boot", Spring.io. [Online]. Available: https://spring.io/projects/spring-boot.

[36]A. Wilkinson, "Spring Boot Gradle Plugin Reference Guide", Docs.spring.io. [Online]. Available: https://docs.spring.io/spring-boot/docs/current/gradle-plugin/reference/html/.

[37]"Spring Data", Spring.io. [Online]. Available: https://spring.io/projects/spring-data.

[38]"Spring Data Elasticsearch", Spring.io. [Online]. Available: https://spring.io/projects/spring-data-elasticsearch.

[39]B. Team, "Spring Data Elasticsearch", Docs.spring.io, 2019. [Online]. Available: https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/.

[40]"spring-projects/spring-data-elasticsearch", GitHub spring-projects. [Online]. Available: https://github.com/spring-projects/spring-data-elasticsearch.

[41]"Elasticsearch: RESTful, Distributed Search & Analytics | Elastic", Elastic.co. [Online]. Available: https://www.elastic.co/products/elasticsearch.

[42]"Apache Lucene", Lucene.apache.org. [Online]. Available: https://lucene.apache.org/.

[43]C. Gormley and Z. Tong, Elasticsearch - The Definitive Guide, 1st ed. O'Reilly Media, Inc, 2015.

[44]A. Paro, ElasticSearch Cookbook, 2nd ed. Packt Publishing, 2015.

[45]"Theory Behind Relevance Scoring | Elasticsearch: The Definitive Guide | Elastic", Elastic.co. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/guide/current/scoring-theory.html.

[46]P. Turney and P. Pantel, "From Frequency to Meaning: Vector Space Models of Semantics", Journal of Artificial Intelligence Research, vol. 37, pp. 141-188, 2010. Available: 10.1613/jair.2934.

[47]"Function Score Query | Elasticsearch Reference | Elastic", Elastic.co. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-function-score-query.html

[48]"The Closer, The Better | Elasticsearch: The Definitive Guide | Elastic", Elastic.co. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/guide/current/decay-functions.html.