ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αυτόματη θεματική κατηγοριοποίηση κειμένου με χρήση ευφυών τεχνικών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΡΓΥΡΩ ΠΑΠΑΣΠΥΡΟΥ

Επιβλέπων : Ανδρέας - Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων : Ελένη Βάθη
Υποψήφια Διδάκτωρ Ε.Μ.Π.

Αθήνα, Ιούλιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αυτόματη θεματική κατηγοριοποίηση κειμένου με χρήση ευφυών τεχνικών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΡΓΥΡΩ ΠΑΠΑΣΠΥΡΟΥ

Επιβλέπων : Ανδρέας - Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων : Ελένη Βάθη
Υποψήφια Διδάκτωρ Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 20στη Ιουλίου 2018

.....	.....	.....
Ανδρέας Σταφυλοπάτης	Παναγιώτης Τσανάκας	Γιώργος Στάμου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2018

.....

Παπασπύρου Αργυρώ

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

Περίληψη

Η αυτόματη κατηγοριοποίηση κειμένου είναι η επιστήμη που προσπαθεί να επιλύσει το πρόβλημα της κατηγοριοποίησης ενός κειμένου. Από τις πιο αποτελεσματικές μεθόδους κατηγοριοποίησης είναι η ανάπτυξη συστημάτων επιβλεπόμενης μάθησης. Ταυτόχρονα, τα τελευταία χρόνια η ανάπτυξη των κοινωνικών δικτύων και των διαδικτυακών κοινοτήτων παράγουν δεδομένα τεράστιου όγκου που αυξάνουν την ανάγκη για τη δημιουργία συστημάτων που θα τα κατηγοριοποιούν αυτόματα.

Στην παρούσα εργασία θα μελετηθούν διαφορετικές τεχνικές και συστήματα επιβλεπόμενης μάθησης σε μεγάλο όγκο δεδομένων, που έχουν ληφθεί από τη διαδικτυακή σελίδα ερωτο-απαντήσεων Stack Overflow, με σκοπό να δούμε ποια παράγει τα καλύτερα αποτελέσματα.

Λέξεις κλειδιά: Κατηγοριοποίηση, μηχανική μάθηση, επιβλεπόμενη μάθηση, επεξεργασία φυσικής γλώσσας, Stack Overflow.

Abstract

Automatic text categorization is the science that attempts to resolve the problem of categorizing a text. One of the most effective categorization methods is the development of supervised learning systems. At the same time, in recent years, the development of social networks and online communities is generating massive data that increases the need for systems that automatically categorize them.

In this paper we will study different techniques and systems of supervised learning in a large volume of data, collected from the Stack Overflow questions and answers website, to see what produces the best results.

Key words: Classification, machine learning, supervised learning, natural language processing, Stack Overflow.

Ευχαριστίες

Ευχαριστώ τον καθηγητή Ανδρέα Σταφυλοπάτη και το Εργαστήριο Ευφυών Συστημάτων για την δυνατότητα που μου δόθηκε να εργαστώ πάνω στο συγκεκριμένο θέμα. Ιδιαίτερα θα ήθελα να ευχαριστήσω την Ελένη Βάθη για την απεριόριστη βοήθεια της, τις συμβουλές και την υπομονή της.

Ιδιαίτερες ευχαριστίες στο κύριο Κώστα Ναδάλη, για τις συμβουλές που μου έδωσε σε όλη την διαδικασία πραγματοποίησης της παρούσας εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω τα άτομα που στάθηκαν δίπλα μου και με συντρόφευσαν όλα αυτά τα χρόνια. Ιδιαίτερα θα ήθελα να ευχαριστήσω τα συντρόφι@ και φίλους μου Γ. Γρ., Ει. Δ., Α. Ζ., Κ. Ζ., Η. Λ., Κ. Ντ., Ν. Π., Β. Τζ., Ν. Τζ., Α. Τσ., Δ. Δ., Μ. Ζ., Π. Φ., Θ. Π., Γ. Κ. και M.S. Ξεχωριστές ευχαριστίες στο φίλο μου Θ. Μ., για τη στήριξη του σε όλες τις δυσκολίες των τελευταίων χρόνων.

Περιεχόμενα

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

1. ΕΙΣΑΓΩΓΗ

1.1. Τεχνητή Νοημοσύνη και Μηχανική Μάθηση

1.1.1. Τεχνητή Νοημοσύνη

Για πολλά χρόνια ο όρος τεχνητή νοημοσύνη ήταν συνυφασμένος με την επιστημονική φαντασία.. Πλέον είναι η ερώτηση "Τι είναι η τεχνητή νοημοσύνη?" αναζητείται καθημερινά στο Google από ανθρώπους που συναντούν τον όρο παντού μπροστά τους. Θέλοντας να δώσουμε έναν επίσημο και κατανοητό ορισμό θα μπορούσαμε να ορίσουμε ως τεχνητή νοημοσύνη τον κλάδο της επιστήμης υπολογιστών που ασχολείται με την σχεδίαση και την υλοποίηση υπολογιστικών συστημάτων που μπορούν να μιμούνται στοιχεία και λειτουργίες της ανθρώπινης νόησης.

Αν και η τεχνητή νοημοσύνη εμφανίζεται ως όρος στους μύθους της αρχαιότητας, με χαρακτηριστικό παράδειγμα τον Τάλο, γονέας της τεχνητής νοημοσύνης θεωρείται ο Άλαν Τούρινγκ που με τις θεωρίες του έβαλε τα θεμέλια για την ανάπτυξη του κλάδου. Ο τομέας της τεχνητής νοημοσύνης "κοιτάει το μέλλον" και φαίνεται να είναι η επιστήμη εκείνη που θα καθορίσει την πορεία των επόμενων γενεών της ανθρωπότητας.

Η μελέτη των μηχανισμών μάθησης βρίσκεται στον πυρήνα της ανάπτυξης της τεχνητής νοημοσύνης. Προκειμένου να μπορέσει ένα μηχάνημα να μάθει και να προσομοιώσει ανθρώπινη συμπεριφορά δεν αρκεί απλώς να έχει αρκετά, αλλά ανεπεξέργαστα, δεδομένα. Η κατηγοριοποίησης αυτών των δεδομένων, η αποτύπωση των μεταξύ τους σχέσεων και οι ιδιότητές τους αποτελούν βάση της διαδικασίας αυτής. Για αυτόν τον λόγο έχει αναπτυχθεί ένας ξεχωριστός κλάδος, αυτός της μηχανικής μάθησης.

1.1.2. Μηχανική Μάθηση

Η μηχανική μάθηση είναι ένας κλάδος της επιστήμης υπολογιστών που χρησιμοποιεί στατιστικές τεχνικές για να δώσει σε υπολογιστές την δυνατότητα να μάθουν. Με τον όρο "δυνατότητα να μάθουν¨, εννοούμε τη δυνατότητα να βελτιώσουν, προοδευτικά, την απόδοση τους σε μια συγκεκριμένη εργασία. Η μηχανική μάθηση εστιάζει στην ανάπτυξη προγραμμάτων που έχουν πρόσβαση σε δεδομένα και τα χρησιμοποιούν για να μάθουν.

Ο όρος μηχανική μάθηση επινοήθηκε από τον Arthur Samuel, πρωτοπόρο στον κλάδο της τεχνητής νοημοσύνης, το 1959. Σύμφωνα με αυτόν, "η μηχανική μάθηση είναι το πεδίο μάθησης που δίνει στους υπολογιστές την δυνατότητα να μάθουν, χωρίς να έχουν προγραμματιστεί αναλυτικά". Ένας μάλλον λιγότερα ασαφής ορισμός δόθηκε το 1998, από τον Tom Mitchell, επιστήμονα υπολογιστών. Ο Tom Mitchell όρισε την μηχανική μάθηση ως εξής:

"Ένα υπολογιστικό πρόγραμμα λέγεται ότι μαθαίνει από την εμπειρία Ε όσον αφορά μια εργασία Τ και ένα μέτρο επίδοσης Ρ, αν η απόδοση του στο Τ, όπως αυτή μετράται από το Ρ, βελτιώνεται με την εμπειρία του από το Ε."

Η μηχανική μάθηση ξεκινάει με τη συγκέντρωση δεδομένων ή παρατηρήσεων, ώστε να εντοπιστούν πρότυπα στα δεδομένα που θα επιτρέψουν στα υπολογιστικά προγράμματα να πάρουν καλύτερες αποφάσεις στο μέλλον, με βάση τα δεδομένα που τους παρέχουμε. Ο πρωταρχικός στόχος είναι να καταφέρουμε να εκπαιδεύσουμε τους υπολογιστές ώστε να μαθαίνουν μόνοι τους, χωρίς ανθρώπινη παρέμβαση ή βοήθεια, και να επαναπροσδιορίζουν τις πράξεις τους ανάλογα με τα αποτελέσματα. Για να επιτευχθεί αυτό, δίνεται στο πρόγραμμα ένα σύνολο εκπαίδευσης και το πρόγραμμα εκπαιδεύεται πάνω σε αυτό.

1.2. Κατηγοριοποίηση

Κατηγοριοποίηση είναι η διαδικασία της ανάθεσης σε ένα αντικείμενο μίας ή περισσότερων προκαθορισμένων κατηγοριών. Ένας πιο επίσημος ορισμός είναι ο εξής:

Κατηγοριοποίηση είναι η εργασία της εκπαίδευσης μίας συνάρτησης f, η οποία αντιστοιχεί κάθε σύνολο ιδιοτήτων x σε μία προκαθορισμένη ετικέτα y.

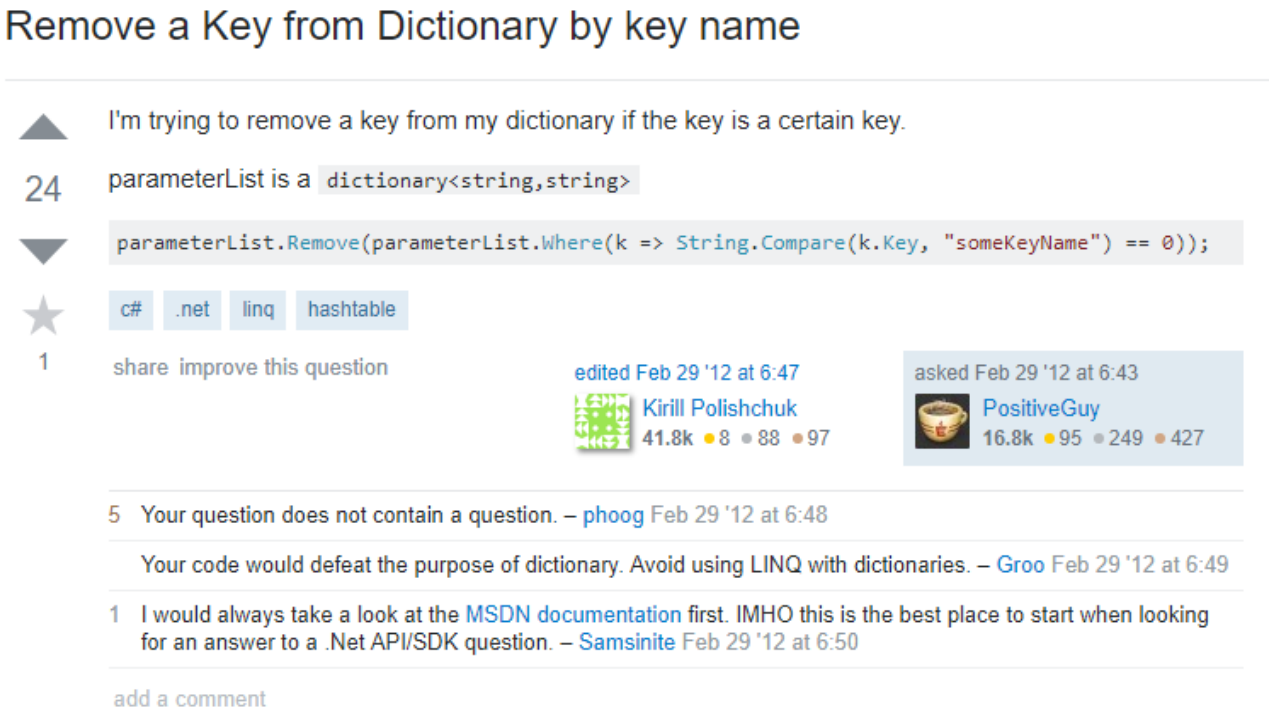
Η συνάρτηση f είναι γνωστή και ως μοντέλο κατηγοριοποίησης. Το μοντέλο κατηγοριοποίησης χρησιμοποιείται για τη διάκριση αντικειμένων διαφορετικών κατηγοριών. Επίσης χρησιμοποιείται για να προβλέψει την κατηγορία καινούργιων, αγνώστων αντικειμένων. Μπορούμε να θεωρήσουμε, δηλαδή, το μοντέλο κατηγοριοποίησης ως ένα μαύρο κουτί που αυτόματα αναθέτει μια κατηγορία σε καινούργια δεδομένα, χρησιμοποιώντας ως στοιχεία τα δεδομένα που έχει ήδη κατηγοριοποιήσει. Η κατηγοριοποίηση είναι πολύ αποτελεσματική όταν καλείται να κατηγοριοποιήσει δεδομένα σε διττές ή αυστηρά διακριτές κατηγορίες, όπως αν ένα ζώο είναι θηλαστικό, πτηνό, ψάρι ή αμφίβιο, αλλά αναποτελεσματικό όταν πρέπει να κατηγοριοποιήσει κάτι σε σειριακές κατηγορίες, όπως αν ένας άνθρωπος είναι κοντός, ψηλός ή μετρίου αναστήματος.

Η κατηγοριοποίηση είναι μια συστηματική προσέγγιση στη δημιουργία ενός μοντέλου κατηγοριοποίησης από ένα σύνολο δεδομένων που παρέχεται ως είσοδος. Οι τεχνικές για την ανάπτυξη ταξινομητών, με σκοπό την κατηγοριοποίηση δεδομένων από ένα αρχικό σύνολο, είναι ποικίλες. Για παράδειγμα τα νευρωνικά δίκτυα, τα δένδρα αποφάσεων, οι γραμμικοί ταξινομητές, οι ταξινομητές naive Bayes είναι όλα παραδείγματα διαφορετικών τεχνικών για τη λύση των προβλημάτων κατηγοριοποίησης. Κάθε τεχνική χρησιμοποιεί διαφορετικούς αλγορίθμους μάθησης, η επιλογή της κατάλληλης τεχνικής εξαρτάται από το είδος του προβλήματος και των δεδομένων. Σκοπός είναι η δημιουργία μοντέλων που θα προβλέπουν με ακρίβεια την κατηγορία νέων δεδομένων.

Η διαδικασία της κατηγοριοποίησης αρχικά χρειάζεται ένα σύνολο εκπαίδευσης (training set), που θα περιέχει τα δεδομένα των οποίων οι κατηγορίες είναι γνωστές. Το σύνολο εκπαίδευσης (training set) χρειάζεται για να χτιστεί το μοντέλο κατηγοριοποίησης. Αφού δημιουργηθεί το μοντέλο θα δοκιμαστεί σε ένα σύνολο ελέγχου (test set), το οποίο αποτελείται από δεδομένα των οποίων οι κατηγορίες δεν είναι γνωστές στο μοντέλο. Η διαδικασία αξιολόγησης του μοντέλου βασίζεται στον αριθμό των δεδομένων από το δοκιμαστικό μοντέλο των οποίων τις κατηγορίες θα καταφέρει να προβλέψει σωστά.

1.3. Stack Overflow

Το Stack Overflow είναι ένας διαδικτυακό φόρουμ υποβολής ερωτήσεων και απαντήσεων πάνω στον κλάδο της πληροφορικής. Δημιουργήθηκε το Σεπτέμβριο του 2008 και πλέον μετράει περισσότερους από 4.000.000 εγγεγραμμένους χρήστες και πάνω από 10.000.000 ερωτήσεις. Η λογική του φόρουμ είναι ότι αποτελεί ένα χώρο μετάδοσης πληροφοριών, όχι ένα κοινωνικό δίκτυο. Χαρακτηριστικό του φόρουμ αυτού είναι ότι οι χρήστες θέτουν ερωτήσεις πάνω σε πραγματικά προβλήματα που μπορεί να αντιμετωπίζουν και παίρνουν απαντήσεις από άλλους χρήστες, που απαντούν με βάση τις γνώσεις τους και την εμπειρία τους. Το φόρουμ λειτουργεί, σε μεγάλο βαθμό, με τη λογική της αυτό-ρύθμισης, αφού οι χρήστες επιλέγουν την καλύτερη απάντηση σε κάθε ερώτηση ψηφίζοντας ανάμεσα στις διαθέσιμες απαντήσεις.

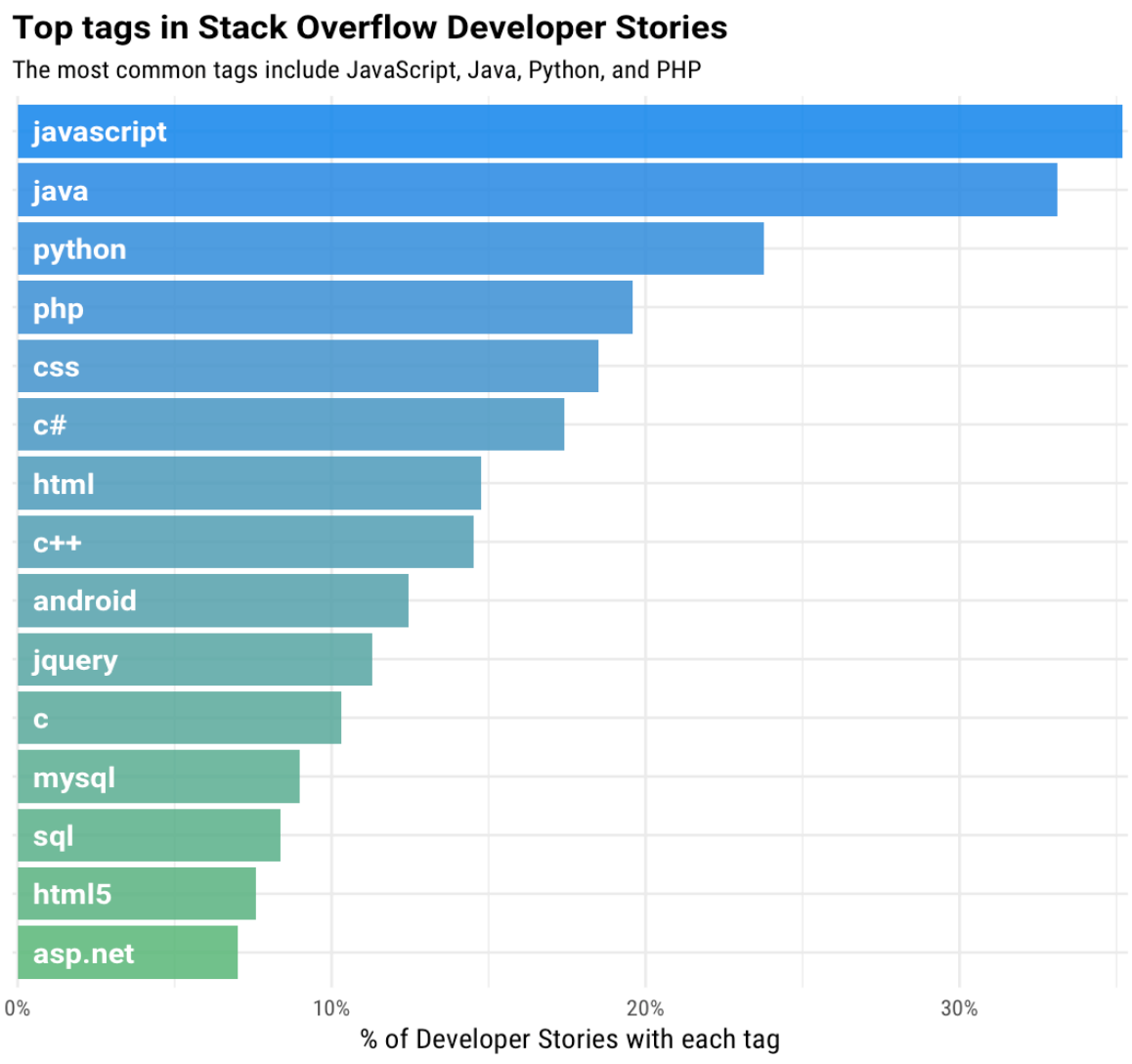


(Μια τυπική ερώτηση στο Stack Overflow)

Παραπάνω βλέπουμε μια τυπική ερώτηση στο Stack Overflow. Κάθε ερώτηση στο Stack Overflow έχει τα εξής χαρακτηριστικά:

- Τον τίτλο της ερώτησης
- Το κείμενο ερώτησης, όπου ο χρήστης επεξηγεί την ερώτηση που έθεσε στον τίτλο.
- Την βαθμολογία της ερώτησης.
- Μία ή περισσότερες κατηγορίες, στις οποίες ο χρήσης κατατάσσει την ερώτηση που έχει θέσει. Οι κατηγορίες μπορεί να είναι η γλώσσα προγραμματισμού στην οποία αναφέρεται η ερώτηση, η δομή που χρησιμοποιείται, το framework, ή ακόμα και το στυλ του προγραμματισμού που χρησιμοποιείται (πχ agile).
- Μία ή περισσότερες απαντήσεις, από άλλους χρήστες, μαζί με τη βαθμολογία τους.

Προκειμένου να δημιουργήσει ένα σύστημα αξιολόγησης το Stack Overflow έχει εφαρμόσει τη λογική του rating. Κάθε ερώτηση και απάντηση μπορεί να αξιολογηθεί από κάθε άλλον εγγεγραμμένο χρήστη μέσω του συστήματος του υπερψήφισης (upvoting) και καταψήφισης (downvoting). Οι απαντήσεις κάτω από μία ερώτηση είναι ταξινομημένες, σε φθίνουσα σειρά, με βάση τις ψήφους τους. Μέσω της συλλογικής γνώσης των χρηστών



(Οι 10 πιο δημοφιλείς κατηγορίες στο Stack Overflow)

και του συστήματος της ψήφισης (voting), το Stack Overflow προσπαθεί να διασφαλίσει ότι οι σωστές απαντήσεις θα είναι και αυτές που θα εμφανίζονται πρώτες σε μια ερώτηση.

Οι κατηγορίες (tags) που χρησιμοποιούνται έχουν σκοπό να κάνουν πιο εύκολη την εύρεση και κατηγοριοποίηση των ερωτήσεων. Κάθε ερώτηση μπορεί να έχει μέχρι και 5 διαφορετικές κατηγορίες, αφού μια ερώτηση μπορεί να αναφέρεται σε πολλά διαφορετικά θέματα. Το Stack Overflow έχει πάνω από 50,000 διαφορετικές κατηγορίες, ενώ αφήνει τους χρήστες – υπό περιπτώσεις - να δημιουργήσουν και καινούργιες κατηγορίες.

Οι ερωτήσεις στο Stack Overflow παρουσιάζουν την εξής δυσκολία. Είναι ερωτήσεις που θέτονται από πραγματικούς ανθρώπους. Αυτό σημαίνει ότι προέρχονται από ανθρώπους με διαφορετικά backgrounds, από ανθρώπους που κάνουν διαφορετική χρήση της γλώσσας, από ανθρώπους με διαφορετικά επίπεδα εκπαίδευσης. Ακόμα πρέπει να λάβουμε υπόψιν, ότι παρόλο που η επίσημη γλώσσα που χρησιμοποιείται στο Stack Overflow είναι τα αγγλικά, οι άνθρωποι που θέτουν τις ερωτήσεις μπορεί να προέρχονται από οποιοδήποτε μέρος στον κόσμο. Οι ερωτήσεις δεν περνάνε από κάποιου τύπου επεξεργασία πριν δημοσιευτούν. Άρα πρέπει να έχουμε πάντα στο μυαλό μας τα συντακτικά, ορθογραφικά και πιο σημαντικά εννοιολογικά λάθη που περιέχονται στα κείμενα, σε πολύ μεγαλύτερο βαθμό από αυτά που θα συναντούσαμε πχ σε ένα αρθογραφικό site.

№	Country	Number of users
1	USA	253452
2	India	67297
3	Great Britain	33395
4	Germany	19706
5	Canada	16685
6	China	14234
7	Australia	12592
8	Brazil	12325
9	France	12217
10	Russia	11319

(Οι 10 χώρες με τον μεγαλύτερο αριθμό χρηστών στο Stack Overflow (Οκτώβριος 2014)

1.4. Εισαγωγή στο Πρόβλημα

Έχοντας κάνει μία σύντομη εισαγωγή στην τεχνητή νοημοσύνη και στην κατηγοριοποίηση κειμένου, μπορούμε να περιγράψουμε το πρόβλημα το οποίο θα προσπαθήσει να αναλύσει αυτή η εργασία. Το πρόβλημα αυτό είναι ένα πρόβλημα αυτόματης κατηγοριοποίησης κειμένου, συγκεκριμένα των ερωτήσεων που υποβάλλονται στο Stack Overflow. Σκοπός είναι η δημιουργία ενός συστήματος που θα είναι σε θέση να προβλέπει, με βάση το κείμενο και τον κώδικα, μιας νέας, προς υποβολή ερώτησης, και να μπορεί να κάνει μια στοχευμένη εκτίμηση για το ποια θα είναι τα πιθανά tags που θα συνοδεύουν την ερώτηση αυτή. Για παράδειγμα θα θέλαμε το σύστημα να μπορεί να αναγνωρίζει ότι η ερώτηση

I'm creating a C application, and I need some data from my SQL server. Does anyone know how can I make an SQL query from my C application under Windows?

θα συνοδεύεται από τα tags C και SQL, ενώ η ερώτηση

What is the difference between a Java EE Web Profile certified server (like JOnAS) and a Java EE Full Platform certified server (like JBoss AS)?

θα συνοδεύεται από το tag java-ee.

2. ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗ ΚΕΙΜΕΝΟΥ

2.1. Τι είναι η κατηγοριοποίηση κειμένου?

2.1.1. Εισαγωγή στην κατηγοριοποίηση κειμένου

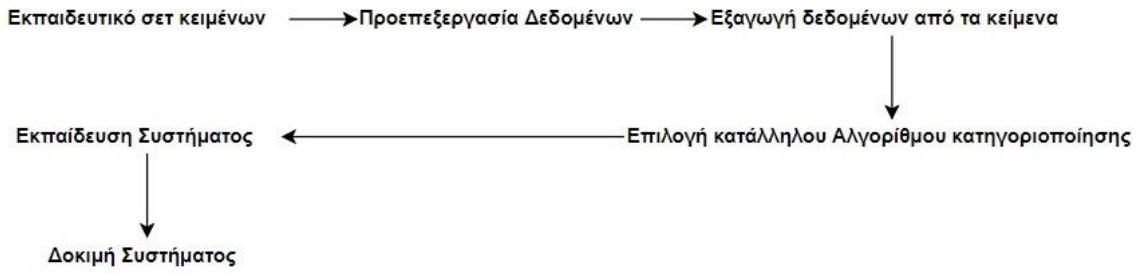
Με την έκρηξη του Internet και την εισαγωγή του σε κάθε πτυχή της καθημερινότητας μας, η ανάγκη για αυτόματη κατηγοριοποίηση του κειμένου σε προκαθορισμένες κατηγορίες έχει γνωρίσει μια ραγδαία ανάπτυξη. Η ανάπτυξη αυτή οφείλεται κυρίως στην αύξηση των διαθέσιμων κείμενων σε ψηφιακή μορφή και στην ανάγκη για την οργάνωση τους. Η κατηγοριοποίηση κειμένου έχει πολλές εφαρμογές όπως η αναγνώριση spam emails, η βελτίωση των αποτελεσμάτων των μηχανών αναζήτησης, η εξαγωγή άποψης ή συναισθήματος από κείμενα, κριτικές κτλ. και η κατηγοριοποίηση προϊόντων σε διαδικτυακά καταστήματα τύπου Amazon. Η ανάγκη για την ανάπτυξη αυτόματης κατηγοριοποίησης κειμένου αυξάνεται όσο αυτά τα site γίνονται ολοένα και πιο δημοφιλή.

Η κατηγοριοποίηση κειμένου (text classification) αναθέτει μία ή περισσότερες κατηγορίες σε ένα κείμενο, ανάλογα με το περιεχόμενο του. Εάν η μεταβλητή di, συμβολίζει ένα κείμενο i από ένα σετ κειμένων D και $\{c_1,c_2,...c_n\}$ είναι το σύνολο των κατηγοριών που εμφανίζονται στο σετ κειμένων, τότε η κατηγοριοποίηση κείμενο αντιστοιχεί μία ή περισσότερες κατηγορίες c_j στο κείμενο d_i. [1]

Τα κείμενα μπορούν να είναι σελίδες στο διαδίκτυο, βιβλία, δημοσιεύσεις σε μέσα κοινωνικής δικτύωσης κ.τ.λ.. Σε διάφορες περιπτώσεις η κατηγοριοποίηση αυτή έχει δυαδική μορφή, ένα αντικείμενο να ανήκει δηλαδή σε μια κατηγορία ή όχι (πχ spam mails). Παρόλα αυτά, οι περιπτώσεις αυτές είναι σπάνιες και τα προβλήματα κατηγοριοποίησης ανήκουν σε περισσότερες από μία κατηγορίες. Μια χαρακτηριστική περίπτωση που ένα κείμενο μπορεί να ανήκει σε περισσότερες από μία κατηγορίες είναι τα διάφορα site υποβολής ερωτήσεων, όπως είναι το Stack Overflow, το Quora ή και το Yahoo Answers. Μία ερώτηση σπανίως ανήκει μόνο σε μία κατηγορία, είτε αφορά ερωτήσεις προγραμματισμού είτε ερωτήσεις γενικού θέματος, π.χ. μουσική ή φιλοσοφία.

Τα βήματα, πίσω από την γενική λογική της κατηγοριοποίησης κειμένου είναι τα εξής [2] :

1. Προ επεξεργασία κειμένων
2. Εξαγωγή επιθυμητών χαρακτηριστικών από το κείμενο
3. Επιλογή μοντέλου εκπαίδευσης
4. Εκπαίδευση των ταξινομητών
5. Αξιολόγηση των ταξινομητών



(Διαδικασία κατηγοριοποίησης)

Υπάρχουν πολλά, διαφορετικά μοντέλα κατηγοριοποίησης κειμένου, ανάλογα με το είδος των κειμένων που έχουμε αλλά και τον αριθμό των διαθέσιμων κατηγοριών. Στον παρακάτω πίνακα βλέπουμε τα διάφορα μοντέλα, ανάλογα με τον αριθμό των διαθέσιμων κατηγοριών και το πόσες κατηγορίες μπορούν να αντιστοιχιστούν σε κάθε είσοδο.

	K = 2	**K > 2**
L = 1	Binary	Multi-class
L > 1	Multi-label	Multi-output

(Όπου K ο αριθμός των διαθέσιμων κατηγοριών και L ο αριθμός κατηγοριών που μπορούν να αποδοθούν σε ένα κείμενο)

Η binary κατηγοριοποίηση χρησιμοποιείται όταν υπάρχουν δύο διαθέσιμες κατηγορίες και σε κάθε κείμενο αποδίδεται μία κατηγορία. Χαρακτηριστικό παράδειγμα είναι ο διαχωρισμός ηλεκτρονικών μηνυμάτων σε “spam” και “not spam”. Η multi-class κατηγοριοποίηση χρησιμοποιείται όταν υπάρχουν πάνω από δύο διαθέσιμες κατηγορίες, αλλά στο κείμενο αποδίδεται μόνο μια. Ένα τέτοια παράδειγμα είναι ο χαρακτηρισμός καταλληλόλητας τηλεοπτικών προγραμμάτων στην τηλεόραση. Η multi-label κατηγοριοποίηση, με την οποία ασχολούμαστε, χρησιμοποιείται όταν υπάρχουν πάνω από δύο διαθέσιμες κατηγορίες και σε κάθε κείμενο αποδίδεται πάνω από μία κατηγορία. Παραδείγματος χάρη, multi-label κατηγοριοποίηση χρησιμοποιείται για την κατηγοριοποίηση μιας ταινίας σε διαφορετικά είδη (π.χ. Drama, Action κτλ.). Τέλος, η multi-output κατηγοριοποίηση χρησιμοποιείται όταν εκπαιδεύεται ένας ταξινομητής ανά διαθέσιμη έξοδο.

Στην παρούσα εργασία θα ασχοληθούμε με την multi-label κατηγοριοποίηση κειμένου.

2.1.2. Multi-label Κατηγοριοποίηση Κειμένου

Όπως αναφέρθηκε και στην προηγούμενη ενότητα, όταν κατηγοριοποιείται ένα κείμενο, πολλές φορές υπάρχει η ανάγκη να του αποδοθούν περισσότερες από μία κατηγορίες. Στην multi-label κατηγοριοποίηση κειμένου το μοντέλο αντιστοιχεί κάθε είσοδο x σε διανύσματα με δυαδικές τιμές y, όπου το y περιέχει κάθε κατηγορία που συναντάται στο σύνολο των κειμένων.

Examples	Adventure	Drama	Fantasy
Fellowship of the Ring	X	X	X
Pan's Labyrinth		X	X
The Avengers	X		X

(Παράδειγμα multi-label κατηγοριοποίησης)

Υπάρχουν διαφορετικές προσεγγίσεις για την υλοποίηση της multi-label κατηγοριοποίησης. Οι υλοποιήσεις αυτές μπορούν να χωριστούν σε δύο κατηγορίες: στις *μεθόδους μετασχηματισμού προβλημάτων* (problem transformation methods) και στις *μεθόδους προσαρμογής αλγορίθμων* (algorithm adaptation methods). Ως μετασχηματισμό προβλημάτων αναφερόμαστε στις μεθόδους που μετασχηματίζουν το πρόβλημα σε ένα ή περισσότερα προβλήματα single label κατηγοριοποίησης ή προβλήματα αναδρομής. Ως προσαρμογή αλγορίθμων αναφερόμαστε στις μεθόδους που επεκτείνουν έναν συγκεκριμένο αλγόριθμο για να μπορεί να χειριστεί multi-label δεδομένα. [3]

2.2. Μέθοδοι μετασχηματισμού προβλημάτων

Οι μέθοδοι μετασχηματισμού προβλημάτων ασχολούνται με την μετατροπή του προβλήματος σε binary ή σε multiclass κατηγοριοποίηση. Η μετατροπή αυτή είναι αρκετά προσαρμοστική, αφού δεν εξαρτάται από τον αλγόριθμο ταξινόμησης που χρησιμοποιείται. Έτσι μας επιτρέπει να χρησιμοποιήσουμε συστήματα single label ταξινόμησης, όπως το k-NN, τα Decision Trees και τα SVMs,.

2.2.1. Binary Relevance

Ένας τρόπος να επιτυγχάνεται αυτό είναι με την χρήση της μεθόδου binary relevance, όπου εκπαιδεύει ανεξάρτητα έναν binary ταξινομητή για κάθε διαθέσιμη κατηγορία που υπάρχει. Το συνδυαστικό μοντέλο μετά προβλέπει για κάθε παράδειγμα όλες τις κατηγορίες στις οποίες ανήκει το παράδειγμα, χρησιμοποιώντας τα αποτελέσματα που παράγουν οι ταξινομητές. Με αυτόν τον τρόπο επιτρέπουμε πολλαπλές κατηγορίες να προβλεφθούν για κάθε παράδειγμα.

Η μέθοδος Binary Relevance θα αναλύσει το πρόβλημα σε q ανεξάρτητα binary learning προβλήματα, όπου για κάθε κατηγορία στο σύνολο των κατηγοριών Y θα εκπαιδευτεί ένας binary ταξινομητής. Για κάθε κατηγορία λj,θα δημιουργηθεί το

αντίστοιχο binary εκπαιδευτικό σετ το Dj, από το αρχικό εκπαιδευτικό σετ, όπου

$$D_j = \{(x^i, y^i_j) \mid 1 \le i \le m\}$$

Δηλαδή, κάθε παράδειγμα (x^i, y^i) μετασχηματίζεται σε ένα binary παράδειγμα, με βάση τη σχετικότητα του με το λ_j. Ένας ταξινομητής g_j θα δημιουργηθεί από το D_j, εφαρμόζοντας κάποιον αλγόριθμο binary μάθησης. Έτσι, όλα τα παραδείγματα (x^i, y^i) θα συνεισφέρουν στην εκπαίδευση όλων των ταξινομητών g_j $(1 \le j \le q)$. Για κάθε σχετική κατηγορία το x_i θα θεωρείται ως θετικό παράδειγμα εκπαίδευσης στη δημιουργία του g_j, και για κάθε μη σχετική κατηγορία αρνητικό. Η διαδικασία αυτή ονομάζεται cross-training.

Έστω x ένα παράδειγμα του αρχικού συνόλου εκπαίδευσης. Η μέθοδος Binary Relevance θα προβλέψει την σχέση στο σύνολο των κατηγοριών Y του παραδείγματος μαζεύοντας τις σχετικότητες του κάθε ταξινομητή και συνδυάζοντας τα αποτελέσματα για όλες τις σχετικές κατηγορίες

$$Y = \{\lambda_j \mid g_j(x) > 0, 1 \le j \le q\} \quad [4]$$

Η μέθοδος αυτή παρουσιάζει μικρή πολυπλοκότητα, καθώς δημιουργεί Y binary ταξινομητές, αλλά επειδή αντιμετωπίζει ξεχωριστά την κάθε κατηγορία δε λαμβάνει υπόψιν της τις σχέσεις μεταξύ των κατηγοριών. [5]

Examples	Adventure
Fellowship of the Ring	X
Pan's Labyrinth	
The Avengers	X

Examples	Drama
Fellowship of the Ring	X
Pan's Labyrinth	X
The Avengers	

Examples	Fantasy
Fellowship of the Ring	X
Pan's Labyrinth	X
The Avengers	X

(Παράδειγμα multi-label κατηγοριοποίησης με binary relevance)

2.2.2. Label Powerset

Η μέθοδος Label Powerset (LP) είναι μια λιγότερο διαδεδομένη μέθοδος μετασχηματισμού προβλήματος. Η μέθοδος αυτή θεωρεί κάθε μοναδικό σετ κατηγοριών που υπάρχει στο σύνολο εκπαίδευσης ως μία νέα κατηγορία ενός νέου single-label ταξινομητή. Για κάθε παράδειγμα του συνόλου ο ταξινομητής αυτός θα παράξει το πιο πιθανό αποτέλεσμα, που στην πραγματικότητα θα είναι ένα σύνολο κατηγοριών. [6]

Το Label Powerset μας δίνει καλά αποτελέσματα αλλά έχει μεγάλη πολυπλοκότητα και παρουσιάζει θέματα με την "αραιότητα" που μπορεί να παρουσιάζουν συγκεκριμένες κατηγορίες. [7]

Examples	Fantasy & Drama	Adventure & Fantasy	Adventure & Drama & Fantasy
Fellowship of the Ring			X
Pan's Labyrinth	X		
The Avengers		X	

(Παράδειγμα λογικής multi-label κατηγοριοποίησης με Label Powerset)

2.3. Μέθοδοι προσαρμογής αλγορίθμων

Οι μέθοδοι προσαρμογής αλγορίθμων ασχολούνται με την προσαρμογή "δημοφιλών" αλγορίθμων κατηγοριοποίησης, ώστε να μπορούν να εκπαιδευτούν σε multi-label δεδομένα. Πολλοί αλγόριθμοι κατηγοριοποίησης, όπως ο k-Nearest Neighbors (k-NN), Decision Trees, έχουν προσαρμοστεί σε multi-label δεδομένα. [8]

Εξετάζεται ως παράδειγμα η προσαρμογή του αλγόριθμου k-NN. Ο αλγόριθμος ML-k-NN (multi-label k-Nearest Neighbors), επεκτείνει τον αλγόριθμο k-NN χρησιμοποιώντας την Bayesian κατανομή. Για κάθε διαθέσιμη κατηγορία y του συνόλου κατηγοριών Y, βρίσκει τα k πιο κοντινά παραδείγματα στο παράδειγμα και θεωρεί ότι αυτά που έχουν τουλάχιστον την κατηγορία y ως θετικά και τα υπόλοιπα ως αρνητικά παραδείγματα.

Έστω παράδειγμα x, και C_j, παραδείγματα στη γειτονιά του x, όπου περιέχουν την κατηγορία y_j. Αν H_j η πιθανότητα το x να ανήκει στην κατηγορία y_j και $P(H_j \mid C_j)$ η δεσμευμένη πιθανότητα το x να ανήκει στην κατηγορία y_j και να έχει Cj γείτονες που ανήκουν επίσης στην κατηγορία y_j. Αντίστοιχα, $P(\neg H_j \mid C_j)$ η πιθανότητα να μην ανήκει το x στην κατηγορία y_j. Έτσι το προβλεπόμενο σετ κατηγοριών δίνεται από τον τύπο:

$$Y = y_j \mid \frac{P(H_j \mid C_j)}{P(\neg Hj \mid Cj)} \; 1<j<q \text{ [9]}$$

Ενώ οι μέθοδοι προσαρμογής αλγορίθμων επιφέρουν καλά αποτελέσματα, πολλοί δημοφιλείς αλγόριθμοι όπως οι SVM, Linear SVM και Naive Bayes δεν έχουν προσαρμοστεί για multi-label δεδομένα.

2.4. Κατηγοριοποίηση με τη χρήση συσταδοποίησης (Clustering)

Ένας άλλος δημοφιλής τρόπος κατηγοριοποίησης κειμένου είναι με τη χρήση συσταδοποίησης (clustering). Η συσταδοποίηση αναφέρεται στη διαδικασία εύρεσης παρόμοιων κειμένων σε ένα σύνολο από κείμενα. Η ομοιότητα μεταξύ των κειμένων βρίσκεται μέσα από μια συνάρτηση ομοιότητας. Ο βαθμός ανάλυσης στον οποίο ψάχνονται οι ομοιότητες μπορεί να είναι σε επίπεδο κειμένου, πρότασης ή και λέξεων.

Υπάρχουν πολλοί αλγόριθμοι συσταδοποίησης οι οποίοι μπορούν να χρησιμοποιηθούν για την κατηγοριοποίηση κειμένου. Το κείμενο μπορεί να αναπαρασταθεί με διανύσματα, όπως με δυαδικά διανύσματα που υποδεικνύουν την ύπαρξη ή όχι μιας λέξης ή και με διανύσματα που υπολογίζουν το βάρος των λέξεων, όπως τα tf-idf.

Παρόλα αυτά, η κατηγοριοποίηση κειμένου με τη χρήση συσταδοποίησης παρουσιάζει δυσκολίες, καθώς το κείμενο έχει διάφορα ιδιαίτερα χαρακτηριστικά τα οποία απαιτούν την χρήση αλγορίθμων σχεδιασμένων για να χειρίζονται κείμενα. Παρακάτω περιγράφονται, περιληπτικά, μερικές από αυτές τις δυσκολίες. [10]

- Η αναπαράσταση του κειμένου γίνεται από διανύσματα πολύ μεγάλων διαστάσεων, τα οποία όμως είναι πολύ αραιά. Το μέγεθος του λεξιλογίου είναι πολύ μεγάλο αλλά ένα κείμενο μπορεί να περιέχει ελάχιστες λέξεις. Το γεγονός αυτό δυσκολεύει πολύ την εύρεση ομοιοτήτων μεταξύ των κειμένων.
- Ο αριθμός των concept σε ένα σύνολο κειμένων είναι πολύ μικρός σε σχέση με το σύνολο των λέξεων, καθώς οι λέξεις που σχετίζονται εμφανίζονται συχνά μαζί. Αυτό σημαίνει ότι υπάρχουν λίγα concept τα οποία μπορεί να ανακαλύψει ο αλγόριθμος, συγκριτικά με το λεξιλόγιο των κειμένων.

3. ΑΛΓΟΡΙΘΜΟΙ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

3.1. Αλγόριθμοι μηχανικής μάθησης

Οι αλγόριθμοι μηχανικής μάθησης χωρίζονται σε δύο βασικές κατηγορίες, στους αλγόριθμους επιβλεπόμενης μάθησης (supervised learning) και μη επιβλεπόμενης μάθησης (unsupervised learning), ανάλογα από το αν το σύστημα περιμένει μια ανατροφοδότηση από το χρήστη ή όχι. Πέρα από τις δύο αυτές βασικές κατηγορίες έχουν αναπτυχθεί και υβριδικές κατηγορίες που εμπεριέχουν στοιχεία και από τις δύο, όπως η ημι-επιβλεπόμενη μάθηση και η ενισχυμένη μάθηση.

3.1.1. Επιβλεπόμενη Μάθηση

Οι αλγόριθμοι επιβλεπόμενης μάθησης χρησιμοποιούνται όταν υπάρχει μια "σωστή απάντηση" για κάθε παράδειγμα, όπως η κατάταξη κειμένου σε κατηγορίες με ετικέτες. Για την εκπαίδευση του συστήματος παρέχεται, μαζί με τα δεδομένα εισόδου και η επιθυμητή έξοδος για το καθένα από αυτά. Με τον τρόπο αυτό, για κάθε είσοδο που δέχεται το πρόγραμμα προβλέπει μια έξοδο και συγκρίνει το αποτέλεσμα που παράγει με το σωστό αποτέλεσμα που του παρέχεται. Με βάση τις λάθος προβλέψεις που κάνει τροποποιεί αναλόγως το μοντέλο. Ο αλγόριθμος θα σταματήσει να κάνει προβλέψεις όταν επιτύχει ένα αποδεκτό επίπεδο απόδοσης. Στους αλγόριθμους επιβλεπόμενης μάθησης το πρόγραμμα εφαρμόζει το τι έχει μάθει στο παρελθόν, από το σύνολο εκπαίδευσης, σε νέα δεδομένα για να προβλέψει τα μελλοντικά γεγονότα. [11]

Οι αλγόριθμοι επιβλεπόμενης μάθησης χωρίζονται σε δύο κατηγορίες, στους αλγόριθμους κατηγοριοποίησης, με τους οποίους και ασχολούμαστε σε αυτήν την εργασία, και στους αλγόριθμους παλινδρόμησης. Οι αλγόριθμοι κατηγοριοποίησης προσπαθούν να αναγνωρίσουν σε ποια από όλες τις διαθέσιμες κατηγορίες ανήκει κάθε καινούργιο δεδομένο που παρέχεται στο σύστημα χρησιμοποιώντας ως βάση το εκπαιδευτικό σετ. Οι αλγόριθμοι παλινδρόμησης προσπαθούν να βρούνε ποια από τις διαθέσιμες εξισώσεις μπορεί να ταιριάξει με τα δεδομένα, ώστε να προβλέψει την τιμή των μελλοντικών εξόδων.

3.1.2. Μη Επιβλεπόμενη Μάθηση

Οι αλγόριθμοι μη-επιβλεπόμενης μάθησης χρησιμοποιούνται όταν δεν υπάρχει μια "σωστή απάντηση" για τα παραδείγματα, όπως σε προβλήματα που το σύστημα πρέπει να εντοπίσει δεδομένα που παρουσιάζουν ομοιότητες μεταξύ τους. Στους αλγόριθμους μη επιβλεπόμενης μάθησης παρέχονται μόνο δεδομένα εισόδου χωρίς κάποια έξοδο. Το πρόγραμμα εξερευνάει τα δεδομένα που του δίνονται και προσπαθεί να αντλήσει συμπεράσματα από αυτά, ψάχνοντας να βρει σχέσεις και να βρει κρυμμένες δομές μέσα σε αυτά.

Οι αλγόριθμοι μη επιβλεπόμενης μάθησης χωρίζονται σε δύο κατηγορίες, στους αλγόριθμους ομαδοποίησης και στους αλγόριθμους σύνδεσης. [12]

3.1.3. Ήμι-επιβλεπόμενη και Ενισχυμένη Μάθηση

Στην ημι-επιβλεπόμενη μάθηση στον αλγόριθμο δίνονται δεδομένα μαζί με την επιθυμητή τους έξοδο καθώς και δεδομένα χωρίς αυτήν. Συνήθως τα στοιχεία που έχουν έξοδο είναι λίγα, ενώ τα στοιχεία που δεν έχουν είναι πολλά. Με αυτόν τον τρόπο τα συστήματα μπορούν να επιτύχουν μεγάλη ακρίβεια. [13]

Στην ενισχυμένη μάθηση, ο αλγόριθμος επιδρά με το περιβάλλον του παράγοντας δράσεις και ανακαλύπτοντας λάθη ή ανταμοιβές. Αυτή η μέθοδος επιτρέπει στα προγράμματα να ανακαλύψουν την ιδανική συμπεριφορά μέσα σε ένα συγκεκριμένο πλαίσιο, ώστε να βελτιστοποιήσουν την απόδοση τους.

Παρακάτω παρουσιάζεται η λογική πίσω από τους αλγόριθμους ταξινομητές που χρησιμοποιήθηκαν για την επίλυση του δικού μας προβλήματος.

3.2. Naive Bayes

Ο αλγόριθμος Naive Bayes είναι ένας από τους πιο διαδεδομένους αλγόριθμους κατηγοριοποίησης, ο οποίος σαν βασική αρχή του έχει τον νόμο του Bayes για την ανεξαρτησία των υποθέσεων μεταξύ των προβλέψεων. Ο αλγόριθμος Bayes θεωρεί ότι όλα τα γνωρίσματα που εξετάζονται είναι μεταξύ τους ανεξάρτητα. Η υπόθεση αυτή βοηθάει στην απλοποίηση της διαδικασίας της κατηγοριοποίησης.

Ο νόμος του Bayes μας επιτρέπει να υπολογίσουμε μεταγενέστερες πιθανότητες, με βάση τις καταστάσεις που γνωρίζουμε ήδη. Έστω C μεταβλητή για την κατηγορία ενός παραδείγματος και διάνυσμα X, με τυχαίες μεταβλητές που αναπαριστούν τις τιμές του χαρακτηριστικού που δίνεται ως είσοδος. Η πιθανότητα ένα διάνυσμα x να ανήκει στην κατηγορία c, δεδομένου ότι το διάνυσμα X ανήκει στην κατηγορία C, δίνεται από τον γνωστό κανόνα Bayes:

$$P(C = c_j, X = x) = \frac{p(C = c)p(X = x|C = c)}{p(X = x)}$$

όπου X=x είναι η περίπτωση $X = x_1 \wedge x_2 \wedge \ldots \wedge x_n$.

Για να υπολογίσει σε ποια από τις n διαθέσιμες κατηγορίες ανήκει, εν τέλει, το διάνυσμα x ο αλγόριθμος εκτιμά την μεγαλύτερη πιθανότητα. Ο αλγόριθμος υποθέτει ότι κάθε χαρακτηριστικό είναι ανεξάρτητο από κάθε άλλο, και άρα μπορεί απλά να πάρει το μέγιστο δυνατό αποτέλεσμα.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y) \text{ [14]}$$

Πάρα την αρκετά απλοποιημένη προσέγγιση των Naive Bayes συστημάτων, δίνουν πολύ καλά αποτελέσματα σε πολλές περιπτώσεις, όπως η κατηγοριοποίηση κειμένων και η αναγνώριση spam μηνυμάτων. Χρειάζονται λίγα δεδομένα για την εκπαίδευσή τους και άρα η διαδικασία εκπαίδευσης είναι πολύ γρήγορη, ειδικά όταν τη συγκρίνουμε με

κάποιους πιο περίπλοκους αλγόριθμους. Θα πρέπει να λάβουμε όμως υπόψιν ότι οι αλγόριθμοι Naive Bayes χάνουν ιδιαιτέρως στην ακρίβεια των προβλέψεων τους. Επεκτάσεις του αλγορίθμου Naive Bayes ενσωματώνουν και άλλες γνωστές κατανομές πιθανοτήτων, όπως ο αλγόριθμος Gaussian Naive Bayes και ο Bernoulli Naive Bayes. [15]

3.3. Decision Trees

Τα δένδρα αποφάσεων (decision trees) είναι από τις πιο απλές και διαδεδομένες τεχνικές κατηγοριοποίησης, καθώς εφαρμόζουν μία σαφή, ξεκάθαρη λογική που μπορεί να προσαρμοστεί σε μία μεγάλη γκάμα προβλημάτων.

Τα δένδρα αποφάσεων θέτουν μία σειρά από ερωτήσεις για τις ιδιότητες των αντικειμένων του συνόλου εκπαίδευσης. Για κάθε αντικείμενο που εισάγεται στο μοντέλο, αυτό του θέτει μια ερώτηση. Κάθε φορά που παίρνει απάντηση θέτει μία νέα ερώτηση, μέχρι να μπορούν να συμπεράνουν την κατηγορία του αντικειμένου. Το σύνολο των ερωτήσεων και των απαντήσεων μπορούν να οργανωθούν σε δομή δένδρου αποφάσεων, όπου είναι μια ιεραρχική δομή από κόμβους και κατευθυνόμενων συνδέσμων. [16]

Η δομή ενός δένδρου αποφάσεων είναι η ακόλουθη. Αποτελείται από εσωτερικούς κόμβους και κόμβους φύλλα. Στη δομή αυτή κάθε κόμβος φύλλο αντιστοιχεί σε μία κατηγορία, όπου αναπαρίσταται με μια περιοχή του χώρου των δεδομένων. Κάθε μητερματικός κόμβος περιέχει ερωτήσεις - συνθήκες για τις ιδιότητες των αντικειμένων, με σκοπό να χωρίσει τα αντικείμενα με διαφορετικές ιδιότητες. Όσο πιο βαθύ είναι το δένδρο, τόσο πιο περίπλοκοι είναι οι κανόνες αποφάσεων και τόσο πιο ακριβές είναι το μοντέλο.

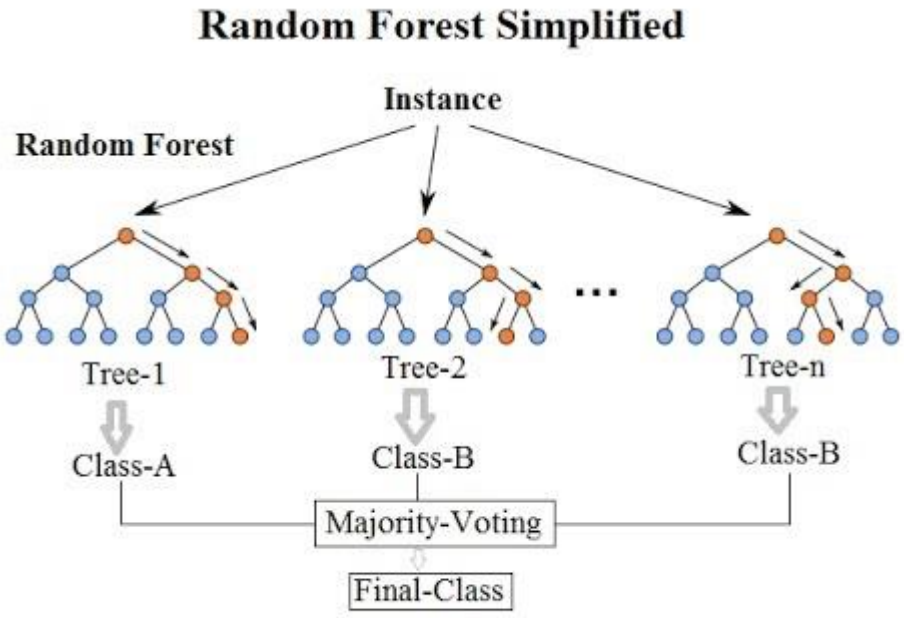
Η χρήση των δένδρων αποφάσεων έχει πολλά πλεονεκτήματα. Είναι ένας αλγόριθμος που γίνεται εύκολα κατανοητός, επίσης είναι πολύ απλό να οπτικοποιηθεί με τη χρήση διαγράμματος. Δεν απαιτεί μεγάλο όγκο δεδομένων για να εκπαιδευτεί και μπορεί να δώσει καλά αποτελέσματα στις περιπτώσεις κατηγοριοποίησης με πολλά δυνατά αποτελέσματα. Τα μειονεκτήματα του δένδρου αποφάσεων συμπεριλαμβάνουν την αστάθεια που μπορεί να προκληθεί στα δένδρα από μικρές αλλαγές στα δεδομένα εισόδου, την περίπτωση να φτιαχτούν υπερβολικά σύνθετα δένδρα που δεν γενικεύουν καλά τα δεδομένα (overfitting).

(Παράδειγμα κατηγοριοποίησης με τη χρήση δένδρου αποφάσεων)

3.4. Random Forest Classifier

Ο αλγόριθμος Random Forest, όπως πιθανώς θα μπορούσε να καταλάβει κανείς από το όνομα του, αποτελεί μία επέκταση των δένδρων αποφάσεων. Είναι και αυτός ένας αλγόριθμος επιβλεπόμενης μάθησης, όπου δημιουργεί ένα «δάσος», ένα σύνολο δηλαδή δένδρων αποφάσεων. Όπως και ένα δάσος είναι πιο πυκνό αν περιέχει πολλά δένδρα, έτσι και ο αλγόριθμος μας δίνει μεγαλύτερη ακρίβεια όσο περισσότερα δένδρα αποφάσεων δημιουργούνται. [17]

Ο αλγόριθμος ανήκει στην κατηγορία του 'Ensemble Learning', όπου συνδυάζει πολλά μοντέλα για να λύσει ένα πρόβλημα. Δημιουργεί πολλούς ταξινομητές, που εκπαιδεύονται και κάνουν προβλέψεις ανεξάρτητα ο ένας από τον άλλο. Αυτές οι προβλέψεις συνδυάζονται, ώστε να πραγματοποιήσουν μια μεγάλη πρόβλεψη, όπου θα είναι καλύτερη ή τουλάχιστον τόσο καλή όσο η πρόβλεψη του κάθε ταξινομητή. Ο αλγόριθμος Random Forest δημιουργεί τυχαία δένδρα αποφάσεων, κάποια από τα οποία είναι χρήσιμα για την πρόβλεψη που θέλουμε να πραγματοποιηθεί και κάποια όχι. Με τον τρόπο αυτό, όταν δίνεται στον αλγόριθμο ένα αντικείμενο όλα τα δένδρα που έχουν δημιουργηθεί θα πραγματοποιήσουν μια πρόβλεψη. Οι προβλέψεις θα συγκεντρωθούν και συμψηφιστούν, προκειμένου να υπολογιστεί η τελική πρόβλεψη. Καθώς τα περισσότερα δένδρα θα δώσουν άκυρες προβλέψεις θα ακυρώνονται μεταξύ τους και άρα θα είναι τα χρήσιμα δένδρα αυτά που θα καθορίσουν, εν τέλει, την τελική πρόβλεψη.



(Παράδειγμα κατηγοριοποίησης με τη Random Forest Classifier)

3.5. K-Nearest Neighbors

Ο αλγόριθμος K Nearest Neighbors (k-NN) είναι ένας αλγόριθμος κατηγοριοποίησης που ταξινομεί τα αντικείμενα με βάση κριτήρια ομοιότητας. Ο αλγόριθμος k-NN είναι ένας μη-παραμετροποιήσιμος αλγόριθμος, το οποίο σημαίνει ότι δεν βασίζεται σε υποθέσεις για την κατανομή των δεδομένων. Ένα αντικείμενο ταξινομείται με βάση την πιο κοινή κατηγορία που συναντάται στους γείτονες τους, όπου ως γείτονες ορίζονται τα αντικείμενα με διανυσματική απόσταση μικρότερη του K. Για K = 1, παραδείγματος χάριν, ένα αντικείμενο κατηγοριοποιείται με βάση τον πιο κοντινό του γείτονα. Γενικά, μια μεγάλη τιμή K μπορεί να δώσει πιο ακριβή αποτελέσματα, καθώς συμπεριλαμβάνει περισσότερους γείτονες και βελτιώνουν το αποτέλεσμα. Από την άλλη, μία μεγάλη τιμή K θα συμπεριλάβει μέσα στο ευρέως των γειτόνων και πιο σπάνιες περιπτώσεις, που επηρεάζουν αρνητικά την απόφαση. Επίσης τέτοιες τιμές K, ώστε να μην προκύπτουν ισοψηφίες μεταξύ των πιθανών κατηγοριών, όπως ζυγή αριθμοί ή πολλαπλάσια του συνόλου των κατηγοριών Y. [18]

Για τον υπολογισμό της απόστασης μπορεί να χρησιμοποιηθεί οποιαδήποτε μαθηματική απόσταση σημείων, με τις συνηθέστερες να αποτελούν την ευκλείδεια απόσταση, την απόσταση Manhattan και την απόσταση Minkowski.

- $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$ (Ευκλείδεια απόσταση)
- $\sum_{i=1}^{k}|x_i - y_i|$ (απόσταση Manhattan)
- $(\sum_{i=1}^{k}(|x_i - y_i|)^q)^{1/q}$ (απόσταση Minkowski)

3.6. Multilayer Perceptron Classifier

Ο Multilayer perceptron (perceptron πολλών επιπέδων ή MLP για συντομία) ταξινομητής βασίζεται στην λογική των νευρωνικών δικτύων εμπρόσθιας διάδοσης. Χρησιμοποιεί τη λογική του απλού perceptron, σύμφωνα με την οποία δημιουργείται ένας νευρώνας που τροφοδοτείται από μια είσοδο x n στοιχείων, η οποία πολλαπλασιάζεται με ένα βάρος w (επίσης n στοιχείων) και τα γινόμενα αθροίζονται μαζί με έναν όρο πόλωσης b. Το τελικό άθροισμα εισάγεται σε μία μη γραμμική συνάρτηση φ, από την οποία προκύπτει η έξοδος y σύμφωνα με τον τύπο:

$$v = x1w1 + x2w2 + \cdots + xnwn + b = wTx + b$$

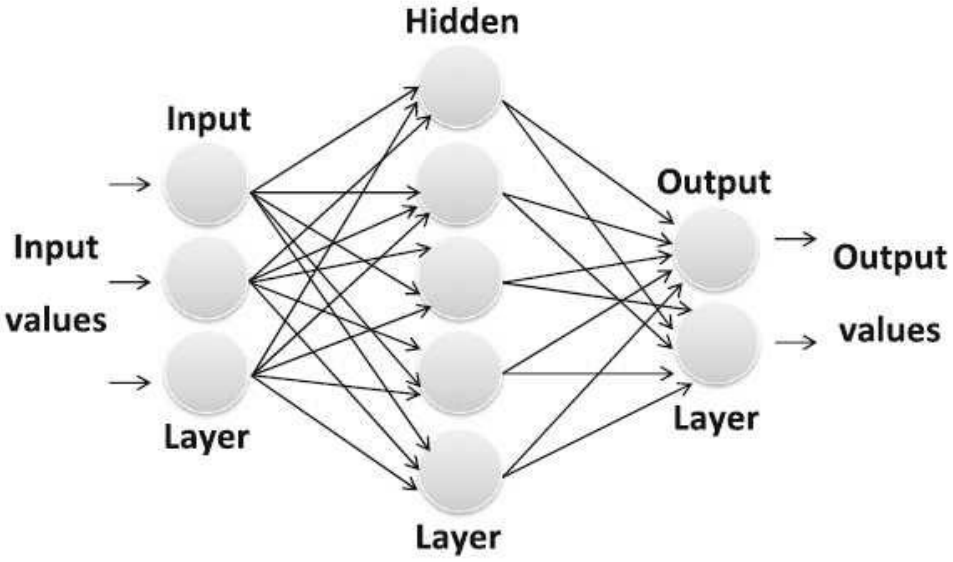
$$y = \varphi(v) = \varphi(wTx + b)\text{ , όπου } w = (w1, w2, \dots, wn).$$

Η συνάρτηση y έχει δύο επιτρεπτές εξόδους, 1 και -1. Άρα έχουμε

$$y=sgn(\mathbf{w}T\mathbf{x}+b)=\{+1\text{ , }\alpha\nu\ \mathbf{w}T\mathbf{x}+b>0\ \ -1\text{ , }\alpha\nu\ \mathbf{w}T\mathbf{x}+b<0$$

Εφαρμόζοντας τη λογική του απλού perceptron στην περίπτωση της κατηγοριοποίησης αρχικοποιούμε τις τιμές των παραμέτρων τυχαία. Για κάθε είσοδο που δίνεται στο σύστημα, τα βάρη και η πόλωση τροποποιούνται αναλόγως, με βάση το αν η έξοδος που δίνεται από το σύστημα είναι σωστή ή όχι. [19]

Στην περίπτωση τώρα του MLP, αντί για ένα επίπεδο με έναν μόνο νευρώνα υπάρχει τουλάχιστον ένα ακόμα επίπεδο (ή περισσότερα), που ονομάζεται κρυφό επίπεδο (hidden layer), ανάμεσα στην είσοδο και στην έξοδο. Τα δεδομένα εισόδου μετασχηματίζονται από το ένα επίπεδο στο άλλο. Τα βάρη και οι πολώσεις του κάθε επιπέδου προσαρμόζονται ανάλογα με την έξοδο. Η γενική λογική πίσω από τον MLP είναι ότι όσο περισσότερα επίπεδα υπάρχουν τόσο καλύτερη η ταξινόμηση που μπορεί να



πραγματοποιήσει το σύστημα.

(Παράδειγμα μοντέλου Multilayer Perceptron)

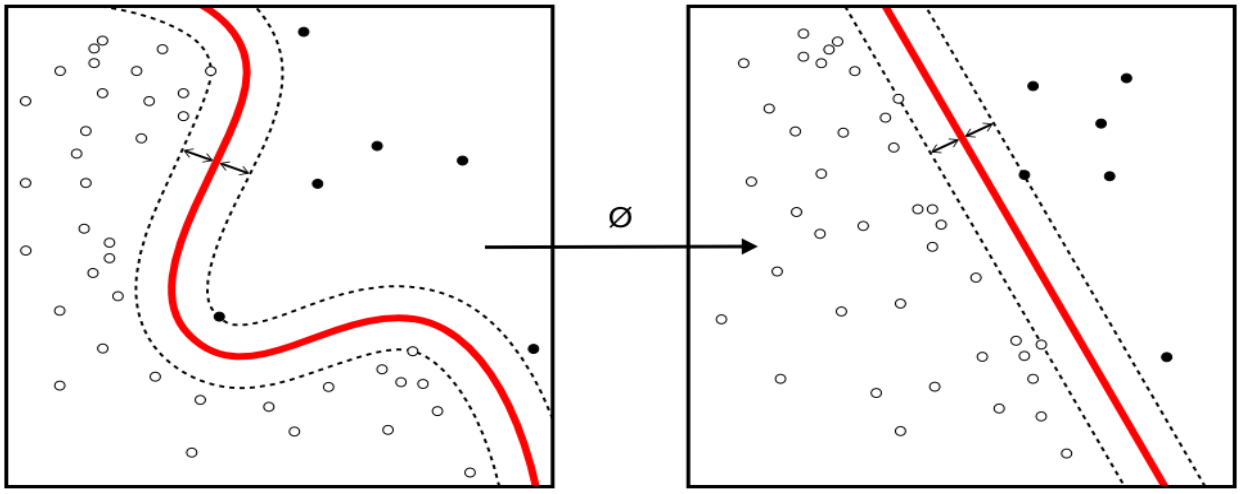
3.7. Support Vector Machines

Οι Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines ή SVM) είναι ένας αλγόριθμος μεγάλου περιθωρίου, που μπορεί να πραγματοποιήσει αποτελεσματικά μη γραμμική κατηγοριοποίηση. Ο αλγόριθμος είναι εφαρμογή των support vectors και είναι ένας από τους πιο διαδεδομένους αλγόριθμους κατηγοριοποίησης για κατηγοριοποίηση εικόνων, κειμένου αλλά και χειρόγραφων χαρακτήρων. Ταυτόχρονα, έχει χρησιμοποιηθεί ευρέως στη βιολογία και σε άλλες επιστήμες.

Ένα support vector κατασκευάζει ένα υπερεπίπεδο ή ένα σετ υπερεπιπέδων σε

πολυδιάστατο χώρο, τα οποία χωρίζουν τα δεδομένα από την μία κατηγορία στην άλλη. Τα επίπεδα διαχωρισμού όμως μπορεί να είναι πάρα πολλά ή και άπειρα. Ο αλγόριθμος ψάχνει να βρει εκείνο το υπερεπίπεδο για το οποίο μεγιστοποιείται η απόσταση μεταξύ των στοιχείων των επιπέδων. Αν υπάρχει αυτό το υπερεπίπεδο ονομάζεται maximum margin hyperplane και ο κατηγοριοποιητής που δημιουργείται από αυτό maximum margin classifier. Η γενική λογική είναι ότι όσο μεγαλύτερη είναι η απόσταση μεταξύ των επιπέδων τόσο μικρότερο και το ποσοστό της λάθος ταξινόμησης των δεδομένων. [20]

Πολύ συχνά όμως, τα προβλήματα που καλείται να αντιμετωπίζει ο SVM αλγόριθμος δεν είναι γραμμικά διαχωρίσιμα. Για αυτόν τον λόγο, τα προβλήματα αυτά αντιστοιχίζονται σε προβλήματα ενός χώρου υψηλότερων διαστάσεων, που κάνει τον διαχωρισμό πιο εύκολο στο διάστημα αυτό. Οι αντιστοιχήσεις αυτές, προκειμένου να μειώσει το υπολογιστικό φορτίο, σχεδιάζονται έτσι ώστε να εξασφαλιστεί ότι τα προϊόντα μπορούν να υπολογιστούν εύκολα από τις μεταβλητές του αρχικού χώρου, ορίζοντας τους με όρους μιας λειτουργίας πυρήνα (kernel function) *k(x,y)* επιλεγμένη ώστε να ταιριάζει στο πρόβλημα.



(Παράδειγμα αντιστοίχισης με τη χρήση λειτουργίας πυρήνα)

3.8. Γραμμικά Support Vector Machines

Ο αλγόριθμος Linear Support Vector Machines είναι μια κατηγορία των SVM αλγορίθμων όπου η λειτουργία πυρήνα έχει γραμμική τιμή.

Έστω ότι δίνεται ένα εκπαιδευτικό σετ αποτελούμενο από n σημεία της μορφής

$$(x_1,y_1),...,(x_n,y_n)$$

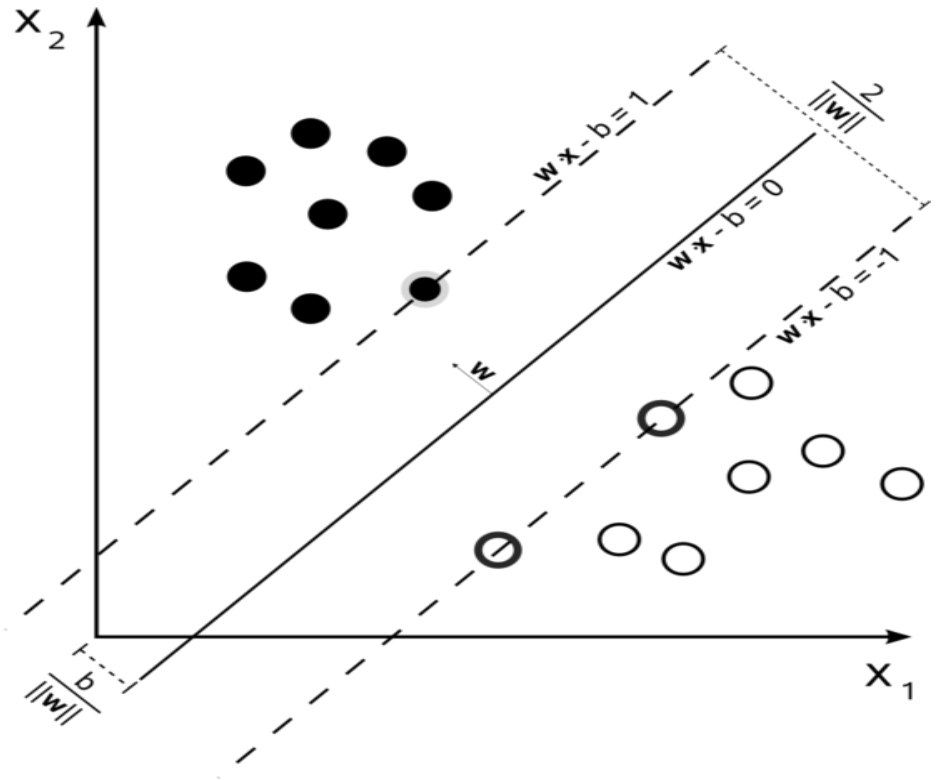
όπου y_i μπορεί να είναι 1 είτε -1, καθένα από αυτά υποδεικνύει σε ποια κατηγορία του σημείο x_i ανήκει. Κάθε στοιχείο x_i είναι ένα διάνυσμα p διαστάσεων. Ο αλγόριθμος ψάχνει να βρει το maximum margin hyperplane όπου χωρίζει τα x_i για τα οποία ισχύει ότι

$y_i = 1$ από τα σημεία για τα οποία ισχύει ότι $y_i = -1$, ώστε η απόσταση μεταξύ του υπερεπιπέδου και του πιο κοντινού σημείου x_i από το κάθε γκρουπ να είναι μέγιστη.

Κάθε υπερεπίπεδο μπορεί να γραφτεί σαν ένα σύνολο από σημεία x για τα οποία ισχύει ότι

$$w \cdot x - b = 0,$$

όπου w είναι κάθετο διάνυσμα στο υπερεπίπεδο. Η παράμετρος $\frac{b}{w}$ καθορίζει το αντιστάθμισμα του υπερεπιπέδου από την αρχή μέχρι το διάστημα w. [21]



(Maximum-margin hyperplane για SVM εκπαιδευμένο με δείγματα από δύο κατηγορίες)

4. ΕΠΕΞΕΡΓΑΣΙΑ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

4.1. Επεξεργασία Φυσικής Γλώσσας

Η επεξεργασία φυσικής γλώσσας (Natural Language Processing ή NLP για συντομία) είναι ένας κλάδος της τεχνητής νοημοσύνης που επιτρέπει στους υπολογιστές να καταλαβαίνουν και να χειρίζονται την ανθρώπινη γλώσσα [22]. Η επεξεργασία φυσικής γλώσσας χρησιμοποιεί στοιχεία από πολλούς κλάδους, όπως η επιστήμη υπολογιστών και η μηχανογραφημένη γλωσσική επιστήμη (computational linguistics), με σκοπό να "κλείσει" το κενό που δημιουργείται ανάμεσα στην ανθρώπινη επικοινωνία και στην νόηση του υπολογιστή.

Η επεξεργασία φυσικής γλώσσας δεν είναι μία νέα επιστήμη, αλλά έχει γνωρίσει μεγάλη ανάπτυξη χάρης στο αυξημένο ενδιαφέρον ως προς την επικοινωνία μεταξύ ανθρώπου – μηχανής. Η επεξεργασία φυσικής γλώσσας βοηθάει τους υπολογιστές να επικοινωνούν με τους ανθρώπους στην γλώσσα τους. Είναι αυτή που καθιστά δυνατό σε έναν υπολογιστή να διαβάζει κείμενα, να ακούει και να κατανοεί ομιλία, να αναγνωρίζει συναισθήματα και να αποφασίζει ποια κομμάτια σε μία συνομιλία είναι σημαντικά και ποια όχι. Η δυσκολία που παρουσιάζεται στην επεξεργασία της φυσικής γλώσσας προέρχεται από την ίδια την φύση των ανθρώπινων γλωσσών. Οι ανθρώπινες γλώσσες είναι ιδιαίτερα περίπλοκες και ποικιλόμορφες, είτε έχουμε προφορική είτε γραπτή αποτύπωσή τους. Αρκεί να σκεφτούμε ότι κάθε γλώσσα έχει χιλιάδες δικούς της κανόνες γραμματικής και συντακτικού, ορολογία και αργκό.

Η επεξεργασία φυσικής γλώσσας, πλέον, έχει ξεπεράσει το στάδιο της δημιουργίας στατιστικών μεθόδων και στατιστικής μηχανικής μάθησης, αλλά προσπαθεί να καλύψει και την ανάγκη για μία συντακτική και σημασιολογική κατανόηση της εκάστοτε γλώσσας. Κάθε κείμενο που αναλύεται έχει τρεις διαφορετικές πλευρές, οι οποίες πρέπει να κατανοηθούν και να αναλυθούν.

4.1.1. Σημασιολογική Πλευρά

Η σημασιολογική πλευρά εξετάζει το συγκεκριμένο νόημα της κάθε λέξης του κειμένου. Παραδείγματος χάριν, η φράση "Εκείνος κρατούσε στο χέρι του ένα πολύ καλό φύλλο" μπορεί να έχει πολλά νοήματα, θα μπορούσαμε να αναφερόμαστε στον αναγνώστη μιας εφημερίδας, σε έναν βοτανολόγο ή σε έναν χαρτοπαίκτη. Το να γνωρίζει κάποιος το πλαίσιο στο οποίο αναφέρεται μια πρόταση είναι πολύ σημαντικό για την κατανόησή της. Αν η φράση ήταν "Εκείνος κρατούσε στο χέρι του ένα πολύ καλό φύλλο και αποφάσισε να ποντάρει" ο αναγνώστης καταλαβαίνει αμέσως ότι το φύλλο στο οποίο αναφερόμαστε είναι φύλλο τράπουλας. Χωρίς όμως τη μηχανική μάθηση της φυσικής γλώσσας ένας υπολογιστής δε θα μπορούσε να το καταλάβει.

4.1.2. Συντακτική Πλευρά

Η συντακτική πλευρά εξετάζει τη συντακτική δομή μίας πρότασης. Παραδείγματος χάριν έστω ότι έχουμε την φράση "Εκείνος εγκατέλειψε τους συμπαίκτες τους λίγο πριν τελειώσουν τα λεφτά". Το ρήμα "τελειώσουν" θα μπορούσε να έχει ως υποκείμενο είτε

τους συμπαίκτες είτε εκείνον, ανάλογα με τους πως θα το διαβάσει κανείς.

4.1.3. Συμφραζόμενη Πληροφορία

Τέλος, πρέπει να γίνεται κατανοητό το γενικό πλαίσιο, μέσα στο οποίο μια λέξη, φράση ή πρόταση εμφανίζεται. Παραδείγματος χάριν, μπορεί κάποιος να περιγράψει κάτι ως "άρρωστο". Γνωρίζουμε όμως, ότι η λέξη μπορεί να έχει θετική ή αρνητική χροιά. Αν αναφερόμαστε σε κάποιον άνθρωπο ως άρρωστο μπορούμε να καταλάβουμε ότι η λέξη έχει αρνητική χροιά. Αν αναφερόμαστε όμως σε μία ταινία ή σε ένα βίντεο-παιχνίδι η λέξη έχει θετική χροιά.

Ενώνοντας τελικά τις αναλύσεις που γίνονται γύρω από αυτές τις 3 πλευρές μπορούμε να πάρουμε πληροφορίες γύρω από την πρόταση που θα μας βοηθήσουν στην κατανόηση της. Ακόμα και αν αυτές οι πληροφορίες δεν είναι από μόνες του πολύ χρήσιμες, χρειάζεται ο συνδυασμός και των τριών πλευρών προκειμένου να καταλάβουμε την πρόταση.

Αυτή η ανάλυση επιτυγχάνεται μέσω μοντέλων μηχανικής μάθησης ή και με κανόνες που μπορούν να δοθούν σε ένα σύστημα για να τους χρησιμοποιήσει όταν αναλύει ένα κείμενο. Για να επιτύχουμε τα καλύτερα δυνατά αποτελέσματα, πρέπει να υπάρχει ένας συνδυασμός των δύο, καθώς η μηχανική μάθηση από μόνη της δυσκολεύεται να κατανοήσει το θέμα ενός κειμένου. Από την άλλη, στα αρχικά της χρόνια η επεξεργασία φυσικής γλώσσας βασιζόταν μόνο σε κανόνες και μοτίβα που εφαρμόζονταν στις προτάσεις. Η δημιουργία όμως και η εφαρμογή αυτών των κανόνων έπαιρνε πάρα πολύ χρόνο, καθώς έπρεπε να προσδιοριστούν με μεγάλη ακρίβεια. Επίσης, όσο εξελισσόταν η γλώσσα, καθώς αναφερόμαστε σε φυσικές, ζωντανές γλώσσες, οι κανόνες αδυνατούσαν να συμβαδίσουν με αυτήν.

Πλέον, τα σύγχρονα συστήματα επεξεργασίας φυσικής γλώσσας συνδυάζουν και τα δύο μοντέλα και αυτή η υβριδική προσέγγιση έχει φέρει πολύ καλύτερα αποτελέσματα.

4.2. Μοντέλο Bag-Of-Words

Το μοντέλο Bag-Of-Words είναι ένας δημοφιλής τρόπος εξαγωγής δεδομένων που χρησιμοποιείται στην επεξεργασία φυσικής γλώσσας. Το μοντέλο αυτό δημιουργεί μια απλοποιημένη και συμπυκνωμένη απεικόνιση του κειμένου. Το κείμενο, ή το σύνολο των κειμένων, αναπαρίσταται με ένα σύνολο αραιών διανυσμάτων. Στο μοντέλο αυτό, το κείμενο αντιμετωπίζεται ως ένας σάκος (bag) από λέξεις, χωρίς να λαμβάνεται υπόψιν η γραμματική, το συντακτικό του κειμένου ή την σειρά των λέξεων. Λαμβάνεται υπόψιν μόνο η παρουσία ή η συχνότητα εμφάνισης των λέξεων. Στην προκειμένη περίπτωση, αυτό που μας ενδιαφέρει είναι το σύνολο των κειμένων (ερωτήσεων) στο οποίο εμφανίζεται η εκάστοτε λέξη του λεξιλογίου μας.

Το bag-of-words μοντέλο είναι ευρέως διαδομένο για μεθόδους κατηγοριοποίησης κειμένου, όπου η συχνότητα εμφάνισης μιας λέξης χρησιμοποιείται για την εκπαίδευση ενός ταξινομητή. Παρόλα αυτά, το μοντέλο αυτό εμφανίζει και διάφορες αδυναμίες. Το

γεγονός ότι αντιμετωπίζει τις λέξεις ως ξεχωριστές οντότητες, αγνοώντας τις σχέσεις μεταξύ τους, όπως την σειρά με την οποία βρίσκονται μέσα σε ένα κείμενο, είναι μία από τις βασικότερες. [23]

Ένας τρόπος επιμέρους αντιμετώπισης του προβλήματος αυτό είναι το σπάσιμο της πρότασης σε n-grams. Η χρήση των n-grams, αντί για μεμονωμένων λέξεων, βοηθάει σε μία διατήρηση των σχέσεων μεταξύ των λέξεων. Η χρήση των n-grams, βέβαια, έχει το μειονέκτημα ότι αυξάνει ιδιαίτερα το μέγεθος του λεξιλογίου.

Παρά τα μειονεκτήματα που έχει το μοντέλο συνεχίζει να είναι ένας από τους πιο απλούς, και δημοφιλής, τρόπους στο πεδία της κατηγοριοποίησης κειμένου.

4.3. N-Grams

Ως n-gram ορίζεται μία ακολουθία n συνεχόμενων όρων από ένα κείμενο. Μια ακολουθία μπορεί να αποτελείται από συλλαβές, γράμματα ή και λέξεις. Τα n-grams χρησιμοποιούνται ευρέως στους τομείς των πιθανοτήτων, της επεξεργασίας δεδομένων και της επεξεργασίας φυσικής γλώσσας. Τα μοντέλα n-grams μπορούν να χρησιμοποιηθούν για οποιοδήποτε τύπο δεδομένων, από δορυφορικές εικόνες μέχρι μικρές σειρές DNA, γεγονός που τα κάνει ιδιαίτερα δημοφιλή. Τα n-grams έχουν αποδειχθεί πολύ αποτελεσματικά στην μοντελοποίηση δεδομένων κειμένου, με ιδιαίτερη εφαρμογή στην τεχνητή μετάφραση.

Παρόλα αυτά, το μοντέλο των n-grams παρουσιάζει και αυτό κάποιες αδυναμίες, με πιο σημαντική από αυτές το μικρό εύρος της εξάρτησης που έχουν μεταξύ τους τα n-grams. Επίσης αδυνατούν να αντιμετωπίσουν τη μεγάλη πολυπλοκότητα των φυσικών γλωσσών. Δύο προτάσεις με τις ίδιες λέξεις και το ίδιο νόημα παράγουν δύο πολύ διαφορετικά σύνολα n-grams. Για την αντιμετώπιση των προβλημάτων αυτών, οι εφαρμογές που χρησιμοποιούν n-grams συνήθως το κάνουν μαζί με μοντέλα Bayesian inference.

Το σπάσιμο ενός κειμένου σε grams γίνεται με τη χρήση ενός κυλιόμενου παραθύρου, μεγέθους n. Μια ακολουθία δύο όρων (λέξεων) ενός κειμένου ονομάζεται bigram, μια ακολουθία τριών trigram, ενώ μια ακολουθία μίας λέξη unigram κ.ο.κ. . Έστω, για παράδειγμα, η πρόταση “The quick brown fox jumps over the lazy dog”. Με τη χρήση bigrams, τα n-grams που θα προκύψουν είναι τα εξής:

- the quick
- quick brown
- brown fox
- fox jumps
- jumps over
- over the
- the lazy
- lazy dog

Ενώ με τη χρήση trigrams, τα n-grams που θα προκύψουν είναι τα εξής:

- the quick brown
- quick brown fox
- brown fox jumps
- fox jumps over
- jumps over the
- over the lazy
- the lazy dog

Για την χρήση των bigrams δημιουργήθηκε ένα καινούργιο bag-of-words, παρόμοιο με αυτό των unigrams. Στην ακολουθούμενη υλοποίηση χρησιμοποιήθηκαν bigrams.

4.4. Term Frequency – Inverse Document Frequency

Το μοντέλο Term Frequency – Inverse Document Frequency (tf-idf) είναι ένα μοντέλο στατιστικής απεικόνισης του κειμένου, βασισμένο πάνω στη συχνότητα εμφάνισης μιας λέξης, τόσο στο κείμενο που βρίσκεται, όσο και στο σύνολο των κειμένων. Η γενική λογική πίσω από το μοντέλο αυτό είναι ότι οι λέξεις που εμφανίζονται σε πάρα πολλά κείμενα και που ανήκουν σε πολλές διαφορετικές κατηγορίες, δεν είναι χρήσιμες για την κατηγοριοποίηση τους. Σκοπός αυτού το μοντέλου είναι να αποδίδει χαμηλό σκορ σε αυτές τις λέξεις. [24]

Το μοντέλο αυτό χρησιμοποιείται ευρέως και πέρα από την επεξεργασία φυσικής γλώσσας. Το μοντέλο αυτό έχει χρησιμοποιηθεί και για την κατηγοριοποίηση των αναφορών που μπορεί να περιέχει ένα paper.

Το μοντέλο αποτελείται από δύο όρους, το Term Frequency (tf) και το Inverse Document Frequency (idf).

- Term Frequency (tf): Μετράει πόσα συχνά ένας όρος συναντάται σε ένα κείμενο. Μιας και η συχνότητα εξαρτάται από το μέγεθος του κειμένου (πχ ένας όρος είναι πιο πιθανό να συναντηθεί περισσότερες φορές σε ένα μεγάλο κείμενο από ένα μικρό), το σύνολο των φορών που εμφανίζεται ο όρος διαιρείται με το σύνολο των λέξεων που περιέχει το κείμενο, ως τρόπος κανονικοποίησης. Η σχέση που χρησιμοποιείται για τον υπολογισμό του tf είναι η εξής:

$$\mathrm{TF}(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

- Inverse Document Frequency (idf): Ουσιαστικά μετράει πόσο σημαντικός είναι ένας όρος στην κατηγοριοποίηση του κειμένου. Όπως ειπώθηκε και παραπάνω, υπάρχουν όροι όπως “is”, “are”, “code” που μπορεί να εμφανίζονται σε πάρα πολλά κείμενα, αλλά έχουν λίγη σημασία για την κατηγοριοποίηση. Για αυτό το λόγο οφείλει να δοθεί βαρύνουσα σημασία στους όρους που εμφανίζονται λιγότερες φορές. Η σχέση που χρησιμοποιείται για τον υπολογισμό του idf είναι η εξής:

$$\text{IDF(t)} = \log_e\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it}\right)$$

Το διάνυσμα tf-idf υπολογίζεται ως το γινόμενο των δύο όρων.

Έστω ότι σε ένα κείμενο 100 λέξεων η λέξη fox εμφανίζεται 4 φορές. Αυτό σημαίνει ότι το term frequency της λέξης fox είναι $\frac{4}{100}$ = 0,04. Αν τώρα έχουμε ένα σύνολο 100.000 κειμένων στο οποίο η λέξη fox σε 100 από αυτά αυτό σημαίνει ότι το inverse term frequency της λέξης ισούται με $\log_{10}(\frac{100.000}{100})$ = 4. Συνεπώς το διάνυσμα tf-idf ισούται με 0,16.

4.5. Αναπαράσταση Αραιών Πινάκων

Οι αραιοί πίνακες (sparse matrix) είναι διανυσματικοί πίνακες, όπου τα περισσότερα στοιχεία τους είναι 0. Ένας πίνακας που τα περισσότερα στοιχεία του είναι μη-μηδενικά ονομάζεται πυκνός. Η αραιότητα του πίνακα, που ισούται με 1 μείον την πυκνότητα του πίνακα, είναι το σύνολο των μηδενικών στοιχείων διά το σύνολο των στοιχείων του πίνακα.

Ένας πίνακας, συνήθως αποθηκεύεται ως ένας πίνακας δύο διαστάσεων. Κάθε στοιχείο του πίνακα προσδιορίζεται από δύο δείκτες, τυπικά i και j, όπου αντιπροσωπεύουν τη γραμμή και τη στήλη όπου βρίσκεται το στοιχείο. Σε έναν αραιό, όμως, πίνακα η μνήμη που χρειάζεται για την αποθήκευση του μειώνεται σημαντικά αν αποθηκεύσουμε μόνο τα μη-μηδενικά του στοιχεία. Η λογική αυτή αποφέρει πολύ σημαντικά οφέλη τόσο στη μνήμη όσο και στην υπολογιστική ισχύ που χρειάζεται η επεξεργασία του πίνακα.

Υπάρχουν πολλοί τρόποι να αποθηκευτεί ένας αραιός πίνακας. Μερικοί από αυτούς είναι το Dictionary of Keys (DoK), που αποθηκεύει σε ένα dictionary το mapped pair των δύο δεικτών, και την τιμή του στοιχείου, το Coordinate List (COO), που αποθηκεύει σε μία λίστα τούπλες τριών στοιχείων τους δείκτες και την τιμή, και το List of lists (LIL), όπου αποτελείται από τρεις παράλληλες λίστες για τους δείκτες και την τιμή του στοιχείου.

5. ΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ

5.1. Συλλογή Δεδομένων

Το σύνολο των δεδομένων που χρησιμοποιήθηκε ως σύνολο εκπαίδευσης προήλθε από το Stack Overflow. Το Stack Overflow, όπως αναφέρθηκε και εισαγωγικά, είναι ένα site ερωτήσεων και απαντήσεων από και προς προγραμματιστές. Έχει δημιουργηθεί και τρέχει ως κομμάτι του Stack Exchange network.

Τα δεδομένα αποτελούνται από το 10% των ερωτήσεων από το σύνολο των προγραμματιστικών ερωτήσεων που υπήρχαν στο Stack Overflow, με ημερομηνίες δημιουργίας πριν από τις 20 Οκτωβρίου 2016, το οποίο παρέχεται από το Kaggle. Από το σύνολο των δεδομένων χρησιμοποιήθηκαν μόνο οι ερωτήσεις και τα tags που περιέχουν. Ο όγκος των δεδομένων ξεπερνούσε τα 2 GB σε μέγεθος. Ο αριθμός των ερωτήσεων ξεπερνά τις 1.264.212 ενώ περιλαμβάνονται πάνω από 35.000 μοναδικά tags. Κάθε ερώτηση περιλαμβάνει ένα ή και περισσότερα tags.

5.1.1. Αρχική Μορφή Δεδομένων

Τα δεδομένα στη μορφή που κατέβηκαν από το Kaggle βρίσκονταν σε μορφή αρχείου csv file format (comma-separated values). Τα csv αρχεία περιλαμβάνουν μια εγγραφή ανά σειρά και οι ενότητες της κάθε εγγραφής χωρίζονται με κόμματα.

Η κάθε εγγραφή ερώτησης περιλαμβάνει τις εξής ενότητες:

- Id - Αποκλειστικός κωδικός αναγνώρισης της ερώτησης, μοναδικός στο σύνολο των ερωτήσεων.
- OwnerUserId - Κωδικός αναγνώρισης του χρήστη που έθεσε την ερώτηση, μοναδικός ανά χρήστη αλλά όχι στο σύνολο των ερωτήσεων.
- CreationDate – Η ημερομηνία που ο χρήστης έθεσε την ερώτηση.
- ClosedDate – Η ημερομηνία που έκλεισε η ερώτηση, είτε γιατί απαντήθηκε.
- Score – Η βαθμολογία που συγκέντρωσε η ερώτηση από άλλους χρήστες.
- Title – Ο τίτλος της ερώτησης.
- Body – Το κύριο σώμα της ερώτησης.
-

Η κάθε εγγραφή tag περιλαμβάνει στις εξής ενότητες:

- Id - Αποκλειστικός κωδικός αναγνώρισης της ερώτησης, μοναδικός στο σύνολο των ερωτήσεων.
- OwnerUserId - Κωδικός αναγνώρισης του χρήστη που έθεσε την ερώτηση, μοναδικός ανά χρήστη αλλά όχι στο σύνολο των ερωτήσεων.
- CreationDate – Η ημερομηνία που ο χρήστης έθεσε την ερώτηση.
- ClosedDate – Η ημερομηνία που έκλεισε η ερώτηση, είτε γιατί απαντήθηκε.
- Score – Η βαθμολογία που συγκέντρωσε η ερώτηση από άλλους χρήστες.
- Title – Ο τίτλος της ερώτησης.
- Body – Το κύριο σώμα της ερώτησης.

Από τα αρχεία χρησιμοποιήθηκαν το αρχείο των ερωτήσεων (Questions.csv) και το αρχείο των tags (Tags.csv). Για την δημιουργία του συνόλου εκπαίδευσης δεν

χρειαστήκαν όλα τα δεδομένα που περιέχει η κάθε εγγραφή, συγκεκριμένα δεν χρειάστηκαν οι ημερομηνίες , ο κωδικός αναγνώρισης του χρήστη και η βαθμολογία της ερώτησης.

Ως κείμενο της κάθε ερώτησης χρησιμοποιήθηκε τόσο ο τίτλος όσο και το σώμα της ερώτησης.

5.1.2. Εισαγωγή Δεδομένων Σε Βάση Δεδομένων

Προκειμένου να αποφευχθεί το επαναλαμβανόμενο διάβασμα των δεδομένων από τα αρχεία, πριν την οποιαδήποτε επεξεργασία του κειμένου, οι εγγραφές εισάχθηκαν σε μια βάση δεδομένων. Στη βάση δεδομένων κατασκευάστηκαν 2 tables, ένα table Questions, στο οποίο εισάχθηκαν οι ερωτήσεις, και ένα table Tags, στο οποίο εισάχθηκαν οι εγγραφές των tags.

Για καθένα από τα δύο αρχεία, τα δεδομένα της κάθε εγγραφής χωρίστηκαν και περάστηκαν στα αντίστοιχα tables της βάσης, ένα για τις ερωτήσεις (Questions) και ένα για τα tags (Tags). Με τον τρόπο αυτό μειώθηκε σημαντικά ο χρόνος που χρειάζεται για το διάβασμα των δεδομένων. Από εδώ και πέρα, για κάθε συλλογή δεδομένων που θα χρειαζόμαστε το πρόγραμμα μπορεί να αναφέρεται στην βάση, αντί να ανατρέχει στα csv. Η δημιουργία βάσης έχει και το επιπρόσθετο πλεονέκτημα ότι μπορούμε να διατηρήσουμε, προσθέτοντας επιπλέον fields, το κείμενο σε διάφορες μορφές (πχ στην αρχική μορφή του, μετά τον καθαρισμό του από ανεπιθύμητα δεδομένα, αφού έχει γίνει stemmed, K.O.K.). Έτσι σε περίπτωση λάθους σε κάποια φάση της προεργασίας μπορεί εύκολα ο χρήστης να έχει πρόσβαση στην προηγούμενη φάση του κειμένου και να την διορθώσει.

Στην βάση εισήχθησαν, για την πληρότητα της εγγραφής, και τα στοιχεία που δεν χρησιμοποιήθηκαν για το σύνολο εκπαίδευσης. Επίσης σε κάθε εγγραφή αποδόθηκε ένα unique Id (Guid) προκειμένου να μπορούν να βρεθούνε στη βάση δεδομένων.

Για την εισαγωγή των στοιχείων από τα csv αρχεία στη βάση δεδομένων χρησιμοποιήθηκαν ως τεχνολογίες η C# και η SQL.

Κάθε ερώτηση αποθηκεύτηκε ως ξεχωριστή εγγραφή στη βάση, με τα ανάλογα columns. Όσον αφορά τα tags, επιλέχθηκε να εισαχθούν όχι ανά ερώτηση, αλλά ανά tag. Έτσι, η κάθε εγγραφή tag στο αντίστοιχο table, περιείχε τις ερωτήσεις στις οποίες εμφανιζόταν το αντίστοιχο tag.

Για την αποθήκευση των tags χρησιμοποιήθηκε η δομή του dictionary. Ως κλειδί του dictionary χρησιμοποιούταν το tag, ενώ ως τιμή το σύνολο των κωδικών αναγνώρισης των ερωτήσεων που εμφανιζόταν. Για κάθε tag που διαβαζόταν το πρόγραμμα ανέτρεχε στο dictionary των tags. Αν το tag υπήρχε ήδη πρόσθετε στην τιμή του τον αποκλειστικός κωδικός αναγνώρισης της ερώτησης. Το dictionary αυτό αποθηκεύτηκε τελικά στον πίνακα Tags. Το σύνολο των ερωτήσεων περιείχε πάνω από 30.000 μοναδικά Tags.

5.2. Καθαρισμός του κειμένου των ερωτήσεων

Προκειμένου να μπορέσουμε να χρησιμοποιήσουμε τις ερωτήσεις ως σύνολο εκπαίδευσης για τα συστήματα έπρεπε να περάσουν πρώτα από μια διαδικασία καθαρισμού και προεργασίας του κειμένου που περιείχαν. Οι ερωτήσεις του dataset βρίσκονταν στην HTML μορφή τους, όπως βρίσκονται και στο site του Stack Overflow. Για τον καθαρισμό των ερωτήσεων χρησιμοποιήθηκε η C#.

Κάθε ερώτηση διαβαζόταν από τη βάση δεδομένων που δημιουργήθηκε στο προηγούμενο βήμα μαζί με το unique Id της.

Πριν ξεκινήσει η διαδικασία καθαρισμού του κειμένου των ερωτήσεων έπρεπε να αφαιρεθούν τα HTML Tags και τα HTML Entities από το κείμενο. Για την αφαίρεση των HTML Entities χρησιμοποιήθηκε μια λίστα που περιέχει όλα τα γνωστά HTML Entities. Το πρόγραμμα ψάχνει να βρει στο κείμενο κάποιο Entity που να περιέχεται στη λίστα και να βρίσκεται ανάμεσα από τους χαρακτήρες & και ;. Αν βρεθεί κάποιο Entity το αντικαθιστά με κενό. Η αντικατάσταση γινόταν με κενό προκειμένου να αποφευχθεί η ένωση λέξεων που χωριζόταν με Entities. Για τα HTML Tags η διαδικασία ήταν πιο περίπλοκη, καθώς υπάρχουν tags τα οποία περιέχουν ανάμεσα τους κείμενο το οποίο θα πρέπει να αφαιρεθεί. Συγκεκριμένα το HTML Tag <code> χρησιμοποιείται για την συμπερίληψη και μορφοποίηση κώδικα μέσα στις ερωτήσεις. Το κείμενο αυτό έπρεπε να αφαιρεθεί και να μην συμπεριληφθεί στο λεξιλόγιο για το εκπαιδευτικό σετ. Ο κώδικας που περιείχε μια ερώτηση απομονώθηκε, προκειμένου να δημιουργηθεί ένα ξεχωριστό λεξιλόγιο για τον κώδικα, και κρατήθηκε σε ένα ξεχωριστό column της εγγραφής, με την ονομασία Code. Η επεξεργασία που έγινε στον κώδικα καθώς και η δημιουργία του λεξιλογίου του κώδικα εξηγούνται παρακάτω. Το κείμενο που βρισκόταν ανάμεσα στο <a href> tag, που αναφέρεται σε συνδέσμους και υπερ-συνδέσμους μεταξύ σελίδων, αφαιρέθηκε μαζί με τα αντίστοιχα tags. Τα υπόλοιπα tags αφαιρέθηκαν χρησιμοποιώντας το εξής Regular Expression "`<.*?>| &.*?;`" και αντικαθιστώντας τα με κενό. Η αντικατάσταση και εδώ γινόταν με κενό προκειμένου να αποφευχθεί η ένωση λέξεων. Τα πολλαπλά κενά που μπορεί να προέκυψαν στο τελικό κείμενο αντικαταστάθηκαν με ένα.

Για την διαδικασία του καθαρισμού του κειμένου των ερωτήσεων ακολουθήθηκαν τα εξής βήματα. Αρχικά το κείμενο μετατράπηκε όλο σε μικρά γράμματα, χρησιμοποιώντας την ρουτίνα string.ToLower(). Από το κείμενο αφαιρέθηκαν όλα τα νούμερα και τα σημεία στίξης, χρησιμοποιώντας τις ρουτίνες ελέγχου της C# char.IsPunctuation() και char.IsNumber() που ελέγχουν αν ένας χαρακτήρας είναι νούμερο ή σημείο στίξης. Αν ήταν το πρόγραμμα τον αντικαθιστούσε με κενό. Η αντικατάσταση γινόταν με κενό προκειμένου να αποφευχθεί η ένωση λέξεων που χωριζόταν με σημεία στίξης. Αν στο τελικό κείμενο εμφανίζονταν πολλαπλά κενά αντικαθίσταται με ένα.

Από το κείμενο αφαιρέθηκαν επίσης όσες λέξεις αποτελούνταν από ένα γράμμα, εκτός και αν αυτό ήταν "c", καθώς αναφέρεται στην γλώσσα προγραμματισμού. Επίσης αφαιρέθηκαν όλοι οι μη λατινικοί χαρακτήρες, ώστε να μην προκύψουν στο λεξιλόγιο λέξεις από άλλες γλώσσες. Συγκεκριμένα πολλές ερωτήσεις περιείχαν κινέζικους, ταιλανδέζικους και ιαπωνικούς χαρακτήρες.

Τέλος, αφαιρέθηκαν από το κείμενο όλα τα stop words καθώς εμφανίζονται με μεγάλη συχνότητα και δεν προσφέρουν κάποια επιπλέον πληροφορία στο ήδη υπάρχον

κείμενο.

Το καθαρισμένο κείμενο της ερώτησης αποθηκεύτηκε, χρησιμοποιώντας το unique Id της ερώτησης, στο καινούργιο Field της εγγραφής με το όνομα “Cleaned” του πίνακα Questions της βάσης. Το μέγεθος του λεξιλογίου που προέκυψε, μετά και τον καθαρισμό του συνόλου των ερωτήσεων, ήταν 176.619 λέξεις.

5.2.1. Αποκοπή Καταλήξεων

Στο καθαρισμένο κείμενο των ερωτήσεων εφαρμόστηκε η τεχνική της αποκοπής καταλήξεων από τις λέξεις, προκειμένου να μειώσουμε περαιτέρω το μέγεθος του λεξιλογίου. Για την αποκοπή των καταλήξεων των λέξεων χρησιμοποιήθηκε ο αλγόριθμος Porter2-stemmer [1]. Κάθε ερώτηση, μετά τον καθαρισμό της, υποβαλλόταν και στη διαδικασία της αποκοπής καταλήξεων.

Η ερώτηση, μετά και την αποκοπή των καταλήξεων, αποθηκευόταν στη βάση στο καινούργιο Field της εγγραφής με το όνομα “Stemmed” του πίνακα Questions της βάσης. Το μέγεθος του λεξιλογίου που προέκυψε, μετά και την αποκοπή των καταλήξεων, ήταν 147.432 λέξεις.

5.3. Δημιουργία Λεξιλογίου Κειμένου

Για την δημιουργία του λεξιλογίου χρησιμοποιήθηκε το σύνολο των καθαρών ερωτήσεων, όπως αυτό βρισκόταν αποθηκευμένο στη βάση δεδομένων, στο πεδίο “Stemmed” του πίνακα Questions της βάσης.

Προκειμένου να δημιουργηθεί το λεξιλόγιο δημιουργήθηκε ένα dictionary, το οποίο σαν κλειδί είχε λέξεις και σαν τιμές αριθμό εμφάνισης. Κάθε ερώτηση χωρίστηκε στις λέξεις που αποτελούταν. Σε περίπτωση που μια λέξη εμφανιζόταν πολλές φορές σε μια ερώτηση κρατιόταν μόνο μια φορά, χρησιμοποιώντας τη συνάρτηση string.Distinct() της C#. Για κάθε λέξη της ερώτησης το πρόγραμμα ανέτρεχε στο dictionary. Αν η λέξη υπήρχε ήδη, τότε ανέβαζε τον αριθμό εμφάνισης κατά 1. Αν δεν υπήρχε πρόσθετε στο dictionary μια νέα εγγραφή με τη λέξη και αριθμό εμφάνισης 1.

Για αυξήσουμε την ευκολία πρόσβασης και την ταχύτητα ανάκτησης δεδομένων το λεξιλόγιο, στην καθαρή μορφή του, αποθηκεύτηκε στον πίνακα BagOfWords. Κάθε εγγραφή του πίνακα BagOfWords περιέχει τη λέξη και τον αριθμό εμφάνισης της. Από το λεξιλόγιο αφαιρέθηκαν όσες λέξεις είχαν αριθμό εμφάνισης μικρότερο του 3 καθώς θα έδιναν ελάχιστα παραπάνω στοιχεία για την κάθε ερώτηση. Το μέγεθος του τελικού λεξιλογίου, μετά και την αφαίρεση των σπάνιων λέξεων ήταν 147.432 λέξεις.

Δημιουργήθηκαν επίσης, ένα μικρότερο λεξιλόγιο προκειμένου να δούμε πιο λεξιλόγιο δίνει τα πιο ακριβή αποτελέσματα. Δημιουργήθηκε ένα λεξιλόγιο που

[1] https://github.com/nemec/porter2-stemmer

χρησιμοποιούσε τις λέξεις με συχνότητα μεγαλύτερη ή ίση του 10. Το μέγεθος του λεξιλογίου αυτού ήταν 37.171 λέξεις.

5.4. Καθαρισμός κώδικα των ερωτήσεων

Για τον καθαρισμό του κώδικα που περιείχε η κάθε ερώτηση ακολουθήθηκε παρόμοια διαδικασία με τον καθαρισμό του κειμένου. Ο κώδικας συλλέχθηκε στο βήμα το καθαρισμού του κειμένου των ερωτήσεων και είχε αποθηκευτεί στη βάση, στο column Code. Η διαδικασία που ακολουθήθηκε για τον καθαρισμό του κώδικα ήταν παρόμοια με τον καθαρισμό του κειμένου. Τα σημεία στίξης και εδώ αντικαταστάθηκαν με κενό, με σκοπό να χωριστούν οι μεταβλητές από τα functions (πχ mystring.IsNullOrEmpty()).

5.5. Δημιουργία Λεξιλογίου Κώδικα

Για την δημιουργία του λεξιλογίου κώδικα χρησιμοποιήθηκε αντίστοιχη διαδικασία με τη δημιουργία λεξιλογίου κειμένου. Για την αφαίρεση των μεταβλητών από το λεξιλόγιο χρησιμοποιήθηκε πάλι ο αριθμός εμφάνισης της λέξης στο σύνολο των ερωτήσεων. Αν μια λέξη εμφανιζόταν σε λιγότερο από 3 κείμενα θεωρήθηκε μεταβλητή και αφαιρέθηκε.

Το τελικό μέγεθος του λεξιλογίου του κώδικα ήταν 141.667 λέξεις. Επίσης δημιουργήθηκε ένα λεξιλόγιο που χρησιμοποιούσε μόνο τις λέξεις με συχνότητα μεγαλύτερη ή ίση του 10. Το μέγεθος του λεξιλογίου αυτού ήταν 53.782 λέξεις.

5.6. Δημιουργία Διανυσμάτων Tf-Idf κειμένου

Κάθε ερώτηση του συνόλου εκπαίδευσης μετατράπηκε σε διάνυσμα tf-idf. Για την μετατροπή των ερωτήσεων σε διανύσματα ακολουθήθηκε η εξής διαδικασία. Αρχικά διαβάστηκε από τον πίνακα της βάσης BagOfWords το λεξιλόγιο, μαζί με τον αριθμό εμφάνισης της κάθε λέξης, και αποθηκεύτηκε σε ένα dictionary, με τρόπο αντίστοιχο με αυτόν που περιεγράφηκε στο προηγούμενο βήμα.

Κάθε ερώτηση που βρισκόταν στη βάση διαβάστηκε από το Field “Stemmed” και χωρίστηκε στις λέξεις από τις οποίες αποτελούταν. Για να υπολογιστεί ο αριθμός εμφάνισης της κάθε λέξης στην ερώτηση χρησιμοποιήθηκε ξανά η δομή του dictionary. Για κάθε λέξη της ερώτησης το πρόγραμμα ανέτρεχε στο dictionary. Αν η λέξη υπήρχε ήδη, τότε ανέβαζε τον αριθμό εμφάνισης κατά 1. Αν δεν υπήρχε πρόσθετε στο dictionary μια νέα εγγραφή με τη λέξη και αριθμό εμφάνισης 1. Το Term Frequency της κάθε λέξης υπολογίστηκε από το dictionary της ερώτησης ενώ το Inverse Document Frequency από το dictionary του BagOfWords.

5.6.1. Δημιουργία Τελικού Αρχείου Κειμένου

Λόγω του μεγάλου όγκου του λεξιλογίου του κειμένου επιλέχθηκε η δομή του αραιού "sparse" πίνακα. Έτσι αντί να δημιουργηθούν διανύσματα με πάνω από 100.000 στοιχεία το καθένα δημιουργήθηκαν διανύσματα για έναν sparse πίνακα. Στη δομή του αραιού πίνακα αντί να κρατείται όλο το διάνυσμα κρατούνται μόνο τα μη μηδενικά στοιχεία. Κάθε διάνυσμα του πίνακα αποτελείται από 3 στήλες. Η πρώτη στήλη εκπροσωπεύει τον αριθμό της ερώτησης (counter), η δεύτερη το index της λέξης στο λεξιλόγιο και η τρίτη την τιμή του tf-idf διανύσματος.

Για κάθε λέξη της κάθε ερώτησης του συνόλου εκπαίδευσης δημιουργήθηκε μια εγγραφή με τα τρία αυτά στοιχεία στο FinalFile_Text.csv.

5.7. Δημιουργία Διανυσμάτων Tf-Idf Κώδικα

Ο κώδικας της κάθε ερώτησης του συνόλου εκπαίδευσης μετατράπηκε σε διάνυσμα tf-idf. Για την μετατροπή του κώδικα σε διανύσματα ακολουθήθηκε η ίδια διαδικασία με το κείμενο των ερωτήσεων, αλλά χρησιμοποιήθηκε το λεξιλόγιο του κώδικα.

5.7.1. Δημιουργία Τελικού Αρχείου Κώδικα

Λόγω του μεγάλου όγκου του λεξιλογίου του κώδικα επιλέχθηκε η δομή του αραιού "sparse" πίνακα. Η διαδικασία που ακολουθήθηκε για τη δημιουργία του πίνακα ήταν αντίστοιχη με αυτή που περιγράφηκε για το τελικό αρχείο κειμένου.

Για κάθε λέξη της κάθε ερώτησης του εκπαιδευτικού σετ δημιουργήθηκε μια εγγραφή με τα τρία αυτά στοιχεία στο FinalFile_Code.csv.

5.8. Δημιουργία Τελικού Αρχείου Tags

Για τη δημιουργία του τελικού αρχείου Tags χρησιμοποιήθηκε ο πίνακας της βάσης Tags, χρησιμοποιώντας όμως μόνο τα 20 πιο δημοφιλή tags και όχι το σύνολο των tags.

Από τα δεδομένα του πίνακα δημιουργήθηκε ένα dictionary, που σαν κλειδί είχε τον κωδικό αναγνώρισης της κάθε ερώτησης και σαν τιμή τα tags που είχε η κάθε ερώτηση. Από την δομή αυτού του dictionary δημιουργήθηκε το τελικό αρχείο FinalFile_Tags.csv που χρησιμοποιήθηκε ως δεύτερη παράμετρος για τα συστήματα.

Στο αρχείο αυτό δημιουργήθηκε μια γραμμή για κάθε ερώτηση του συνόλου εκπαίδευσης, που κάθε μία αποτελούταν από ένα διάνυσμα με 20 στοιχεία. Αν η ερώτηση περιείχε το συγκεκριμένο tag η τιμή στο συγκεκριμένο index ήταν 1, αλλιώς ήταν 0.

Λόγω του μικρού αριθμού των tags δεν κρίθηκε απαραίτητο να χρησιμοποιηθεί

αραιός πίνακας, ανάλογος με αυτόν των ερωτήσεων.

6. ΠΕΙΡΑΜΑΤΙΚΗ ΔΙΑΔΙΚΑΣΙΑ

6.1. Πειραματική Διαδικασία

Στο κεφάλαιο αυτό θα εξεταστεί η πειραματική διαδικασία που ακολουθήθηκε με σκοπό την εξαγωγή συμπερασμάτων γύρω από την κατηγοριοποίηση των ερωτήσεων του Stack Overflow. Λόγω του μεγάλου όγκου δεδομένων οι απαιτήσεις σε υπολογιστική ισχύ αλλά και στον χρόνο εκτέλεσης ήταν ιδιαίτερα αυξημένες. Τα πειράματα έγιναν σε υπολογιστή με 11,7 GB Ram και επεξεργαστή Inter Core i7.

Τα προγράμματα που αναπτύχθηκαν γράφτηκαν σε Python. Η Python επιλέχθηκε με κριτήριο τον μεγάλο όγκο βιβλιοθηκών που υπάρχουν για την κατηγοριοποίηση κειμένου στην Python αλλά και λόγω της απλότητας και ευκολίας του κώδικα. Όλοι οι αλγόριθμοι υλοποιήθηκαν με τη βοήθεια του sklearn module[2] και της βιβλιοθήκης scikit-multilearn[3]. Παρακάτω παρουσιάζονται οι παράμετροι που χρησιμοποιήθηκαν για τον κάθε κατηγοριοποιητή.

Sklearn.tree.DecisionTreeClassifier

1. criterion='gini'
2. splitter='best'
3. max_depth=None,
4. min_samples_split=2,
5. min_samples_leaf=1
6. *min_weight_fraction_leaf=0.0*
7. max_features=None
8. random_state=0
9. max_leaf_nodes=None
10. min_impurity_decrease=0.0
11. min_impurity_split=None
12. class_weight=None
13. presort=False

Sklearn.tree.RandomForestClassifier

1. *n_estimators=10*
2. *criterion='gini'*
3. *max_depth=None*
4. *min_samples_split=2*
5. *min_samples_leaf=1*
6. *min_weight_fraction_leaf=0.0*
7. *max_features='auto'*
8. *max_leaf_nodes=None*
9. *min_impurity_decrease=0.0*
10. *min_impurity_split=None*

[2] http://scikit-learn.org/stable/

[3] http://scikit.ml/

11. *bootstrap=True*
12. *oob_score=False*
13. *n_jobs=1*
14. *random_state=None*
15. *verbose=0*
16. *warm_start=False*
17. *class_weight=None*

Sklearn.ensemble.MLPClassifier

1. *hidden_layer_sizes=(100,100,100)*
2. *activation='relu'*
3. *solver='sgd'*
4. *alpha=0.0001*
5. *batch_size='auto'*
6. *learning_rate='constant'*
7. *learning_rate_init=0.001*
8. *power_t=0.5*
9. *max_iter=100*
10. *shuffle=True*
11. *random_state=21*
12. *tol=0.005*
13. *verbose=10*
14. *warm_start=False*
15. *momentum=0.9*
16. *nesterovs_momentum=True*
17. *early_stopping=False*
18. *validation_fraction=0.1*
19. *beta_1=0.9*
20. *beta_2=0.999*
21. *epsilon=1e-08*

Sklearn.neighbors.KNeighborsClassifier

1. *n_neighbors=5*
2. *weights='uniform'*
3. *algorithm='auto'*
4. *leaf_size=30*
5. *p=2*
6. *metric='minkowski'*
7. *metric_params=None*
8. *n_jobs=1*

Sklearn.svm.SVC

1. *C=1.0*
2. *kernel='rbf'*
3. *degree=3*
4. *gamma='auto'*

5. *coef0=0.0*
6. *shrinking=True*
7. *probability=False*
8. *tol=0.001*
9. *cache_size=200*
10. *class_weight=None*
11. *verbose=False*
12. *max_iter=-1*
13. *decision_function_shape='ovr'*
14. *random_state=None*

Sklearn.svm.LinearSVC

1. *penalty='l2'*
2. *loss='squared_hinge'*
3. *dual=True*
4. *tol=0.0001*
5. *C=1.0*
6. *multi_class='ovr'*
7. *fit_intercept=True*
8. *intercept_scaling=1*
9. *class_weight=None*
10. *verbose=0*
11. *random_state=None*
12. *max_iter=1000*

Sklearn.naive_bayes.MultinomialNB

1. alpha=1.0
2. fit_prior=True
3. class_prior=None

Επίσης χρησιμοποιήθηκαν οι εξής μέθοδοι μετασχηματισμού προβλήματος:

- BinaryRelevance
- LabelPowerset
- LabelPowerset With Clusterer

6.2. Προσέγγιση

Ο σκοπός αυτής της διπλωματικής είναι να δούμε ποια μεθοδολογία μας δίνει τα καλύτερα αποτελέσματα. Για να το επιτύχουμε δοκιμάστηκαν διαφορετικές τεχνικές.

Ο κάθε αλγόριθμος εκτελέστηκε αρχικά χωρίς κάποια μέθοδο μετασχηματισμού και μετά με κάθε μία από αυτές, προκειμένου να μπορέσουμε να συγκρίνουμε τα αποτελέσματα που προκύπτουν από την κάθε μια.

Οι αλγόριθμοι εκτελέστηκαν σε τρία διαφορετικά εκπαιδευτικά σετ. Αρχικά σαν εκπαιδευτικό σετ χρησιμοποιήθηκε μόνο το κείμενο, μετά την προεξεπεργασία που περιεγράφηκε στο προηγούμενο κεφάλαιο, των ερωτήσεων – χωρίς τον κώδικα που αυτές συμπεριλαμβάνανε. Ύστερα, χρησιμοποιήθηκε μόνο ο κώδικας των ερωτήσεων, χωρίς το κείμενο. Τέλος χρησιμοποιήθηκε τόσο το κείμενο όσο και ο κώδικας των ερωτήσεων.

Σκοπός της επεξεργασίας αυτής είναι να δούμε αν προκύπτουν καλύτερα αποτελέσματα από την αφαίρεση του κώδικα από τις ερωτήσεις, αν ο κώδικας από μόνος του μας δίνει επαρκής πληροφορίες για την σωστή κατηγοριοποίηση του κειμένου και τέλος αν το συνδυασμένο κείμενο που προκύπτει από τα δύο βελτιώνει τα αποτελέσματά μας. Ταυτόχρονα οι αλγόριθμοι δοκιμάστηκαν σε δύο διαφορετικά σετ λεξιλογίων, όπως εξηγήθηκε και στην προηγούμενη ενότητα, ένα μικρότερο και ένα μεγαλύτερο προκειμένου να δούμε πιο παράγει τα καλύτερα αποτελέσματα.

Τέλος, δοκιμάστηκε, σε όλους τους συνδυασμούς αλγορίθμων και μεθόδων μετασχηματισμού η χρήση του Variance Threshold. Η χρήστη του Variance Threshold αφαιρεί όλα εκείνα τα δεδομένα, η απόκλιση των οποίων, σε σχέση με τα υπόλοιπα δεδομένα, δεν ξεπερνάει ένα κατώφλι, το οποίο έχουμε ορίσει εμείς εκ των προτέρων.

Να αναφερθεί εδώ ότι αν και ο αλγόριθμος k-NN έχει συμπεριληφθεί δεν καταφέραμε να εξάγουμε αποτελέσματα, καθώς η εκτέλεση του είχε ιδιαίτερα αυξημένες υπολογιστικές απαιτήσεις. Επίσης να υπογραμμιστεί, ότι πως αναφέρθηκε και παραπάνω οι αλγόριθμοι SVM, Linear SVM και Naive Bayes δεν έχουν προσαρμοστεί για multi-label δεδομένα και άρα δεδομένα υπάρχουν μόνο με κάποια μέθοδο μετασχηματισμού.

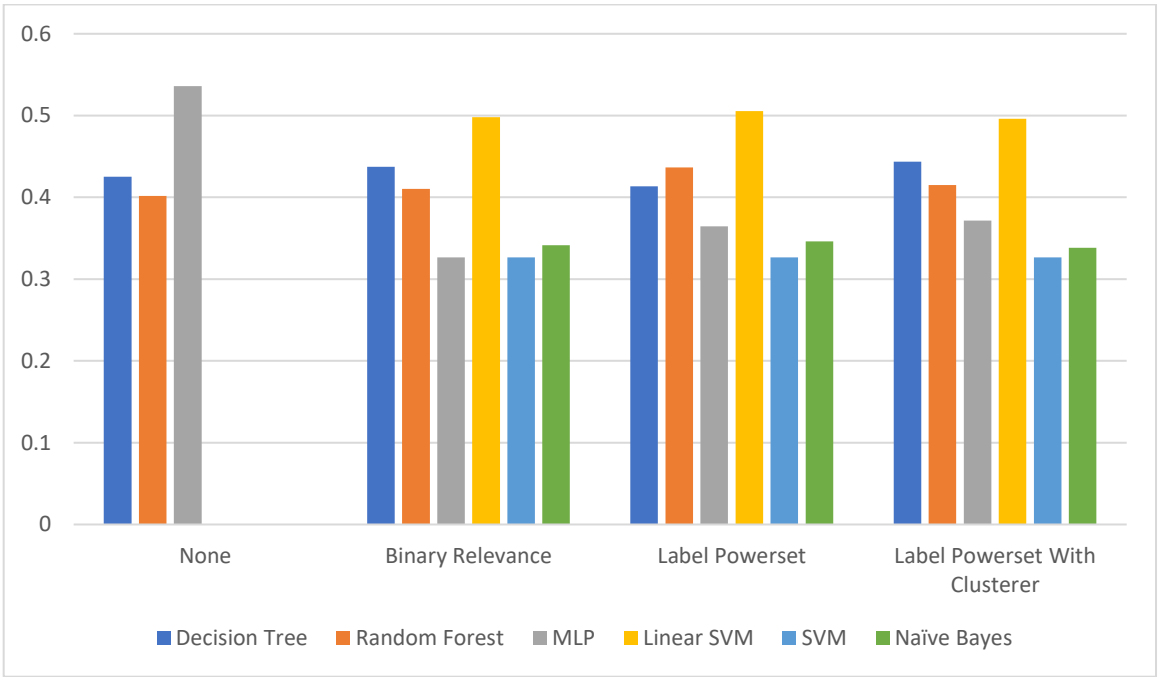
6.3. Αποτελέσματα

6.3.1. Χρήση λεξιλογίου με συχνότητα πάνω από 2 χωρίς Variance Threshold

Αρχικά ως εκπαιδευτικό σετ δοκιμάστηκε μόνο το κείμενο των ερωτήσεων. Για τα δεδομένα αυτά λάβαμε τα εξής αποτελέσματα:

	Decision Tree	Random Forest	MLP	Linear SVM	SVM	Naïve Bayes
None	0.42539	0.40165	0.53597	0	0	0
Binary Relevance	0.4372	0.41038	0.32657	0.4981	0.32657	0.3414
Label Powerset	0.41357	0.43676	0.36473	0.50535	0.32657	0.34614
Label Powerset With Clusterer	0.4436	0.41505	0.37152	0.49589	0.32657	0.33821

(Αποτελέσματα των ταξινομητών με τη χρήση μόνο του κειμένου των ερωτήσεων)

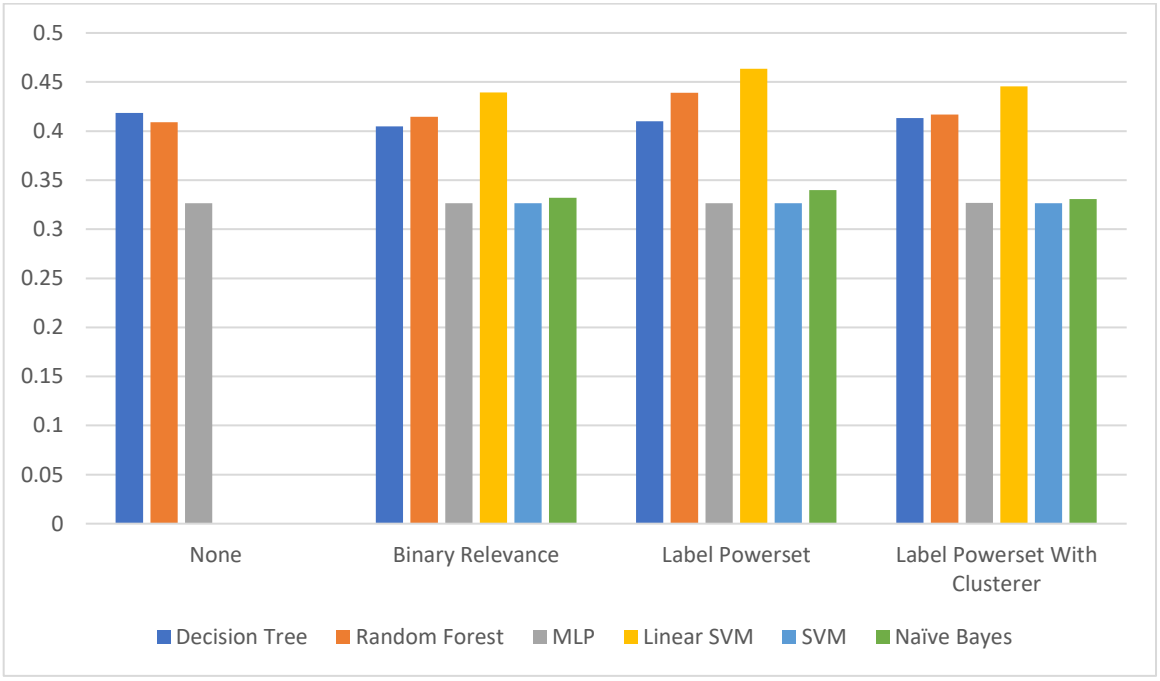


(Γραφική απεικόνιση των αποτελεσμάτων)

Αντίστοιχα, δοκιμάστηκε και ως εκπαιδευτικό σετ μόνο ο κώδικας που περιλαμβάναν οι ερωτήσεις. Τα αποτελέσματα ήταν τα εξής:

	Decision Tree	Random Forest	MLP	Linear SVM	SVM	Naïve Bayes
None	0.41865	0.40901	0.32657	0	0	0
Binary Relevance	0.40468	0.41461	0.32657	0.43945	0.32657	0.33203
Label Powerset	0.40998	0.43894	0.32657	0.46362	0.32657	0.34
Label Powerset With Clusterer	0.41314	0.41699	0.32693	0.44556	0.32657	0.33068

(Αποτελέσματα των ταξινομητών με τη χρήση μόνο του κώδικα των ερωτήσεων)

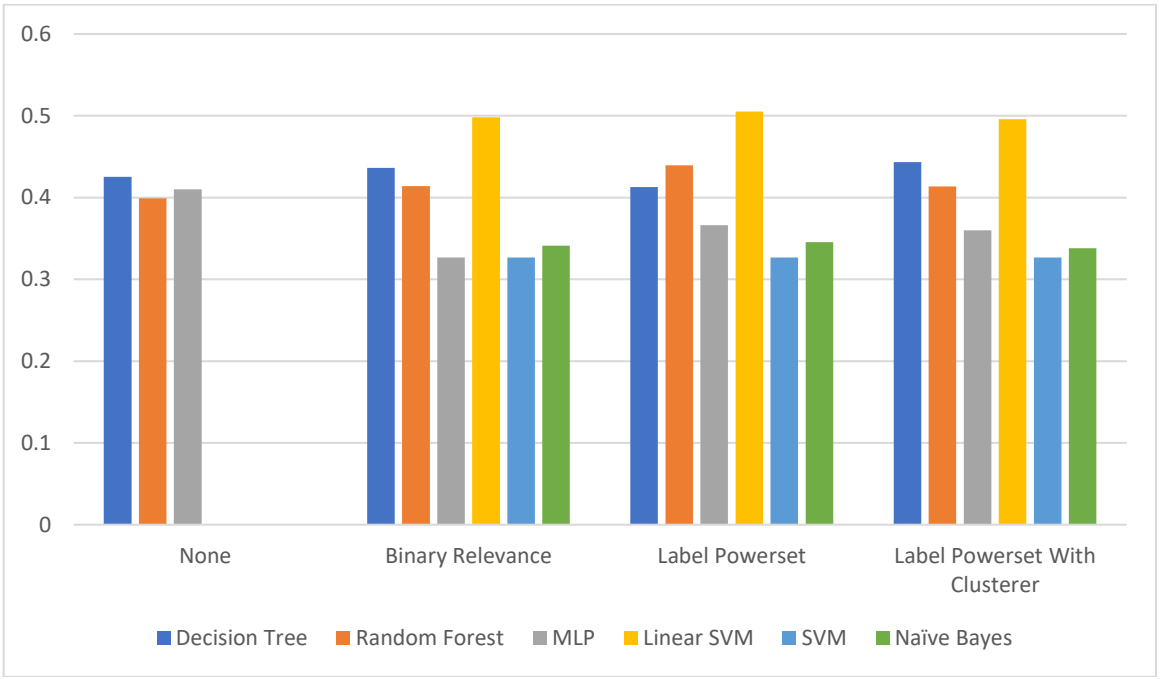


(Γραφική απεικόνιση των αποτελεσμάτων)

Τέλος ως εκπαιδευτικό σετ χρησιμοποιήθηκε τόσο ο κώδικας όσο και το κείμενο των ερωτήσεων. Τα αποτελέσματα που πήραμε ήταν τα εξής:

	Decision Tree	Random Forest	MLP	Linear SVM	SVM	Naïve Bayes
None	0.42531	0.39925	0.41004	0	0	0
Binary Relevance	0.43627	0.41398	0.32657	0.4981	0.32657	0.341
Label Powerset	0.4126	0.43957	0.36605	0.50535	0.32657	0.34553
Label Powerset With Clusterer	0.4433	0.41357	0.36013	0.4959	0.32657	0.33809

(Αποτελέσματα των ταξινομητών με τη χρήση του κώδικα και του κειμένου των ερωτήσεων)



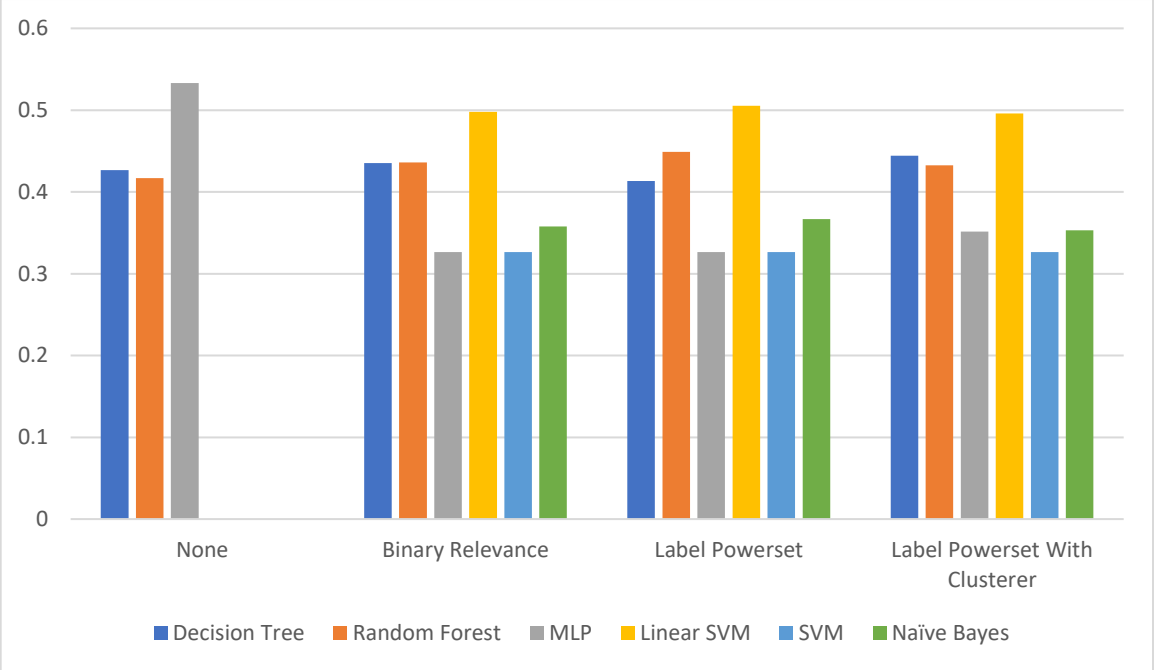
(Γραφική απεικόνιση των αποτελεσμάτων)

6.3.2. Χρήση λεξιλογίου με συχνότητα πάνω από 2 με Variance Threshold

Αρχικά ως εκπαιδευτικό σετ δοκιμάστηκε μόνο το κείμενο των ερωτήσεων. Για τα δεδομένα αυτά λάβαμε τα εξής αποτελέσματα:

	Decision Tree	Random Forest	MLP	Linear SVM	SVM	Naïve Bayes
None	0.42678	0.41705	0.53321	0	0	0
Binary Relevance	0.43527	0.43588	0.32657	0.4981	0.32657	0.35777
Label Powerset	0.41338	0.44912	0.32657	0.50535	0.32657	0.36687
Label Powerset With Clusterer	0.44447	0.43261	0.35148	0.49589	0.32657	0.35322

(Αποτελέσματα των ταξινομητών με τη χρήση μόνο του κειμένου των ερωτήσεων)

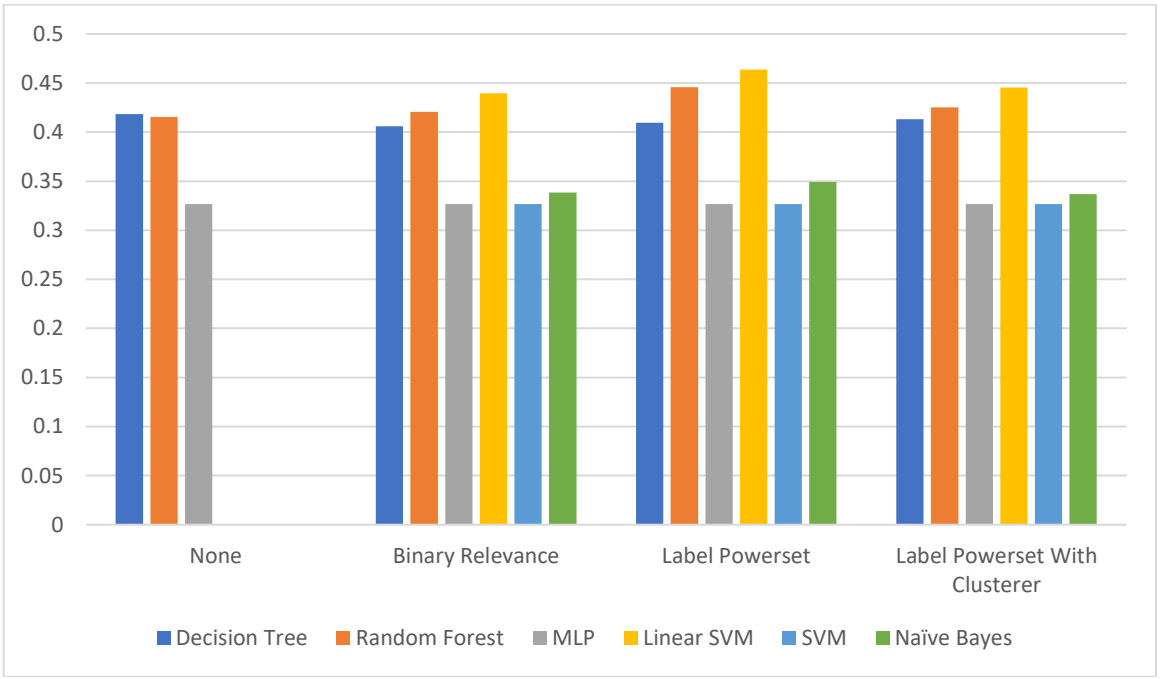


(Γραφική απεικόνιση των αποτελεσμάτων)

Αντίστοιχα, δοκιμάστηκε και ως εκπαιδευτικό σετ μόνο ο κώδικας που περιλαμβάναν οι ερωτήσεις. Τα αποτελέσματα ήταν τα εξής:

	Decision Tree	Random Forest	MLP	Linear SVM	SVM	Naïve Bayes
None	0.41848	0.41537	0.32657	0	0	0
Binary Relevance	0.40609	0.42069	0.32657	0.43945	0.32657	0.33842
Label Powerset	0.40958	0.44581	0.32657	0.4636	0.32657	0.34923
Label Powerset With Clusterer	0.41329	0.42522	0.32685	0.44556	0.32657	0.33692

(Αποτελέσματα των ταξινομητών με τη χρήση μόνο του κώδικα των ερωτήσεων)

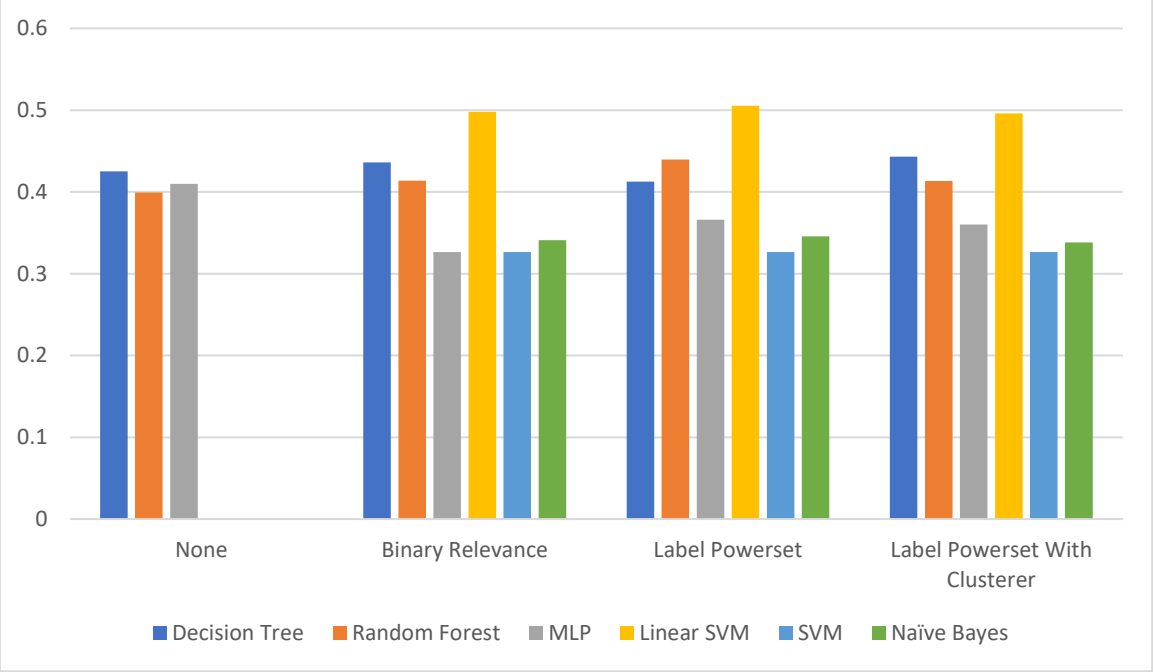


(Γραφική απεικόνιση των αποτελεσμάτων)

Τέλος ως εκπαιδευτικό σετ χρησιμοποιήθηκε τόσο ο κώδικας όσο και το κείμενο των ερωτήσεων. Τα αποτελέσματα που πήραμε ήταν τα εξής:

	Decision Tree	Random Forest	MLP	Linear SVM	SVM	Naïve Bayes
None	0.42531	0.39925	0.41004	0	0	0
Binary Relevance	0.43627	0.41398	0.32657	0.4981	0.32657	0.341
Label Powerset	0.4126	0.43957	0.36605	0.50535	0.32657	0.34553
Label Powerset With Clusterer	0.4433	0.41357	0.36013	0.4959	0.32657	0.33809

(Αποτελέσματα των ταξινομητών με τη χρήση του κώδικα και του κειμένου των ερωτήσεων)



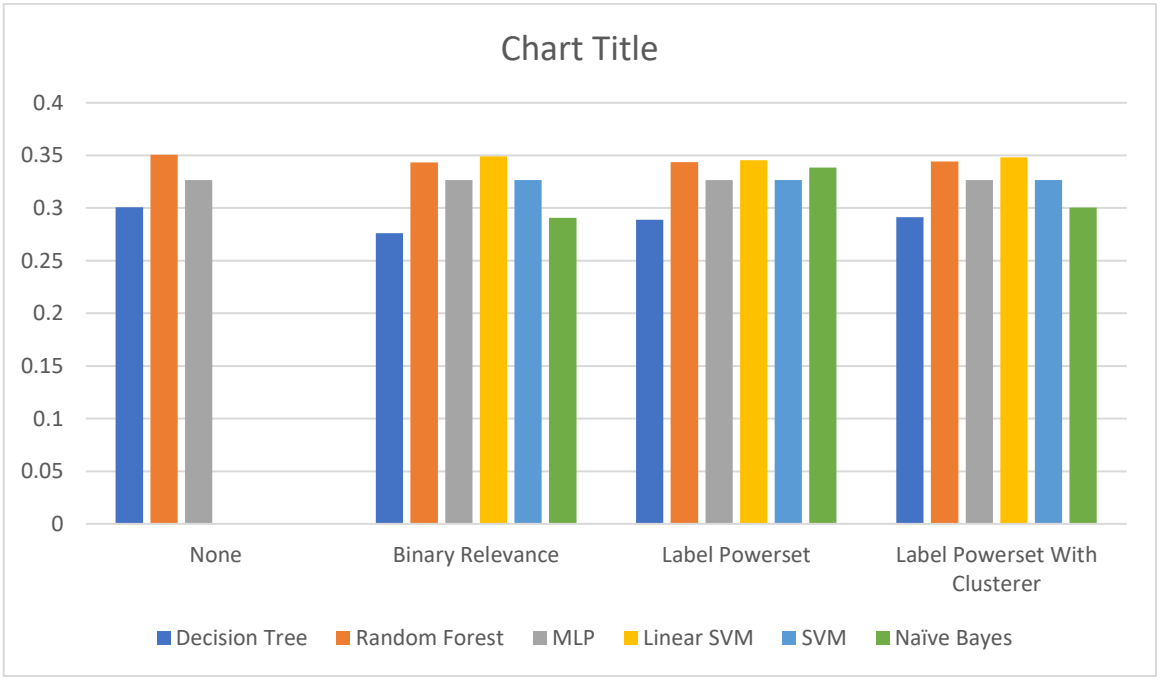
(Γραφική απεικόνιση των αποτελεσμάτων)

6.3.3. Χρήση λεξιλογίου n-grams με συχνότητα πάνω από 2 με Variance Threshold

Ως εκπαιδευτικό σετ δοκιμάστηκε μόνο το κείμενο των ερωτήσεων και ως λεξιλόγιο τα bigrams που προέκυψαν. Για τα δεδομένα αυτά λάβαμε τα εξής αποτελέσματα:

	Decision Tree	Random Forest	MLP	Linear SVM	SVM	Naïve Bayes
None	0.30075	0.35051	0.32657	0	0	0
Binary Relevance	0.27608	0.34318	0.32657	0.34903	0.32657	0.29085
Label Powerset	0.28887	0.34363	0.32657	0.34548	0.32657	0.33837
Label Powerset With Clusterer	0.29119	0.34432	0.32657	0.34806	0.32657	0.3004

(Αποτελέσματα των ταξινομητών με τη χρήση των bigrams)



(Γραφική απεικόνιση των αποτελεσμάτων)

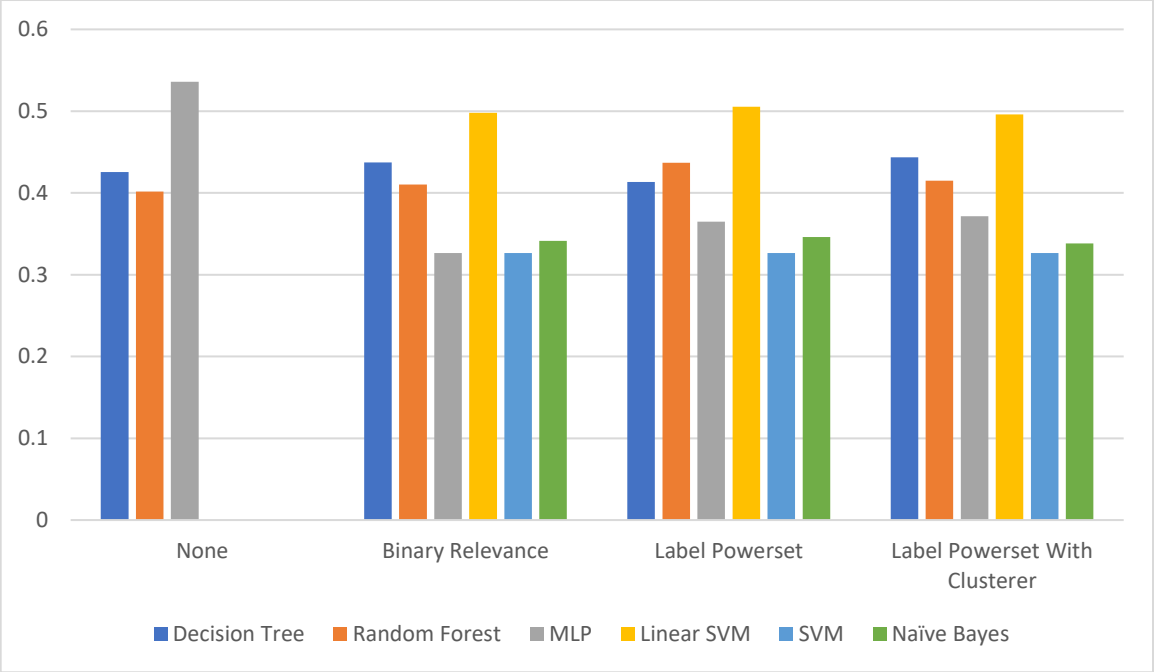
Λόγω των κακών αποτελεσμάτων που προέκυψαν από τα πειράματα που εφαρμόστηκαν στα bigrams δεν κρίθηκε απαραίτητο να δοκιμαστεί και η εκπαίδευση του συστήματος με bigrams του κώδικα των ερωτήσεων.

6.3.4. Χρήση λεξιλογίου με συχνότητα πάνω από 9 χωρίς Variance Threshold

Αρχικά ως εκπαιδευτικό σετ δοκιμάστηκε μόνο το κείμενο των ερωτήσεων. Για τα δεδομένα αυτά λάβαμε τα εξής αποτελέσματα:

	Decision Tree	Random Forest	MLP	Linear SVM	SVM	Naïve Bayes
None	0.42539	0.40165	0.53597	0	0	0
Binary Relevance	0.4372	0.41038	0.32657	0.4981	0.32657	0.3414
Label Powerset	0.41357	0.43676	0.36473	0.50535	0.32657	0.34614
Label Powerset With Clusterer	0.4436	0.41505	0.37152	0.49589	0.32657	0.33821

(Αποτελέσματα των ταξινομητών με τη χρήση μόνο του κειμένου των ερωτήσεων)

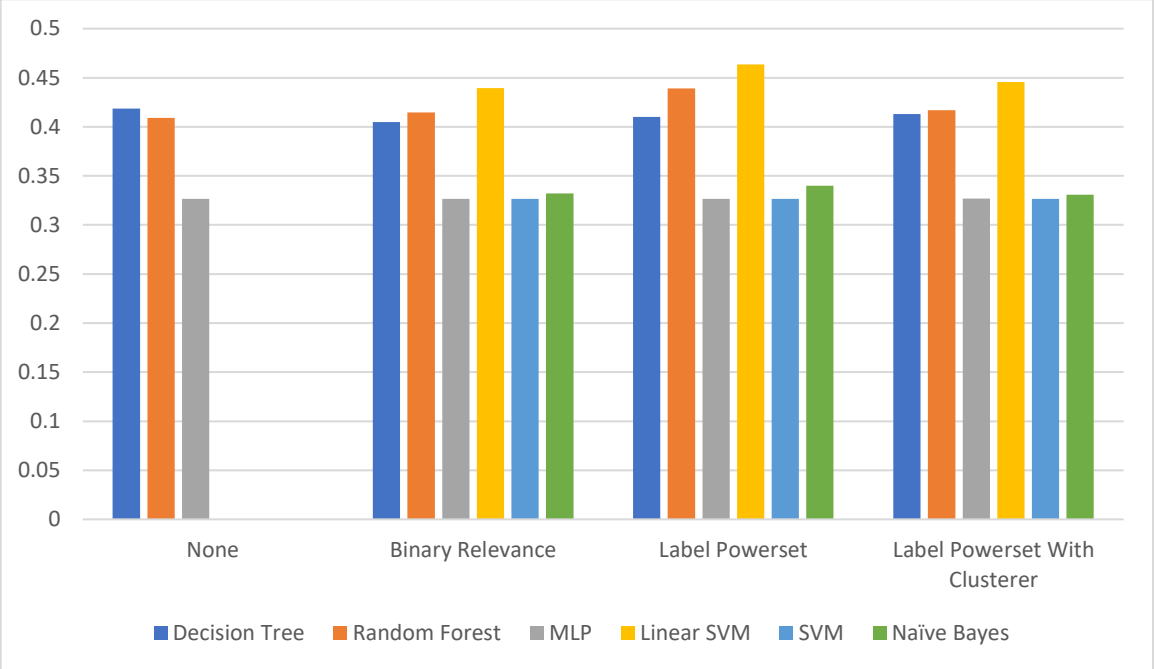


(Γραφική απεικόνιση των αποτελεσμάτων)

Αντίστοιχα, δοκιμάστηκε και ως εκπαιδευτικό σετ μόνο ο κώδικας που περιλαμβάναν οι ερωτήσεις. Τα αποτελέσματα ήταν τα εξής:

	Decision Tree	Random Forest	MLP	Linear SVM	SVM	Naïve Bayes
None	0.40528	0.40372	0.36436	0	0	0
Binary Relevance	0.40466	0.4078	0.32657	0.491	0.32657	0.33398
Label Powerset	0.39398	0.43818	0.32657	0.50737	0.32657	0.33736
Label Powerset With Clusterer	0.41541	0.41553	0.35334	0.48624	0.32657	0.33191

(Αποτελέσματα των ταξινομητών με τη χρήση μόνο του κώδικα των ερωτήσεων)

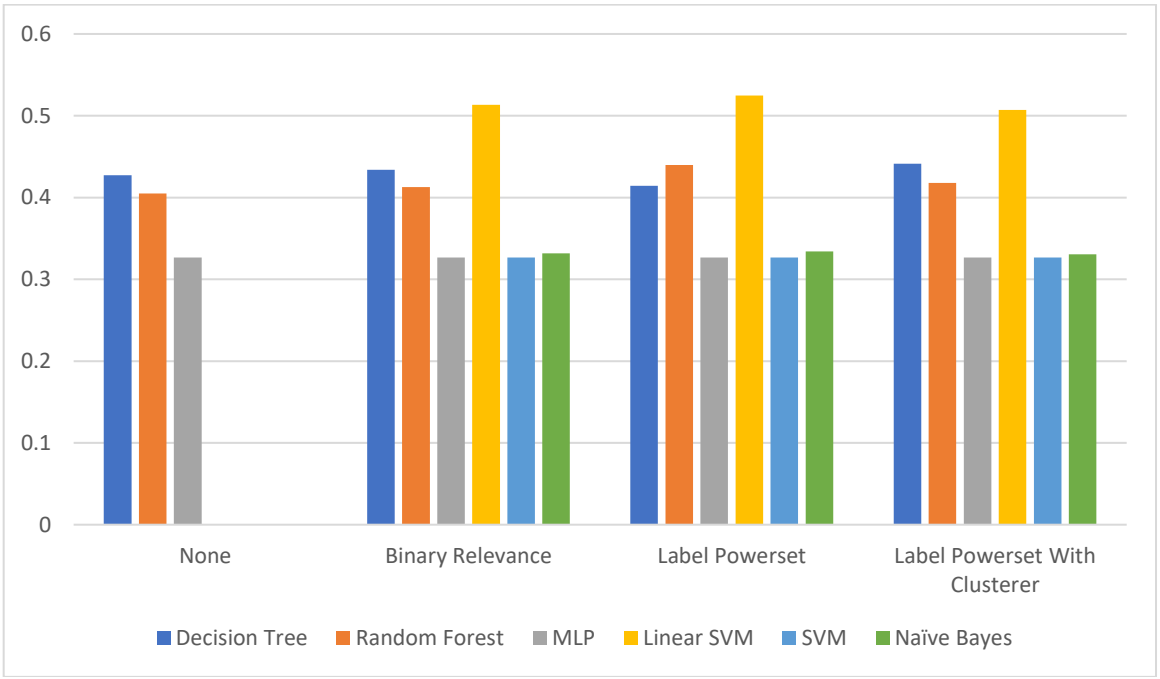


(Γραφική απεικόνιση των αποτελεσμάτων)

Τέλος ως εκπαιδευτικό σετ χρησιμοποιήθηκε τόσο ο κώδικας όσο και το κείμενο των ερωτήσεων. Τα αποτελέσματα που πήραμε ήταν τα εξής:

	Decision Tree	Random Forest	MLP	Linear SVM	SVM	Naïve Bayes
None	0.42732	0.40484	0.32657	0	0	0
Binary Relevance	0.43375	0.4127	0.32657	0.51338	0.32657	0.3318
Label Powerset	0.41417	0.43969	0.32657	0.5247	0.32657	0.33422
Label Powerset With Clusterer	0.44154	0.41805	0.32657	0.50718	0.32657	0.33046

(Αποτελέσματα των ταξινομητών με τη χρήση του κώδικα και του κειμένου των ερωτήσεων)



(Γραφική απεικόνιση των αποτελεσμάτων)

Καθώς είδαμε, στην προηγούμενη ενότητα ότι η χρήση του Variance Threshold επιφέρει πολύ μικρή αύξηση στην ποιότητα των αποτελεσμάτων μας δε θεωρήθηκε απαραίτητο να δοκιμαστεί και στο μειωμένο λεξιλόγιο.

7. ΣΥΜΠΕΡΑΣΜΑΤΑ & ΕΠΕΚΤΑΣΕΙΣ

7.1. Γενικά Συμπεράσματα

Ολοκληρώνοντας την παρούσα εργασία, επικεντρωνόμαστε στα εξής συμπεράσματα:

- Η διαδικασία συλλογής και καθαρισμού των δεδομένων, ώστε να μπορούν να χρησιμοποιηθούν από τα προγράμματα είναι πολύ χρονοβόρα. Η εισαγωγή των δεδομένων σε βάση δεδομένων μείωσε πολύ τον χρόνο αυτό. Επίσης έδωσε την δυνατότητα να επεξεργαστούν επανειλημμένα τα δεδομένα και να μπορέσουν να επανεπεξεργαστούν για να βελτιστοποιηθούν περεταίρω τα δεδομένα.

- Η διαδικασία κατηγοριοποίησης των ερωτήσεων του Stack Overflow παρουσιάζει πολύ μεγαλύτερη δυσκολία από την διαδικασία κατηγοριοποίησης εξίσου σύντομων κειμένων από ένα άλλου τύπου φόρουμ. Το γεγονός αυτό φαίνεται να οφείλεται τόσο στην συντομία των κειμένων, στην κακή χρήση της γλώσσας, στην επαναληψιμότητα του λεξιλογίου και στο πολύ εξειδικευμένο λεξιλόγιο.

- Είδαμε ότι το μέγεθος του λεξιλογίου παίζει πολύ σημαντικό ρόλο στην ακρίβεια των αποτελεσμάτων. Μικραίνοντας το λεξιλόγιο μας η ακρίβεια αυξήθηκε για κάποιους αλγορίθμους αλλά μειώθηκε για κάποιους άλλους. Αυτό σημαίνει ότι η επιλογή του μεγέθους του λεξιλογίου εξαρτάται από τον αλγόριθμο που επιλέγεται για την κατηγοριοποίηση.
-
- Ο μεγάλος όγκος του λεξιλογίου δημιούργησε προβλήματα στην εκτέλεση των αλγορίθμων. Αλγόριθμοι όπως ο k-NN δεν είναι κατάλληλοι για προβλήματα με τόσο μεγάλο όγκο λεξιλογίου, καθώς είναι πολύ απαιτητικοί από άποψη υπολογιστικών πόρων.

- Η χρήση των n-grams δεν ενδείκνυται για την κατηγοριοποίηση των ερωτήσεων στο Stack Overflow. Αυτό έχει να κάνει τόσο με τον μεγάλο όγκο του λεξιλογίου όσο και με την μεγαλύτερη πολυπλοκότητα που εμφανίζει η κατηγοριοποίηση με τη χρήση n-grams.

- Η χρήση clustering πάνω στις ετικέτες δίνει χειρότερα αποτελέσματα από τη χρήση του απλού powerset. Σημαντικό ρόλο έπαιξε σε αυτό και το μεγάλο μέγεθος του λεξιλογίου, σε συνάρτηση με το μικρό μέγεθος των κειμένων αλλά και το ότι fastgreedy αλγόριθμο.

- Γενικά είδαμε ότι τα καλύτερα αποτελέσματα τα δίνει ο αλγόριθμος Linear SVM. Καλά αποτελέσματα δίνει επίσης και ο αλγόριθμος Random Forest καθώς επίσης και ο Decision Trees.

- Τέλος, μπορούμε με ασφάλεια να καταλήξουμε στο συμπέρασμα ότι η διαδικασία κατηγοριοποίησης των ερωτήσεων του Stack Overflow είναι ένα ιδιαίτερα δύσκολο πρόβλημα. Επιβεβαιώνεται η αρχική παραδοχή για κακή, μη συνεκτική χρήση της γλώσσας από τους χρήστες, καθώς δεν περνάει από κάποια φάση επιμέλειας πριν δημοσιευτεί.

7.2. Επεκτάσεις

Κάθε μέρα δημιουργούνται πάνω από 5.000 καινούργιες ερωτήσεις στο Stack Overflow. Στο μέλλον θα μπορούσε να προχωρήσει η δημιουργία ενός συστήματος που θα λειτουργεί σε πραγματικό χρόνο, προτείνοντας στους χρήστες κατηγορίες για την ερώτηση τους πριν την υποβάλλουν. Το σύστημα αυτό θα μπορούσε να λειτουργεί αξιοποιώντας την λογική της επιβλεπόμενης μάθησης, καθώς ο χρήστης θα μπορεί να δέχεται ή να αρνείται τα προτεινόμενα tags του προγράμματος και άρα να συνεχίζει να μαθαίνει. Ταυτόχρονα θα μπορούσε να διερευνηθεί η εφαρμογή της προσέγγισης μας το σύστημα για την αναγνώριση ερωτήσεων που έχουν κατηγοριοποιηθεί λάθος από τους χρήστες. Να παρακολουθεί δηλαδή τις νέες και τις υπάρχουσες ερωτήσεις και αν θεωρεί ότι υπάρχουν μεγάλες αποκλίσεις μεταξύ των δικών του αποτελεσμάτων και των κατηγοριών της ερώτησης να θέτει την ερώτηση ως "υπό εξέταση" για πιθανή επανα-κατηγοριοποίηση.

Όπως αναφέρθηκε και εισαγωγικά ένα σημαντικό πρόβλημα που αντιμετωπίζουμε στην κατηγοριοποίηση των ερωτήσεων του Stack Overflow είναι το εξιδεικευμένο λεξιλόγιο και τις μικρές διαφορές μεταξύ των ερωτήσεων. Μια ερώτηση που ανήκει στην κατηγορία SQL και μία άλλη που ανήκει στην κατηγορία MySQL είναι πολύ πιθανό να περιέχουν πανομοιότυπο κείμενο και πολύ μικρές διαφορές στον κώδικα τους, αν αυτός υπάρχει. Για την αντιμετώπιση του προβλήματος αυτού θα μπορούσε να εξεταστεί η ομαδοποίηση σχετικών κατηγοριών και η κατηγοριοποίηση των ερωτήσεων στις ομάδες αυτές. Παραδείγματος χάριν, κατηγορίες όπως java, java ee, JavaScript, θα μπορούσαν να ανήκουν σε μία ομάδα, ενώ οι κατηγορίες SQL, MySQL, PL\SQL σε μία άλλη. Μια ευρεία και συγκροτημένη ομαδοποίηση αυτή θα μείωνε πολύ τον αριθμό των διαθέσιμων κατηγοριών αλλά επίσης θα αντιμετώπιζε το πρόβλημα του κοινού και πολύ εξιδεικευμένου λεξιλογίου που εμφανίζεται στις ερωτήσεις. Για την επιμέρους κατηγοριοποίηση, ανάμεσα στις κατηγορίες μίας ομάδας θα μπορούσαν να εκπαιδευτούν ανάλογα συστήματα εκπαιδευμένα στις συγκεκριμένες κατηγορίες.

Τέλος, μια επέκταση του προβλήματος θα μπορούσε να λαμβάνει υπόψιν της και τους χρήστες που θέτουν την ερώτηση. Στην παρούσα υλοποίηση ο χρήστης που έθετε μία ερώτηση αγνοούταν εντελώς. Αν όμως ένας χρήστης δείχνει, μέσω από τα στατιστικά στοιχεία που διατηρεί το Stack Overflow, δείχνει μια προτίμηση σε συγκεκριμένες κατηγορίες το σύστημα θα μπορούσε να αντιμετωπίζει τις κατηγορίες αυτές ως αύξουσας βαρύτητας σε σχέση με κατηγορίες για τις οποίες ο χρήστης δε θέτει συχνά ερωτήσεις.

8. ΚΩΔΙΚΑΣ

8.1. C# Programs

8.1.1. PreprocessFiles

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.IO;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace PreprocessFiles
{
   class Program
   {
      ////Utilities Functions Not included

      #region Clean Up Text

      #region Variables

      static string Replacement = "";
      static string CleanText = "";
      static string Element = "";
      static string ElementCore = "";
      static char Letter = ' ';
      static string Letters = "";
      static bool StartElement = false;

      #region Known Entities

      static Dictionary<string, string> KnownEntities = new
      Dictionary<string, string>()
      {
         {"&#x0;", ""},       // Control Character: NUL Null character
.
.
.
.

         {"&{Θ&;", " "},
         {"&Θ(;", " "}


      };
      #endregion

      #region Remove Entities

      private static string RemoveEntities(string Text)
      {
         #region Initialize Vars

         Letter = ' ';
         StartElement = false;
         CleanText = "";
         Element = "";
```

```
            #endregion

            try
            {
                for (int i = 0; i < Text.Length; i++)
                {
                    Letter = Text[i];

                    #region Text

                    if (!StartElement) // Text
                    {
                        if (Letter != '&') CleanText += Letter;
                        else
                        {
                            StartElement = true;
                            Element += Letter;
                        }
                    }
                    #endregion

                    #region Element

                    else // StartElement - Element
                    {
                        if (Letter != ';') Element += Letter;
                        else
                        {
                            StartElement = false;
                            Element += Letter;

                            #region Clean Known or All Entities

                            if (KnownEntities.TryGetValue(Element, out
Replacement)) Element = Replacement;
                            else if (!Element.Contains(" "))
                            {
                                if (!UnknownElements.ContainsKey(Element))
UnknownElements.Add(Element, Element);
                                if (cleanAllTags) Element = " "; // Remove
Unknown Elements
                            }
                            CleanText += Element;
                            Element = "";

                            #endregion
                        }
                    }
                    #endregion
                }
                if (Element != "") CleanText += Element;

                #region Check

                //WriteText("Remove Entities Before   : " + Text);
                //WriteText("Remove Entities After    : " + CleanText);

                #endregion

                return CleanText;
```

```
            }
            catch (Exception ex)
            {
                WriteError("RemoveEntities - Could not cleanup found entities
in the following text. Text used as is." + cRLF + Text, ex.ToString());
                return Text;
            }
        }
        #endregion

        #region Remove Tags

        public static bool cleanAllTags = true;

        private static string RemoveTags(string Text)
        {
            #region Initialize Vars

            CleanText = "";
            Letter = ' ';
            StartElement = false;
            Element = "";
            ElementCore = "";

            #endregion
            try
            {
                return Regex.Replace(Text, "<.*?>| &.*?;", " ");

            }
            catch (Exception ex)
            {
                WriteError("RemoveTags - Could not cleanup tags in the
following text. Text left as is." + cRLF + Text, ex.ToString());
                return Text;
            }
        }

        #endregion

        #region Clean Up String

        public static bool returnedClean;

        public static string vowels = "aeiouy";
        public static string consonants = "bcdfghjklmnpqrstvwxz";

        public static string CleanString(string uncleanString)
        {
            try
            {
                returnedClean = true;

                string cleanedupString = uncleanString;
                string code = "";
                cleanedupString = cleanedupString.ToLower();

                #region Remove Entities

                cleanedupString = RemoveEntities(cleanedupString);
```

```
            #endregion

            #region Remove Tags

            cleanedupString = RemoveTags(cleanedupString);

            #endregion

            #region Remove Numbers & Punctuation

            for (int i = 0; i < cleanedupString.Length; i ++)
            {
               if (char.IsPunctuation(cleanedupString[i]))
cleanedupString = cleanedupString.Replace(cleanedupString[i].ToString(),
" ");

               if (char.IsNumber(cleanedupString[i])) cleanedupString =
cleanedupString.Replace(cleanedupString[i].ToString(), " ");
            }

            while (cleanedupString.Contains("  ")) cleanedupString =
cleanedupString.Replace("  ", " ");

            #endregion

            if (string.IsNullOrEmpty(cleanedupString)) return
cleanedupString;

            #region Remove Single Letter words

            for (int i = 0; i < cleanedupString.Length; i++)
            {
               if (cleanedupString[i] != 'c')
               {
                  if (i > 0 && i < cleanedupString.Length - 1)
                  {
                     if (cleanedupString[i - 1].ToString() == " " &&
cleanedupString[i + 1].ToString() == " ")
                     {
                        cleanedupString = cleanedupString.Substring(0, i
- 1) + " " + cleanedupString.Substring(i + 1);
                     }
                  }
                  else if (i == 0 && cleanedupString.Length > 1)
                  {
                     if (cleanedupString[i + 1].ToString() == " ")
                     {
                        cleanedupString = " " +
cleanedupString.Substring(1);
                     }
                  }
                  else if (i == cleanedupString.Length - 1)
                  {
                     if (cleanedupString[i - 1].ToString() == " ")
                     {
                        cleanedupString = cleanedupString.Substring(0, i
- 1) + " ";
                     }
                  }
```

```
                }

            }

            while (cleanedupString.Contains("  ")) cleanedupString =
cleanedupString.Replace("  ", " ");

            #endregion

            #region Remove non english characters

            for (int i = 0; i < cleanedupString.Length; i++)
            {
               if (!vowels.Contains(cleanedupString[i]) &&
!consonants.Contains(cleanedupString[i]))
               {
                  cleanedupString =
cleanedupString.Replace(cleanedupString[i].ToString(), " ");
               }
            }

            while (cleanedupString.Contains("  ")) cleanedupString =
cleanedupString.Replace("  ", " ");

            #endregion

            return cleanedupString;
         }
         catch (Exception ex)
         {
            WriteError("Failed to clean up string : " + uncleanString +
cRLF , "Exception : " + ex.ToString());
            returnedClean = false;

            return uncleanString;

         }
      }
      #endregion

      #endregion

      #region Add Questions CSV To DB

      #region Question Class

      public class Question
      {
         public string Id;
         public string OwnerUserId;
         public string CreationDate;
         public string ClosedDate;
         public string Score;
         public string Title;
         public string Body;
         public string UncleanBody;
         public string Code;
         public string Link;
      }
```

```
        #endregion

        #region Get Question From String Element

        public static Question GetQuestionFromElement(string element)
        {
            Question newQuestion = new Question();
            try
            {
                newQuestion.Id = element.Substring(0, element.IndexOf(","));
                element = element.Substring(element.IndexOf(",") + 1);

                newQuestion.OwnerUserId = element.Substring(0,
element.IndexOf(","));
                element = element.Substring(element.IndexOf(",") + 1);

                newQuestion.CreationDate = element.Substring(0,
element.IndexOf(","));
                element = element.Substring(element.IndexOf(",") + 1);

                newQuestion.ClosedDate = element.Substring(0,
element.IndexOf(","));
                element = element.Substring(element.IndexOf(",") + 1);

                newQuestion.Score = element.Substring(0,
element.IndexOf(","));
                element = element.Substring(element.IndexOf(",") + 1);

                newQuestion.Title = element.Substring(0,
element.IndexOf(","));
                element = element.Substring(element.IndexOf(",") + 1);

                newQuestion.UncleanBody = newQuestion.Title + " " + element;
                newQuestion.Code = "";

                while (element.Contains("<code>") &&
element.Contains("</code>"))
                {
                    string code =
element.Substring(element.IndexOf("<code>"));
                    code = code.Substring(0, code.IndexOf("</code>") + 7);
                    element = element.Replace(code, " ");
                    newQuestion.Code += code + "\r\n";
                }

                if (element.Contains("<a href")) newQuestion.Link = "True";
                else newQuestion.Link = "False";

                newQuestion.Body = CleanString(newQuestion.Title +" " +
element);

                AddToDictionary(newQuestion.Body);

                return newQuestion;

            }
            catch (Exception ex)
            {
                WriteError("",ex.ToString());
                Console.WriteLine("Exception To String : " + ex.ToString());
```

```
            Console.WriteLine(newQuestion.Id);
            Console.WriteLine(newQuestion.OwnerUserId);
            Console.WriteLine(newQuestion.CreationDate);
            Console.WriteLine(newQuestion.ClosedDate);
            Console.WriteLine(newQuestion.Score);
            Console.WriteLine(newQuestion.Title);
            Console.WriteLine(newQuestion.Body);
            Console.WriteLine(element);
            Console.WriteLine("Press any key...");
            Console.ReadKey();
            return null;
         }
      }

      #endregion

      #region Add Questions from CSV to DB

      public static void QuestionsToDB(string path)
      {
         WriteTitle("Reading lines from Files : " +
Path.GetFileName(path));

         using (var conn = new SqlConnection(connString))
         {
            conn.Open();
            int elementCounter = 0;
            int elementNotAdded = 0;
            using (var file = new StreamReader(path))
            {
               var line = file.ReadLine();

               line = file.ReadLine();

               string element = "";

               while (line != null)
               {
                  if (line.EndsWith("\"") && !line.EndsWith("\"\"") &&
!line.EndsWith(",\""))
                  {

                     element += line;
                     Question QuestionFound = new Question();

                     if (element.Contains("̃"))
                        element = element.Replace("̃", "");
                     if (element.Contains("Ą"))
                        element = element.Replace("Ą", "A");

                     QuestionFound = GetQuestionFromElement(element);
                     if (QuestionFound != null)
                     {
                        string query = "INSERT INTO dbo.Questions (GID,
ID, BODY, UNCLEAN, CODE, LINK) " +
                                       "VALUES (@Gid, @Id, @Body,
@UncleanBody, @Code, @Link) ";

                        Guid gid = Guid.NewGuid();
```

```
                        using (SqlCommand cmd = new SqlCommand(query,
conn))
                        {
                            cmd.Parameters.AddWithValue("@Gid", gid);
                            cmd.Parameters.AddWithValue("@Id",
QuestionFound.Id);
                            cmd.Parameters.AddWithValue("@Body",
QuestionFound.Body);
                            cmd.Parameters.AddWithValue("@UncleanBody",
QuestionFound.UncleanBody);
                            cmd.Parameters.AddWithValue("@Code",
QuestionFound.Code);
                            cmd.Parameters.AddWithValue("@Link",
QuestionFound.Link);
                            cmd.ExecuteNonQuery();
                        }

                        if (!returnedClean)
                        {
                            WriteText("Question Entry with GID " +
gid.ToString() + " contains unclean Body.");
                        }
                    }
                    else

                    {
                        elementNotAdded++;
                    }
                    element = "";
                    elementCounter++;
                    Console.Write("Elements Added : " + elementCounter +
"\r");
                }
                else
                {
                    element += line;
                }

                line = file.ReadLine();
            }
            WriteText("Elements found : " +
elementCounter.ToString());
            WriteText("Elements not added : " +
elementNotAdded.ToString());
        }
        conn.Close();
    }
}

#endregion

#endregion

#region Run

public static string answersCSV = "";
public static string questionsCSV = "";
public static string tagCSV = "";
public static string vocabulary = "";
```

```
        public static void Run()
        {
            try
            {

                #region Start - Open Log - Variables


                workDir = @"C:\Users\babis\Desktop\Iro\NTUA\Stack Sample\";

                OpenLog();

                WriteText(ProgInfo);

                ProgStartTime = DateTime.Now;

                WriteTitle("Program started at       : " + ProgStartTime);

                #endregion

                QuestionsToDB(@"C:\Users\babis\Desktop\Iro\NTUA\Stack
Sample\Originals\Questions.csv");

            }
            catch(Exception ex)
            {
                WriteError("Run - Failed to run program", ex.ToString());
            }
        }
        #endregion

        #region Main

        static bool Continue = true;

        static void Main(string[] args)
        {
            #region Run Clean TMX Files Program

                Run();

            #endregion

            if (!Continue)
                Console.WriteLine("Program terminated with errors. Press any
key to continue...");
            else
                Console.WriteLine("OK. Program terminated without errors.");
        }

        #endregion
    }
}
```

8.1.2. CreateVectors.cs

```
#region Using

using System;
using System.IO;
using System.Net;
using System.Collections.Generic;
using System.Diagnostics;
using System.Data.SqlClient;
using System.Data;
using System.Net.Mail;
using System.Linq;
using System.Text.RegularExpressions;

#endregion

namespace CreateVectors
{
   public class CreateSparse
   {
      //Utilities  Functions Missing

      #region Collect Tags

      static Dictionary<string, int> tags = new Dictionary<string,
int>();

      public static void GetTagsFromServer()
      {
         WriteTitle("Collectin Tags from Server");

         commString = "SELECT TOP 20 TAG FROM dbo.Tags500 ORDER BY COUNT
DESC";

         int counter = 0;

         using (comm = new SqlCommand(commString, conn))
         {

            using (SqlDataReader reader = comm.ExecuteReader())
            {
               while (reader.Read())
               {


                  string tag = Convert.ToString(reader["TAG"]);

                  tags.Add(tag, counter);

                  Console.Write("Tags read : " + counter.ToString() +
"\r");

                  counter++;
               }
            }
         }

         foreach (string tag in tags.Keys)
```

```
            {
                WriteText("Tag", tag);
            }
            WriteText("Tags read", counter.ToString() + "\r");

        }

        #endregion

        #region Create Bag Of Words

        static Dictionary<string, int> bagOfCodes = new Dictionary<string,
int>();
        static Dictionary<string, int> bagOfCodesIndex = new
Dictionary<string, int>();

        static void CreateBagOfWords()
        {
            try
            {
                WriteTitle("Creating Bag Of Words");
                int counter = 0;

                commString = "SELECT Term, Frequency FROM
dbo.BagOfWords_Stemmed WHERE Frequency > 9";

                using (comm = new SqlCommand(commString, conn))
                {
                    using (SqlDataReader reader = comm.ExecuteReader())
                    {
                        while (reader.Read())
                        {

                            string term = Convert.ToString(reader["Term"]);
                            int frequency =
Convert.ToInt32(reader["Frequency"]);

                            bagOfCodes.Add(term, frequency);
                            bagOfCodesIndex.Add(term, counter);
                            counter++;

                            Console.Write("Elements read : " +
counter.ToString() + "\r");


                        }
                    }
                }
                Console.ReadKey();
                WriteText("Elements in DB " + counter.ToString());
            }
            catch (Exception ex)
            {
                WriteError("Failed to get Data from SQL Server",
ex.ToString());
            }
        }

        #endregion
```

```
        #region Create Sparse Code Vectors

        static List<string> vectorsList = new List<string>();
        static List<string> vectorsCodeList = new List<string>();
        static List<string> tagsList = new List<string>();

        static void CreateSparseVector()
        {
            try
            {
                WriteTitle("Collectin Questions from Server");

                int counter = 0;
                int questionsProcessed = 0;
                int unpopular = 0;
                commString = "SELECT STEMM, TDIDF, TAGS FROM Questions";

                using (comm = new SqlCommand(commString, conn))
                {
                    using (SqlDataReader reader = comm.ExecuteReader())
                    {
                        while (reader.Read())
                        {

                            string stemm = Convert.ToString(reader["STEMM"]);
                            string tdidf = Convert.ToString(reader["TDIDF"]);
                            string tagsInDB = Convert.ToString(reader["TAGS"]);

                            #region Process Questions

                            string[] stemmWords = stemm.Split(new char[] { ' '
}, StringSplitOptions.RemoveEmptyEntries).Distinct().ToArray();

                            string[] vectors = tdidf.Split(',');

                            if (vectors.Length != stemmWords.Length)
                            {
                                WriteText("Inconsistencies found between " +
stemm + "\n and " + tdidf + ".\n Counter : " + counter);
                                WriteText("Will not process");
                                continue;
                            }

                            #region Process Code

                            if (!string.IsNullOrEmpty(stemm))
                            {
                                string[] stringWords = stemm.Split(new char[] { '
', '\t' }, StringSplitOptions.RemoveEmptyEntries);

                                Dictionary<string, int> sentenceDictionary = new
Dictionary<string, int>();

                                foreach (string word in stringWords)
                                {
                                    if (sentenceDictionary.ContainsKey(word))
sentenceDictionary[word]++;
                                    else sentenceDictionary.Add(word, 1);
                                }
```

```
                        foreach (KeyValuePair<string, int> word in
sentenceDictionary)
                        {
                            double td = (double)word.Value /
(double)stringWords.Count();

                            double idf = 0;

                            if (bagOfCodes.ContainsKey(word.Key))
                            {
                                idf = Math.Log(1264212 /
bagOfCodes[word.Key]);

                                string tdIDF = Convert.ToString(td * idf);

                                int position;

                                if (bagOfCodesIndex.ContainsKey(word.Key))
                                {
                                    position = bagOfCodesIndex[word.Key];

                                    string vector = counter + " , " +
position + " , " + tdIDF;

                                    vectorsList.Add(vector);
                                }
                            }
                            else
                            {
                                continue;
                            }

                        }

                    tagsList.Add(tagsInDB);

                    }
                    #endregion

                    #endregion

                    if ((counter % 200000) == 0)
                    {

File.AppendAllLines(@"D:\FinalLess\FinalFile10_Code.csv", vectorsList);
                        vectorsList.Clear();
                    }

                    counter++;
                    questionsProcessed++;

                    Console.Write("Questions read : " +
counter.ToString() + "\r");

                }
            }
        }

        WriteText();
```

```
            WriteText("Questions in DB " + counter.ToString());

            File.AppendAllLines(@"D:\FinalLess\FinalFile10_Code.csv",
vectorsList);

            vectorsList.Clear();
         }
         catch (Exception ex)
         {
            WriteError("Failed to get Data from SQL Server",
ex.ToString());
         }
      }

      #endregion

      #region Write Tags File

      public static void WriteTagsFile()
      {
         WriteTitle("Turning Tags To Vector");

         int counter = 0;

         string pathTags = @"D:\FinalCSV20\FinalFile_Tags.csv";


         using (var w = new StreamWriter(pathTags))
         {
            foreach (string key in tagsList)
            {
               char[] vector = defaultVector.ToCharArray();

               counter++;

               List<string> values = key.Split(',').ToList();
               List<int> indexes = new List<int>();

               foreach (string value in values)
               {
                  if (tags.ContainsKey(value))
indexes.Add(tags[value.Trim()]);
               }

               if (indexes.Count > 0)
               {
                  try
                  {
                     foreach (int i in indexes)
                        vector[(2 * i)] = '1';
                  }
                  catch (Exception ex)
                  {
                     WriteText("Failed on question " + key +
ex.ToString() + ". Indexes : ");
                     foreach (int i in indexes) WriteText("I : " +
i.ToString());
                     WriteText("");
                     continue;
                  }
```

```
                }
                string newString = new string(vector);
                newString = newString.TrimEnd(',');

                string line = newString;

                w.WriteLine(line);
                w.Flush();
                Console.Write("Tags written " + counter.ToString() +
"\r");
            }
         }
         WriteText("Elements vectorised " + counter.ToString());
      }

      #endregion

      #region Run

      public static string jobStatus = "";
      public static string requestStatus = "";
      public static string runError = "";
      public static string connString = "Data
Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=\"C:\\USERS\\BABIS\\DESKTOP\\IRO\\NTUA\\STACK
SAMPLE\\STACKDB.MDF\";Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=True;ApplicationIntent=Re
adWrite;MultiSubnetFailover=False;";
      public static string defaultVector = "";

      public static void Run()
      {
         try
         {
            #region Get and Report Program Info

            if (!progError) ProgInfo();

            MoveProgLog(progDir + "\\" + progName + ".log");


            #endregion

            #region Connect to SQL

            SQLConnect();

            #endregion

            #region Collect Bag of Codes

            CreateBagOfCodes();

            #endregion

            #region Collect Tags

            GetTagsFromServer();

            for (int i = 0; i < 20; i++)
```

```
            {
                defaultVector += "0,";
            }

            #endregion

            #region Create Sparse Vectors

            CreateSparseVector();

            #endregion

            #region Move Log

            string tempLog = Path.GetDirectoryName(progLog) + "\\" +
Path.GetFileNameWithoutExtension(progLog);

            if (!progError) tempLog += ".log";
            else
            {
                tempLog += "_ERROR.log";

                MoveProgLog(tempLog);
            }
            #endregion

            #region End - Close Log

            progStopTime = DateTime.Now;
            TimeSpan TimeTaken = progStopTime - progStartTime;
            string StrTimeTaken = TimeTaken.ToString();
            StrTimeTaken = StrTimeTaken.Substring(0, StrTimeTaken.Length
- 5);
            WriteTitle("Program finished at : " + progStopTime + " after
" + StrTimeTaken);
            CloseProgLog();

            #endregion
        }
        catch (Exception ex)
        {
            WriteError("MTPreTrans - Could not run program.",
ex.ToString());
        }
    }
    #endregion

    #region Main - Get Parameters

    public static string[] progParams = null;

    static int Main(string[] args)
    {
        try
        {
            #region Set Program Parameters

            progParams = args;

            #endregion
```

```
            #region Run Program

            Run();

            #endregion

            #region Wait and Return Exit Code

            if (!progError) Console.WriteLine("Program completed.");
            else Console.WriteLine("Program terminated with ERRORS.");

            if (!progError) return 0;
            else return 1;

            #endregion
         }
         catch (Exception ex)
         {
            Console.WriteLine("Main - Could not get and check given
parameters.", ex.ToString());
            return 1;
         }
      }
      #endregion
   }
}
```

8.2. Python Programs

8.2.1. Final.py

```
import sys

import numpy as np
import pandas as pd
import csv
from scipy.sparse import coo_matrix

import os

import time
start_time = time.time()

documents_no = 100000 # -1
tags_no = 20 #100

variance_threshold = True

classifier = sys.argv[1] #"decision_tree", "random_forest", "MLP"
method = sys.argv[2] #"none", "binary_relevance", "label_powerset",
"LabelPowerset_with_Clusterer"
features = sys.argv[3] # "Text", "Code", "All"

if documents_no == -1:
    if tags_no == -1:
        if method == "none":
            row = []
            col = []
            data = []
            with open('./csvs/FinalFile_' + features + '.csv','r') as f:
                for line in f:
                    array =
line.replace('\r','').replace('\n','').split(',')
                    row.append(int(array[0]))
                    col.append(int(array[1]))
                    data.append(float(array[2]))

            X = coo_matrix((data, (row, col)),
shape=(max(row)+1,max(col)+1))

            Y = pd.read_csv('./csvs/FinalFile_Tags.csv',
sep=',',header=None, engine='python')
        else:
            row1 = []
            col1 = []
            data1 = []
            with open('./csvs/FinalFile_' + features + '.csv','r') as f:
                for line in f:
                    array =
line.replace('\r','').replace('\n','').split(',')
                    row1.append(int(array[0]))
                    col1.append(int(array[1]))
                    data1.append(float(array[2]))
            row2 = []
            col2 = []
            data2 = []
```

```
            with open('./csvs/FinalFile_Tags-sparse.csv','r') as f:
                for line in f:
                    array =
line.replace('\r','').replace('\n','').split(',')
                    row2.append(int(array[0]))
                    col2.append(int(array[1]))
                    data2.append(float(array[2]))

            max_row_1 = max(row1)
            max_row_2 = max(row2)

            if max_row_1 > max_row_2:
                X = coo_matrix((data1, (row1, col1)), shape=(max_row_1+1,
max(col1)+1))
                Y = coo_matrix((data2, (row2, col2)), shape=(max_row_1+1,
max(col2)+1))
            else:
                X = coo_matrix((data1, (row1, col1)), shape=(max_row_2+1,
max(col1)+1))
                Y = coo_matrix((data2, (row2, col2)), shape=(max_row_2+1,
max(col2)+1))
    else:
        if method == "none":
            row = []
            col = []
            data = []
            with open('./csvs/FinalFile_' + features + '.csv','r') as f:
                for line in f:
                    array =
line.replace('\r','').replace('\n','').split(',')
                    row.append(int(array[0]))
                    col.append(int(array[1]))
                    data.append(float(array[2]))

            X = coo_matrix((data, (row, col)),
shape=(max(row)+1,max(col)+1))

            Y = pd.read_csv('./csvs/FinalFile_Tags.csv',
sep=',',header=None, usecols=range(tags_no), engine='python')
        else:
            row1 = []
            col1 = []
            data1 = []
            with open('./csvs/FinalFile_' + features + '.csv','r') as f:
                for line in f:
                    array =
line.replace('\r','').replace('\n','').split(',')
                    row1.append(int(array[0]))
                    col1.append(int(array[1]))
                    data1.append(float(array[2]))
            row2 = []
            col2 = []
            data2 = []
            with open('./csvs/FinalFile_Tags-sparse.csv','r') as f:
                for line in f:
                    array =
line.replace('\r','').replace('\n','').split(',')
                    if int(array[1]) < tags_no:
                        row2.append(int(array[0]))
                        col2.append(int(array[1]))
```

```
                        data2.append(float(array[2]))

            max_row_1 = max(row1)
            max_row_2 = max(row2)

            if max_row_1 > max_row_2:
                X = coo_matrix((data1, (row1, col1)), shape=(max_row_1+1,
max(col1)+1))
                Y = coo_matrix((data2, (row2, col2)), shape=(max_row_1+1,
max(col2)+1))
            else:
                X = coo_matrix((data1, (row1, col1)), shape=(max_row_2+1,
max(col1)+1))
                Y = coo_matrix((data2, (row2, col2)), shape=(max_row_2+1,
max(col2)+1))
else:
    if tags_no == -1:
        row = []
        col = []
        data = []
        with open('./csvs/FinalFile_' + features + '.csv','r') as f:
            for line in f:
                array = line.replace('\r','').replace('\n','').split(',')
                if int(array[0]) < documents_no:
                    row.append(int(array[0]))
                    col.append(int(array[1]))
                    data.append(float(array[2]))

        X = coo_matrix((data, (row, col)), shape=(documents_no,
max(col)+1))
        if method == "none":
            Y = pd.read_csv('./csvs/FinalFile_Tags.csv',
sep=',',header=None, nrows=documents_no, engine='python')
        else:
            row = []
            col = []
            data = []
            with open('./csvs/FinalFile_Tags-sparse.csv','r') as f:
                for line in f:
                    array =
line.replace('\r','').replace('\n','').split(',')
                    if int(array[0]) < documents_no:
                        row.append(int(array[0]))
                        col.append(int(array[1]))
                        data.append(float(array[2]))

            Y = coo_matrix((data, (row, col)), shape=(documents_no,
max(col)+1))
    else:
        row = []
        col = []
        data = []
        with open('./csvs/FinalFile_' + features + '.csv','r') as f:
            for line in f:
                array = line.replace('\r','').replace('\n','').split(',')
                if int(array[0]) < documents_no:
                    row.append(int(array[0]))
                    col.append(int(array[1]))
                    data.append(float(array[2]))
```

```
        X = coo_matrix((data, (row, col)), shape=(documents_no,
max(col)+1))
        if method == "none":
            Y = pd.read_csv('./csvs/FinalFile_Tags.csv',
sep=',',header=None, nrows=documents_no, usecols=range(tags_no),
engine='python')
        else:
            row = []
            col = []
            data = []
            with open('./csvs/FinalFile_Tags-sparse.csv','r') as f:
                for line in f:
                    array =
line.replace('\r','').replace('\n','').split(',')
                    if int(array[0]) < documents_no:
                        if int(array[1]) < tags_no:
                            row.append(int(array[0]))
                            col.append(int(array[1]))
                            data.append(float(array[2]))
            Y = coo_matrix((data, (row, col)), shape=(documents_no,
max(col)+1))

#print X.shape
#print Y.shape

if variance_threshold:
    from sklearn.feature_selection import VarianceThreshold
    selector = VarianceThreshold()
    selector.fit(X)
    X = selector.transform(X)

    #print X.shape
    #print train_instances.shape

if classifier == "decision_tree":
    from sklearn.tree import DecisionTreeClassifier
    clf = DecisionTreeClassifier(random_state = 0)
elif classifier == "random_forest":
    from sklearn.ensemble import RandomForestClassifier
    clf = RandomForestClassifier()
elif classifier == "MLP":
    from sklearn.neural_network import MLPClassifier
    clf = MLPClassifier(hidden_layer_sizes=(100,100,100), max_iter=100,
alpha=0.0001, solver='sgd', verbose=10,  random_state=21,tol=0.005)
elif classifier == "KNeighbors":
    from sklearn.neighbors import KNeighborsClassifier
    clf = KNeighborsClassifier(5)
elif classifier == "SVM":
    from sklearn import svm
    clf = svm.SVC()
elif classifier == "LinearSVM":
    from sklearn import svm
    clf = svm.LinearSVC()
elif classifier == "Naive_Bayes":
    from sklearn.naive_bayes import MultinomialNB
    clf = MultinomialNB()

if method == "none":
    clf2 = clf
    #from sklearn.model_selection import train_test_split
```

```
    #X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2)
    #clf.fit(X_train, y_train)
    #y_pred = clf.predict(X_test)
    #scores = accuracy_score(y_test, y_pred)
elif method == "binary_relevance":
    from skmultilearn.problem_transform import BinaryRelevance
    clf2 = BinaryRelevance(classifier=clf, require_dense=[False,True])
elif method == "classifier_chain":
    from skmultilearn.problem_transform import ClassifierChain
    # initialize classifier chains multi-label classifier
    # with a gaussian naive bayes base classifier
    clf2 = ClassifierChain(classifier=clf)
elif method == "label_powerset":
    from skmultilearn.problem_transform import LabelPowerset
    clf2 = LabelPowerset(classifier=clf, require_dense=[False, False])
elif method == "LabelPowerset_with_Clusterer":
    from skmultilearn.problem_transform import LabelPowerset
    from skmultilearn.cluster import IGraphLabelCooccurenceClusterer
    from skmultilearn.ensemble import LabelSpacePartitioningClassifier
    problem_transform_classifier = LabelPowerset(classifier=clf,
require_dense=[False, False])
    clusterer = IGraphLabelCooccurenceClusterer('fastgreedy',
weighted=True, include_self_edges=True)
    clf2 = LabelSpacePartitioningClassifier(problem_transform_classifier,
clusterer)

#from sklearn.model_selection import cross_val_score
#scores = cross_val_score(clf2, X, Y, cv=5, n_jobs = 1)
from sklearn.model_selection import cross_validate
scores = cross_validate(clf2, X, Y, cv=5, n_jobs =
1,scoring=['accuracy','precision_macro','recall_macro'])

if documents_no == -1:
    if tags_no == -1:
        results_file = "results_" + method + "_all.txt"
    else:
        results_file = "results_" + method + "_" + str(tags_no) +
"tags_all.txt"
else:
    if tags_no == -1:
        results_file = "results_" + method + "_" + str(documents_no) +
"docs.txt"
    else:
        results_file = "results_" + method + "_" + str(documents_no) +
"docs" + "_" + str(tags_no) + "tags.txt"

if variance_threshold:
    variance_folder = "VarianceThreshold"
else:
    variance_folder = "noVarianceThreshold"

if not os.path.exists("./results/"  + variance_folder + '/'):
    os.makedirs("./results/"  + variance_folder + '/')


if not os.path.exists("./results/"  + variance_folder + '/' + features +
'/'):
    os.makedirs("./results/"  + variance_folder + '/'  + features + '/')
```

```
if not os.path.exists("./results/"  + variance_folder + '/'  + features +
'/' + classifier + "/"):
    os.makedirs("./results/"  + variance_folder + '/'  + features + '/'
+ classifier + "/")

with open("./results/"  + variance_folder + '/'  + features + '/' +
classifier + "/" + results_file, 'w') as f:
    f.write(str(scores))
    f.write("\n")
    f.write("--- %s seconds ---" % (time.time() - start_time))
```

8.2.2. Plot.py

```
import os.path

import numpy as np
import matplotlib.pyplot as plt

documents_no = 100000
tags_no = 20

methods = ["none", "binary_relevance", "label_powerset",
"LabelPowerset_with_Clusterer"]
classifiers = ["decision_tree","random_forest",
"MLP","LinearSVM","SVM","Naive_Bayes"]

variance_threshold = True

if variance_threshold:
    variance_folder = "VarianceThreshold"
else:
    variance_folder = "noVarianceThreshold"

def score(features,classifier,method):
    if documents_no == -1:
        if tags_no == -1:
            results_file = "./results/"  + variance_folder + '/' +
features + '/' + classifier + "/" + "results_" + method + "_all.txt"
        else:
            results_file = "./results/"  + variance_folder + '/' +
features + '/' + classifier + "/" + "results_" + method + "_" +
str(tags_no) + "tags_all.txt"
    else:
        if tags_no == -1:
            results_file = "./results/"  + variance_folder + '/' +
features + '/' + classifier + "/" + "results_" + method + "_" +
str(documents_no) + "docs.txt"
        else:
            results_file = "./results/"  + variance_folder + '/' +
features + '/' + classifier + "/" + "results_" + method + "_" +
str(documents_no) + "docs" + "_" + str(tags_no) + "tags.txt"
    if os.path.isfile(results_file):
        with open(results_file, 'r') as f:
            text = ''
            for line in f:
                if "seconds" not in line:
                    text += line.replace('\n','')
                text = ' '.join(text.split())
            split_text = text.split('), ')
            for a in split_text:
                if a.startswith('\'test_accuracy'):
                    array = [float(x) for x in
a.replace("'test_accuracy': array([",'').replace("]","").replace("
","").split(',')]
            return (sum(array)/len(array))
    else:
        return 0.0

#https://matplotlib.org/gallery/statistics/barchart_demo.html

n_groups = len(classifiers)
```

```
for features in ["Text","Code","All"]:
#for features in ["Grams"]:
    fig, ax = plt.subplots()

    ax.grid(b=True,which='both')

    index = np.arange(n_groups)
    bar_width = 0.2

    opacity = 0.4
    error_config = {'ecolor': '0.3'}

    ax.set_xticklabels(classifiers)

    f = open("results_" + features + "_" + variance_folder + ".csv","w")
    f.write(",")
    for classifier in classifiers[:-1]:
        f.write(classifier)
        f.write(",")
    f.write(classifiers[-1])
    f.write("\n")

    scores = []
    for classifier in classifiers:
        scores.append(score(features,classifier,"none"))
    ax.bar(index, scores, bar_width, alpha=opacity, color='b',
label="none")
    f.write("none,")
    for sc in scores[:-1]:
        f.write(str(sc))
        f.write(",")
    f.write(str(scores[-1]))
    f.write("\n")

    scores = []
    for classifier in classifiers:
        scores.append(score(features,classifier,"binary_relevance"))
    ax.bar(index + bar_width, scores, bar_width, alpha=opacity,
color='r', label="binary_relevance")

    f.write("binary_relevance,")
    for sc in scores[:-1]:
        f.write(str(sc))
        f.write(",")
    f.write(str(scores[-1]))
    f.write("\n")

    scores = []
    for classifier in classifiers:
        scores.append(score(features,classifier,"label_powerset"))
    ax.bar(index + 2 * bar_width, scores, bar_width, alpha=opacity,
color='g', label="label_powerset")

    f.write("label_powerset,")
    for sc in scores[:-1]:
        f.write(str(sc))
        f.write(",")
    f.write(str(scores[-1]))
    f.write("\n")
```

```
    scores = []
    for classifier in classifiers:

scores.append(score(features,classifier,"LabelPowerset_with_Clusterer"))
    ax.bar(index + 3 * bar_width, scores, bar_width, alpha=opacity,
color='y', label="label_powerset_with_clusterer")

    f.write("label_powerset_with_clusterer,")
    for sc in scores[:-1]:
        f.write(str(sc))
        f.write(",")
    f.write(str(scores[-1]))
    f.write("\n")

    ax.set_xlabel('Classifiers')
    ax.set_ylabel('Accuracy')
    plt.axis([-0.5,6,0,0.6])
    #ax.set_title('Scores by group and gender')
    ax.set_xticks(index + bar_width + bar_width / 2)
    ax.legend()

    fig.tight_layout()
    plt.show()
    f.close()
```

8.2.3. Run_all.py

```
import os

for classifier in ["decision_tree","random_forest","MLP"]:
    for method in ["none","binary_relevance", "label_powerset",
"LabelPowerset_with_Clusterer"]:
        for features in ["Text", "Code", "All"]:
            print "Classifier: ", classifier, ", Method: ", method,
",Features: ", features
            command = "python final.py " + classifier + " " + method + "
" + features
            os.system(command)

for classifier in ["Naive_Bayes","LinearSVM","SVM"]:
    for method in ["binary_relevance", "label_powerset",
"LabelPowerset_with_Clusterer"]: #
        for features in ["Text", "Code", "All"]:
            print "Classifier: ", classifier, ", Method: ", method,
",Features: ", features
            command = "python final.py " + classifier + " " + method + "
" + features
            os.system(command)
```

9. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] M. IKONOMAKIS, S. KOTSIANTIS και V. TAMPAKAS, Text Classification Using Machine Learning Techniques, WSEAS International Conference on Computers, 2005.

[2] M. K. Dalal και M. A. Zaveri, Automatic Text Classification: A Technical Review, International Journal of Computer Applications, 2011.

[3] J. Nam, J. Kim και E. L. Mencia, Large-scale Multi-label Text Classification - Revisiting Neural Networks, Part of the Lecture Notes in Computer Science book series, 2014.

[4] M.-L. ZHANG, Y.-K. LI, X.-Y. LIU και X. GENG, Binary Relevance for Multi-Label Learning: An Overview, Frontiers of Computer Science, 2017.

[5] O. Luaces, J. Díez, J. Barranquero, J. J. d. Coz και A. Bahamonde, Binary relevance efficacy for multilabel classification, Progress in Artificial Intelligence.

[6] G. Tsoumakas και I. Katakis, Multi-Label Classification: An Overview, Int J Data Warehousing and Mining, 2009.

[7] H. Modi και M. Panchal, Experimental Comparison of Different Problem : Transformation Methods for Multi-Label Classification using MEKA, International Journal of Computer Applications, 2012.

[8] A. M. Santos, A. M. P. Canuto και A. F. Neto, A Comparative Analysis of Classification Methods to Multi-label Tasks in Different Application Domains, International Journal of Computer Information Systems and Industrial Management Applications, 2011.

[9] M.-L. Zhang και Z.-H. Zhou, A Review on Multi-Label Learning Algorithms, IEEE Transactions on Knowledge and Data Engineering, 2013.

[10] M. Allahyari, S. A. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez και K. J. Kochut, A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques, Survey of Text Mining, 2017.

[11] P. Cunningham και M. Cord, Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval, 2008.

[12] P. Dayan, Unsupervised Learning, The MIT Encyclopedia of the Cognitive Sciences, 1999.

[13] V. J. Prakash και D. L. Nithya, A Survey on Semi-Supervised Learning Techniques, International Journal of Computer Trends and Technology, 2014.

[14] T. M. Mitchell, Machine Learning, 1997.

[15] H. Zhang και D. Li, Naïve Bayes Text Classifier, IEEE International Conference on Granular Computing, 2007 .

[16] M. Steinbach, P.-N. Tan και V. Kumar, Introduction to Data Mining, 2005.

[17] S. Bharathidason και P. C. Jothi Venkataeswaran, Improving Classification Accuracy based on Random Forest Model with Uncorrelated High Performing Trees, International Journal of Computer Applications, 2014.

[18] J. R. Herrero και J. J. Navarro, Efficient Implementation of Nearest Neighbor, Computer Recognition Systems, 2005.

[19] P. Thomas, Perceptron learning for classification problems, 2015.

[20] B. Pang, L. Lee και S. Vaithyanathan, Thumbs up? Sentiment Classification using Machine Learning, Proceedings of EMNLP, 2002.

[21] Y. Tang, Deep Learning using Linear Support Vector Machines, ICML 2013 Challenges in Representation Learning Workshop, 2013.

[22] R. Collobert και J. Weston, A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning, Proceedings of the 25th international conference on Machine learning, 2008.

[23] S. Wang και C. D. Manning, Baselines and Bigrams: Simple, Good Sentiment and Topic Classification, 2012.

[24] W. Zhang, T. Yoshida και X. Tang, A comparative study of TF*IDF, LSI and multi-words for text classification, 2011.

[25] R. Venkatesan και M. J. Er, Multi-Label Classification Method Based on Extreme Learning Machines.