ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υβριδική Προσέγγιση της Μεθόδου Γράφων Λέξεων με POS Tagging και Ανατροφοδότηση του Μοντέλου Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αναστάσιος Μ. Αλεξόπουλος

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υβριδική Προσέγγιση της Μεθόδου Γράφων Λέξεων με POS Tagging και Ανατροφοδότηση του Μοντέλου Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αναστάσιος Μ. Αλεξόπουλος

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 13η Δεκεμβρίου 2017.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Θ. Βαρβαρίγου	*Ε. Βαρβαρίγος*	*Δ. Ασκούνης*
Καθηγήτρια Ε.Μ.Π.	*Καθηγητής Ε.Μ.Π.*	*Καθηγητής Ε.Μ.Π.*

Αθήνα, Δεκέμβριος 2017

.....

Αναστάσιος Μ. Αλεξόπουλος

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

Ευχαριστίες

Καταρχάς, θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια Θεοδώρα Βαρβαρίγου που μου ανέθεσε αυτή την εργασία, δίνοντάς μου τη δυνατότητα να ασχοληθώ με ένα πολύ ενδιαφέρον αντικείμενο, την Ανάλυση Συναισθήματος.

Επίσης, ευχαριστώ θερμά τον υποψήφιο Διδάκτορα του Ε.Μ.Π., Βρεττό Μουλό και τον Πέτρο Κληροδέτη, για το χρόνο που μου αφιέρωσαν, τη βοήθεια, τις οδηγίες και κατευθύνσεις που μου έδιναν σε όλα τα στάδια εκπόνησης της εργασίας.

Ευχαριστώ πολύ τον κ. Χάρη Πολίτη, διότι μέσα από τη συνεργασία μας τα τελευταία πέντε περίπου χρόνια έγινα, πιστεύω, καλύτερος επιστήμονας και άνθρωπος.

Ένα μεγάλο ευχαριστώ στους συμφοιτητές και φίλους μου, την Ανθούσα, το Ζαχαρία και τον Παναγιώτη, για αυτά τα πολύ όμορφα φοιτητικά χρόνια που μου χάρισαν.

Το μεγαλύτερο ευχαριστώ, όμως, οφείλω προπαντός στην οικογένειά μου, για την υπομονή και τη στήριξή τους κατά τη διάρκεια των σπουδών μου.

Αναστάσιος Μ. Αλεξόπουλος
Αθήνα, Δεκέμβριος 2017

Περίληψη

Ανάλυση συναισθήματος είναι ένας ερευνητικός κλάδος της Επεξεργασίας Φυσικής γλώσσας που αφορά στην υπολογιστική διαχείριση της κοινής γνώμης, των συναισθημάτων και της υποκειμενικότητας, από απλά κείμενα. Το αντικείμενο παρουσιάζει ολοένα και μεγαλύτερο ερευνητικό ενδιαφέρον τα τελευταία χρόνια, χάρη στην εξάπλωση της χρήσης του διαδικτύου.

Οι αλγόριθμοι ανάλυσης συναισθήματος διακρίνονται σε τρεις βασικές κατηγορίες ανάλογα με την προσέγγιση που χρησιμοποιούν. Υπάρχουν μέθοδοι μηχανικής μάθησης, μέθοδοι λεξιλογικής προσέγγισης, με χρήση λεξικού, και υβριδικές μέθοδοι, που αποτελούν συνδυασμό αυτών των δύο.

Μέχρι τα τελευταία χρόνια, οι υβριδικές μέθοδοι δεν ήταν ιδιαίτερα διαδεδομένες, εξαιτίας της πολυπλοκότητάς τους, όμως πλέον έχουν αρχίσει να αναπτύσσονται διάφορες μέθοδοι αυτής της κατηγορίας.

Στόχος της παρούσας εργασίας είναι η βελτίωση του υβριδικού αλγορίθμου ανάλυσης συναισθήματος με γράφους λέξεων, με την εισαγωγή νέων χαρακτηριστικών που ερείδονται στην ανάλυση του κειμένου σε μέρη του λόγου και την εξαγωγή του συναισθήματος για κάθε λέξη. Προτείνεται επίσης μια διαδικασία ανατροφοδότησης του αλγορίθμου, με σκοπό να μπορεί να ανταποκριθεί στις μεταβολές της γλώσσας.

Λέξεις Κλειδιά: ανάλυση συναισθήματος, γράφοι λέξεων, προεπεξεργαστής κειμένου, χαρακτηρισμός μερών του λόγου, λεξικό συναισθήματος, υβριδική προσέγγιση, ταξινομητής Naive Bayes, sentiment analysis, word graphs, document preprocessor, part of speech (POS) tagging, SentiWordNet, hybrid method, Naive Bayes classifier.

Abstract

Sentiment Analysis is an ongoing field of research in Natural Language Processing, that refers to the computational treatment of opinions, sentiments and subjectivity of texts. This field has recently enjoyed a huge burst of research activity, mostly due to the spread in use of the World Wide Web.

Sentiment classification techniques can be divided into machine learning approach, lexicon based approach and hybrid approach, which refers to a combined use of the former approaches. Until recently, the use of hybrid techniques was not frequent, because of their higher computational complexity, but now a lot of hybrid methods have started to develop.

This diploma thesis aims to optimize the hybrid Word Graph sentiment analysis method, by inserting new features based on part of speech tagging and extracting the sentiment of words individually. The thesis also suggests the implementation of a feedback mechanism for the method, in order for it to be able to adapt to the changes of language over time.

Keywords: sentiment analysis, word graphs, document preprocessor, part of speech (POS) tagging, SentiWordNet, hybrid method, Naive Bayes classifier.

Πίνακας περιεχομένων

Κατάλογος σχημάτων

Κατάλογος πινάκων

1

Εισαγωγή

1.1 Ανάλυση συναισθήματος

Η σημερινή εποχή χαρακτηρίζεται από τη ραγδαία αύξηση του όγκου των πληροφοριών που διατίθενται στο διαδίκτυο, ένα φαινόμενο που πλέον περιγράφεται με τον όρο Big Data. Χάρη στα μέσα κοινωνικής δικτύωσης (social media, λ.χ. Facebook, Twitter, κ.λπ.), ιστοσελίδες με κριτικές (λ.χ. IMDB, Yelp, TripAdvisor κ.λπ.), συζητήσεις σε forum, blog κ.λπ., διακινείται στο διαδίκτυο τεράστιος όγκος δεδομένων σε ψηφιακή μορφή, με τα οποία εκφράζονται κρίσεις, γνώμες ή και συναισθήματα των χρηστών του [24].

Έχοντας ως αντικείμενο τη διαχείριση αυτών των δεδομένων, η ανάλυση συναισθήματος (sentiment analysis) αποτελεί έναν ερευνητικό κλάδο της επεξεργασίας φυσικής γλώσσας (Natural Language Processing, NLP) που συγκεντρώνει ολοένα και μεγαλύτερο ενδιαφέρον από τις αρχές της προηγούμενης δεκαετίας.

1.1.1 Ορισμός

Με τον όρο «ανάλυση συναισθήματος» νοείται η χρησιμοποίηση τεχνικών επεξεργασίας φυσικής γλώσσας, ανάλυσης κειμένου, υπολογιστικής γλωσσολογίας, με σκοπό τη συστηματική αναγνώριση, εξαγωγή, ποσοτικοποίηση και μελέτη συναισθηματικών καταστάσεων, καθώς και υποκειμενικών πληροφοριών [18, 22].

Ουσιαστικά πρόκειται για μια διαδικασία προσδιορισμού της πολικότητας της γνώμης που εκφράζεται, ταξινόμησής της σε κλάσεις (classification) ή, με άλλα λόγια, του χαρακτηρισμού της λ.χ. ως θετική, αρνητική, ουδέτερη κ.ο.κ. [15].

Γίνεται δεκτό ότι η οι έννοιες της ανάλυσης συναισθήματος και της εξόρυξης γνώμης (opinion mining) εμφανίστηκαν σχεδόν παράλληλα και ότι διαφέρουν μεταξύ τους, όμως σε γενικές γραμμές χρησιμοποιούνται ως ταυτόσημες [32].

1.1.2 Εφαρμογές

Παράγοντες που οδήγησαν στην ανάπτυξη του ερευνητικού τομέα της ανάλυσης συναισθήματος αποτελούν, μεταξύ άλλων, α. η ανάπτυξη τεχνικών μηχανικής μάθησης στην επεξεργασία φυσικής γλώσσας και την ανάκτηση πληροφορίας, β. η μεγαλύτερη διαθεσιμότητα συνόλου δεδομένων για την εκπαίδευση αλγορίθμων μηχανικής μάθησης, χάρη στην άνθηση του παγκόσμιου ιστού (World Wide Web), και συγκεκριμένα η ανάπτυξη ιστοσελίδων όπου υπάρχει συσσώρευση κριτικών, και γ. η συνειδητοποίηση των εμπορικών εφαρμογών που προσφέρει το συγκεκριμένο ερευνητικό αντικείμενο [32].

Επιχειρήσεις και πάροχοι υπηρεσιών μελετούν κριτικές για τα προϊόντα τους εφαρμόζοντας τεχνικές ανάλυσης συναισθήματος, ώστε να προσαρμόζονται στις ανάγκες και τις απαιτήσεις των καταναλωτών και να αυξάνουν τα κέρδη τους.

Πολιτικοί αναλυτές εκτιμούν την πρόθεση ψήφου των πολιτών, ή τη γνώμη τους για κάποια υποψηφιότητα, κατά τη διάρκεια μιας πολιτικής εκστρατείας. Είναι επίσης δυνατό να προβλεφθεί το εκλογικό αποτέλεσμα, από σχετικές δημοσιεύσεις στα μέσα κοινωνικής δικτύωσης [27].

Η ανάλυση συναισθήματος σε χρηματιστηριακές ειδήσεις αποτελεί μια ανερχόμενη μέθοδο για την πρόβλεψη των χρηματιστηριακών τάσεων, γεγονός που βοηθά τις αποφάσεις των επενδυτών [44].

Ας σημειωθεί ότι για την υλοποίηση αυτών των εφαρμογών έχουν δημιουργηθεί πολλές νεοφυείς επιχειρήσεις (start-up companies), ενώ επίσης μεγάλες επιχειρήσεις στελεχώνουν το ανθρώπινο δυναμικό τους, μόνο για το σκοπό αυτό. Η ανάλυση συναισθήματος εφαρμόζεται σχεδόν σε οποιοδήποτε επιχειρηματικό και κοινωνικό τομέα [24].

1.2 Αλγόριθμοι για την ανάλυση συναισθήματος

Οι αλγόριθμοι για την ανάλυση συναισθήματος μπορούν να διακριθούν σε τρεις βασικές κατηγορίες, ανάλογα με την προσέγγιση που χρησιμοποιούν, και συγκεκριμένα σε:

α. προσέγγιση μηχανικής μάθησης (machine learning approach),

β. προσέγγιση βασισμένη στη χρήση λεξικού (lexicon-based approach),

γ. υβριδική προσέγγιση (hybrid approach) [27].

1.2.1 Προσέγγιση μηχανικής μάθησης (machine learning approach)

Η μηχανική μάθηση αποτελεί έναν ευρύτερο κλάδο της επιστήμης υπολογιστών, που σχετίζεται με την ικανότητα των υπολογιστών να «μαθαίνουν», να εκπαιδεύονται, χωρίς να έχουν προγραμματιστεί ρητώς.

Όσον αφορά στην ανάλυση συναισθήματος, η προσέγγιση μηχανικής μάθησης περιλαμβάνει την εκπαίδευση ενός μοντέλου επιβλεπόμενης μάθησης με ένα σύνολο δεδομένων εκπαίδευσης (training set). Από κάθε στιγμιότυπο εκπαίδευσης (training instance) εξάγονται κοινά πρότυπα (patterns) ή χαρακτηριστικά (features), τα οποία στη συνέχεια χρησιμοποιούνται για την αξιολόγηση του μοντέλου.

Η αξιολόγηση του μοντέλου γίνεται με την εκτίμηση της πολικότητας σε ένα σύνολο δεδομένων αξιολόγησης (testing set). Σε ένα στιγμιότυπο άγνωστης κλάσης από το νέο σύνολο δεδομένων, η απόκριση του αλγορίθμου μπορεί να είναι μια αυστηρή εκτίμηση της κλάσης (hard classification), ή ένα σύνολο τιμών που εκφράζουν την πιθανότητα το άγνωστο στιγμιότυπο να ανήκει σε κάθε κλάση (soft classification) [27].

Εάν για την εκπαίδευση του αλγορίθμου τα δεδομένα που χρησιμοποιούνται είναι επισημειωμένα (labeled) με την κλάση στην οποία ανήκουν, δηλαδή αν στο μοντέλο παρέχεται η πληροφορία της κλάσης πολικότητας των δεδομένων, τότε γίνεται λόγος για επιβλεπόμενη μάθηση (supervised learning). Στην αντίθετη περίπτωση πρόκειται για μη επιβλεπόμενη μάθηση (unsupervised learning). Στον αλγόριθμο δεν είναι γνωστό ποια πρέπει να είναι η απόκρισή του, όμως αναγνωρίζει κοινά πρότυπα στην είσοδο, και έτσι οι αλγόριθμοι αυτής της κατηγορίας χρησιμοποιούνται ιδίως για την ομαδοποίηση (clustering) των δεδομένων. Είναι επίσης δυνατό ο αλγόριθμος να εκπαιδεύεται τόσο με επισημειωμένα όσο και με μη επισημειωμένα δεδομένα, οπότε στην περίπτωση αυτή γίνεται λόγος για ημι-επιβλεπόμενη μάθηση (semi-supervised learning).

Πολλοί αλγόριθμοι ταξινόμησης έχουν σχεδιαστεί για καθεμία από αυτές τις κατηγορίες. Στην επιβλεπόμενη μάθηση οι πιο δημοφιλείς ταξινομητές (classifiers) είναι, μεταξύ άλλων, οι Naive Bayes (βλ. ενότητα 3.2.4), Maximum Entropy, Support Vector Machines.

1.2.2 Προσέγγιση βασισμένη στη χρήση λεξικού (lexicon-based approach)

Η προσέγγιση κάνει την παραδοχή ότι ο συναισθηματικός χαρακτηρισμός του κειμένου μπορεί να προκύψει από τον συναισθηματικό χαρακτηρισμό των επιμέρους λέξεων ή φράσεων του, και επομένως βασίζεται στην κατασκευή και χρήση μιας συλλογής λέξεων και φράσεων για τις οποίες είναι γνωστό ότι εκφράζουν κάποιο συναίσθημα. Η πολικότητα των λέξεων ή φράσεων εκφράζεται με μια αριθμητική τιμή η οποία έχει δοθεί εκ των προτέρων, και όλες αυτές καταχωρούνται σε ένα λεξικό συναισθήματος (sentiment lexicon).

Το λεξικό συναισθήματος μπορεί να κατασκευάζεται αυτόματα [12, 15, 42] ή και χειροκίνητα [38]. Σε κάθε περίπτωση έχει ασκηθεί κριτική ότι τα λεξικά που κατασκευάζονται είναι αναξιόπιστα. Στην πρώτη περίπτωση απαιτείται τεράστιος όγκος δεδομένων προκειμένου το λεξικό να παρέχει αξιόπιστα αποτελέσματα. Στη δεύτερη περίπτωση, ο χρόνος που απαιτείται για την επισημείωση της πολικότητας των λέξεων κρίνεται απαγορευτικός, ενώ

μπορεί στα αποτελέσματα να υπεισέρχεται και ουσιώδης υποκειμενικός παράγοντας [3]. Επομένως, σε γενικές γραμμές, η χρήση λεξικού συναισθημάτων δεν είναι επαρκής για την ανάλυση συναισθήματος.

Ακόμη, όσον αφορά την προσέγγιση με χρήση λεξικού, θα πρέπει να σημειωθούν τα εξής χαρακτηριστικά παραδείγματα:

α. Είναι δυνατό λέξεις με θετική ή αρνητική σημασία να έχουν αντίθετη πολικότητα ανάλογα με το θέμα στο οποίο αναφέρονται.

β. Μπορεί μια πρόταση να περιέχει λέξεις με θετική ή αρνητική πολικότητα, αλλά η πρόταση καθαυτή να μην εκφράζει κάποιο συναίσθημα.

γ. Μια πρόταση με ειρωνικό/σαρκαστικό περιεχόμενο δεν μπορεί να γίνει αντιληπτή μόνο από τις λέξεις που χρησιμοποιούνται σε αυτή.

δ. Μια πρόταση μπορεί να εκφράζει κάποια γνώμη ή συναίσθημα, χωρίς να περιέχει λέξεις με θετική ή αρνητική πολικότητα [24].

Υπάρχουν πολλά ευρέως διαδεδομένα κατασκευασμένα λεξικά, όπως τα WordNet, SentiWordNet (βλ. παρακάτω ενότητα 3.2.3), Multi Perspective Question Answering (MPQA) Subjectivity Lexicon κ.λπ. [32].

1.2.3 Υβριδική προσέγγιση (hybrid approach)

Η υβριδική προσέγγιση περιλαμβάνει συνδυασμό των ως άνω δύο προσεγγίσεων, με τα λεξικά συναισθημάτων να διαδραματίζουν ουσιώδη ρόλο στην πλειοψηφία των εφαρμοζόμενων τεχνικών. Υφιστάμενες τεχνικές χρησιμοποιούνται συνδυαστικά για να ξεπεραστούν οι περιορισμοί και να αξιοποιηθούν τα πλεονεκτήματα κάθε μιας, ώστε να αυξηθεί η ακρίβεια της ταξινόμησης [26, 37]. Λόγω της μεγάλης υπολογιστικής πολυπλοκότητας, η προσέγγιση μέχρι και λίγα χρόνια πριν δεν ήταν ιδιαίτερα διαδεδομένη [27].

1.3 Το αντικείμενο της παρούσας εργασίας

Ένας ήδη εφαρμοσμένος υβριδικός αλγόριθμος ανάλυσης συναισθήματος με χρήση n-grams έχει χρησιμοποιηθεί για την εξαγωγή χαρακτηριστικών από κριτικές ταινιών από βάση δεδομένων του IMDB [25]. Κάθε κριτική είναι μια παράγραφος κειμένου στην οποία γίνεται δεκτό ότι εκφράζεται γνώμη για κάποια ταινία, επομένως γίνεται προσπάθεια να εξαχθεί η πολικότητα της κριτικής. Τα χαρακτηριστικά που εξάγονται από το κείμενο κάθε κριτικής χρησιμοποιούνται για την εκπαίδευση και αξιολόγηση ενός μοντέλου μηχανικής μάθησης. Στην παρούσα εργασία ερευνώνται περαιτέρω χαρακτηριστικά τα οποία μπορούν να εξάγονται από το κείμενο και να παρέχονται ως είσοδος στο μοντέλο, καθώς και η εισαγωγή διαδικασίας ανατροφοδότησης (feedback), για την περαιτέρω βελτίωση της επίδοσής του.

1.3.1 Δομή της εργασίας

Στο Κεφάλαιο 2 γίνεται εκτενής περιγραφή ήδη υφιστάμενων αλγορίθμων ανάλυσης συναισθήματος, με γράφους n-grams και γράφους λέξεων. Περιγράφονται επίσης διάφορα προβλήματα λειτουργίας, που καθιστούν σκόπιμη τη διεύρυνσή και βελτίωσή τους.

Στο Κεφάλαιο 3 παρουσιάζεται μια νέα προτεινόμενη υβριδική μέθοδος ανάλυσης συναισθήματος, τα χαρακτηριστικά και ο τρόπος υλοποίησής της. Προτείνεται επίσης μια πρωτογενής διαδικασία ανατροφοδότησης του αλγορίθμου.

Στο Κεφάλαιο 4 γίνεται προσπάθεια αξιολόγησης της νέας προτεινόμενης μεθόδου μέσα από διάφορες μετρήσεις, καθώς και της λειτουργικότητας της ανατροφοδότησης, και περιγράφονται ορισμένα συμπεράσματα.

1.4 Σχετικές ερευνητικές εργασίες

Κάθε χρόνο δημοσιεύονται δεκάδες εργασίες σχετικά με την ανάλυση συναισθήματος. Όσον αφορά σε εργασίες με σύνολα δεδομένων από κριτικές ταινιών, ενδεικτικά παρατίθενται οι εξής:

Σε [33] έγινε πρωτότυπο έργο στον τομέα της ανάλυσης συναισθήματος. Οι συγγραφείς εργάστηκαν με πολλούς αλγόριθμους (Naive Bayes, Support Vector Machines) σε σύνολο δεδομένων από κριτικές ταινιών, όπου κάθε κριτική αναπαρίσταται με τη μέθοδο bag-of-words. Στη μέθοδο αυτή, σε κάθε λέξη αντιστοιχίζεται ένα βάρος, ανάλογα με την ύπαρξή της στο κείμενο. Γίνεται προσπάθεια αξιοποίησης στοιχείων όπως η άρνηση, καθώς και πληροφορίες για τη θέση των λέξεων ή το μέρος του λόγου (part of speech) που αποτελούν, χωρίς όμως ιδιαίτερη επιτυχία. Οι συγγραφείς καταλήγουν στο συμπέρασμα ότι πολλές τεχνικές, ενώ είναι τυπικά χρήσιμες για τον εντοπισμό του θέματος του κειμένου, επιδρούν αρνητικά στην ανίχνευση της πολικότητας του συναισθήματος [34].

Σε άλλη εργασία [31], οι ίδιοι συγγραφείς προτείνουν τη δημιουργία ενός φίλτρου προ-επεξεργασίας των κριτικών ταινιών, για την αφαίρεση όλων των μη υποκειμενικών προτάσεων που περιγράφουν την υπόθεση της ταινίας, χωρίς να αποδίδουν οποιαδήποτε ένδειξη για το συναίσθημα των χρηστών. Εκπαίδευσαν το φίλτρο με ένα σύνολο 5.000 υποκειμενικών και 5.000 μη υποκειμενικών προτάσεων από κριτικές ταινιών του IMDB. Το φίλτρο είχε ακρίβεια 92%, ενώ ο αλγόριθμος για την ταξινόμηση συναισθήματος με ταξινομητή Naive Bayes είχε ακρίβεια 86,4%, βελτιωμένη κατά περίπου 3% σε σχέση με προηγούμενη εργασία (με ταξινομητή Support Vector Machines δεν παρατηρήθηκε βελτίωση της ακρίβειας) [34].

Σε [14] προτείνεται η εκπαίδευση ενός γενικού ταξινομητή συναισθήματος με την ενσωμάτωση πληροφοριών από προϋπάρχον λεξικό συναισθήματος. Οι συγγραφείς

αναφέρονται στις πληροφορίες από το λεξικό ως επισημειωμένα χαρακτηριστικά, τα οποία χρησιμοποιούν για να περιορίσουν τις προβλέψεις του ταξινομητή σε μη ταξινομημένα στιγμιότυπα. Στην εργασία τους εξάγουν αυτόματα λέξεις με πολικότητα που σχετίζονται έντονα με μια συγκεκριμένη θεματική (domain-specific) και επιβεβαιώνουν την ιδέα ότι η πολικότητα των λέξεων μπορεί να διαφέρει, ανάλογα με το θέμα στο οποίο αυτές αναφέρονται. Στην προκείμενη περίπτωση η έρευνα διενεργήθηκε με σύνολα δεδομένων από κριτικές ταινιών και από πολυ-θεματικά (multi-domain) σύνολα δεδομένων, από το IMDB και το Amazon. Η προσέγγισή τους είχε καλύτερη επίδοση σε σύγκριση με άλλες μεθόδους ταξινόμησης συναισθήματος με πολύ μικρή επίβλεψη (weakly supervised sentiment classification methods), και μπορεί να εφαρμοσθεί για οποιαδήποτε περίπτωση ταξινόμησης σε κείμενο, όπου κάποια σχετική πληροφορία είναι γνωστή πριν την ταξινόμηση.

Σε [4] εξετάζονται δύο γενικές προσεγγίσεις μεθόδων στην ανάλυση συναισθήματος, μια βασισμένη σε λεξικό συναισθημάτων, και μια επιβλεπόμενης μάθησης. Οι μέθοδοι αξιολογήθηκαν σε ένα τυπικό σύνολο δεδομένων από κριτικές ταινιών και προέκυψε ότι η μέθοδος επιβλεπόμενης μάθησης ήταν καταφανώς καλύτερη σε σχέση με την μέθοδο που χρησιμοποιούσε λεξικό συναισθημάτων.

Εκτενέστερη περιγραφή και αναφορά σε μελέτες για ποικίλες μεθόδους ανάλυσης συναισθήματος σε διάφορες εφαρμογές μπορεί να βρεθεί σε [27].

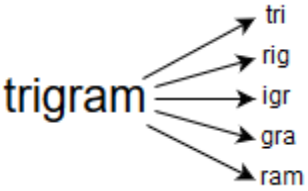
2

Ανάλυση συναισθήματος με n-grams και γράφους λέξεων

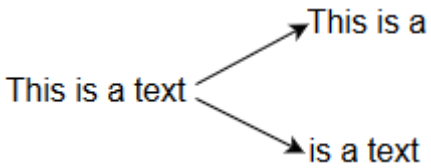
2.1 N-grams

Το n-gram είναι μια ακολουθία *n* συνεχόμενων στοιχείων που εξάγονται από μια ακολουθία χαρακτήρων. Τα στοιχεία μπορεί να είναι χαρακτήρες (character n-grams), ή ακόμα και λέξεις (word n-grams). Για την περίπτωση *n=1* τα εξαγόμενα n-grams ονομάζονται unigrams, ενώ επίσης για την περίπτωση *n=2* τα εξαγόμενα n-grams ονομάζονται bigrams.

Για την εξαγωγή των n-grams λαμβάνουμε επικαλυπτόμενες ακολουθίες των *n* στοιχείων, κατά περίπτωση, όπως φαίνεται ενδεικτικά στα κάτωθι σχήματα (για *n=3*):



Σχήμα 2.1: Τα character 3-grams που σχηματίζονται από την ακολουθία 'trigram'.



Σχήμα 2.2: Τα word 3-grams που σχηματίζονται από την ακολουθία 'This is a text'.

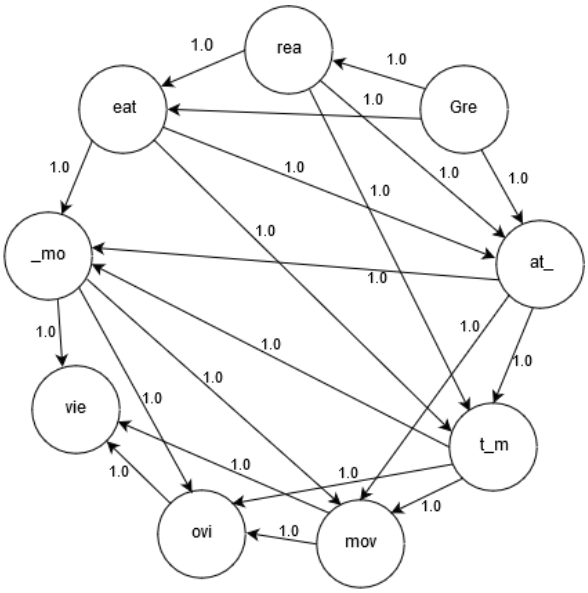
2.1.1 Γράφοι n-gram

Σε μελέτες για μοντέλα εξαγωγής περίληψης κειμένου (text summarization) [8, 9] έχει αναπτυχθεί η ιδέα αναπαράστασης κειμένων με γράφους, οι κόμβοι των οποίων είναι n-grams. Κατά τη διαδικασία επιλέγεται μια τιμή παραθύρου *W* (window) σύμφωνα με την οποία καθορίζεται ότι κάθε κόμβος συνδέεται με μια ακμή με τα *W* επόμενα n-grams (εφόσον υπάρχουν). Οι ακμές έχουν βάρη ανάλογα με το πλήθος εμφανίσεων του εκάστοτε ζεύγους n-grams.

Για παράδειγμα, για *n=3* σε κείμενο 'Great movie' σχηματίζονται τα εξής επικαλυπτόμενα 3-grams (ο κενός χαρακτήρας λαμβάνεται υπ' όψιν και συμβολίζεται με «_»):

1. Gre
2. rea
3. eat
4. at
5. t_m
6. mo
7. mov
8. ovi
9. vie

Με παράθυρο *W=3* για το ως άνω κείμενο σχηματίζεται ο εξής γράφος:



Σχήμα 2.3: Character 3-gram γράφος για το κείμενο 'Great movie'.

2.2 Ο αλγόριθμος ανάλυσης συναισθήματος με n-grams

Σε [1, 2] παρουσιάσθηκε η ιδέα χρησιμοποίησης της αναπαράστασης n-grams γράφων για ανάλυση συναισθήματος. Η διαδικασία εκκινεί με τη δημιουργία n-gram γράφου

για το πρώτο κείμενο του συνόλου δεδομένων. Στη συνέχεια, για κάθε επόμενο κείμενο, ο n-gram γράφος που δημιουργείται συγχωνεύεται με τον αρχικό, με αποτέλεσμα τη δημιουργία ενός μεγάλου γράφου μοντέλου (model graph).

Κατά τη συγχώνευση των γράφων εάν επανεμφανίζονται ζεύγη n-grams θα πρέπει να μεταβάλλεται το βάρος της αντίστοιχης ακμής. Μια ακμή που εμφανίζεται $n-1$ φορές στο γράφο μοντέλου και άλλη μια φορά σε γράφο κειμένου, θα πρέπει να συγχωνεύεται με βάρος που είναι ο μέσος όρος των n βαρών.

Η νέα τιμή του βάρους της ακμής υπολογίζεται ως εξής:

$$new\_average = old\_average + \frac{weight - old\_average}{n}$$

Στον ως άνω μαθηματικό τύπο οι συμβολισμοί αναλύονται ως εξής:

weight: το βάρος της ακμής στον υπό συγχώνευση γράφο

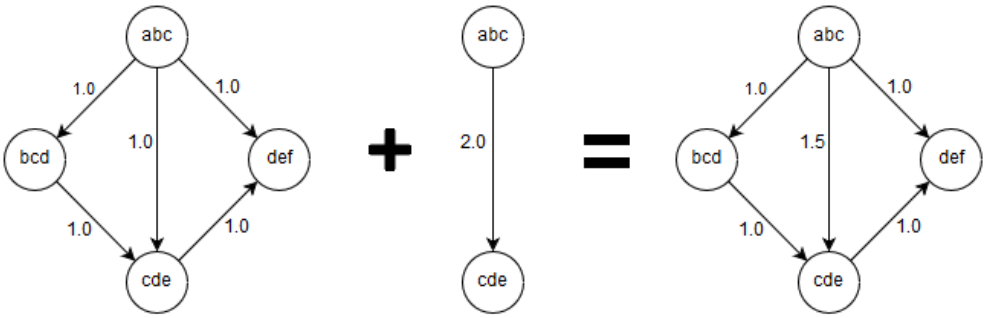
n: ο αριθμός των γράφων που έχουν ήδη συγχωνευθεί

new_average: το νέο βάρος που θα λάβει η ακμή στο νέο συγχωνευμένο γράφο.

old_average: είναι η τιμή της ακμής στον model graph, και συγκεκριμένα

$$old\_average = \frac{w_1 + w_2 + \ldots + w_{n-1}}{n-1}$$

Τα παρακάτω μπορούν να γίνουν κατανοητά με το εξής παράδειγμα:



Σχήμα 2.4: Συγχώνευση ήδη υπάρχουσας ακμής σε γράφο.

Με βάση τα ανωτέρω, δημιουργούνται δύο μεγάλοι γράφοι, ένας με θετική πολικότητα (positive model graph) και ένας με αρνητική πολικότητα (negative model graph). Οι γράφοι αυτοί χρησιμοποιούνται για την εξαγωγή χαρακτηριστικών (features) που δίδονται ως είσοδος σε ένα μοντέλο μηχανικής μάθησης.

Συγκεκριμένα, για κάθε κείμενο που δίδεται στον αλγόριθμο προς ταξινόμηση σχηματίζεται ο αντίστοιχος n-gram γράφος. Ο γράφος αυτός (μικρός γράφος, G^i) συγκρίνεται με τους δύο κατασκευασμένους model graphs (μεγάλοι γράφοι, G^j) και εξάγονται τα ακόλουθα χαρακτηριστικά [9]:

1. Co-occurrence/Containment Similarity (CS): Το πηλίκο του αριθμού ακμών του μικρού γράφου G^i που εμφανίζονται στον μεγάλο γράφο G^j, προς το σύνολο των ακμών του.

Ας σημειωθεί ότι δεν λαμβάνονται υπ' όψιν τα βάρη των ακμών, αλλά η τυχόν ύπαρξή τους σε κάποιο γράφο. Η τιμή του χαρακτηριστικού βρίσκεται μεταξύ 0 και 1. Στον παρακάτω μαθηματικό τύπο μ είναι συνάρτηση συμμετοχής (membership function) που λαμβάνει τιμή 1 εάν η ακμή e ανήκει στο γράφο G^j, διαφορετικά λαμβάνει τιμή 0. Επίσης, με $|G|$ συμβολίζεται το πλήθος ακμών του γράφου G.

$$CS(G^i, G^j) = \frac{\sum_{e \in G^i} \mu(e, G^j)}{\max(|G^i|, |G^j|)}$$

2. Size Similarity (SS): Το πηλίκο του αριθμού ακμών του μικρού γράφου προς τον αριθμό ακμών του μεγάλου γράφου. Η τιμή του χαρακτηριστικού βρίσκεται μεταξύ 0 και 1.

$$SS(G^i, G^j) = \frac{\min(|G^i|, |G^j|)}{\max(|G^i|, |G^j|)}$$

3. Value Similarity (VS): Ο αριθμός των ακμών που εμφανίζονται στον μικρό γράφο, που εμφανίζονται και στον μεγάλο, προς τον αριθμό ακμών του μεγάλου γράφου, λαμβάνοντας υπ' όψιν και τα βάρη των ακμών. Αν μια ακμή υπάρχει και στους δύο γράφους, τότε χρησιμοποιείται το μικρότερο από τα δύο βάρη. Η τιμή του χαρακτηριστικού είναι μεταξύ 0 και 1. Στον παρακάτω μαθηματικό τύπο είναι w_e^i το βάρος της ακμής e στον γράφο G^i, και επίσης w_e^j είναι το βάρος της ακμής e στον γράφο G^j.

$$VS(G^i, G^j) = \frac{\sum_{e \in G^i} (\mu(e, G^j) \times \frac{\min(w_e^i, w_e^j)}{\max(w_e^i, w_e^j)})}{\max(|G^i|, |G^j|)}$$

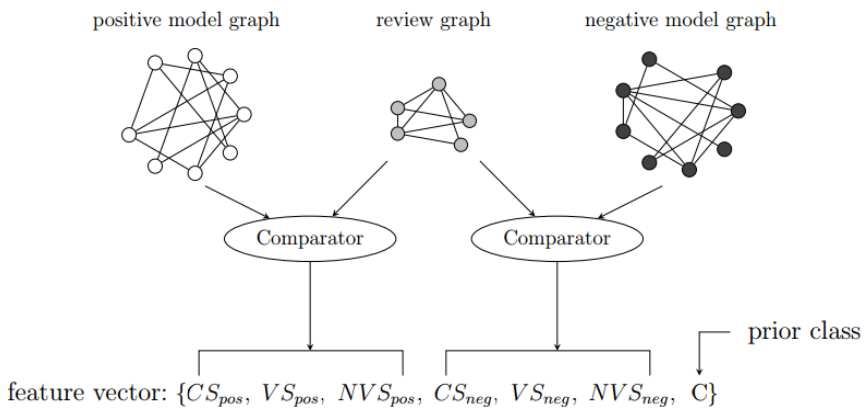
4. Normalized Value Similarity (NVS): Το χαρακτηριστικό αυτό είναι ενδεικτικό της ομοιότητας των δύο γράφων, πλην όμως, στην περίπτωση αυτή το μέγεθος των γράφων δεν διαδραματίζει κάποιο ρόλο. Η τιμή του χαρακτηριστικού είναι μεταξύ 0 και 1, και υπολογίζεται ως εξής:

$$NVS(G^i, G^j) = \frac{VS}{\frac{\min(|G^i|, |G^j|)}{\max(|G^i|, |G^j|)}}$$

Συγκρίνοντας τον μικρό γράφο με τους δύο μεγάλους γράφους θετικής και αρνητικής πολικότητας αποθηκεύουμε σε ένα διάνυσμα (feature vector) τις τιμές των χαρακτηριστικών CS, VS, NVS, συνολικά δηλαδή έξι αριθμητικές τιμές μεταξύ 0 και 1.

Αποθηκεύουμε επίσης στο διάνυσμα την τιμή της κλάσης στην οποία ανήκει το κείμενο ανάλογα με την πολικότητα (0 για αρνητική, 1 για θετική).

Στο τέλος για κάθε στιγμιότυπο που δίδεται στο μοντέλο μηχανικής μάθησης για την εκπαίδευσή του, δημιουργείται ένα διάνυσμα επτά τιμών, όπως εξηγείται και στο ακόλουθο σχήμα.



Σχήμα 2.5: Αναπαράσταση της σύγκρισης γράφου με τους γράφους πολικότητας για την εξαγωγή χαρακτηριστικών.

Όταν ο αλγόριθμος εκπαιδευθεί, εξετάζεται η ακρίβειά του με νέα στιγμιότυπα. Για αυτά ο αλγόριθμος δέχεται ως είσοδο τις 6 πρώτες τιμές του διανύσματος και αποκρίνεται με μια εκτίμηση για την πολικότητα του κειμένου που έχει δοθεί.

Σε προηγούμενη εργασία σχετικά με την επίδοση του ως άνω αλγορίθμου σε κριτικές ταινιών διατυπώθηκαν προβληματισμοί ερειδόμενοι στη λειτουργία των n-grams [45]. Για το λόγο αυτό προτάθηκε η χρησιμοποίηση μιας νέας μεθόδου ανάλυσης συναισθήματος, και συγκεκριμένα η αναπαράσταση των κειμένων με n-grams λέξεων. Η μέθοδος αυτή αποκαλείται μέθοδος ανάλυσης συναισθήματος με γράφους λέξεων (word graphs) [43].

2.3 *Ανάλυση συναισθήματος με γράφους λέξεων (word graphs)*

Στην προτεινόμενη νέα μέθοδο, κάθε λέξη του κειμένου αναπαρίσταται με έναν κόμβο στον σχηματιζόμενο γράφο. Και πάλι επιλέγεται μια τιμή παραθύρου W (window), με την οποία καθορίζεται ο αριθμός γειτονικών λέξεων, δηλαδή κάθε κόμβος του γράφου συνδέεται με μια ακμή με τους κόμβους που αντιστοιχούν στις W επόμενες λέξεις (εφόσον υπάρχουν).

Για παράδειγμα, σε κείμενο ‘I thought this was a quite good movie’, με παράθυρο *W=3*, η λέξη *was* γειτνιάζει με τις λέξεις: ‘a’, ‘quite’, ‘good’.

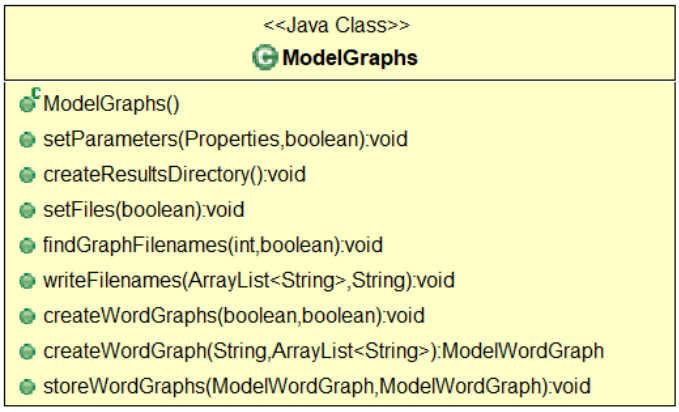
Και σε αυτή την περίπτωση οι ακμές έχουν βάρη, ανάλογα με το πλήθος εμφανίσεων ενός ζεύγους λέξεων εντός του επιλεγμένου παραθύρου, στο κείμενο.

Η μέθοδος ακολουθεί παρόμοια λογική με την ανάλυση συναισθήματος με n-grams. Η υλοποίηση της μεθόδου, όσον αφορά στην ανάλυση συναισθήματος με γράφους λέξεων, μπορεί να διακριθεί σε τρία στάδια όπως περιγράφεται ακολούθως.

2.3.1 *Πρώτο στάδιο*

Στο πρώτο στάδιο δημιουργούνται οι γράφοι μοντέλα (λέξεων), θετικής και αρνητικής πολικότητας (positive/negative model word graph). Οι γράφοι πολικότητας

αποθηκεύονται ώστε να μπορούν να ανακτηθούν στο μέλλον, για να χρησιμοποιηθούν σε διαφορετικά πειράματα. Το στάδιο αυτό δεν ασχολείται με την εκπαίδευση ή αξιολόγηση του μοντέλου μηχανικής μάθησης και ως εκ τούτου θα μπορούσε να χαρακτηρισθεί ως στάδιο προεπεξεργασίας. Για το στάδιο αυτό δημιουργείται μια κλάση με όνομα ModelGraphs, με τις μεθόδους που φαίνονται στο ακόλουθο σχήμα.



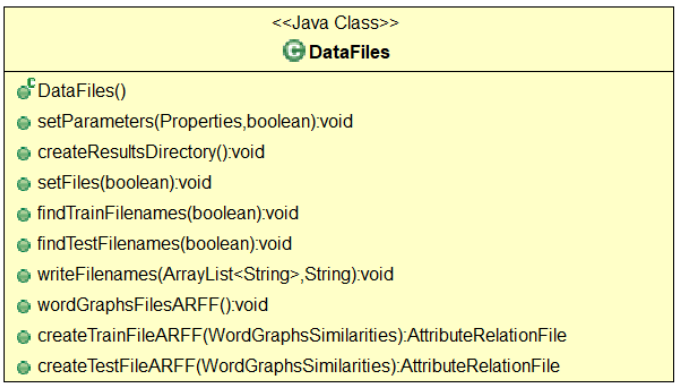
Σχήμα 2.6: Διάγραμμα UML με τις βασικές συναρτήσεις της κλάσης ModelGraphs.

2.3.2 Δεύτερο στάδιο

Το δεύτερο στάδιο περιλαμβάνει τη σύγκριση γράφων από κείμενα κριτικής που θα χρησιμοποιηθούν για την εκπαίδευση ή την αξιολόγηση της μεθόδου, με τους γράφους πολικότητας. Δημιουργούνται κατ' αυτό τον τρόπο διανύσματα χαρακτηριστικών (instances), τα οποία αποθηκεύονται σε αρχεία της μορφής Attribute Relation File Format (ARFF), ένα για την εκπαίδευση και ένα για την αξιολόγηση της μεθόδου.

Μετά το σημείο αυτό ο αλγόριθμος δεν έχει πλέον γνώση του κειμένου της εκάστοτε κριτικής, αλλά ουσιαστικά έχει στη διάθεσή του έξι αριθμούς που προέκυψαν από τις συγκρίσεις με τους γράφους, από το συνδυασμό των οποίων θα πρέπει στη συνέχεια να αποκριθεί αν πρόκειται για θετική ή αρνητική κριτική.

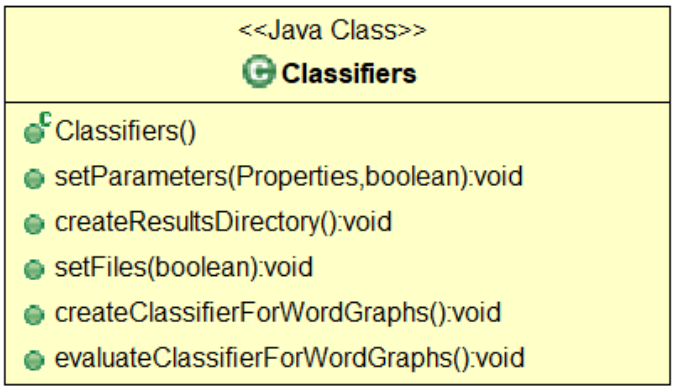
Για το σκοπό αυτό δημιουργείται μια κλάση με όνομα DataFiles, με τις μεθόδους που φαίνονται στο παρακάτω σχήμα (ο κώδικας παρατίθεται στο παράρτημα).



Σχήμα 2.7: Διάγραμμα UML με τις βασικές συναρτήσεις της κλάσης DataFiles.

2.3.3 *Τρίτο στάδιο*

Στο τρίτο στάδιο δίδονται τα αρχεία που κατασκευάστηκαν στο δεύτερο στάδιο, ως είσοδος σε έναν ταξινομητή, για την εκπαίδευση και την αξιολόγησή του. Για το σκοπό αυτό δημιουργείται μια κλάση με όνομα Classifiers, με τις μεθόδους που φαίνονται στο παρακάτω σχήμα.



Σχήμα 2.8: Διάγραμμα UML της κλάσης Classifiers.

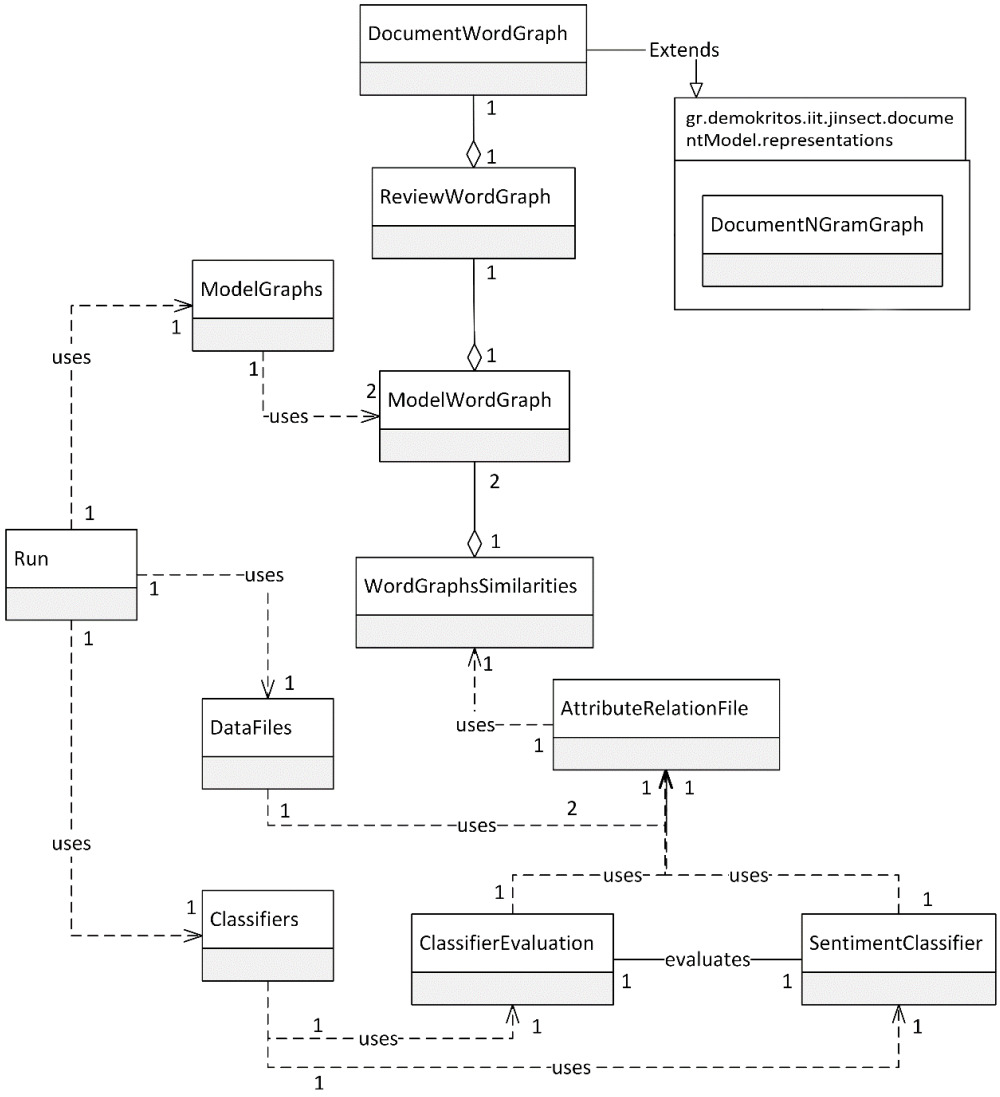
2.3.4 *Συνολική εκτέλεση του αλγορίθμου*

Το κείμενο κάθε κριτικής αναπαρίσταται σε γράφο λέξεων με τη χρησιμοποίηση της κλάσης με όνομα ReviewWordGraph. Κατά τα περιγραφόμενα ανωτέρω, στο πρώτο στάδιο συγχωνεύονται οι γράφοι από πολλές κριτικές σε γράφους μοντέλα, θετικής και αρνητικής πολικότητας. Οι γράφοι πολικότητας περιγράφονται από την κατασκευασθείσα κλάση με το όνομα ModelWordGraph. Για τη συγχώνευση γράφων και λοιπές ενέργειες σε γράφους (λ.χ. αφαίρεση κοινού τμήματος γράφων, δημιουργία αντιγράφων κ.λπ.), η κλάση ReviewWordGraph υλοποιεί την κλάση DocumentWordGraph, μια επέκταση της κλάσης DocumentNGramGraph που υπάρχει στη βιβλιοθήκη JInsect[1] και υλοποιεί n-gram γράφους. Η κλάση DocumentWordGraph πρόκειται στην ουσία για επέκταση της κλάσης DocumentNGramGraph ώστε να χειρίζεται λέξεις, αντί n-άδες χαρακτήρων. Για την υλοποίηση της σύγκρισης γράφων λέξεων χρησιμοποιείται η κλάση με το όνομα WordGraphsSimilarities. Η κλάση WordGraphsSimilarities χρησιμοποιεί την κλάση NGramCachedGraphComparator για την αποδοτική σύγκριση γράφων, που υλοποιείται στη βιβλιοθήκη JInsect.

Για την δημιουργία των αρχείων εκπαίδευσης και αξιολόγησης του ταξινομητή χρησιμοποιείται η κλάση AttributeRelationFile. Η κλάση αποθηκεύει τις τιμές των διανυσμάτων χαρακτηριστικών σε αρχεία ARFF.

[1] http://sourceforge.net/projects/jinsect/.

Για τη δημιουργία του ταξινομητή χρησιμοποιείται η κλάση SentimentClassifier. Η κλάση αυτή χρησιμοποιεί τη βιβλιοθήκη weka[2] που ενσωματώνει υλοποιήσεις πολλών ταξινομητών και μεθόδους αξιολόγησής τους. Η κλάση SentimentClassifier χρησιμοποιεί το αρχείο εκπαίδευσης που δημιουργείται από την κλάση AttributeRelationFile για την εκπαίδευση του ταξινομητή. Η αξιολόγησή του ταξινομητή πραγματοποιείται με την κλάση ClassifierEvaluation, χρησιμοποιώντας το αρχείο ελέγχου που δημιουργήθηκε, επίσης από την κλάση AttributeRelationFile.



Σχήμα 2.9: Διάγραμμα UML των κλάσεων που υλοποιούν τον αλγόριθμο ανάλυσης συναισθήματος με γράφους λέξεων.

[2] http://www.cs.waikato.ac.nz/~ml/weka/.

2.3.5 Παράμετροι για την εκτέλεση του αλγορίθμου

Για το πρώτο στάδιο κατασκευής των γράφων πολικότητας έχουμε τις εξής παραμέτρους:

1. graph reviews: ο αριθμός των κριτικών ταινιών (reviews), το κείμενο των οποίων θα ενσωματωθεί σε γράφο πολικότητας.

2. window: μήκος παραθύρου.

3. remove: καθορίζει αν θα αφαιρεθεί ο κοινός υπογράφος των δυο γράφων πολικότητας στην περίπτωση που τίθεται σε true.

4. preprocess: καθορίζει αν θα γίνεται προεπεξεργασία των κειμένων προκειμένου να αφαιρεθούν ειδικοί χαρακτήρες, προτού αυτό ενσωματωθεί στο γράφο, όταν τίθεται σε true.

Όσον αφορά τη μηχανική μάθηση, το δεύτερο και τρίτο στάδιο:

5. training reviews: ο αριθμός των κριτικών που θα χρησιμοποιηθεί στο σύνολο δεδομένων για την εκπαίδευση του μοντέλου μηχανικής μάθησης (training set).

6. test reviews: ο αριθμός των κριτικών που θα χρησιμοποιηθεί στο σύνολο δεδομένων για την αξιολόγηση του μοντέλου μηχανικής μάθησης (test set).

7. positive rate: το ποσοστό των κριτικών των δυο παραπάνω συνόλων που θα ανήκουν στη θετική κλάση (positive).

8. classifier: το όνομα του ταξινομητή weka ο οποίος θα χρησιμοποιηθεί.

Περιλαμβάνονται και γενικές παράμετροι με τις οποίες καθορίζονται τα διαστήματα στα οποίο θα κυμαίνονται οι βαθμολογίες των κριτικών που χρησιμοποιούνται για τους γράφους πολικότητας, το σύνολο εκπαίδευσης ή το σύνολο αξιολόγησης κ.λπ.:

9. minPosRating: η μικρότερη τιμή που μπορεί να έχει κριτική ώστε να είναι θετική.

10. maxPosRating: η μεγαλύτερη τιμή που μπορεί να έχει κριτική ώστε να είναι θετική.

11. minNegRating: η μικρότερη τιμή που μπορεί να έχει κριτική ώστε να είναι αρνητική.

12. maxNegRating: η μεγαλύτερη τιμή που μπορεί να έχει κριτική ώστε να είναι αρνητική.

13. shuffle: καθορίζει αν οι κριτικές που χρησιμοποιούνται για τη δημιουργία των γράφων πολικότητας και των συνόλων εκπαίδευσης και αξιολόγησης θα δίδονται με τυχαία σειρά.

2.3.6 Μελέτη επίδοσης του αλγορίθμου - Προβλήματα λειτουργίας

Η μέθοδος ανάλυσης συναισθήματος με γράφους λέξεων έχει αξιολογηθεί στο παρελθόν ως προς την επίδοσή της, η οποία φτάνει σε ακρίβεια πρόβλεψης περί το 80%. Θα πρέπει ωστόσο να παρατηρηθούν μερικά προβλήματα λειτουργίας:

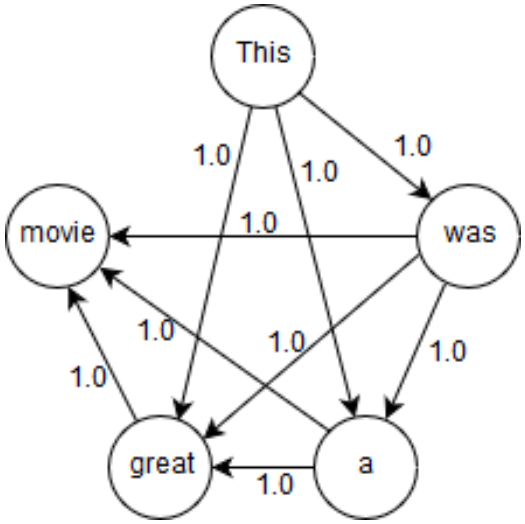
1. Η μέθοδος δεν έχει τη δυνατότητα να διακρίνει την άρνηση σε μια πρόταση, με ικανοποιητικό τρόπο.

Για παράδειγμα ας ληφθεί υπ' όψιν χαρακτηριστικά η πρόταση 'This was a great movie'. Σε αυτήν σχηματίζονται τα εξής word-grams: 'This', 'was', 'a', 'great', 'movie'.

Αντίστοιχα, αν θεωρήσουμε την πρόταση 'This was not a great movie', σε αυτή σχηματίζονται τα εξής word-grams: 'This', 'was', 'not', 'a', 'great', 'movie'. Στην περίπτωση αυτή, πέντε από τα έξι word-grams που σχηματίζονται είναι ίδια με αυτά της προηγούμενης πρότασης.

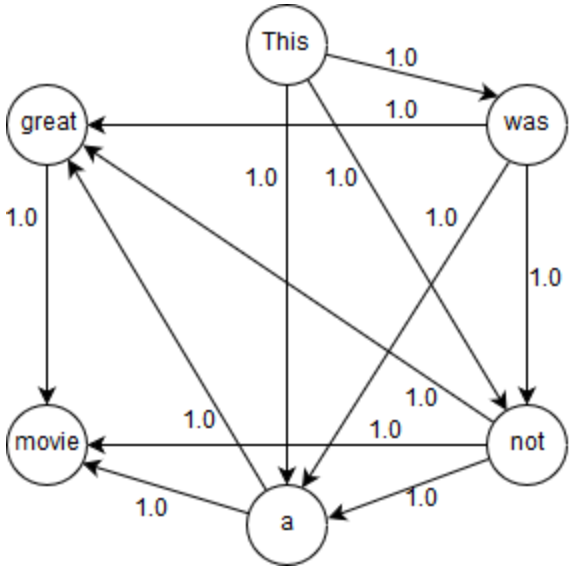
Το ίδιο θα ισχύει και για τους κόμβους του γράφου που θα δημιουργηθούν, είναι δηλαδή σχεδόν ίδιοι με τους κόμβους του γράφου που σχηματίζεται από την προηγούμενη πρόταση. Συγκεκριμένα, με παράθυρο $W=3$ θα δημιουργηθούν οι εξής γράφοι.

Για την πρόταση 'This was a great movie':



Σχήμα 2.10: Word 3-gram γράφος για το κείμενο 'This was a great movie', με παράθυρο 3.

Για την πρόταση 'This was not a great movie':



Σχήμα 2.11: Word 3-gram γράφος για το κείμενο 'This was not a great movie'.

Στο γράφο της πρώτης πρότασης υπάρχουν 9 ακμές, ενώ σε αυτόν της δεύτερης πρότασης υπάρχουν 12 ακμές. Από αυτές τις ακμές, οι 7 είναι κοινές και για τους δύο γράφους. Επομένως, καταρχάς, υπάρχει μεγάλη ομοιότητα των γράφων μεταξύ τους, παρά το γεγονός ότι εκφράζουν συναίσθημα αντίθετης πολικότητας.

Ας αναλογιστούμε, όμως, επίσης, ότι στην πράξη αυτοί οι μικροί γράφοι θα έπρεπε να συγκριθούν με γράφους πολικότητας. Οι γράφοι πολικότητας, κατά τα ανωτέρω, έχουν κατασκευαστεί από μεγάλο σύνολο δεδομένων κριτικών ταινιών, των οποίων η πολικότητα είναι γνωστή εκ των προτέρων.

Η μέθοδος ανάλυσης συναισθήματος με γράφους λέξεων έχει κατά την υλοποίησή της παράμετρο remove, με την οποία καθορίζεται αν θα αφαιρεθεί ο κοινός υπογράφος των γράφων πολικότητας. Στην περίπτωση που δεν αφαιρείται ο κοινός υπογράφος, τότε με μεγάλη πιθανότητα οι προτάσεις αυτές θα έχουν μεγάλη ομοιότητα και με τους δύο γράφους, με μεγάλο κίνδυνο ο αλγόριθμος να δώσει επισφαλές αποτέλεσμα. Στην περίπτωση που αφαιρεθεί ο κοινός υπογράφος των γράφων πολικότητας, τότε η πρόταση θα είχε πολύ μικρή ομοιότητα και με τους δύο γράφους. Η βασική πρόταση, δηλαδή, από την οποία θα χαρακτήριζε κανείς την πολικότητα της κριτικής, δεν έχει καμία χρηστικότητα για τον αλγόριθμο.

Με βάση τα ανωτέρω, είναι προφανές ότι η μέθοδος ανάλυσης συναισθήματος χρειάζεται κάποιο τρόπο με τον οποίο να μπορεί να αναγνωρίζει την άρνηση σε μια πρόταση και να τη χειρίζεται κατάλληλα.

2. Οι γράφοι πολικότητας κατασκευάζονται άπαξ και στη συνέχεια παραμένουν αμετάβλητοι.

Η γλώσσα αλλάζει με το χρόνο, νέες λέξεις ή φράσεις δημιουργούνται, το νόημα ή ακόμα και η χρήση των λέξεων μεταβάλλεται. Οι αλλαγές αυτές μπορεί να είναι πρακτικές (λ.χ. λέξεις είναι υπερβολικά μεγάλες ή ηχούν όμοια με προσβλητικές λέξεις, ή αποτελούν ταμπού κ.λπ.) [11], ή να οφείλονται σε παράγοντες γλωσσολογικούς, κοινωνικούς, πολιτισμικούς, ψυχολογικούς [6], πολιτικές ή ιστορικές εξελίξεις κ.ο.κ.

Για παράδειγμα, όταν επρόκειτο να διεξαχθεί δημοψήφισμα στο Ηνωμένο Βασίλειο για την παραμονή ή την αποχώρησή της χώρας από την Ευρωπαϊκή Ένωση τον Ιούνιο του 2016, δημιουργήθηκε η λέξη Brexit, σύνθετη λέξη από τις λέξεις ‘British’ και ‘exit’ (αντίστοιχο παράδειγμα αποτελεί και η δημιουργία της λέξης Grexit, για την αποχώρηση της Ελλάδας από την Ευρωπαϊκή Ένωση).

Ως δεύτερο παράδειγμα, στις εκλογές των ΗΠΑ το 2016, ο εκλεγείς Πρόεδρος χρησιμοποιούσε στην προεκλογική του εκστρατεία το σλόγκαν ‘Make America Great Again’. Δημιουργήθηκε έτσι η συντομογραφία MAGA, η οποία πλημμύρισε τα μέσα κοινωνικής δικτύωσης.

Ως τρίτο παράδειγμα, η αγγλική λέξη ‘terrific’ παλαιότερα χρησιμοποιείτο με αρνητική σημασία (τρομακτικός, κάτι που τρομοκρατεί), και μόνο πρόσφατα άρχισε να ερμηνεύεται με θετική σημασία (φοβερός, τέλειος, φανταστικός, απίθανος[3]) [30].

Σε μακροχρόνια βάση όλες οι μεταβολές της γλώσσας θα είχαν αρνητική επίδραση στην επίδοση του αλγορίθμου. Νέες λέξεις ή φράσεις δεν θα έβρισκαν αντιστοίχιση στους γράφους πολικότητας, θα αποτελούσαν δηλαδή «θόρυβο» για το σύστημα, ή δυνητικά θα αντιστοιχίζονταν σε λάθος πολικότητα.

Επομένως, θα ήταν σκόπιμο να προστεθεί κάποια διαδικασία ανατροφοδότησης (feedback) στον αλγόριθμο, με την οποία θα μπορούσε να προσαρμόζεται και να εναρμονίζεται με τις μεταβολές της γλώσσας.

Θα μπορούσε για παράδειγμα ο αλγόριθμος να ανατροφοδοτεί τους γράφους πολικότητας με νέες κριτικές. Όμως θα πρέπει να σημειωθεί ότι οι γράφοι πολικότητας καταλαμβάνουν πολύ χώρο στη μνήμη του υπολογιστή. Όσο πιο πολλές κριτικές χρησιμοποιούνται για τη δημιουργία τους, τόσο περισσότερο χώρος απαιτείται για την αποθήκευσή τους. Αυτό το πρόβλημα ήταν πολύ πιο σημαντικό στη μέθοδο ανάλυσης συναισθήματος με n-grams, αν αναλογιστεί κανείς ότι στην περίπτωση εκείνη είχαμε πολύ περισσότερους κόμβους για μια λέξη, όμως και στην περίπτωση των word-grams έχει τη σημειολογία του. Επομένως, τυχόν ανατροφοδότηση στους γράφους πολικότητας θα επιβάρυνε ακόμη περισσότερο τη χρήση μνήμης για την εκτέλεση του αλγόριθμου.

Ας σημειωθεί επίσης ότι για να μπορούν να χρησιμοποιηθούν οι νέοι γράφοι θα έπρεπε να αποθηκεύονται συνεχώς σε νέα binary αρχεία εξ αρχής, το γεγονός δε αυτό θα ήταν χρονικά ασύμφορο.

Στην επόμενη ενότητα παρουσιάζεται μια τροποποίηση της περιγραφείσας μεθόδου, με την εισαγωγή μιας διαδικασίας ανατροφοδότησης.

[3] Βλ. μετάφραση από http://www.wordreference.com/engr/terrific.

3

Παρουσίαση μιας νέας υβριδικής μεθόδου

Προτείνεται η ενσωμάτωση στον προηγούμενο αλγόριθμο τεχνικών ανάλυσης κειμένου σε μέρη του λόγου (part of speech tagging, POS tagging) και η εισαγωγή στοιχείων προσέγγισης με λεξικό. Σκοπός είναι η ανάλυση κάθε κειμένου κριτικής ταινίας σε προτάσεις, και η αναζήτηση προτύπων που να οδηγούν στο συμπερασμό της πολικότητας της κριτικής.

3.1 Μέρη του λόγου

3.1.1 Θεωρητικό υπόβαθρο

Ως μέρη του λόγου (parts of speech, POS) ή κλάσεις λέξεων ή συντακτικές κατηγορίες ορίζονται οι κλάσεις στις οποίες ομαδοποιούνται οι λέξεις, από τις οποίες αντλούνται αρκετές πληροφορίες σχετικά με τη λέξη αλλά και τις γειτονικές της [17]. Για παράδειγμα, άρθρα και επίθετα συχνά προηγούνται των ουσιαστικών, και ουσιαστικά συχνά προηγούνται των ρημάτων, επομένως η πληροφορία ότι μια λέξη είναι ουσιαστικό ή ρήμα αποτελεί σημαντική ένδειξη για τις γειτονικές της λέξεις.

Στην αγγλική γλώσσα τα μέρη του λόγου διακρίνονται σε κλειστές και ανοικτές κλάσεις. Κλειστές κλάσεις είναι αυτές στις οποίες πολύ σπάνια (αν όχι ποτέ) προστίθενται καινούργιες λέξεις. Κλειστές κλάσεις αποτελούν οι προθέσεις (prepositions), οι «προσδιοριστές» (determiners), οι αντωνυμίες (pronouns), οι σύνδεσμοι (conjunctions), τα βοηθητικά ρήματα (auxiliary verbs), τα μόρια (particles) και οι αριθμοί (numerals).

Εξ αντιδιαστολής, ανοικτές κλάσεις είναι αυτές που μπορεί να εμπλουτίζονται με νέες λέξεις που είτε επινοούνται είτε δανείζονται από άλλες γλώσσες. Στις ανοικτές κλάσεις, ανεξαρτήτως γλώσσας, ανήκουν τα ουσιαστικά (nouns), τα επίθετα (adjectives), τα ρήματα (verbs) και τα επιρρήματα (adverbs).

3.1.2 Part of speech (POS) tagging

Η διαδικασία ανάθεσης σε κάθε λέξη ενός μέρους του λόγου ονομάζεται part of speech tagging (POS tagging). Σε φυσικό επίπεδο, η διαδικασία αυτή μπορεί να ταυτιστεί σε επίπεδο φυσικής γλώσσας με τη διαδικασία διαίρεσης σε σύμβολα. Σε γενικές γραμμές tags ανατίθενται και στα σημεία στίξης, και γι' αυτό το λόγο γίνεται προεπεξεργασία του κειμένου για το διαχωρισμό των λέξεων [17].

Υπάρχουν πολλές κατηγοριοποιήσεις (σύνολα) μερών του λόγου (tagsets), ωστόσο οι περισσότεροι σύγχρονοι αλγόριθμοι επεξεργασίας φυσικής γλώσσας χρησιμοποιούν το σύνολο του Penn Treebank Project, η οποία περιλαμβάνει 36 tags (χωρίς τα σημεία στίξης) και έχει ως εξής [36]:

Tag	**Description**	**Tag**	**Description**
CC	Coordinating conjunction	PRP$	Possessive pronoun
CD	Cardinal number	RB	Adverb
DT	Determiner	RBR	Adverb, comparative
EX	Existential *there*	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	*to*
JJR	Adjective, comparative	UH	Interjection
JJS	Adjective, superlative	VB	Verb, base form
LS	List item marker	VBD	Verb, past tense
MD	Modal	VBG	Verb, gerund or present participle
NN	Noun, singular or mass	VBN	Verb, past participle
NNS	Noun, plural	VBP	Verb, non-3rd person singular present
NNP	Proper noun, singular	VBZ	Verb, 3rd person singular present
NNPS	Proper noun, plural	WDT	Wh-determiner
PDT	Predeterminer	WP	Wh-pronoun
POS	Possessive ending	WP$	Possessive wh-pronoun
PRP	Personal pronoun	WRB	Wh-adverb

Πίνακας 3.1: Λίστα μερών του λόγου και των αντίστοιχων tags σύμφωνα με το Penn Treebank Project.

Το πρόβλημα σε αυτή τη διαδικασία ερείδεται στο γεγονός ότι οι λέξεις μπορεί να είναι διφορούμενες (ambiguous). Επομένως σε κάθε περίπτωση σκοπός είναι να βρεθεί το σωστό μέρος του λόγου που αντιστοιχεί σε μια λέξη.

Για παράδειγμα, η λέξη ‘content’ μπορεί να σημαίνει «περιεχόμενο», αν πρόκειται για ουσιαστικό, ή «χαρούμενος»[4], αν πρόκειται για επίθετο, και μάλιστα η λέξη προφέρεται διαφορετικά ανάλογα με το μέρος του λόγου που αποτελεί.

Για την εκπαίδευση και την αξιολόγηση μοντέλων ανάθεσης μερών του λόγου (POS taggers) χρησιμοποιούνται προκατασκευασμένα σώματα κειμένων (corpora) με επισημειωμένα μέρη του λόγου. Ορισμένα βασικά σώματα κειμένων που χρησιμοποιούνται για το σκοπό αυτό στην αγγλική γλώσσα είναι τα εξής:

1. Brown corpus [21]: Πρόκειται για ένα σύνολο περίπου ενός εκατομμυρίου λέξεων από 500 κείμενα (περίπου 2.000 λέξεις το καθένα) με διαφορετικές πηγές. Δημοσιεύθηκε για πρώτη φορά το 1961 στις ΗΠΑ και είχε ευρεία εφαρμογή στην υπολογιστική γλωσσολογία, ενώ για πολλά χρόνια αποτελούσε την πιο αναφερόμενη πηγή σε ερευνητικές εργασίες του τομέα.

2. Switchboard corpus: Πρόκειται για ένα σύνολο περίπου δύο εκατομμυρίων λέξεων που συγκεντρώθηκαν από τηλεφωνικές συνομιλίες (περίπου 2.400) κατά τα έτη 1990-1991. Δημοσιεύθηκε για πρώτη φορά το 1992-1993 [10]. Ο χαρακτηρισμός των μερών του λόγου σε κάθε λέξη έγινε με αυτόματο τρόπο και στη συνέχεια τα tags διορθώθηκαν μηχανικά από υπομνηματιστές.

3. North American News Text Corpus (AP corpus): Πρόκειται για συλλογή κειμένων που δημοσιεύθηκαν από το ειδησεογραφικό πρακτορείο Associated Press. Ουσιαστικά περιλαμβάνει δημοσιεύματα αρθρογράφων σε καθημερινές εφημερίδες όπως το New York Times ή το Washington Post. Η συλλογή υπάρχει σε διάφορες εκδόσεις. Στην έκδοση του 1987 το σύνολο περιέχει περίπου δεκαπέντε εκατομμύρια λέξεις, ενώ σε αυτή του 1988 περιέχει τριάντα έξι εκατομμύρια λέξεις [7].

4. WSJ corpus: Η συλλογή αυτή υπάρχει επίσης σε πολλές εκδόσεις. Στο αρχικό σύνολο [23] περιλαμβάνονταν 313,1 MB δεδομένων από συλλογή δημοσιεύσεων στο Wall Street Journal (WSJ) κατά τα έτη 1987-1989.

Οι λέξεις στο WSJ corpus είναι λιγότερο διφορούμενες ως προς τα μέρη του λόγου σε σχέση με το Brown corpus. Το γεγονός αυτό οφείλεται μάλλον στον προσανατολισμό των δημοσιευμάτων της εφημερίδας σε οικονομικά νέα που περιορίζει τη χρήση των λέξεων, σε σύγκριση με λέξεις από ένα σύνολο κειμένων με ευρύ φάσμα πηγών.

Όταν μια λέξη είναι διφορούμενη ως προς το μέρος του λόγου που αποτελεί, βασική ιδέα είναι να χρησιμοποιείται ως σημείο αναφοράς το μέρος του λόγου που έχει τη μεγαλύτερη συχνότητα εμφάνισης για τη συγκεκριμένη λέξη, κατά την εκπαίδευση του μοντέλου. Ένα τέτοιο

[4] Βλ. μετάφραση από http://www.wordreference.com/engr/content.

μοντέλο είχε εκπαιδευθεί σε σύνολο δεδομένων του WSJ corpus και παρουσίαζε ακρίβεια 92,34%.

Από την άλλη πλευρά υπάρχουν στατιστικά μοντέλα ανάθεσης μερών του λόγου, όπως τα Κρυφά Μαρκοβιανά Μοντέλα (Hidden Markov Models, HMMs), τα Μαρκοβιανά Μοντέλα Μέγιστης Εντροπίας (Maximum Entropy Markov Models, MEMMs), άλλα λογαριθμικά γραμμικά (log-linear) μοντέλα κ.ο.κ., τα οποία επιτυγχάνουν ακρίβεια της τάξης του 97%.

3.1.3 Εφαρμογή στην ανάλυση συναισθήματος

Έχει υποστηριχθεί ότι ορισμένα μέρη του λόγου εμπεριέχουν συναίσθημα. Σε [13] καθίσταται προσπάθεια πρόβλεψης της πολικότητας επιθέτων, ενώ σε [41] η μελέτη επεκτείνεται με τον επιπρόσθετο συνδυασμό ουσιαστικών και επιρρημάτων.

Επομένως, με βάση τα ανωτέρω, η γνώση του μέρους του λόγου που αποτελεί μια λέξη σε συνδυασμό με την πολικότητά της μπορεί να αποτελούν ένδειξη της πολικότητας ενός κειμένου.

3.2 Η αρχιτεκτονική της μεθόδου

Στις επόμενες υποενότητες περιγράφονται τα εργαλεία που χρησιμοποιούνται για την υλοποίηση της περιγραφόμενης μεθόδου.

3.2.1 Document preprocessor

Για το διαχωρισμό ενός κειμένου κριτικής σε προτάσεις χρησιμοποιείται η κλάση DocumentPreprocessor[5] που υλοποιείται από βιβλιοθήκη του Stanford για την επεξεργασία κειμένων. Δίδεται ως είσοδος το αρχείο κειμένου μιας κριτικής και επιστρέφεται λίστα με τις προτάσεις του κειμένου. Για τη λειτουργία της κλάσης δίδεται επίσης μια σειρά από σύμβολα και ακολουθίες συμβόλων, με τα οποία ορίζεται το τέλος μιας πρότασης (sentence delimiters).

Το προκαθορισμένο σύνολο συμβόλων με βάση τα οποία ορίζεται το τέλος μιας πρότασης (default sentence delimiters), σύμφωνα με την υλοποίηση της κλάσης είναι το εξής:

‘.’, ‘?’, ‘!’, ‘!!’, ‘!!!’, ‘??’, ‘?!’, ‘!?’

Μετά από μελέτη όμως των αρχείων κριτικής ταινιών, το ως άνω σύνολο κρίθηκε ανεπαρκές, καθώς τα κείμενα δεν αναλύονταν σε προτάσεις με ορθό τρόπο. Ένα πολύ αξιόλογο σύνολο συμβόλων που ορίζουν το τέλος μιας πρότασης, το οποίο και χρησιμοποιήθηκε για τους σκοπούς της παρούσας εργασίας είναι το εξής:

‘.’, ‘..’, ‘...’, ‘?’, ‘??’, ‘???’, ‘!’, ‘!!’, ‘!!!’, ‘
’, ‘

’

[5] https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/process/DocumentPreprocessor.html.

Επισημαίνεται καταρχάς ότι θα ήταν σκόπιμο να χρησιμοποιηθούν κανονικές εκφράσεις (regular expressions, regex) για την κομψότερη αναπαράσταση του συνόλου, όμως, με σημερινά δεδομένα, αυτό δεν επιτρέπεται (ακόμη) από την υλοποίηση της κλάσης.

Επίσης, θα ήταν δυνατό στα κείμενα κριτικής να εμφανίζονται ακολουθίες περισσότερων των τριών τελειών, ερωτηματικών ή θαυμαστικών, όμως κάτι τέτοιο διαπιστώθηκε ότι ήταν εξαιρετικά σπάνιο, επομένως παρέλκει η προσθήκη περισσότερων συμβόλων στο ως άνω σύνολο, καθώς θα ήταν άσκοπη και καταχρηστική.

Επίσης διαπιστώθηκε ότι τα κείμενα κριτικής ταινιών περιέχουν το HTML tag για το διαχωρισμό γραμμής
, προφανώς ως ένδειξη μιας νέας παραγράφου στο κείμενο, το οποίο μπορεί να εμφανίζεται μέχρι και δύο φορές συνεχόμενα στο κείμενο, γι' αυτό και προστέθηκε στο παραπάνω σύνολο συμβόλων.

Ας σημειωθεί τέλος ότι ο προεπεξεργαστής κειμένου αναγνωρίζει την άρνηση των ρημάτων στη συνοπτική μορφή "n't" και τη διαχωρίζει από το ρήμα σε ξεχωριστή λέξη.

3.2.2 POS Taggers

Σε [39, 40] περιγράφονται δύο λογαριθμικά γραμμικά (log-linear) μοντέλα Μέγιστης Εντροπίας για τον αυτόματο χαρακτηρισμό των λέξεων σε μέρη του λόγου (Maximum Entropy Tagger), χρησιμοποιώντας το tagset του Penn Treebank Project (Πίνακας 3.1).

Η υλοποίηση των μοντέλων αυτών σε Java γίνεται με την κλάση MaxentTagger[6] που παρέχεται από βιβλιοθήκη του Stanford. Ένα αντικείμενο αυτής της κλάσης αρχικοποιείται με ένα υπάρχον μοντέλο εκπαιδευμένο για το χαρακτηρισμό λέξεων ως προς τα μέρη του λόγου.

Για τους σκοπούς της παρούσας εργασίας χρησιμοποιείται το μοντέλο 'wsj-0-18-left3words.tagger' είχε ακρίβεια 96,97% κατά τον έλεγχο σε σύνολο δεδομένων από το WSJ corpus, και 88,85% ακρίβεια σε άγνωστο σύνολο.

Ας σημειωθεί ότι η λίστα μερών του λόγου που παρέχεται από το Penn Treebank Project είναι ιδιαίτερα εξαντλητική, και για το λόγο αυτό έγινε προσπάθεια απλοποίησης, με βάση τις εξής παρατηρήσεις:

1. Όλα τα επίθετα έχουν ως πρώτο γράμμα του tag το γράμμα 'J'.
2. Όλα τα ουσιαστικά έχουν ως πρώτο γράμμα του tag το γράμμα 'N'.
3. Όλα τα ρήματα έχουν ως πρώτο γράμμα του tag το γράμμα 'V'.
4. Όλα τα επιρρήματα έχουν ως πρώτο γράμμα του tag το γράμμα 'R'.

Τα ως άνω τέσσερα μέρη του λόγου είναι πιο πιθανό να εκφράζουν κάποιο συναίσθημα, επομένως θα μπορούσαμε με αρκετή ασφάλεια να αγνοήσουμε τα υπόλοιπα (λ.χ. άρθρα, προθέσεις

[6] https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/tagger/maxent/MaxentTagger.html.

κ.λπ.). Για λόγους καθαρά εμφάνισης, τα επίθετα αναγράφονται ως ‘ADJECTIVE’, τα ουσιαστικά ως ‘NOUN’, τα ρήματα ως ‘VERB’ και τα επιρρήματα ως ‘ADVERB’.

3.2.3 Λεξικό συναισθημάτων SentiWordNet 3.0

Το λεξικό SentiWordNet αναπτύχθηκε το 2010 [5] και χρησιμοποιείται για την υποστήριξη συστημάτων ταξινόμησης συναισθήματος και εξόρυξης δεδομένων. Χρησιμοποιεί ως βασικό δομικό στοιχείο τα σύνολα συνωνύμων (synsets) της λεξιλογικής βάσης δεδομένων WordNet [28].

Το WordNet είναι μία μεγάλη λεξιλογική βάση δεδομένων στα αγγλικά που δημιουργήθηκε στο Πανεπιστήμιο Princeton το 1985. Σκοπός του είναι ο συνδυασμός των χαρακτηριστικών ενός ερμηνευτικού λεξικού (dictionary) και ενός λεξικού συνωνύμων (thesaurus), για τη δημιουργία ενός «λεξικού νοημάτων» [20]. Για το λόγο αυτό, ουσιαστικά, ρήματα, επίθετα και επιρρήματα εντάσσονται σε ομάδες-σύνολα συνωνύμων (synsets) με το καθένα να εκφράζει μία διακριτή έννοια. Το WordNet δίνει σύντομους ορισμούς των λέξεων και παραδείγματα χρήσης τους και περιλαμβάνει σχέσεις μεταξύ των synsets όπως σχέσεις υπερωνυμίας (hyperonymy) ή υπωνυμίας (hyponymy) ή και αντωνυμίας μεταξύ των επιθέτων (antonymy).

Το SentiWordNet είναι το αποτέλεσμα της αυτόματης επισήμανσης των συνόλων συνωνύμων του WordNet σύμφωνα με τις έννοιες του θετικού, του αρνητικού και ουδέτερου, που περιγράφουν τους όρους που το απαρτίζουν [29]. Οι τρεις βαθμολογίες παράγονται από το συνδυασμό των αποτελεσμάτων ενός συνόλου οκτώ τριμερών ταξινομητών οι οποίοι χαρακτηρίζονται από παρόμοια επίπεδα ακρίβειας, αλλά παρουσιάζουν διαφορετική συμπεριφορά ταξινόμησης. Η χρήση των συνόλων συνωνύμων (synsets) και όχι των ίδιων των όρων ως βασικών μονάδων ανάπτυξης του λεξικού προσφέρει τη δυνατότητα διερεύνησης των διαφορετικών ερμηνειών του ίδιου όρου καθώς και των ιδιοτήτων που λαμβάνουν σε σχέση με την έκφραση γνώμης. Οι τρεις βαθμολογίες που υποδεικνύουν το συναίσθημα κάθε συνόλου συνωνύμων (synsets) κυμαίνονται από 0 έως 1, το δε σύνολό τους είναι πάντοτε ίσο με 1.

Ας σημειωθεί ότι το SentiWordNet περιλαμβάνει μόνο ουσιαστικά, επίθετα, ρήματα και επιρρήματα, και είναι ο δεύτερος λόγος για τον οποίο επιλέγουμε στην παρούσα εργασία να αγνοήσουμε μέρη του λόγου από κλειστές κλάσεις.

Τα ως άνω μέρη του λόγου χαρακτηρίζονται με τους εξής συμβολισμούς:

1. Τα επίθετα χαρακτηρίζονται με το γράμμα ‘a’.
2. Τα ουσιαστικά χαρακτηρίζονται με το γράμμα ‘n’.
3. Τα ρήματα χαρακτηρίζονται με το γράμμα ‘v’.
4. Τα επιρρήματα χαρακτηρίζονται με το γράμμα ‘r’.

Όλες οι γραμμές του λεξικού περιέχουν τα εξής στοιχεία:

POS	ID	PosScore	NegScore	SynsetTerms	Gloss
a	194924	0	0.75	shuddery#1 shivery#2 scary#1 scarey#1 chilling#1	provoking fear terror; "a scary movie"; "the most terrible and shuddery...tales of murder and revenge"
a	2359789	0.5	0.25	astonishing#1 amazing#1	surprising greatly; "she does an amazing amount of work"; "the dog was capable of astonishing tricks"
r	32180	0.25	0	incredibly#2 fantastically#1 fabulously#1	exceedingly; extremely; "she plays fabulously well"

Πίνακας 3.2: Παραδείγματα γραμμών του λεξικού SentiWordNet.

Για την χρησιμοποίηση του λεξικού χρησιμοποιείται η κλάση SentiWordNet, με συναρτήσεις για την εξαγωγή του συναισθήματος κάθε λέξης ανάλογα με το μέρος του λόγου που αποτελεί, ο κώδικας της οποίας παρατίθεται στο παράρτημα.

Για λόγους συντομίας χρησιμοποιούμε επίσης τους εξής συμβολισμούς:

1. Οι θετικές λέξεις (positive) θα χαρακτηρίζονται ‘POS’.
2. Οι αρνητικές (negative) λέξεις θα χαρακτηρίζονται ‘NEG’.
3. Οι ουδέτερες (neutral) λέξεις θα χαρακτηρίζονται ‘NEU’.

3.2.4 Ταξινομητής Naive Bayes

Ο ταξινομητής Naive Bayes δέχεται ως είσοδο τα χαρακτηριστικά ενός στιγμιότυπου δεδομένων και αναθέτει σε αυτά την κλάση που κρίνει ότι είναι πιο πιθανό να ανήκουν. Συγκεκριμένα, εάν $X = (x_1, x_2, \dots, x_n)$ είναι το διάνυσμα χαρακτηριστικών που δίδεται ως είσοδος στον ταξινομητή, τότε ο ταξινομητής αναθέτει σε κάθε κλάση πολικότητας $C_i \in C$ την πιθανότητα υπό συνθήκη, ή εκ των υστέρων πιθανότητα (a posteriori) $P(C_i|x_1, x_2, \dots, x_n)$, και αναθέτει στο στιγμιότυπο την κλάση που εκτιμά ως πιο πιθανή:

$$y = \underset{C_i \in C}{argmax} \; P(C_i|X)$$

Στον παραπάνω μαθηματικό τύπο μπορεί να εφαρμοσθεί ο γνωστός κανόνας του Bayes:

$$P(A|B) = \frac{P(A) \times P(B|A)}{P(B)}$$

Επομένως, με βάση τα ανωτέρω η πιθανότητα υπό συνθήκη κάθε κλάσης υπολογίζεται ως εξής:

$$y = \underset{C_i \in C}{argmax}\, P(C_i|X) = \underset{C_i \in C}{argmax}\, \frac{P(X|C_i) \times P(C_i)}{P(X)}$$

Ο παρονομαστής, για κάθε στιγμιότυπο δεδομένων, είναι σταθερός, άρα δεν έχει πρακτική σημασία για το χαρακτηρισμό της κλάσης. Στην ουσία η πιθανότητα υπό συνθήκη, και επομένως ο χαρακτηρισμός της κλάσης, εξαρτάται από τον αριθμητή.

$$y = \underset{C_i \in C}{argmax}\ P(C_i|X) = \underset{C_i \in C}{argmax}\ P(X|C_i) \times P(C_i)$$

Επομένως υπολογίζουμε την πιο πιθανή κλάση χρησιμοποιώντας δύο πιθανότητες, την εκ των προτέρων πιθανότητα (prior) κάθε κλάσης $P(C_i)$, και την πιθανοφάνεια (likelihood) των δεδομένων με δεδομένη την κλάση $P(X|C_i)$:

$$y = \underset{C_i \in C}{argmax}\ \overbrace{P(X|C_i)}^{likelihood} \times \overbrace{P(C_i)}^{prior}$$

$$y = \underset{C_i \in C}{argmax}\ \overbrace{P(x_1, x_2, \dots, x_n|C_i)}^{likelihood} \times \overbrace{P(C_i)}^{prior}$$

Ο ταξινομητής Naive Bayes κάνει την «αφελή» παραδοχή ότι όλα τα χαρακτηριστικά εισόδου είναι ανεξάρτητα μεταξύ τους:

$$P(x_1, x_2, \dots, x_n|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \dots \times P(x_n|C_i)$$

Επομένως τελικά η πιο πιθανή κλάση υπολογίζεται ως εξής:

$$y = \underset{C_i \in C}{argmax}\ P(C_i) \prod_{j=1}^{n} P(x_j|C_i)$$

Στον ανωτέρω μαθηματικό τύπο, η πιθανότητα $P(C_i)$ είναι γνωστή εκ των προτέρων και αποτελεί το ποσοστό των στιγμιοτύπων εκπαίδευσης που ανήκουν στην κλάση C_i. Επίσης, όσον αφορά στον υπολογισμό των πιθανοτήτων $P(x_j|C_i)$, κατά την εκπαίδευση του μοντέλου κάνουμε μια υπόθεση για την κατανομή των χαρακτηριστικών x_j με βάση τα δεδομένα του συνόλου εκπαίδευσης (μοντέλο γεγονότων, event model) και χρησιμοποιούμε αυτή κατά την αξιολόγηση του ταξινομητή.

Παρά το γεγονός ότι η παραδοχή που κάνει ο Naive Bayes είναι, θα έλεγε κανείς, μη ρεαλιστική, ωστόσο στην πράξη παρουσιάζει ιδιαίτερα μεγάλη επιτυχία, και μάλιστα έχει βρεθεί καλύτερος σε σχέση με άλλες πιο περίπλοκες τεχνικές [35]. Σε κάθε περίπτωση, ο ταξινομητής Naive Bayes είχε χρησιμοποιηθεί και για τη μελέτη του αλγόριθμου ανάλυσης συναισθήματος με γράφους λέξεων, και σε σύγκριση με άλλους ταξινομητές (δένδρο αποφάσεων J48, Multinomial Naive Bayes) βρέθηκε ότι ήταν ο πιο ακριβής.

Με βάση τα ανωτέρω, επιλέγουμε και για την παρούσα εργασία τον ταξινομητή Naive Bayes για την υλοποίηση της μεθόδου.

3.3 Η φιλοσοφία της μεθόδου

Καταρχάς για την εύρεση προτύπων εκτελούμε σε πρώτη φάση πρόγραμμα ανεξάρτητο από τον αλγόριθμο ανάλυσης συναισθήματος, που ουσιαστικά αποτελεί προπαρασκευαστικό στάδιο. Το πρόγραμμα διαχειρίζεται κριτικές ταινιών ως εξής:

1. Αναλύει το κείμενο κάθε κριτικής σε προτάσεις με τη χρήση του Document Preprocessor.

2. Αναλύει κάθε πρόταση σε μέρη του λόγου με τη χρήση του POS tagger. Ας πάρουμε για παράδειγμα την εξής πρόταση:

'The film looks great and the animation is sometimes jaw-dropping'.

Η πρόταση αναλύεται σε μέρη του λόγου ως εξής:

'The_DT film_NN looks_VBZ great_JJ and_CC the_DT animation_NN is_VBZ sometimes_RB jaw-dropping_JJ'.

3. Για κάθε λέξη εξάγεται το συναίσθημα που εκφράζει (θετικό, αρνητικό ή ουδέτερο), με τη χρήση του λεξικού συναισθημάτων SentiWordNet.

Με βάση τα ανωτέρω, για κάθε λέξη της πρότασης προκύπτει ένας συνδυασμός του συναισθήματος που εκφράζει, και του μέρους του λόγου που αποτελεί (λ.χ. POS_VERB, NEG_ADJECTIVE κ.λπ.). Αν η λέξη που εξετάζεται είναι άρνηση 'not' ή 'n't', τότε χαρακτηρίζεται ως 'NOT'. Επικουρικά προσθέτουμε και έναν αύξοντα αριθμό σε κάθε «λέξη» που σχηματίζεται.

Όπως αναφέρθηκε και ανωτέρω, αγνοούμε όλα τα μέρη του λόγου που ανήκουν σε κλειστές κλάσεις. Ήδη μόνο με τέσσερα μέρη του λόγου (ουσιαστικά, επίθετα, ρήματα, επιρρήματα) μια λέξη μπορεί να χαρακτηρίζεται ως:

1. POS_ NOUN
2. NEG_NOUN
3. NEU_NOUN
4. POS_ADJECTIVE
5. NEG_ADJECTIVE
6. NEU_ADJECTIVE
7. POS_VERB
8. NEG_VERB
9. NEU_VERB
10. POS_ADVERB
11. NEG_ADVERB
12. NEU_ADVERB
13. NOT

Επομένως ήδη ο αριθμός των συνδυασμών είναι εξαντλητικός και αυξάνεται εκθετικά σε σχέση με το μήκος της πρότασης σε λέξεις (θεωρητικά 13 για πρόταση με μια λέξη, 169 για πρόταση με δύο λέξεις κ.λπ.).

Σύμφωνα με τα παραπάνω σχηματίζεται για ολόκληρη την πρόταση μια ακολουθία (tag sentence). Για την πρόταση του παραδείγματος θα αγνοηθούν οι λέξεις “the” και “and” (που πράγματι δεν εκφράζουν κάποιο συναίσθημα) και η ακολουθία που σχηματίζεται είναι η εξής:

NEU_NOUN1 NEU_VERB2 POS_ADJECTIVE3 POS_NOUN4 NEU_VERB5 NEU_ADVERB6 NEU_ADJECTIVE7.

Τις ακολουθίες αυτές (κανόνες) τις αποθηκεύουμε σε έναν πίνακα κατακερματισμού (πίνακας κανόνων) και εξετάζουμε την παρουσία αυτών σε κείμενα κριτικής ταινιών, ως χαρακτηριστικό της πολικότητας των κειμένων. Ο πίνακας κανόνων έχει ως κλειδί (key) τον κανόνα που δημιουργείται σε μορφή κειμένου (String), και αντιστοιχείται σε ένα σύνολο δύο τιμών, δύο ακέραιων αριθμών που αντιστοιχούν στο πλήθος εμφανίσεων του κανόνα σε θετική ή σε αρνητική κριτική. Στον πίνακα υπάρχει ως παράμετρος και μια λογική μεταβλητή με την οποία καθορίζεται εάν κατά την εκτέλεση ο κανόνας χρησιμοποιείται ή είναι ανενεργός.

Για την απόφαση αν ένας κανόνας θα χρησιμοποιείται ή θα είναι ανενεργός, το πρόγραμμα λαμβάνει υπ’ όψιν δύο παραμέτρους:

1. ένα κατώφλι που αντιστοιχεί στον ελάχιστο αριθμό εμφανίσεων του κανόνα (threshold number of appearances). Αν μια ακολουθία εμφανίζεται πολύ λίγες φορές, είναι πιθανό η ύπαρξη αυτής να είναι τυχαία, και επομένως να μην έχει την πρακτική σημασία ενός «κανόνα».

2. ένα κατώφλι ποσόστωσης των εμφανίσεων του κανόνα σε θετικές ή αρνητικές κριτικές (threshold percentage). Αν μια ακολουθία εμφανίζεται με την ίδια συχνότητα σε θετικές και αρνητικές κριτικές, είναι πιθανό να μην έχει πρακτική σημασία, εφόσον δεν αντιστοιχείται (σχεδόν) μονοσήμαντα σε κάποια πολικότητα.

4

Υλοποίηση της νέας υβριδικής μεθόδου

4.1 Παράμετροι για την εκτέλεση του προπαρασκευαστικού προγράμματος

Το προπαρασκευαστικό πρόγραμμα χρησιμοποιεί τις εξής παραμέτρους:

1. train rules reviews: ο αριθμός των κριτικών που θα χρησιμοποιηθεί για την αναζήτηση κανόνων.

2. positive rate: το ποσοστό των χρησιμοποιούμενων κριτικών που θα ανήκουν στη θετική κλάση (positive).

3. minPosRating: η μικρότερη τιμή που μπορεί να έχει κριτική ώστε να είναι θετική.

4. maxPosRating: η μεγαλύτερη τιμή που μπορεί να έχει κριτική ώστε να είναι θετική.

5. minNegRating: η μικρότερη τιμή που μπορεί να έχει κριτική ώστε να είναι αρνητική.

6. maxNegRating: η μεγαλύτερη τιμή που μπορεί να έχει κριτική ώστε να είναι αρνητική.

Για την απόφαση αν ένας κανόνας θα χρησιμοποιείται ή όχι, το πρόγραμμα λαμβάνει υπ' όψιν επίσης τις εξής δύο παραμέτρους:

7. thresholdNoOfAppearances: το κατώφλι ελάχιστου απαιτούμενου αριθμού εμφανίσεων του κανόνα.

8. thresholdPercentage: το κατώφλι ποσόστωσης των εμφανίσεων του κανόνα σε θετικές ή αρνητικές κριτικές.

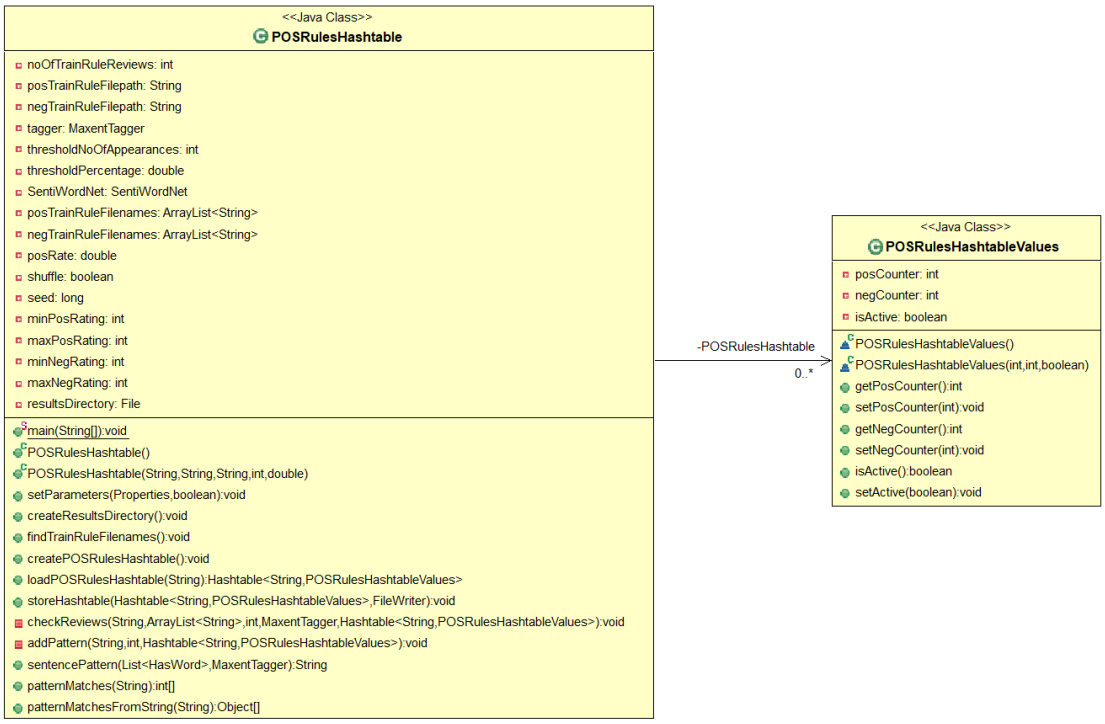
4.2 Περιγραφή του κώδικα

Για την υλοποίηση δημιουργήθηκε η κλάση POSRulesHashtable. Το πρόγραμμα αρχικοποιείται με τις παραμέτρους που αναγράφονται στην προηγούμενη ενότητα και στη συνέχεια δέχεται ως είσοδο κριτικές ταινιών, θετικές και αρνητικές.

Η συνάρτηση checkReviews δέχεται τα κείμενα των κριτικών ένα-ένα, και υλοποιεί την ανάλυση της κάθε κριτικής σε προτάσεις.

Η συνάρτηση sentencePattern φορτώνει τον tagger και το λεξικό SentiWordNet, αποφασίζει για κάθε λέξη το μέρος του λόγου που αποτελεί, καθώς και για το συναίσθημα που εκφράζει, αντιστοιχεί δηλαδή σε κάθε λέξη έναν από τους 13 δυνατούς συνδυασμούς ή την αγνοεί (αν το μέρος του λόγου που αποτελεί είναι κλειστή κλάση). Στη συνέχεια για ολόκληρη την πρόταση επιστρέφει τον κανόνα που σχηματίζεται, σε μορφή String.

Η συνάρτηση addPattern. προσθέτει τον κανόνα στον πίνακα, αυξάνοντας τον κατάλληλο μετρητή, υπό την προϋπόθεση ότι ο κανόνας που επέστρεψε η sentencePattern δεν είναι κενός.



Σχήμα 4.1: Διάγραμμα UML των κλάσεων POSRulesHashtable και POSRulesHashtableValues.

Για την υλοποίηση του πίνακα κανόνων, και συγκεκριμένα για το σύνολο τιμών στο οποίο αντιστοιχίζεται κάθε κανόνας, δημιουργήθηκε η κλάση POSRulesHashtableValues. Η κλάση ενθυλακώνει τρεις τιμές, το πλήθος των εμφανίσεων του κανόνα σε θετικές κριτικές (posCounter), ή αντίστοιχα σε αρνητικές κριτικές (negCounter), και μια λογική μεταβλητή με την οποία προσδιορίζεται αν ο κανόνας θα χρησιμοποιείται ή όχι (isActive). Κάθε φορά που ένας κανόνας προστίθεται για πρώτη φορά στον πίνακα κανόνων είναι ανενεργός. Σε κάθε περίπτωση αυξάνεται ο κατάλληλος μετρητής, αναλόγως αν η κριτική είναι θετική ή αρνητική.

Στο τέλος του προγράμματος, διατρέχεται ο πίνακας κανόνων και οι κανόνες που ικανοποιούν τα δύο κριτήρια κατά τα ανωτέρω (ποσόστωσης και ελάχιστου αριθμού εμφανίσεων) θεωρούνται ως ενεργοί και αποθηκεύονται σε ένα αρχείο .txt με συγκεκριμένη

μορφή. Σε κάθε γραμμή του αρχείου υπάρχει ένας κανόνας και οι δύο τιμές των μετρητών, όλα διαχωρισμένα με χαρακτήρα TAB (‘\t’), ακολουθώντας τη φιλοσοφία του λεξικού SentiWordNet. Οι υπόλοιποι κανόνες που δεν πληρούν τα κριτήρια διαγράφονται.

4.3 Εισαγωγή νέων χαρακτηριστικών στον αλγόριθμο ανάλυσης συναισθήματος

Ο αλγόριθμος ανάλυσης συναισθήματος μετέτρεπε το κείμενο σε γράφο, τον οποίο συνέκρινε με τους δύο γράφους πολικότητας για την εξαγωγή χαρακτηριστικών ομοιότητας. Τώρα, το κείμενο κριτικής αναλύεται επιπλέον σε προτάσεις και εφαρμόζεται σε αυτό η ίδια διαδικασία για την εξαγωγή κανόνων.

Στο διάνυσμα έξι στοιχείων (συν της πολικότητας της κριτικής) προστίθενται δύο ακόμη χαρακτηριστικά. Πρόκειται για δύο ακέραιες τιμές, έναν μετρητή «θετικών» κανόνων και έναν μετρητή «αρνητικών» κανόνων, που εξετάζουν την παρουσία κανόνων στο κείμενο κριτικής.

Η συνάρτηση patternMatches εφαρμόζει ανάλυση στο κείμενο μιας κριτικής (δεχόμενη ως είσοδο το όνομα του αρχείου) και επιστρέφει τους δύο ως άνω μετρητές. Ένας κανόνας που έχει εμφανιστεί τις περισσότερες φορές σε θετική κριτική θα αυξάνει τον μετρητή «θετικών» κανόνων, ενώ αντίστοιχα ένας κανόνας που έχει εμφανιστεί τις περισσότερες φορές σε αρνητική κριτική θα αυξάνει τον μετρητή «αρνητικών» κανόνων.

Οι τιμές των μετρητών δίδονται μαζί με τα χαρακτηριστικά ομοιότητας για την εκπαίδευση του μοντέλου μηχανικής μάθησης. Έτσι, δημιουργείται το αρχείο ARFF για την εκπαίδευση του μοντέλου μηχανικής μάθησης (βλ. ιδίως ενότητες 2.3.2, 2.3.4), το οποίο περιέχει πλέον σε κάθε γραμμή το νέο σύνολο 8 χαρακτηριστικών (συν της πολικότητας του κειμένου).

4.4 Ανατροφοδότηση

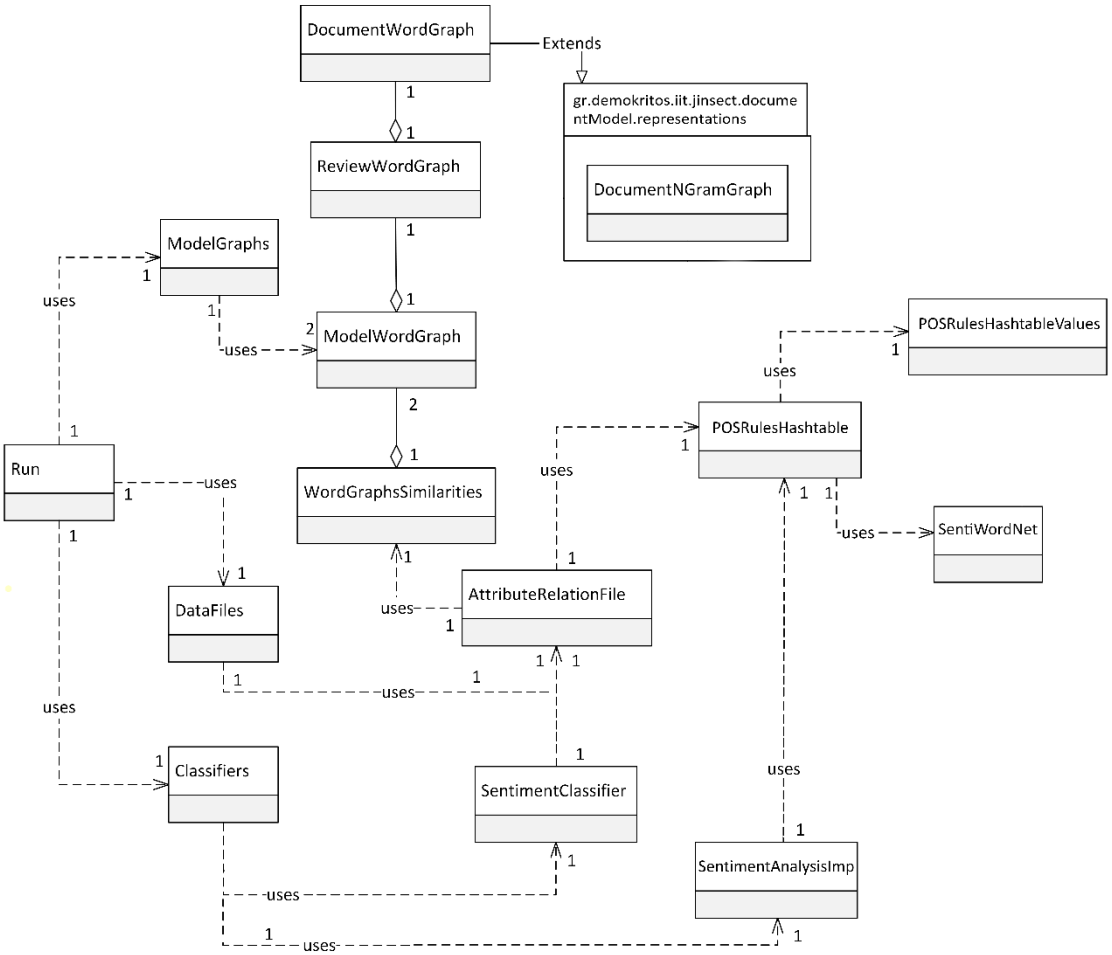
Όπως τονίστηκε και προηγουμένως, σκοπός της παρούσας εργασίας είναι και η εισαγωγή ανατροφοδότησης στον αλγόριθμο ανάλυσης συναισθήματος. Αυτό μπορεί να συμβεί μόνο κατά το στάδιο ελέγχου της επίδοσης του ταξινομητή και, ως εκ τούτου, δεν είναι δυνατό η διαδικασία να παραμείνει αυτοματοποιημένη, όπως έχει περιγραφεί.

Συγκεκριμένα, ο αλγόριθμος ανάλυσης συναισθήματος στο δεύτερο στάδιο έκανε τις συγκρίσεις των κειμένων κριτικής για την εκπαίδευση και την αξιολόγηση του ταξινομητή και δημιουργούσε τα αρχεία ARFF.

Για τους σκοπούς της παρούσας εργασίας, η χρησιμοποίηση του συνόλου δεδομένων για την αξιολόγηση του ταξινομητή μεταφέρεται στο τρίτο στάδιο και γίνεται λιγότερο αυτοματοποιημένη. Δηλαδή πλέον, ο αλγόριθμος στο δεύτερο στάδιο επιλέγει μόνο τα αρχεία

των κειμένων κριτικής που θα χρησιμοποιηθούν για την εκπαίδευση του ταξινομητή και εξάγει τα χαρακτηριστικά τους σε ένα αρχείο ARFF.

Στο τρίτο στάδιο ο αλγόριθμος επιλέγει τα αρχεία κριτικής για την αξιολόγηση του ταξινομητή. Για καθένα από αυτά χρησιμοποιεί την κλάση SentimentAnalysisImp (και όχι την κλάση ClassifierEvaluation) για την εξαγωγή των 8 χαρακτηριστικών, και κάνει μια εκτίμηση της πολικότητας του κειμένου. Η συνάρτηση patternMatchesFromString είναι υπεύθυνη αυτή τη φορά για την ανάλυση του κειμένου σε προτάσεις, και στο τέλος επιστρέφει τους δύο μετρητές κανόνων, καθώς και μια λίστα με τους κανόνες που βρέθηκαν. Οι κανόνες που βρέθηκαν στο κείμενο της κριτικής θα προστεθούν στον πίνακα κανόνων, αυξάνοντας τον μετρητή που αντιστοιχεί στην πολικότητα που εκτιμήθηκε προηγουμένως.



Σχήμα 4.2: UML διάγραμμα υλοποίησης της νέας υβριδικής μεθόδου.

Για τη λειτουργία της ανατροφοδότησης χρησιμοποιούμε πάλι τις δύο παραμέτρους κατωφλίου που περιγράφηκαν νωρίτερα, το κατώφλι ελάχιστου απαιτούμενου αριθμού εμφανίσεων του κανόνα και το κατώφλι ποσόστωσης των εμφανίσεων του κανόνα σε θετικές ή αρνητικές κριτικές. Προστίθενται δηλαδή στον αλγόριθμο ανάλυσης συναισθήματος δύο ακόμη παράμετροι (σε σχέση με εκείνες που περιγράφηκαν στην ενότητα 2.3.5).

Διατρέχοντας τον πίνακα κανόνων κατά διαστήματα, μπορούμε να ελέγχουμε ποιοι κανόνες εξακολουθούν να πληρούν το κριτήριο ποσόστωσης εμφανίσεων, σε περίπτωση δε που δεν το πληρούν θα πρέπει να διαγράφονται. Ακόμη, μπορούμε να εξετάζουμε αν υπάρχουν νέοι κανόνες που πληρούν τα απαραίτητα κατά τα ανωτέρω κριτήρια για να χρησιμοποιούνται ως ενεργοί. Με την ανατροφοδότηση μπορούμε δηλαδή να προσθέτουμε στον πίνακα κανόνες, ή ακόμη και να αφαιρούμε από αυτόν.

Στη συνέχεια ο πίνακας κανόνων διαγράφει τους υπόλοιπους ανενεργούς κανόνες και συνεχίζει κανονικά τη διαδικασία.

5

Αξιολόγηση της μεθόδου

5.1 Δεδομένα

Τα χρησιμοποιούμενα δεδομένα κριτικής ταινιών προέρχονται από μια βάση δεδομένων του IMDB[7]. Τα διαθέσιμα δεδομένα είναι ένα σύνολο 50.000 κριτικών, διαχωρισμένων σε δύο ίσα μέρη, ένα για τη διαδικασία εκπαίδευσης και ένα για τη διαδικασία αξιολόγησης της μεθόδου. Οι κριτικές ταινιών που βρίσκονται σε κάθε κατηγορία είναι επίσης διαχωρισμένες σε θετικές και αρνητικές. Επομένως κάθε σύνολο περιλαμβάνει 12.500 θετικές και 12.500 αρνητικές κριτικές.

Κάθε κριτική βρίσκεται σε ένα αρχείο .txt. Το όνομα κάθε αρχείου περιέχει έναν μοναδικό αριθμό (id) και τη βαθμολογία της κριτικής (rating), έναν ακέραιο αριθμό από 0 έως 10. Δηλαδή κάθε κριτική είναι αποθηκευμένη σε αρχείο της μορφής id_rating.txt. Τα αρχεία που περιλαμβάνουν θετικές κριτικές έχουν βαθμολογία από 7 έως 10. Αντίστοιχα, τα αρχεία που περιλαμβάνουν αρνητικές κριτικές έχουν βαθμολογία από 0 έως 4.

5.2 Τιμές παραμέτρων

Για τις μετρήσεις που περιγράφονται σε αυτό το κεφάλαιο σταθεροποιούμε τις γνωστές παραμέτρους στις εξής τιμές (για τη σημασία των παραμέτρων βλ. ενότητα 2.3.5), οι οποίες δεν θα επαναλαμβάνονται άσκοπα στη συνέχεια:

- graph reviews: 2.000.
- window: 3.
- remove: true.

[7] http://ai.stanford.edu/~amaas/data/sentiment/.

- preprocess: false.
- positive rate: 50.
- classifier: weka.classifiers.bayes.NaiveBayes.
- minPosRating: 7.
- maxPosRating: 10.
- minNegRating: 0.
- maxNegRating: 4.
- shuffle: true.

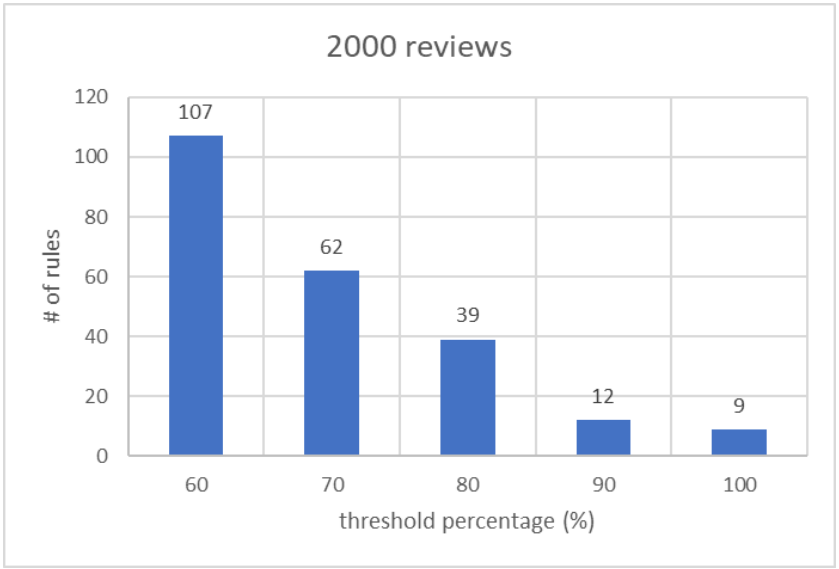
5.3 *Πίνακας κανόνων*

5.3.1 *Κατώφλι ποσόστωσης*

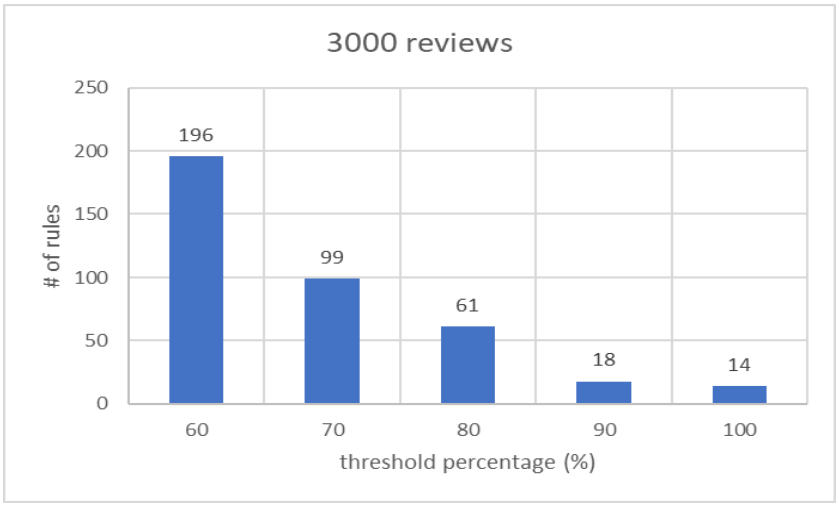
Σε αυτή την υποενότητα παρουσιάζονται μετρήσεις σχετικά με το κατώφλι ποσόστωσης, ένα από τα κριτήρια εισαγωγής κανόνα στον πίνακα κανόνων. Για τις περιγραφόμενες μετρήσεις διατηρούμε σταθερή την παράμετρο που αντιστοιχεί στο δεύτερο κριτήριο, τον ελάχιστο απαιτούμενο αριθμό εμφανίσεων ενός κανόνα:

- thresholdNoOfAppearances=5.

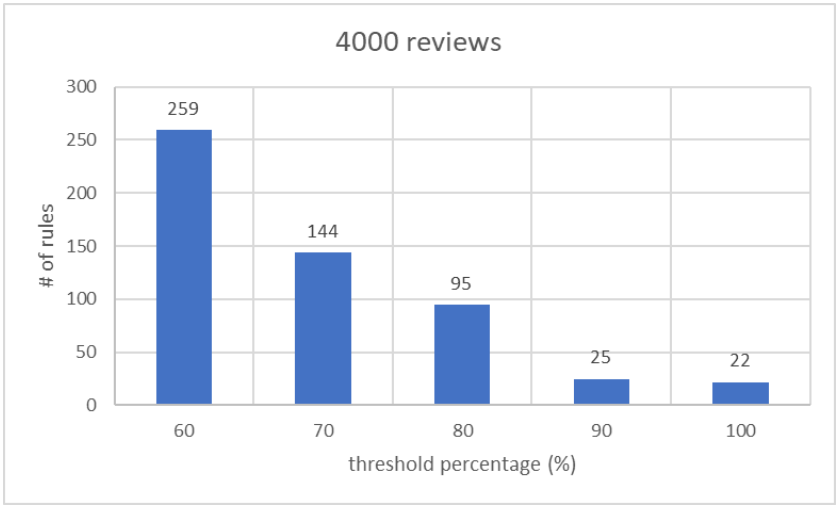
Για τη μελέτη του πλήθους των σχηματιζόμενων κανόνων, σε σχέση με το κατώφλι ποσόστωσης, μεταβάλαμε την παράμετρο POS rules train reviews, από 2000 έως 5000, με βήμα 1000. Τα αποτελέσματα φαίνονται στα ακόλουθα γραφήματα (σχήματα Σχήμα 5.1 έως Σχήμα 5.4):



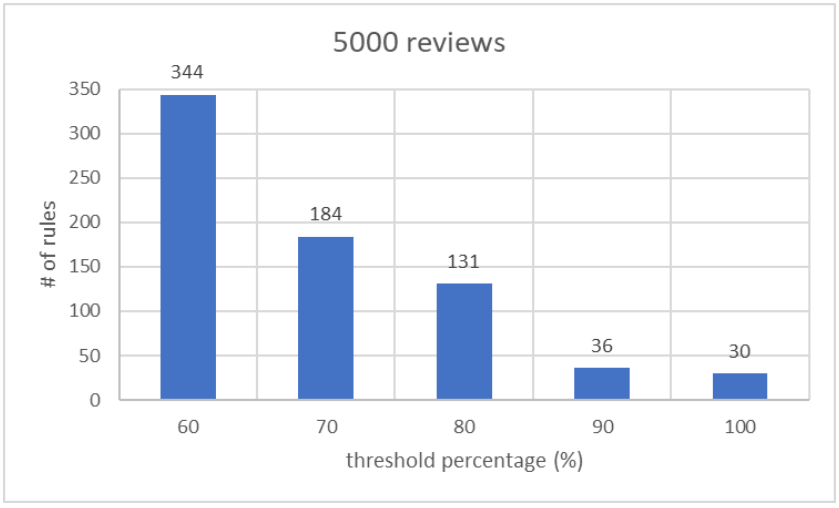
Σχήμα 5.1: Μεταβολή του πλήθους των σχηματιζόμενων κανόνων σε σχέση με το κατώφλι ποσόστωσης, για είσοδο 2000 reviews.



Σχήμα 5.2: Μεταβολή του πλήθους των σχηματιζόμενων κανόνων σε σχέση με το κατώφλι ποσόστωσης, για είσοδο 3000 reviews.



Σχήμα 5.3: Μεταβολή του πλήθους των σχηματιζόμενων κανόνων σε σχέση με το κατώφλι ποσόστωσης, για είσοδο 4000 reviews.



Σχήμα 5.4: Μεταβολή του πλήθους των σχηματιζόμενων κανόνων σε σχέση με το κατώφλι ποσόστωσης, για είσοδο 5000 reviews.

Όπως είναι αναμενόμενο, όσο περισσότερες κριτικές ταινιών δίδονται στον αλγόριθμο, τόσο περισσότεροι κανόνες σχηματίζονται.

Σε κάθε περίπτωση το πλήθος των κανόνων που εμφανίζονται για κατώφλι ποσόστωσης 90% ή 100% είναι εξαιρετικά μικρό. Αντίθετα, για μικρότερες τιμές της παραμέτρου, το πλήθος των κανόνων που σχηματίζονται αυξάνεται σημαντικά.

Με βάση τις μετρήσεις αυτές, πιθανολογείται ότι για πολύ μικρό κατώφλι ποσόστωσης οι κανόνες που σχηματίζονται δεν είναι ακριβείς, επομένως η χρησιμοποίησή τους για την εκπαίδευση του μοντέλου μηχανικής μάθησης δεν ενδείκνυται.

Θα πρέπει επομένως να εξετάσουμε τον αριθμό σφαλμάτων κατά την εκπαίδευση του ταξινομητή, όπως προκύπτουν από τη μελέτη των αρχείων εκπαίδευσης ARFF. Στα αρχεία αυτά, κάθε κριτική που χρησιμοποιείται για την εκπαίδευση αποτελεί μια γραμμή με εννέα χαρακτηριστικά (βλ. ενότητα 4.3).

Ελέγχουμε σε κάθε γραμμή τα τρία τελευταία χαρακτηριστικά, που είναι:

1. ο μετρητής του πλήθους εμφανίσεων ενός θετικού κανόνα.

2. ο μετρητής του πλήθους εμφανίσεων ενός αρνητικού κανόνα.

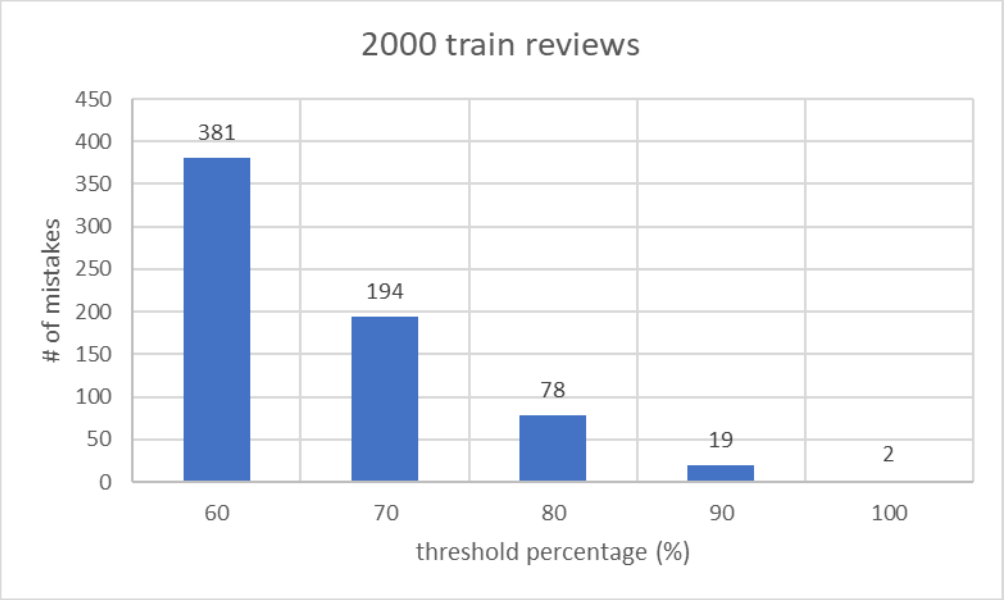
3. η πολικότητα (γνωστή εκ των προτέρων) μιας κριτική.

Θεωρούμε ότι υπάρχει σφάλμα σε κάθε περίπτωση που ένας ή περισσότεροι θετικοί κανόνες εμφανίζονται σε κριτική αρνητικής πολικότητας, ή το αντίστροφο.

Θα ήταν δυνατό, βεβαίως, να υπάρχουν κριτικές στις οποίες εμφανίζονται τόσο θετικοί όσο και αρνητικοί κανόνες, ενδεχομένως μάλιστα με διαφορετικό πλήθος, δηλαδή κάποιος μετρητής να έχει μεγαλύτερη τιμή από τον άλλο. Σε θεωρητικό επίπεδο, εάν λ.χ. σε μια κριτική θετικής πολικότητας εμφανίζονταν τόσο θετικοί όσο και αρνητικοί κανόνες, αλλά περισσότεροι θετικοί από ό,τι αρνητικοί, ή το αντίστροφο, δεν θα επρόκειτο περί σφάλματος. Στην περίπτωση αυτή όμως μας ενδιαφέρει η ακρίβεια της αντιστοίχισης κανόνων σε κάποια πολικότητα, και, επομένως, τέτοιες περιπτώσεις για το πείραμα που διενεργείται σε αυτή την υποενότητα θα θεωρούνται ως σφάλματα. Ας σημειωθεί επίσης ότι η ύπαρξη σφάλματος δεν οδηγεί κατ' ανάγκη σε λανθασμένη εκτίμηση της πολικότητας της κριτικής από τον αλγόριθμο (για την ακρίβεια του αλγορίθμου βλ. ιδίως ενότητα 5.4), διότι τα στοιχεία αυτά λειτουργούν επικουρικά στον αλγόριθμο ανάλυσης συναισθήματος με γράφους λέξεων.

Για το πείραμα αυτό χρησιμοποιήσαμε τις εξής παραμέτρους:

- train reviews=2000.
- Ως πίνακα κανόνων, το αρχείο που σχηματίστηκε νωρίτερα ανάλογα με το κατώφλι ποσόστωσης, με 3000 POS rules train reviews.



Σχήμα 5.5: Μεταβολή του πλήθους των σφαλμάτων που παρουσιάζονται κατά την εκπαίδευση του ταξινομητή, σε σχέση με το κατώφλι ποσόστωσης, για είσοδο 2000 train reviews.

Από τους ελέγχους που διενεργήσαμε σε αυτή την υποενότητα, αλλά και όπως φαίνεται από τα γραφήματα, ο αριθμός των σφαλμάτων κατά την εκπαίδευση του ταξινομητή εξαρτάται άμεσα από το κατώφλι ποσόστωσης.

Συγκεκριμένα φαίνεται ότι, πράγματι, για σχετικά μικρό κατώφλι ποσόστωσης παρουσιάζονται πολλά σφάλματα, δηλαδή κανόνες εμφανίζονται με μεγάλη συχνότητα σε κριτικές ανεξαρτήτως πολικότητας. Γίνεται δεκτό, επομένως, ότι κανόνες που πληρούν πολύ χαμηλό κατώφλι ποσόστωσης δεν έχουν πρακτική σημασία για τον υβριδικό αλγόριθμο, μάλιστα θα μπορούσαν να μειώσουν την απόδοσή του εκπαιδεύοντας λανθασμένα το μοντέλο μηχανικής μάθησης.

Κρίσιμο είναι το γεγονός ότι για κατώφλι ποσόστωσης >90% ο αριθμός των σφαλμάτων μειώνεται σημαντικά. Για τις επόμενες ενότητες, ωστόσο, σταθεροποιούμε την τιμή της παραμέτρου στο 100%, για την εξάλειψη, κατά το δυνατό, των σφαλμάτων.

5.3.2 Κατώφλι ελάχιστου αριθμού εμφανίσεων

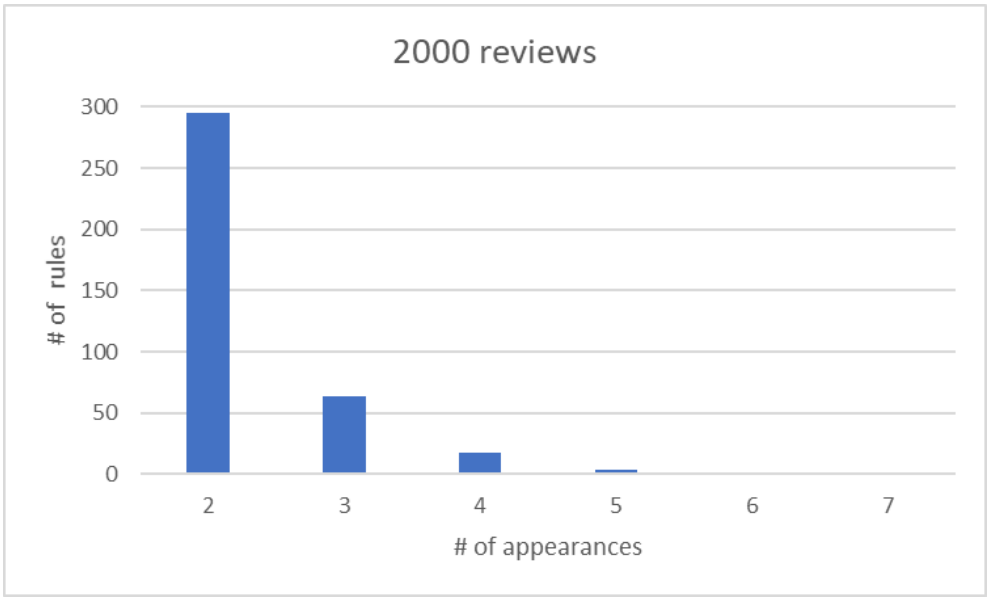
Σε αυτή την υποενότητα παρουσιάζονται μετρήσεις σχετικά με το δεύτερο κριτήριο εισαγωγής κανόνα στον πίνακα κανόνων, το κατώφλι ελάχιστου αριθμού εμφανίσεων ενός κανόνα. Για τις περιγραφόμενες μετρήσεις διατηρούμε σταθερή την παράμετρο που αντιστοιχεί στο προηγούμενο κριτήριο, το κατώφλι ποσόστωσης:

- thresholdPercentage=100.

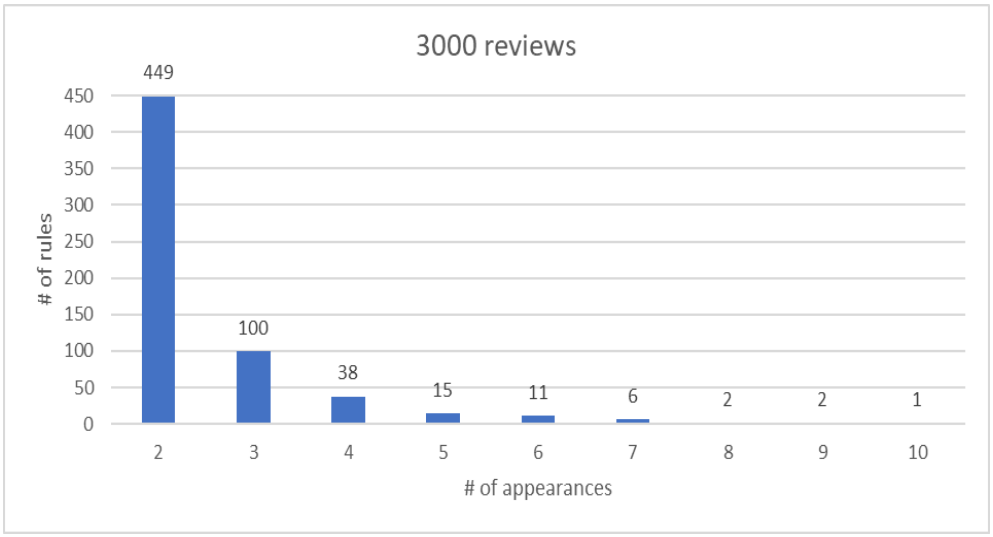
Για τη μελέτη του πλήθους των σχηματιζόμενων κανόνων, σε σχέση με το κατώφλι ποσόστωσης, μεταβάλαμε την παράμετρο POS rules train reviews, από 2000 έως 5000, με βήμα

1000. Αγνοούμε τις ακολουθίες που εμφανίζονται μόνο μια φορά, καθώς είναι άνευ ουσίας και σε καμία περίπτωση δεν μπορεί να θεωρηθεί ότι στοιχειοθετούν «κανόνα».

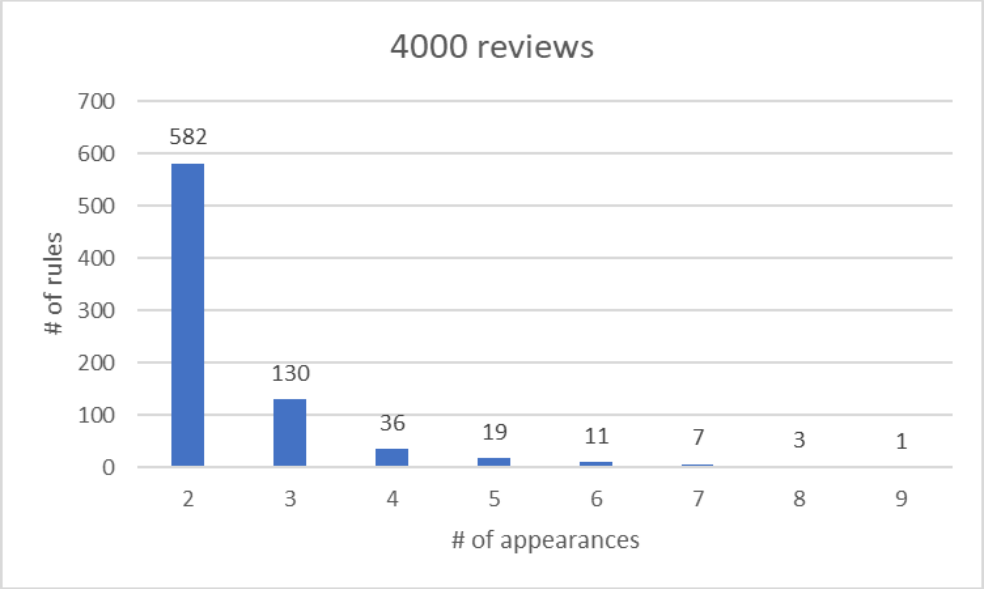
Τα αποτελέσματα φαίνονται στα ακόλουθα γραφήματα (σχήματα Σχήμα 5.6 έως Σχήμα 5.9):



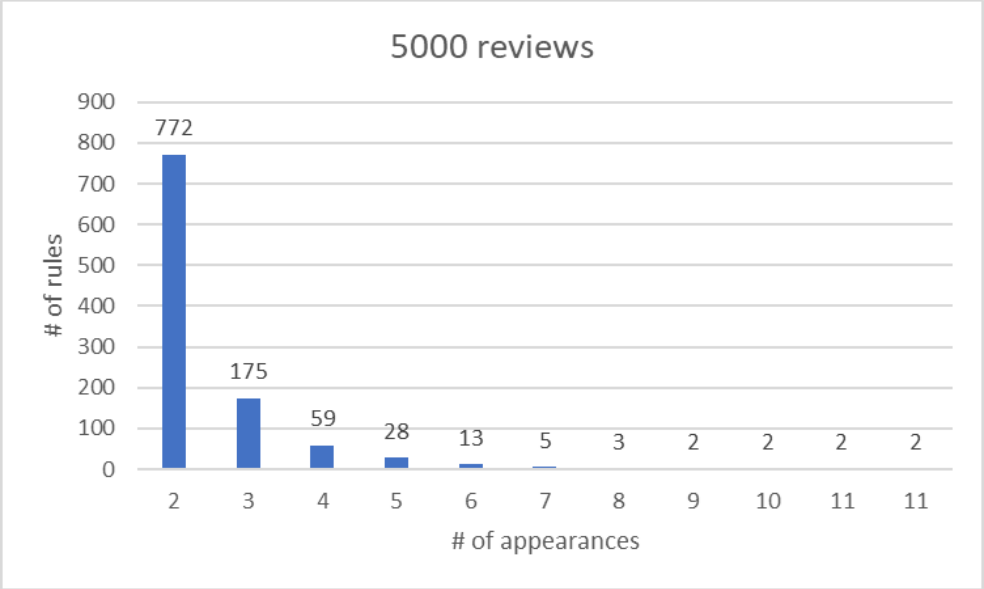
Σχήμα 5.6: Μεταβολή του πλήθους των σχηματιζόμενων κανόνων σε σχέση με τον ελάχιστο απαιτούμενο αριθμό εμφανίσεων ενός κανόνα, για είσοδο 2000 reviews.



Σχήμα 5.7: Μεταβολή του πλήθους των σχηματιζόμενων κανόνων σε σχέση με τον ελάχιστο απαιτούμενο αριθμό εμφανίσεων ενός κανόνα, για είσοδο 3000 reviews.



Σχήμα 5.8: Μεταβολή του πλήθους των σχηματιζόμενων κανόνων σε σχέση με τον ελάχιστο απαιτούμενο αριθμό εμφανίσεων ενός κανόνα, για είσοδο 4000 reviews.



Σχήμα 5.9: Μεταβολή του πλήθους των σχηματιζόμενων κανόνων σε σχέση με τον ελάχιστο απαιτούμενο αριθμό εμφανίσεων ενός κανόνα, για είσοδο 5000 reviews.

Όπως είναι αναμενόμενο, όσο περισσότερες κριτικές ταινιών δίδονται στον αλγόριθμο, τόσο περισσότεροι κανόνες σχηματίζονται.

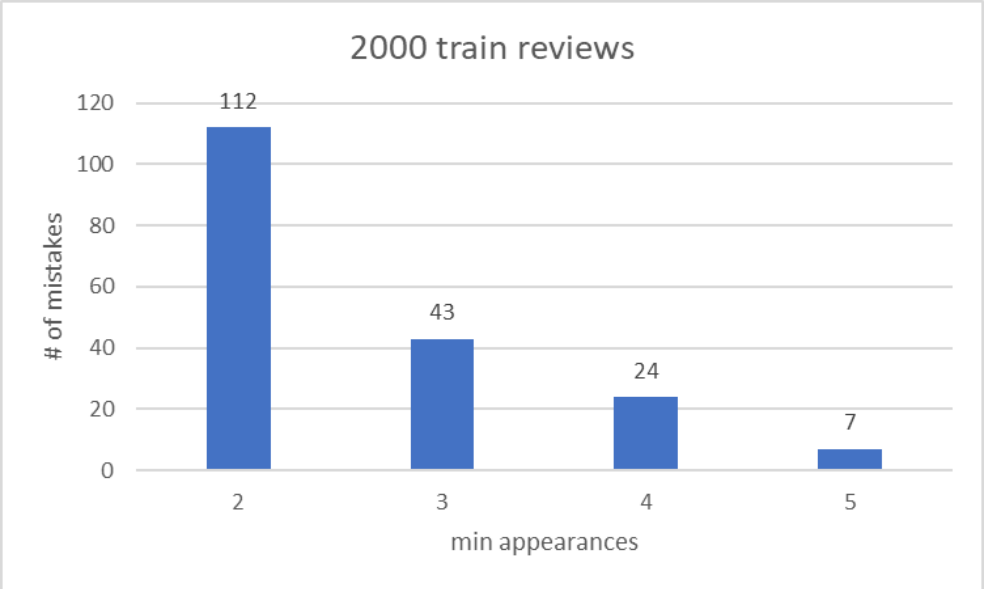
Βλέπουμε ότι το πλήθος των κανόνων που εμφανίζονται για κατώφλι ελάχιστου αριθμού εμφανίσεων ίσο με 5 μειώνεται σε μεγάλο βαθμό, και συγκεκριμένα είναι μικρότερο από το 5% του συνολικού αριθμού κανόνων. Από την άλλη πλευρά, για μικρότερες τιμές της παραμέτρου, το πλήθος των κανόνων που σχηματίζονται είναι υπερβολικά μεγάλος.

Πιθανολογούμε επομένως, με βάση τα ανωτέρω, ότι για πολύ μικρό κατώφλι ελάχιστου αριθμού εμφανίσεων οι κανόνες που σχηματίζονται δεν είναι ακριβείς, επομένως η χρησιμοποίησή τους για την εκπαίδευση του ταξινομητή δεν ενδείκνυται.

Θα πρέπει επομένως και για αυτό το κριτήριο να εξετάσουμε τον αριθμό σφαλμάτων κατά την εκπαίδευση του ταξινομητή, όπως προκύπτουν από τη μελέτη των αρχείων εκπαίδευσης ARFF, κατά τον ίδιο τρόπο που περιγράφηκε στην ενότητα 5.3.1.

Για το πείραμα αυτό χρησιμοποιήσαμε τις εξής παραμέτρους:

- train reviews=2000.
- Ως πίνακα κανόνων, το αρχείο που σχηματίστηκε νωρίτερα ανάλογα με το κατώφλι ελάχιστου αριθμού εμφανίσεων, με 5000 POS rules train reviews.



Σχήμα 5.10: Μεταβολή του πλήθους των σφαλμάτων που παρουσιάζονται κατά την εκπαίδευση του ταξινομητή, σε σχέση με το κατώφλι ελάχιστου αριθμού εμφανίσεων, για είσοδο 2000 train reviews.

Παρατηρούμε και σε αυτή την περίπτωση, όπως φαίνεται από τα γραφήματα, ότι ο αριθμός των σφαλμάτων κατά την εκπαίδευση του ταξινομητή εξαρτάται άμεσα από το κατώφλι ελάχιστου αριθμού εμφανίσεων ενός κανόνα.

Συγκεκριμένα φαίνεται ότι για σχετικά μικρό κατώφλι ελάχιστου αριθμού εμφανίσεων παρουσιάζονται πολλά σφάλματα, δηλαδή κανόνες εμφανίζονται με μεγάλη συχνότητα σε κριτικές ανεξαρτήτως πολικότητας. Γίνεται δεκτό, επομένως, ότι κανόνες που πληρούν πολύ χαμηλό κατώφλι ελάχιστου αριθμού εμφανίσεων μπορεί να έχουν εμφανισθεί τυχαία και δεν έχουν πρακτική σημασία για τον υβριδικό αλγόριθμο, μάλιστα θα μπορούσαν να μειώσουν την απόδοσή του εκπαιδεύοντας λανθασμένα το μοντέλο μηχανικής μάθησης.

Παρατηρείται ότι για κατώφλι αριθμού εμφανίσεων >5 ο αριθμός των σφαλμάτων μειώνεται σημαντικά. Η τιμή αυτή φαίνεται να εξασφαλίζει αφενός ότι δεν πρόκειται για μια πρόταση που επαναλαμβάνεται αυτούσια, και επομένως παράγει τον ίδιο κανόνα, και αφετέρου

ότι δεν πρόκειται για μια τυχαία εμφάνιση του κανόνα. Επομένως στη συνέχεια σταθεροποιούμε την τιμή της παραμέτρου σε 5, για την εξάλειψη, κατά το δυνατό, των σφαλμάτων.

5.3.3 Δημιουργία του πίνακα κανόνων

Στην ενότητα αυτή περιγράφεται η δημιουργία ενός πίνακα κανόνων με κανόνες που έχουν μεγάλη ακρίβεια. Για το σκοπό αυτό χρησιμοποιούμε τις εξής παραμέτρους:

- POS rules train reviews=25000
- thresholdNoOfAppearances=5
- thresholdPercentage=100

Καταρχάς ορίσαμε ένα πολύ μεγάλο σύνολο δεδομένων, το μέγιστο δυνατό, για την εξαγωγή όσο το δυνατόν καλύτερων αποτελεσμάτων.

Θέσαμε το κατώφλι ποσόστωσης εμφανίσεων στο 100%, ορίσαμε δηλαδή την απαίτηση ότι ένας κανόνας εμφανίζεται αυστηρά σε κριτικές θετικής πολικότητας ή σε κριτικές αρνητικής πολικότητας.

Τέλος, το κατώφλι ελάχιστου αριθμού εμφανίσεων ενός κανόνα ορίστηκε στην τιμή 5, μια τιμή που μειώνει σε μεγάλο βαθμό τα σφάλματα κατά την εκπαίδευση του ταξινομητή.

Από τους 253.070 διαφορετικούς κανόνες που καταγράφηκαν συνολικά, μόνο 127 πληρούσαν τα παραπάνω δύο αυστηρά κριτήρια (ο πίνακας με τους κανόνες αυτούς παρατίθεται στο Παράρτημα Β).

5.3.4 Λειτουργικότητα της ανατροφοδότησης

Στην υποενότητα αυτή εξετάζεται η λειτουργικότητα της ανατροφοδότησης, με τα δύο κριτήρια που έχουν ορισθεί στην παρούσα εργασία (ενότητα 4.4).

Για το σκοπό αυτό, και έχοντας πλέον σταθεροποιήσει τις τιμές των κριτηρίων, εξετάζουμε με διαδοχικές εκτελέσεις τα αποτελέσματα του πίνακα κανόνων. Κάθε φορά δίδεται στον αλγόριθμο ως είσοδος ο πίνακας κανόνων που κατασκευάστηκε με την προηγούμενη εκτέλεση. Την πρώτη φορά που εκτελούμε τον αλγόριθμο δίδεται ως είσοδος ο πίνακας κανόνων με τους 127 κανόνες, που παράχθηκε στην ενότητα 5.3.3.

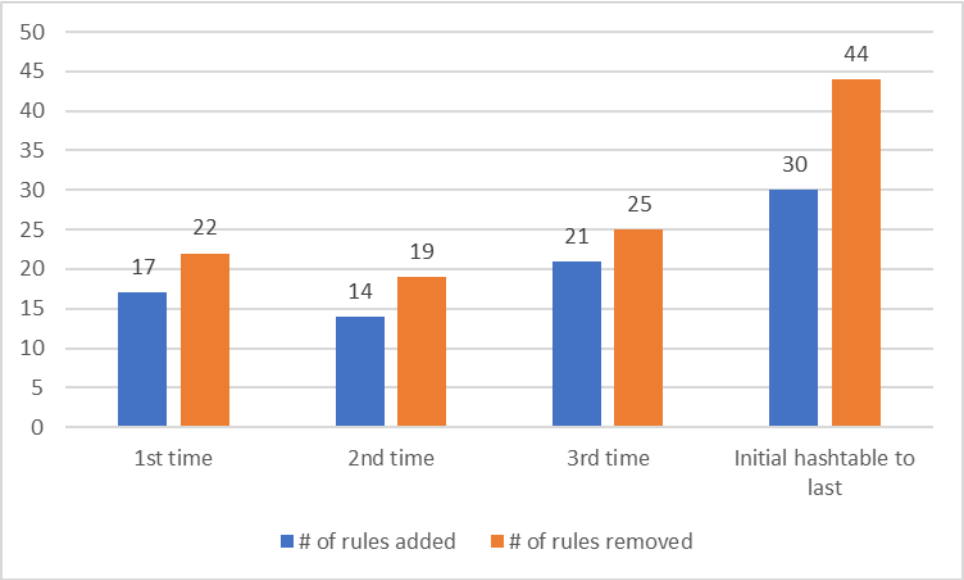
Λαμβάνουμε υπ’ όψιν επίσης ότι θα πρέπει να δίδεται αρκετά μεγάλος αριθμός test reviews στον αλγόριθμο, ώστε να υπάρχει η δυνατότητα να προστίθενται νέοι κανόνες, ή και να αφαιρούνται, από τον πίνακα κανόνων.

Συγκεκριμένα για την εκτέλεση των μετρήσεων σε αυτή την ενότητα χρησιμοποιούμε τις εξής παραμέτρους:

- thresholdPercentage=100.
- thresholdNoOfAppearances=5.

- test reviews=5000.

Εξετάζουμε τον αριθμό των κανόνων που προστίθενται στον πίνακα κανόνων ή αφαιρούνται από αυτόν κάθε φορά (στήλες 1-3). Επίσης εξετάζουμε τον αριθμό των κανόνων που έχουν προστεθεί/αφαιρεθεί στο τέλος, σε σύγκριση με τον αρχικό πίνακα κανόνων (στήλη 4). Τα αποτελέσματα φαίνονται στο ακόλουθο γράφημα:



Σχήμα 5.11: Πλήθος κανόνων που προστίθενται στον πίνακα κανόνων ή αφαιρούνται από αυτόν, σε διαδοχικές εκτελέσεις του αλγορίθμου, για είσοδο 5000 test reviews.

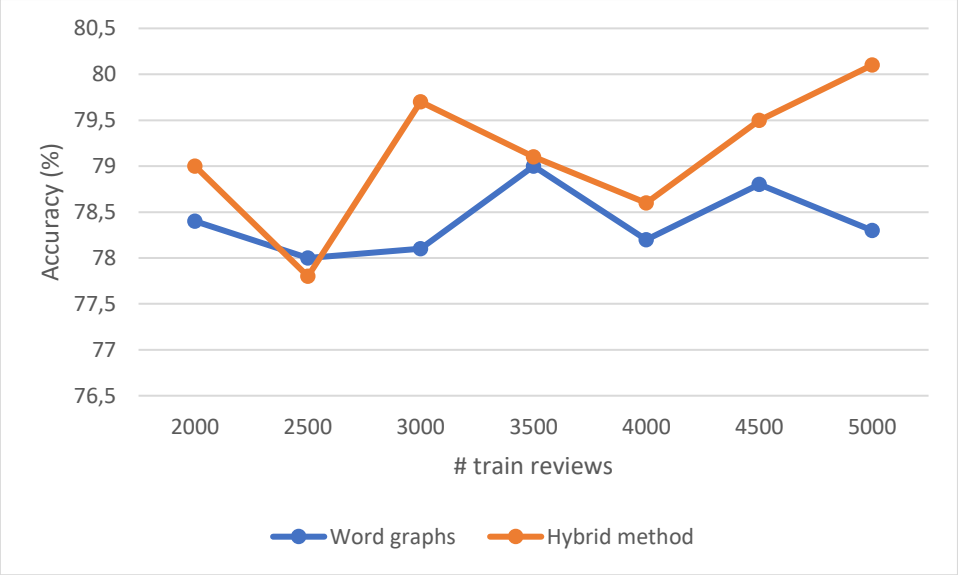
Από το ως γράφημα φαίνεται ότι υπάρχουν πολλοί κανόνες οι οποίοι προστίθενται και αφαιρούνται από τον πίνακα κανόνων, μερικοί από αυτούς μάλιστα κατ' επανάληψη. Η ανατροφοδότηση φαίνεται ότι λειτουργεί σε σταθερό σύνολο δεδομένων από κριτικές ταινιών.

5.4 Ακρίβεια των μεθόδων

Σε αυτή την ενότητα γίνεται σύγκριση της ακρίβειας του ταξινομητή, με χρήση της μεθόδου γράφων λέξεων και της νέας υβριδικής μεθόδου. Για το συγκεκριμένο πείραμα εκτελέσαμε διάφορες μετρήσεις, διατηρώντας σταθερό τον αριθμό των test reviews. Συγκεκριμένα χρησιμοποιήσαμε τις παρακάτω παραμέτρους:

- test reviews=2000.
- thresholdPercentage=100.
- thresholdNoOfAppearances=5.

Τα αποτελέσματα των μετρήσεων φαίνονται στο ακόλουθο διάγραμμα:



Σχήμα 5.12: Μεταβολή της ακρίβειας των δύο συγκρινόμενων μεθόδων, σε σχέση με τον αριθμό των train reviews.

Παρατηρούμε ότι, με εξαίρεση οριακές περιπτώσεις, η νέα υβριδική μέθοδος φαίνεται να είναι συγκριτικά καλύτερη από τη μέθοδο γράφων λέξεων.

Σε γενικές γραμμές είναι λογικό ότι όσο περισσότερα κείμενα δέχεται ο αλγόριθμος για την εκπαίδευσή του, τόσο μεγαλύτερη ακρίβεια ταξινόμησης θα επιτυγχάνει, πλησιάζοντας τα όρια μέγιστης απόδοσής του.

Στην προκείμενη περίπτωση είναι φανερό ότι καθώς αυξάνεται το πλήθος των train reviews αυξάνεται σημαντικά και η διαφορά στην απόδοση των δύο μεθόδων, με τη νέα υβριδική μέθοδο να έχει το προβάδισμα.

5.5 *Σύγκριση του χρόνου εκτέλεσης των δύο μεθόδων*

Εκτελούμε τον κώδικα της παλιάς μεθόδου ανάλυσης συναισθήματος με γράφους λέξεων, καθώς και της προτεινόμενης μεθόδου με χρήση ανάδρασης, εξετάζοντας το χρόνο εκτέλεσης, με σταθερό αριθμό train reviews:

- train reviews=2000.

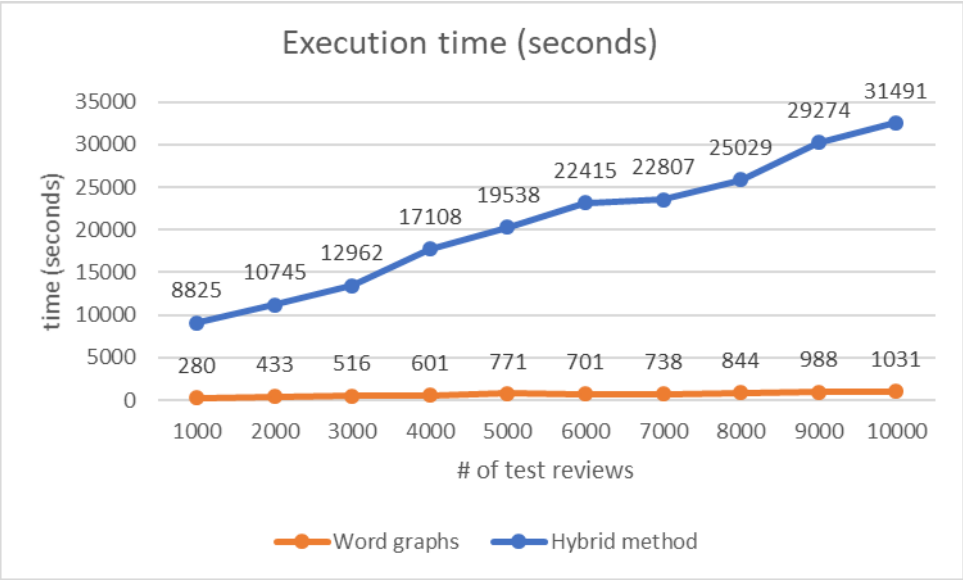
Για τον κώδικα της προτεινόμενης μεθόδου, χρησιμοποιούμε επιπρόσθετα τις εξής παραμέτρους:

- thresholdNoOfAppearances=5
- thresholdPercentage=100

Δίνουμε επίσης ως είσοδο τον πίνακα κανόνων με τους 127 κανόνες που δημιουργήθηκε στην ενότητα 5.3.3.

Για το πείραμα μεταβάλαμε την παράμετρο test reviews από 1000 έως 10000, με βήμα 1000. Στα αποτελέσματα δεν συμπεριλαμβάνεται ο χρόνος που απαιτείται για την

κατασκευή των γράφων πολικότητας, καθώς είναι κατά μέσο όρο ίδιος και για τις δύο μεθόδους.



Σχήμα 5.13: Μεταβολή του χρόνου εκτέλεσης των δύο μεθόδων, καθώς αυξάνεται ο αριθμός των test reviews.

Βλέπουμε ότι ο χρόνος εκτέλεσης της υβριδικής μεθόδου, και συγκεκριμένα μόνο για την εκπαίδευση του ταξινομητή και την αξιολόγησή του, είναι πολύ μεγαλύτερος σε σχέση με τη μέθοδο γράφων λέξεων. Μάλιστα η μέθοδος γράφων έχει χρόνο εκτέλεσης που αυξάνεται με πολύ μικρό ρυθμό σε σχέση με τον αριθμό των test reviews.

5.6 *Συμπεράσματα*

Στην παρούσα εργασία έγινε πρωτογενής προσπάθεια προσέγγισης της ανάλυσης συναισθήματος. Από το συνδυασμό δύο ανεξάρτητων ομάδων χαρακτηριστικών για το συμπερασμό της πολικότητας κειμένων κριτικής ταινιών, χαρακτηριστικά από τους γράφους λέξεων των κειμένων κριτικής και από την ανάλυση των προτάσεων των κειμένων σε μέρη του λόγου και την εξαγωγή συναισθήματος των λέξεων.

Από τις μετρήσεις που διεξήχθησαν στις προηγούμενες ενότητες του Κεφαλαίου 4 προέκυψαν δύο βασικά συμπεράσματα.

Καταρχάς η νέα υβριδική μέθοδος είναι καλύτερη σε ακρίβεια από τη μέθοδο γράφων λέξεων. Μάλιστα όσο περισσότερο εκπαιδεύεται ο ταξινομητής, τόσο περισσότερο αυξάνει η διαφορά στην απόδοση της υβριδικής μεθόδου σε σύγκριση με τη μέθοδο γράφων λέξεων.

Θα πρέπει να σημειωθεί ότι οι δύο μετρητές, ως χαρακτηριστικά της κριτικής που δίδονται στον ταξινομητή, λειτουργούν επικουρικά προς την ανάλυση με γράφους λέξεων. Προφανώς δεν είναι αναγκαία η παρουσία κανόνων σε κάποιο κείμενο κριτικής ώστε ο

ταξινομητής να αποφανθεί για την κλάση πολικότητας. Ωστόσο, στο βαθμό που υπάρχουν, βοηθούν στην αύξηση της ακρίβειάς του.

Ένα δεύτερο συμπέρασμα είναι ότι ο χρόνος εκτέλεσης της υβριδικής μεθόδου είναι τάξεις μεγέθους μεγαλύτερος από αυτόν της μεθόδου γράφων λέξεων. Το εύρημα αυτό ήταν αναμενόμενο. Οι δύο μέθοδοι δημιουργούν σε κάθε κριτική το γράφο λέξεων που θα συγκριθεί με τους γράφους πολικότητας για την εξαγωγή χαρακτηριστικών. Η υβριδική μέθοδος, επιπρόσθετα, θα εφαρμόσει ανάλυση του κειμένου σε προτάσεις και θα κάνει ανάλυση σε κάθε πρόταση του κειμένου για την εξαγωγή περαιτέρω χαρακτηριστικών. Η ανάλυση αυτή προφανώς είναι πολύ πιο χρονοβόρα από μια σύγκριση μεταξύ γράφων.

Ένα σημαντικό στοιχείο που επίσης επηρεάζει το χρόνο εκτέλεσης είναι η ανατροφοδότηση. Στη μέθοδο γράφων έχουμε ως στάδιο προεπεξεργασίας τη δημιουργία των γράφων πολικότητας, όμως στη συνέχεια ο αλγόριθμος ασχολείται μόνο με την ανάλυση συναισθήματος σε κριτικές ταινιών, την εξαγωγή χαρακτηριστικών για την εκπαίδευση και αξιολόγηση του ταξινομητή. Στην περίπτωση της υβριδικής μεθόδου ο αλγόριθμος ασχολείται επιπλέον με τη συνεχή ανανέωση του πίνακα κανόνων, που αποτελεί πρόσθετη χρονική επιβάρυνση της μεθόδου.

Όλα αυτά γίνονται όμως με σκοπό τη βελτίωση της ακρίβειας του ταξινομητή. Επομένως για την εφαρμογή της υβριδικής μεθόδου θα πρέπει σε μια ανάλυση κόστους-οφέλους (cost-benefit analysis) το όφελος από τη βελτίωση της ακρίβειας του ταξινομητή να υπερτερεί έναντι του κόστους σε χρόνο.

5.7 Μελλοντικές εργασίες

Σε επόμενες μελέτες θα είχε μεγάλο ενδιαφέρον η χρησιμοποίηση του πίνακα κανόνων σε διαφορετικά σύνολα δεδομένων. Για παράδειγμα, θα μπορούσε να διερευνηθεί κατά πόσο οι κανόνες που δημιουργήθηκαν με το σύνολο δεδομένων της παρούσας εργασίας μπορούν να εφαρμοστούν σε σύνολα δεδομένων από το Twitter όπου, τουλάχιστον μέχρι σήμερα [16], τα κείμενα είναι μικρά, το πολύ 140 χαρακτήρες. Διαφορετικά, θα μπορούσε να μελετηθεί η εξαγωγή νέων κανόνων που να βρίσκουν εφαρμογή σε αυτά τα σύνολα δεδομένων, και η εξέταση της ακρίβειας του ταξινομητή σε αυτές τις περιπτώσεις.

Επίσης θα είχε ενδιαφέρον η μελέτη της λειτουργικότητας της διαδικασίας ανατροφοδότησης που προστέθηκε στην υβριδική μέθοδο σε βάθος χρόνου. Θα μπορούσε να εξετασθεί κατά πόσο υπάρχει ανταπόκριση στις μεταβολές της γλώσσας και κατά πόσο διατηρείται η απόδοση του ταξινομητή. Θα μπορούσε να μελετηθεί επίσης η τροποποίηση που επέρχεται σε βάθος χρόνου στον πίνακα κανόνων. Τέλος, μπορεί να μελετηθεί η χρησιμοποίηση διαφορετικών κριτηρίων εισαγωγής και αφαίρεσης κανόνων, και η επίδρασή αυτών στην απόδοση του αλγορίθμου.

Παράρτημα Α - Κώδικας

Σε αυτό το παράρτημα παρατίθεται ο κώδικας των νέων κλάσεων, καθώς και των κλάσεων στις οποίες προστέθηκαν νέα στοιχεία λειτουργίας, σε σχέση με την προηγούμενη υλοποίηση [45].

Α.1. Η κλάση POSRulesHashtable

```
package sentimentanalysis;

import java.util.List;
import java.util.Properties;
import java.util.Hashtable;
import java.util.Iterator;

import edu.stanford.nlp.ling.HasWord;
import edu.stanford.nlp.ling.TaggedWord;
import edu.stanford.nlp.process.DocumentPreprocessor;
import edu.stanford.nlp.tagger.maxent.MaxentTagger;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.Reader;
import java.io.StringReader;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;

public class POSRulesHashtable{

    private int noOfTrainRuleReviews;

    private String posTrainRuleFilepath;
    private String negTrainRuleFilepath;

    private MaxentTagger tagger;
    private Hashtable <String, POSRulesHashtableValues>
                POSRulesHashtable;
    private int thresholdNoOfAppearances;
    private double thresholdPercentage;
```

```
private SentiWordNet SentiWordNet;

private ArrayList<String> posTrainRuleFilenames;
private ArrayList<String> negTrainRuleFilenames;

private double posRate;
private boolean shuffle;
private long seed;

private int minPosRating;
private int maxPosRating;
private int minNegRating;
private int maxNegRating;

private File resultsDirectory;

public static void main(String[] args)
        throws IOException, ClassNotFoundException {

    if (args.length != 1) return;

    InputStream reader = new FileInputStream(args[0]);
    Properties properties = new Properties();
    if (args[0].contains(".xml"))
        properties.loadFromXML(reader);
    else
        properties.load(reader);
    long start = System.currentTimeMillis();

    System.out.println("Creating files for setting
        POS rules hashtable.\n");

    POSRulesHashtable object = new POSRulesHashtable();

    object.setParameters(properties, true);
    object.createResultsDirectory();
    object.findTrainRuleFilenames();
    object.createPOSRulesHashtable();
    long end = System.currentTimeMillis();
    NumberFormat formatter = new DecimalFormat("#0.00000");
    System.out.print("\nExecution time is " +
       formatter.format(end - start) / 1000d) + " seconds");
}

public POSRulesHashtable(){ }

public POSRulesHashtable(String taggerFilepath, String
       POSRulesHashtableFilepath,String SentiWordNetFilepath,
       int thresholdNoOfAppearances, double thresholdPercentage)
                throws ClassNotFoundException, IOException{

    setTagger(new MaxentTagger(taggerFilepath));
```

```
    setThresholdNoOfAppearances(thresholdNoOfAppearances);
    setThresholdPercentage(thresholdPercentage);
    setSentiWordNet(new SentiWordNet
            (SentiWordNetFilepath));
    setPOSRulesHashtable(loadPOSRulesHashtable
            (POSRulesHashtableFilepath));
}

public void setParameters(Properties properties, boolean
        onlyThisStage) throws ClassNotFoundException, IOException{

    System.out.println("Setting parameters...");

    setNoOfTrainRuleReviews(Integer.parseInt(properties.
            getProperty("noOfTrainRuleReviews")));
    setMinPosRating(Integer.parseInt(properties.
            getProperty("minPositiveRating")));
    setMaxPosRating(Integer.parseInt(properties.
            getProperty("maxPositiveRating")));
    setMinNegRating(Integer.parseInt(properties.
            getProperty("minNegativeRating")));
    setMaxNegRating(Integer.parseInt(properties.
            getProperty("maxNegativeRating")));
    setPosRate(Double.parseDouble(properties.
            getProperty("positiveRate"))/100);

    if (properties.getProperty("seed") != null)
        setSeed(Long.parseLong(properties.getProperty("seed")));
    else setSeed(-1);
    setShuffle(Boolean.parseBoolean(properties.
            getProperty("shuffle")));

    setPosTrainRuleFilepath(properties.
            getProperty("positiveTrainRuleFilepath"));
    setNegTrainRuleFilepath(properties.
            getProperty("negativeTrainRuleFilepath"));
    setTagger(new MaxentTagger(properties.
            getProperty("taggerFilepath")));
    setSentiWordNet(new SentiWordNet(properties.
            getProperty("SentiWordNetFilepath")));
    setThresholdNoOfAppearances(Integer.parseInt(properties.
            getProperty("thresholdNoOfAppearances")));
    setThresholdPercentage(Integer.parseInt(properties.
            getProperty("thresholdPercentage")));
}

public void createResultsDirectory() {
    System.out.println("Creating directory for results...");

    Date date = new Date();
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(date);
    int day = calendar.get(Calendar.DAY_OF_MONTH);
```

```
    int month = calendar.get(Calendar.MONTH) + 1;
    int year = calendar.get(Calendar.YEAR);
    int hours = calendar.get(Calendar.HOUR_OF_DAY);
    int minutes = calendar.get(Calendar.MINUTE);
    int seconds = calendar.get(Calendar.SECOND);

    String results;

    results = getNoOfTrainRuleReviews() + "_";
    results+= getThresholdPercentage() +"PER";
    results+= getThresholdNoOfAppearances();
    results+= "_" + day + "" + month + "" + year;
    results+= "_" + hours + "." + minutes + "." + seconds;

    setResultsDirectory(new File(results));
    getResultsDirectory().mkdir();
}

public void findTrainRuleFilenames() throws IOException {
    System.out.println("Selecting reviews for creating POS
            rules...");

    FilenamePattern filePattern = new FilenamePattern
            (getMinPosRating(), getMaxPosRating());
    setPosTrainRuleFilenames(filePattern.findFilenames(
            getPosTrainRuleFilepath(),
            (int) (getNoOfTrainRuleReviews() * posRate),
            isShuffle(), getSeed()));

    filePattern = new FilenamePattern(getMinNegRating(),
            getMaxNegRating());
    setNegTrainRuleFilenames(filePattern.findFilenames(
            getNegTrainRuleFilepath(),
            (int) (getNoOfTrainRuleReviews() * (1 - posRate)),
            isShuffle(), getSeed()));

    writeFilenames(getPosTrainRuleFilenames(),
            "FilenamesForTrainRule.txt");
    writeFilenames(getNegTrainRuleFilenames(),
            "FilenamesForTrainRule.txt");
}

public void createPOSRulesHashtable() throws
            ClassNotFoundException, IOException{

    MaxentTagger tagger = getTagger();
    Hashtable<String, POSRulesHashtableValues>
    SentimentRulesMap = new Hashtable<String,
            POSRulesHashtableValues>();

    System.out.println("Creating positive rules...");
    checkReviews (getPosTrainRuleFilepath(),
           getPosTrainRuleFilenames(),1,tagger,SentimentRulesMap);
```

```
    System.out.println("Creating negative rules...");
    checkReviews (getNegTrainRuleFilepath(),
           getNegTrainRuleFilenames(),0,tagger,SentimentRulesMap);
    FileWriter writer = new FileWriter(new File
           (getResultsDirectory()+"/" +
           "POSRulesHashtable.txt"), true);
    storeHashtable(SentimentRulesMap, writer);
    writer.close();
}

public Hashtable <String, POSRulesHashtableValues>
           loadPOSRulesHashtable(String POSRulesHashtableFilepath)
           throws ClassNotFoundException, IOException{

    Hashtable <String, POSRulesHashtableValues> hash =
           new Hashtable <String, POSRulesHashtableValues>();
    BufferedReader br = new BufferedReader (new
           FileReader(POSRulesHashtableFilepath));
    String key;
    int pos, neg;
    for(String line; (line = br.readLine()) != null; ) {
        String[] pattern = line.split("\t");
        key = pattern[0];
        pos = Integer.parseInt(pattern[1]);
        neg = Integer.parseInt(pattern[2]);
        int thresholdNoOfAppearances =
            getThresholdNoOfAppearances();
        double thresholdPercentage = getThresholdPercentage();
        int total_appearances = pos + neg;
        if (total_appearances >= thresholdNoOfAppearances)
            if (((pos * 1.0 / total_appearances) * 100)>=
            thresholdPercentage || ((neg*1.0 / total_appearances)
            * 100)>= thresholdPercentage)
                hash.put(key, new POSRulesHashtableValues(pos,
                    neg, true));
    }
    br.close();
    return hash;
}

public void storeHashtable(Hashtable<String,
           POSRulesHashtableValues> SentimentRulesHashtable,
           FileWriter writer) throws ClassNotFoundException,
           IOException{

    List<String> list = new ArrayList<String>();

    for (String key : SentimentRulesHashtable.keySet()){
        POSRulesHashtableValues values =
            SentimentRulesHashtable.get(key);
        int pos = values.getPosCounter();
        int neg = values.getNegCounter();
        int total_appearances = pos + neg;
```

```
        int thresholdNoOfAppearances =
            getThresholdNoOfAppearances();
        double thresholdPercentage = getThresholdPercentage();

        if (total_appearances < thresholdNoOfAppearances)
            list.add(key);
        else{
            if (((pos*1.0/total_appearances)*100)>=
                            thresholdPercentage){
            values.setActive(true);
            SentimentRulesHashtable.put(key, values);
            writer.write(key + "\t" + pos + "\t" + neg + "\n");
            }
            else if (((neg*1.0/total_appearances)*100)>=
                            thresholdPercentage){
            writer.write(key + "\t" + pos + "\t" + neg + "\n");
            values.setActive(true);
            SentimentRulesHashtable.put(key, values);
            }
            else
                list.add(key);
        }
    }
    for (Iterator<String> iterator = list.iterator();
            iterator.hasNext();) {
        String key = iterator.next();
        SentimentRulesHashtable.remove(key);
    }
}

private void checkReviews (String reviewsFilepath,
        ArrayList <String > reviewFilenames, int sentiment,
        MaxentTagger tagger, Hashtable<String,
        POSRulesHashtableValues> SentimentRulesMap)
                throws IOException, ClassNotFoundException {

    String[] array = {".","..","...","?","??","???","!","!!",
            "!!!","<br /><br />"};
    String filepath;
    if (sentiment == 1) filepath = getPosTrainRuleFilepath();
    else filepath = getNegTrainRuleFilepath();
    for (String s_file: reviewFilenames){
        Reader reader = new BufferedReader(new
            FileReader(filepath +"/" + s_file));
        DocumentPreprocessor dp = new
            DocumentPreprocessor (reader);
        dp.setSentenceFinalPuncWords(array);
        for (List<HasWord> sentence : dp) {
            String tagSentence =
                sentencePattern(sentence, tagger);
            addPattern(tagSentence, sentiment, SentimentRulesMap);
        }
        reader.close();
```

```
    }
}

private void addPattern(String tagSentence, int sentiment,
        Hashtable<String, POSRulesHashtableValues>
        SentimentRulesMap) throws IOException{

    if (!(tagSentence.equals("") ||tagSentence.equals(" "))){
        POSRulesHashtableValues values;
        if (SentimentRulesMap.containsKey(tagSentence))
            values = SentimentRulesMap.get(tagSentence);
        else
            values = new POSRulesHashtableValues();
        if(sentiment==1)
            values.setPosCounter(values.getPosCounter()+1);
        else
            values.setNegCounter(values.getNegCounter()+1);
        SentimentRulesMap.put(tagSentence,values);
    }
}

public String sentencePattern(List<HasWord> sentence,
        MaxentTagger tagger) throws IOException{
    List<TaggedWord> tSentence = tagger.tagSentence(sentence);
    SentiWordNet sent = getSentiWordNet();
    String strword, wordtag, wtag, modified_tSentence ="";
    int counter=1;
    for(TaggedWord word : tSentence){
        strword = word.word();
        if (strword.equals("<br />") ||
                strword.equals("<br /><br />")) continue;
        if (!(strword.equals("n't") || strword.equals("not"))){
            wordtag = word.tag().toString();
            wtag = wordtag.substring(0,1).toLowerCase();
            if (wordtag.equals("RP")) continue;
            else if(wtag.equals("j")) {
                wtag = "a";
                wordtag = "ADJECTIVE";
            }
            else if (wtag.equals("n")) wordtag="NOUN";
            else if (wtag.equals("r")) wordtag="ADVERB";
            else if (wtag.equals("v")) wordtag="VERB";
            else continue;
            double sent_value = sent.extract(strword, wtag);
            String sent_str;
            if (sent_value > 0) sent_str = "POS_";
            else if (sent_value < 0) sent_str = "NEG_";
            else sent_str = "NEU_";
            String tag = sent_str + wordtag;
            if (!(modified_tSentence.equals("")))
                modified_tSentence+=" " + tag + counter;
            else
                modified_tSentence+= tag + counter;
```

```
            }
            else{
                if (!modified_tSentence.equals(""))
                    modified_tSentence+=" NOT"+counter;
                else
                    modified_tSentence+="NOT"+counter;
            }
            counter++;
        }
        return modified_tSentence;
    }

    public int[] patternMatches(String filepath) throws IOException{
        String[] array = {".","..","...","?","??","???","!","!!",
            "!!!","<br /><br />"};
        Reader reader = new BufferedReader(new FileReader(filepath));
        DocumentPreprocessor dp = new DocumentPreprocessor(reader);
        dp.setSentenceFinalPuncWords(array);

        int pos_counter=0;
        int neg_counter=0;
        String pattern;

        for (List<HasWord> sentence : dp) {
            pattern = sentencePattern(sentence, tagger);
            if (getPOSRulesHashtable().containsKey(pattern)){
                POSRulesHashtableValues values =
                        getPOSRulesHashtable().get(pattern);
                if (values.isActive()){
                  if (values.getPosCounter() >values.getNegCounter())
                        pos_counter++;
                  else neg_counter++;
                }
            }
        }
        reader.close();
        int[] patterns = new int[2];
        patterns[0] = pos_counter;
        patterns[1] = neg_counter;

        return patterns;

    }

    public Object[] patternMatchesFromString(String text) throws
IOException{
        String[] array = {".","..","...","?","??","???","!","!!",
            "!!!","<br />","<br /><br />"};
        Reader reader = new StringReader(text);
        DocumentPreprocessor dp = new DocumentPreprocessor(reader);
        dp.setSentenceFinalPuncWords(array);

        int pos_counter=0;
```

```
    int neg_counter=0;
    String pattern;
    ArrayList<String> patterns_list = new ArrayList<String>();
    for (List<HasWord> sentence : dp) {
        pattern = sentencePattern(sentence, tagger);
        patterns_list.add(pattern);
        if (POSRulesHashtable.containsKey(pattern)){
            POSRulesHashtableValues values =
                    POSRulesHashtable.get(pattern);
            if (values.isActive()){
              if (values.getPosCounter() >values.getNegCounter())
                    pos_counter++;
              else
                    neg_counter++;
            }
        }
    }
    reader.close();
    Object[] patterns = new Object[3];
    patterns[0] = pos_counter;
    patterns[1] = neg_counter;
    patterns[2] = patterns_list;
    return patterns;
}

public int getMinPosRating() {
    return minPosRating;
}

public void setMinPosRating(int minPosRating) {
    this.minPosRating = minPosRating;
}

public int getMaxPosRating() {
    return maxPosRating;
}

public void setMaxPosRating(int maxPosRating) {
    this.maxPosRating = maxPosRating;
}

public int getMinNegRating() {
    return minNegRating;
}

public void setMinNegRating(int minNegRating) {
    this.minNegRating = minNegRating;
}

public int getMaxNegRating() {
    return maxNegRating;
}
```

```
public void setMaxNegRating(int maxNegRating) {
    this.maxNegRating = maxNegRating;
}

public double getPosRate() {
    return posRate;
}

public void setPosRate(double posRate) {
    this.posRate = posRate;
}

public ArrayList<String> getPosTrainRuleFilenames() {
    return posTrainRuleFilenames;
}

public void setPosTrainRuleFilenames(ArrayList<String>
       posTrainRuleFilenames) {
    this.posTrainRuleFilenames = posTrainRuleFilenames;
}

public ArrayList<String> getNegTrainRuleFilenames() {
    return negTrainRuleFilenames;
}

public void setNegTrainRuleFilenames(ArrayList<String>
        negTrainRuleFilenames) {
    this.negTrainRuleFilenames = negTrainRuleFilenames;
}

public boolean isShuffle() {
    return shuffle;
}

public void setShuffle(boolean shuffle) {
    this.shuffle = shuffle;
}

public long getSeed() {
    return seed;
}

public void setSeed(long seed) {
    this.seed = seed;
}

public File getResultsDirectory() {
    return resultsDirectory;
}

public void setResultsDirectory(File resultsDirectory) {
    this.resultsDirectory = resultsDirectory;
}
```

```
public String getPosTrainRuleFilepath() {
    return posTrainRuleFilepath;
}

public void setPosTrainRuleFilepath(String posTrainRuleFilepath){
    this.posTrainRuleFilepath = posTrainRuleFilepath;
}

public String getNegTrainRuleFilepath() {
    return negTrainRuleFilepath;
}

public void setNegTrainRuleFilepath(String negTrainRuleFilepath){
    this.negTrainRuleFilepath = negTrainRuleFilepath;
}

public int getNoOfTrainRuleReviews() {
    return noOfTrainRuleReviews;
}

public void setNoOfTrainRuleReviews(int noOfTrainRuleReviews) {
    this.noOfTrainRuleReviews = noOfTrainRuleReviews;
}

public void writeFilenames(ArrayList<String> filenames,
        String outputFile) throws IOException {
   FileWriter writer = new FileWriter(new File(getResultsDirectory()
        + "/" + outputFile), true);
   for (String s: filenames)
        writer.write(s +  "\n");

   writer.close();
}

public MaxentTagger getTagger() {
    return tagger;
}

public void setTagger(MaxentTagger tagger) {
    this.tagger = tagger;
}

public Hashtable <String, POSRulesHashtableValues>
        getPOSRulesHashtable() {
    return POSRulesHashtable;
}

public void setPOSRulesHashtable(Hashtable<String,
        POSRulesHashtableValues> POSRulesHashtable) {
    this.POSRulesHashtable = POSRulesHashtable;
}
```

```
    public SentiWordNet getSentiWordNet() {
        return SentiWordNet;
    }

    public void setSentiWordNet(SentiWordNet SentiWordNet) {
        SentiWordNet = SentiWordNet;
    }

    public int getThresholdNoOfAppearances() {
        return thresholdNoOfAppearances;
    }

    public void setThresholdNoOfAppearances(int
            thresholdNoOfAppearances) {
        this.thresholdNoOfAppearances = thresholdNoOfAppearances;
    }

    public double getThresholdPercentage() {
        return thresholdPercentage;
    }

    public void setThresholdPercentage(double thresholdPercentage) {
        this.thresholdPercentage = thresholdPercentage;
    }
}
```

A.2. Η κλάση POSRulesHashtableValues

```
package sentimentanalysis;

public class POSRulesHashtableValues {


    private int posCounter;
    private int negCounter;
    private boolean isActive;

    POSRulesHashtableValues(){
        this.posCounter=0;
        this.negCounter=0;
        this.isActive=false;
    }

    POSRulesHashtableValues(int pos, int neg, boolean active){
       this.posCounter=pos;
        this.negCounter=neg;
        this.isActive=active;
    }
    public int getPosCounter() {
        return posCounter;
    }
```

```
    public void setPosCounter(int posCounter) {
        this.posCounter = posCounter;
    }

    public int getNegCounter() {
       return negCounter;
    }
    public void setNegCounter(int negCounter) {
       this.negCounter = negCounter;
    }

    public boolean isActive() {
       return isActive;
    }
    public void setActive(boolean isActive) {
        this.isActive = isActive;
    }
}
```

A.3. Η κλάση SentiWordNet

```
package sentimentanalysis;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class SentiWordNet {

    private Map<String, Double> dictionary;

    public SentiWordNet(String pathToSWN) throws IOException {

        this.dictionary = new HashMap<String, Double>();

        HashMap<String, HashMap<Integer, Double>> tempDictionary =
                new HashMap<String, HashMap<Integer, Double>>();

        BufferedReader csv = null;
        try {
            csv = new BufferedReader(new FileReader(pathToSWN));
            int lineNumber = 0;

            String line;
            while ((line = csv.readLine()) != null) {
                lineNumber++;
                if (!line.trim().startsWith("#")) {
                    String[] data = line.split("\t");
                    String wordTypeMarker = data[0];
                    if (data.length != 6) {
```

```
                        throw new IllegalArgumentException
                                ("Incorrect tabulation format in file,
                                line: "+ lineNumber);
                    }

                     Double synsetScore = Double.parseDouble(data[2])
-
                        Double.parseDouble(data[3]);
                    String[] synTermsSplit = data[4].split(" ");

                    for (String synTermSplit : synTermsSplit) {
                        String[] synTermAndRank =
                                synTermSplit.split("#");
                        String synTerm = synTermAndRank[0] + "#"
                                + wordTypeMarker;

                        int synTermRank = Integer.parseInt
                                (synTermAndRank[1]);
                        if (!tempDictionary.containsKey(synTerm)) {
                            tempDictionary.put(synTerm,
                                    new HashMap<Integer, Double>());
                        }

                        tempDictionary.get(synTerm).put(synTermRank,
                                synsetScore);
                    }
                }
            }

            for (Map.Entry<String, HashMap<Integer, Double>> entry :
                    tempDictionary.entrySet()) {
                String word = entry.getKey();
                Map<Integer, Double> synSetScoreMap =
                        entry.getValue();
                double score = 0.0;
                double sum = 0.0;
                for (Map.Entry<Integer, Double> setScore :
                      synSetScoreMap.entrySet()) {
                    score += setScore.getValue()/(double)
                            setScore.getKey();
                    sum += 1.0 / (double) setScore.getKey();
                }
                score /= sum;

                this.dictionary.put(word, score);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (csv != null) {
                csv.close();
            }
        }
```

```
    }

    public double extract(String word, String pos) {
        if(this.dictionary.get(word + "#" + pos) != null) {
                return this.dictionary.get(word + "#" + pos);
        }
        else {
                return 0;
        }
    }

}
```

A.4. Η τροποποιημένη κλάση AttributeRelationFile

```
package sentimentanalysis;

import weka.core.Attribute;
import weka.core.DenseInstance;
import weka.core.Instances;
import gr. demokritos .iit.jinsect.structs.GraphSimilarity;

import java .io. BufferedReader;
import java .io. BufferedWriter;
import java .io. FileReader;
import java .io. FileWriter;
import java .io. IOException;
import java.util.ArrayList;

public class AttributeRelationFile {

        private String relationName;
        private ArrayList < Attribute > attributes;
        private Instances instances;

        public AttributeRelationFile (String relationName) {
                this.relationName = relationName;
        }

        public void createFile (WordGraphsSimilarities values,
                POSRulesHashtable SentimentPOSRules, String
                posFilepath, String negFilepath, ArrayList <String>
                posReviewFilenames, ArrayList <String>
                negReviewFilenames) throws IOException,
                ClassNotFoundException {

                createAttributes();
                addHeader();
                addData (values, SentimentPOSRules, posFilepath,
                        negFilepath, posReviewFilenames,
                        negReviewFilenames);
        }
```

```
public void createFile (String file) throws IOException {
        createRelativeAttributes();
        addHeader();
        addData (file);
}

private void createAttributes() {
        attributes = new ArrayList <Attribute>();
        attributes.add(new Attribute
                ("PositiveContainmentSimilarity"));
        attributes.add(new Attribute
                ("PositiveNormalizedValueSimilarity"));
        attributes.add(new Attribute
                ("PositiveValueSimilarity"));
        attributes.add(new Attribute
                ("NegativeContainmentSimilarity"));
        attributes.add(new Attribute
                ("NegativeNormalizedValueSimilarity"));
        attributes.add(new Attribute
                ("NegativeValueSimilarity"));
        attributes.add(new Attribute
                ("PositiveRulesMatched"));
        attributes.add(new Attribute
                ("NegativeRulesMatched"));

        ArrayList <String > sentimentValues =
                new ArrayList <String >();
        sentimentValues.add("0");
        sentimentValues.add("1");
        attributes.add(new Attribute
                ("Sentiment", sentimentValues));
}

private void createRelativeAttributes() {
        attributes = new ArrayList < Attribute >();
        attributes.add(new Attribute
                ("RelativeContainmentSimilarity"));
        attributes.add(new Attribute
                ("RelativeNormalizedValueSimilarity"));
        attributes.add(new Attribute
                ("RelativeValueSimilarity "));
        ArrayList <String > sentimentValues =
                new ArrayList <String >();
        sentimentValues.add("0");
        sentimentValues.add("1");
        attributes .add(new Attribute
                ("Sentiment", sentimentValues));
}

private void addHeader() {
        instances = new Instances(relationName,attributes,0);
}
```

```
private void addData (WordGraphsSimilarities values,
        POSRulesHashtable SentimentPOSRules, String
        posFilepath, String negFilepath, ArrayList <String >
        posReviewFilenames,ArrayList <String >
        negReviewFilenames) throws IOException,
        ClassNotFoundException {

        addInstances (values.getPosModelGraph(),
                SentimentPOSRules, posFilepath,
                posReviewFilenames, values, 1);
        addInstances (values.getNegModelGraph(),
                SentimentPOSRules, negFilepath,
                negReviewFilenames, values, 0);
}

private void addData (String file) throws IOException {

        BufferedReader reader = new BufferedReader
                (new FileReader (file));
        for (int line = 0; line < 11; line ++)
                reader.readLine();

        String line = reader.readLine();
        while (line != null) {
                String [] values = line.split (",");
                double [] simValues = new double[values.length];
                for (int index = 0; index < values.length - 1;
                        index ++)
                        simValues [index] =
                        Double.parseDouble (values [index]);
                simValues [values.length-1]= Integer.parseInt
                        values[values.length-1]);
                addInstance (simValues);
                line = reader.readLine();
        }
        reader.close();
}

private void addInstances (ModelWordGraph graph,
        POSRulesHashtable POSRulesHashtable, String
        reviewsFilepath, ArrayList <String > reviewFilenames,
        WordGraphsSimilarities values, int sentiment)
        throws IOException, ClassNotFoundException {

        ReviewWordGraph reviewGraph = new ReviewWordGraph
                (graph.getWindow(), graph.getReviewsGraph().
                        isPreprocess());

        for (String s: reviewFilenames) {
                String full_path = reviewsFilepath.concat (s);
                reviewGraph.createGraph (full_path);
                values.graphsSimilaritiesWith (reviewGraph);
```

```
                int patterns[];
                patterns = POSRulesHashtable.patternMatches
                        (full_path);
                int pos_counter = patterns[0];
                int neg_counter = patterns[1];
                addInstance (values.getPosGraphSimilarities(),
                        values.getNegGraphSimilarities(),
                        pos_counter, neg_counter, sentiment);
        }
}

private void addInstance (GraphSimilarity posGraphSim,
        GraphSimilarity negGraphSim, int pos_pattern, int
        neg_pattern, int sentiment) {

        double [] instance = new double
                [instances.numAttributes() ];

        instance [0] = posGraphSim.ContainmentSimilarity;
        instance [1] = posGraphSim.ValueSimilarity /
                posGraphSim.SizeSimilarity;
        instance [2] = posGraphSim.ValueSimilarity;
        instance [3] = negGraphSim.ContainmentSimilarity;
        instance [4] = negGraphSim.ValueSimilarity /
                negGraphSim.SizeSimilarity;
        instance [5] = negGraphSim.ValueSimilarity;

        instance [6] = pos_pattern;
        instance [7] = neg_pattern;
        instance [8] = sentiment;

        instances .add(new DenseInstance (1.0, instance));
}

private void addInstance (double [] simValues) {
        double [] instance = new double
                [instances.numAttributes() ];

        instance [0] = dsim (simValues [0], simValues [3]);
        instance [1] = dsim (simValues [1], simValues [4]);
        instance [2] = dsim (simValues [2], simValues [5]);
        instance [3] = simValues [6];
        instances.add(new DenseInstance (1.0, instance));
}

private int dsim (double posSim, double negSim) {
        int equal = 0;
        int positive = 1;
        int negative = 2;
        if (posSim < negSim) return negative;
        else if (posSim > negSim) return positive;
        else return equal;
```

```
        }


        Public void storeToFile (String outputFile) throws IOException{
                BufferedWriter writer =
                      new BufferedWriter(new FileWriter (outputFile));
                writer.write (instances.toString());
                writer.flush();
                writer.close();
        }

        public Instances getInstances() {
                return instances;
        }
}
```

A.5. Η τροποποιημένη κλάση Classifiers

```
package sentimentanalysis;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Properties;

public class Classifiers {

        private String wordGraphsTrainFile;

        private String posTestFilepath;
        private String negTestFilepath;
        private String taggerFilepath;
        private String POSRulesHashtableFilepath;
        private String SentiWordNetFilepath;

        private String posWordGraphBinaryFile;
        private String negWordGraphBinaryFile;
        private ArrayList<String> posTestFilenames;
        private ArrayList<String> negTestFilenames;

        private int minPosRating;
        private int maxPosRating;
        private int minNegRating;
```

```
private int maxNegRating;
private double posRate;
private boolean shuffle;
private long seed;
private int window;
private boolean preprocess;

private String wekaClassifierName;
private String classifierName;

private int noOfWordGraphsReviews;
private int noTrainReviews;
private int noTestReviews;
private int thresholdNoOfAppearances;
private double thresholdPercentage;
private String wordGraphsClassifierBinaryFile;

private File resultsDirectory;
private String resultsFile;

public static void main(String[] args) throws Exception {
        if (args.length != 3)
                return;

        InputStream reader = new FileInputStream(args[0]);
        Properties properties = new Properties();

        if (args[0].contains(".xml"))
                properties.loadFromXML(reader);
        else
                properties.load(reader);

        boolean stage3 = Boolean.parseBoolean(args[1]);
        boolean allStages = Boolean.parseBoolean(args[2]);

        long start = System.currentTimeMillis();

        System.out.println("Running Third Stage of Sentiment"
                + "Classification: Creating and evaluating"
                + "classifier\n");

        Classifiers object = new Classifiers();

        object.setParameters(properties, true);
        object.createResultsDirectory();
        object.setFiles(allStages);
        object.findTestFilenames(stage3);
        System.out.println("Creating classifier...\n");

        long wordGraphs = System.currentTimeMillis();
        object.createClassifierForWordGraphs();

        System.out.println("Evaluating classifier...\n");
```

```
        long wordGraphs2 = System.currentTimeMillis();
        object.evaluateClassifierForWordGraphs();

        long end = System.currentTimeMillis();

        NumberFormat formatter = new DecimalFormat("#0.00000");

        FileWriter output = new FileWriter(new File(
                object.getResultsFile()), true);

        output.write("Time for creating classifier: ");
        output.write(formatter.format((wordGraphs2 -
                wordGraphs) / 1000d) + " seconds\n");

        output.write("Time for evaluating classifier: ");
        output.write(formatter.format((end - wordGraphs2) /
                1000d) + " seconds\n");

        output.write("\nExecution time is ");
        output.write(formatter.format((end - start) / 1000d)
                + " seconds");
        output.close();

        System.out.print("\nExecution time is " +
                formatter.format((end - start) / 1000d)
                + " seconds");
}

public void setParameters(Properties properties,
        boolean onlyThisStage) {

setWekaClassifierName(properties.getProperty("classifierName"));
String[] classifier = getWekaClassifierName().split("\\.");
String classifierName = classifier[classifier.length - 1];
setClassifierName(classifierName);
setPosTestFilepath(properties.
        getProperty("positiveTestFilepath"));
setNegTestFilepath(properties.
        getProperty("negativeTestFilepath"));
setTaggerFilepath(properties.getProperty("taggerFilepath"));
setPOSRulesHashtableFilepath(properties.
        getProperty("POSRulesHashtableFilepath"));
setSentiWordNetFilepath(properties.
        getProperty("SentiWordNetFilepath"));
setNoTestReviews(Integer.parseInt(properties.
        getProperty("noOfTestReviews")));
setPositiveRate(Double.parseDouble(properties.getProperty
        ("positiveRate"))/100);
setWindow(Integer.parseInt(properties.getProperty
        ("windowSize")));
if (properties.getProperty("seed") != null)
        setSeed(Long.parseLong(properties.
                getProperty("seed")));
```

```
else
        setSeed(-1);

setShuffle(Boolean.parseBoolean(properties.
        getProperty("shuffle")));
setMinPosRating(Integer.parseInt(properties.
        getProperty("minPositiveRating")));
setMaxPosRating(Integer.parseInt(properties.
        getProperty("maxPositiveRating")));
setMinNegRating(Integer.parseInt(properties.
        getProperty("minNegativeRating")));
setMaxNegRating(Integer.parseInt(properties.
        getProperty("maxNegativeRating")));
setThresholdNoOfAppearances(Integer.parseInt(properties.
        getProperty("thresholdNoOfAppearances")));
setThresholdPercentage(Integer.parseInt(properties.
        getProperty("thresholdPercentage")));
        if (onlyThisStage) {
                System.out.println("Setting parameters...");
                setWordGraphsTrainFile(properties.
                        getProperty("wordGraphsTrainFile"));
                setPosWordGraphBinaryFile(properties.
                        getProperty("posWordGraphBinaryFile"));
                setNegWordGraphBinaryFile(properties.
                        getProperty("negWordGraphBinaryFile"));
        }
}

public void createResultsDirectory() {
        System.out.println("Creating directory for results...");
        String results;

        Date date = new Date();
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(date);
        int day = calendar.get(Calendar.DAY_OF_MONTH);
        int month = calendar.get(Calendar.MONTH) + 1;
        int year = calendar.get(Calendar.YEAR);
        int hours = calendar.get(Calendar.HOUR_OF_DAY);
        int minutes = calendar.get(Calendar.MINUTE);
        int seconds = calendar.get(Calendar.SECOND);

        results=getNoTestReviews()+"_" + getClassifierName();
        results+= "_"+ getThresholdPercentage() +"PER";
        results += "_" + day + "" + month + "" + year;
        results += "_" + hours + "." + minutes + "."
                + seconds;
        setResultsDirectory(new File(results));
        getResultsDirectory().mkdir();
}

public void setFiles(boolean allStages) {
        String filePrefix = getResultsDirectory()
```

```
                + "/Binary" + getClassifierName();
        if (allStages) {
                setWordGraphsClassifierBinaryFile(filePrefix
                                + "ClassifierWordGraphs");
                setResultsFile(getResultsDirectory() + "/" +
                        "Results.txt");
        } else {
                System.out.println("Setting the output
                        filenames...");
                String fileSuffix = getNoOfWordGraphsReviews()
                        + "_" + getNoTrainReviews() + "_"
                        + getNoTestReviews();
                setWordGraphsClassifierBinaryFile(filePrefix +
                        "ClassifierWordGraphs_" + fileSuffix);
                setResultsFile(getResultsDirectory() + "/"
                        + fileSuffix + "_Results.txt");
        }
}

public void findTestFilenames(boolean stage3)
        throws IOException {

        System.out.println("Selecting reviews for the testing
                corpus...");
        FilenamePattern filePattern = new FilenamePattern
                (getMinPosRating(), getMaxPosRating());
        setPosTestFilenames(filePattern.findFilenames
                (getPosTestFilepath(),(int)(getNoTestReviews()
                        * posRate), isShuffle(), getSeed()));
        filePattern = new FilenamePattern
                (getMinNegRating(), getMaxNegRating());
        setNegTestFilenames(filePattern.findFilenames
                (getNegTestFilepath(),(int)(getNoTestReviews()
                (1 - posRate)),isShuffle(), getSeed()));
        if (stage3) {
                writeFilenames(getPosTestFilenames(),
                        "FilenamesForTest" + "_" +
                        getNoTestReviews()+ ".txt");
                writeFilenames(getNegTestFilenames(),
                        "FilenamesForTest" + "_" +
                        getNoTestReviews() + ".txt");
        } else {
                writeFilenames(getPosTestFilenames(),
                        "FilenamesForTest.txt");
                writeFilenames(getNegTestFilenames(),
                        "FilenamesForTest.txt");
                }
}

public void writeFilenames(ArrayList<String> filenames,
        String outputFile) throws IOException {

        FileWriter writer = new FileWriter(new File
```

```
                (getResultsDirectory()+"/"+outputFile),true);
        for (String s: filenames)
                writer.write(s +  "\n");
        writer.close();
    }

    public void createClassifierForWordGraphs() throws Exception
{
        SentimentClassifier sentimentClassifier =
            new SentimentClassifier (getWekaClassifierName(),
            getWordGraphsTrainFile());
        sentimentClassifier.createClassifierInstance();
        sentimentClassifier.trainClassifier();
        sentimentClassifier.storeSentimentClassifier
                (getWordGraphsClassifierBinaryFile());
    }

    public void evaluateClassifierForWordGraphs()
        throws Exception {

        SentimentAnalysisImp analyser =
                new SentimentAnalysisImp();
        setAnalyserParameters(analyser);
        int counter = 0;
        ArrayList<String> posfiles = getPosTestFilenames();
        ArrayList<String> negfiles = getNegTestFilenames();

        for(String file : posfiles) {
                BufferedReader br = new BufferedReader(new
                        FileReader(getPosTestFilepath()+file));
                String text = br.readLine();
                int sentiment = analyser.findSentiment(text);
                if (sentiment == 1) counter++;
                br.close();
        }

        for(String file : negfiles) {
                BufferedReader br = new BufferedReader(new
                        FileReader(getNegTestFilepath()+file));
                String text = br.readLine();
                int sentiment = analyser.findSentiment(text);
                if (sentiment == 0) counter++;
                br.close();
        }

        FileWriter writer = new FileWriter
                (getResultsDirectory() + "/POSRulesHashtable"
                + getThresholdPercentage() + "_"
                + getThresholdNoOfAppearances() + ".txt"),
                true);
        analyser.SentimentPOSRulesHashtable.storeHashtable
                (analyser.SentimentPOSRulesHashtable.
                getPOSRulesHashtable(), writer);
```

```
        writer.close();

        FileWriter writer1 = new FileWriter
                (getResultsFile(), true);
        writer1.write("Success Rate: " +
                (counter*1.0/getNoTestReviews())*100 + "%\n");
        writer1.close();
        System.out.println("Success Rate: " +
                (counter*1.0/getNoTestReviews())*100 + "%");
}

public void setAnalyserParameters(SentimentAnalysisImp
        analyser) throws ClassNotFoundException, IOException{
        analyser.setValues(new (getPosWordGraphBinaryFile(),
                getNegWordGraphBinaryFile()));
        SentimentClassifier s_classifier =
                new SentimentClassifier();
        s_classifier.loadSentimentClassifier
                (getWordGraphsClassifierBinaryFile());
        analyser.setClassifier
                (s_classifier.getClassifierInstance());
        analyser.setSentimentPOSRulesHashtable(new
                POSRulesHashtable(getTaggerFilepath(),
                getPOSRulesHashtableFilepath(),
                getSentiWordNetFilepath(),
                getThresholdNoOfAppearances(),
                getThresholdPercentage()));
        analyser.setWindow(getWindow());
        analyser.setPreprocess(isPreprocess());
}

public String getWordGraphsTrainFile() {
        return wordGraphsTrainFile;
}

public void setWordGraphsTrainFile(String wordGraphsTrainFile){
        this.wordGraphsTrainFile = wordGraphsTrainFile;
}

public String getWekaClassifierName() {
        return wekaClassifierName;
}

public void setWekaClassifierName(String wekaClassifierName) {
        this.wekaClassifierName = wekaClassifierName;
}

public String getClassifierName() {
        return classifierName;
}

public void setClassifierName(String classifierName) {
        this.classifierName = classifierName;
```

```
}

public int getNoTrainReviews() {
	return noTrainReviews;
}

public void setNoTrainReviews(int noTrainReviews) {
	this.noTrainReviews = noTrainReviews;
}

public int getNoTestReviews() {
	return noTestReviews;
}

public void setNoTestReviews(int noTestReviews) {
	this.noTestReviews = noTestReviews;
}


public int getNoOfWordGraphsReviews() {
	return noOfWordGraphsReviews;
}

public void setNoOfWordGraphsReviews(int noOfWordGraphsReviews){
	this.noOfWordGraphsReviews = noOfWordGraphsReviews;
}

public void setId(int id) {
	this.id = id;
}

public File getResultsDirectory() {
	return resultsDirectory;
}

public void setResultsDirectory(File resultsDirectory) {
	this.resultsDirectory = resultsDirectory;
}

public boolean isPreprocess() {
	return preprocess;
}

public void setPreprocess(boolean preprocess) {
	this.preprocess = preprocess;
}

public String getWordGraphsClassifierBinaryFile() {
	return wordGraphsClassifierBinaryFile;
}

public void setWordGraphsClassifierBinaryFile(
		String wordGraphsClassifierBinaryFile) {
```

```
        this.wordGraphsClassifierBinaryFile =
                wordGraphsClassifierBinaryFile;
}

public String getResultsFile() {
        return resultsFile;
}

public void setResultsFile(String resultsFile) {
        this.resultsFile = resultsFile;
}

public int getId() {
        return id;
}

public String getPosTestFilepath() {
        return posTestFilepath;
}

public void setPosTestFilepath(String posTestFilepath) {
        this.posTestFilepath = posTestFilepath;
}

public String getNegTestFilepath() {
        return negTestFilepath;
}

public void setNegTestFilepath(String negTestFilepath) {
        this.negTestFilepath = negTestFilepath;
}

public ArrayList<String> getPosTestFilenames() {
        return posTestFilenames;
}

public void setPosTestFilenames(ArrayList<String>
        posTestFilenames) {
        this.posTestFilenames = posTestFilenames;
}

public ArrayList<String> getNegTestFilenames() {
        return negTestFilenames;
}

public void setNegTestFilenames(ArrayList<String>
        negTestFilenames) {
        this.negTestFilenames = negTestFilenames;
}

public int getMinPosRating() {
        return minPosRating;
}
```

```
public void setMinPosRating(int minPosRating) {
        this.minPosRating = minPosRating;
}

public int getMaxPosRating() {
        return maxPosRating;
}

public void setMaxPosRating(int maxPosRating) {
        this.maxPosRating = maxPosRating;
}

public int getMinNegRating() {
        return minNegRating;
}

public void setMinNegRating(int minNegRating) {
        this.minNegRating = minNegRating;
}

public int getMaxNegRating() {
        return maxNegRating;
}

public void setMaxNegRating(int maxNegRating) {
        this.maxNegRating = maxNegRating;
}

public double getPositiveRate() {
        return posRate;
}

public void setPositiveRate(double positiveRate) {
        this.posRate = positiveRate;
}

public boolean isShuffle() {
        return shuffle;
}

public void setShuffle(boolean shuffle) {
        this.shuffle = shuffle;
}

public long getSeed() {
        return seed;
}

public void setSeed(long seed) {
        this.seed = seed;
}
```

```
public String getTaggerFilepath() {
        return taggerFilepath;
}

public void setTaggerFilepath(String taggerFilepath) {
        this.taggerFilepath = taggerFilepath;
}

public String getPOSRulesHashtableFilepath() {
        return POSRulesHashtableFilepath;
}

public void setPOSRulesHashtableFilepath(String
                POSRulesHashtableFilepath) {
        this.POSRulesHashtableFilepath =
                POSRulesHashtableFilepath;
}

public String getSentiWordNetFilepath() {
        return SentiWordNetFilepath;
}

public void setSentiWordNetFilepath(String
                SentiWordNetFilepath) {
        this.SentiWordNetFilepath = SentiWordNetFilepath;
}

public String getPosWordGraphBinaryFile() {
        return posWordGraphBinaryFile;
}

public void setPosWordGraphBinaryFile(String
                posWordGraphBinaryFile) {
        this.posWordGraphBinaryFile = posWordGraphBinaryFile;
}

public String getNegWordGraphBinaryFile() {
        return negWordGraphBinaryFile;
}

public void setNegWordGraphBinaryFile(String
                negWordGraphBinaryFile) {
        this.negWordGraphBinaryFile = negWordGraphBinaryFile;
}

public int getWindow() {
        return window;
}

public void setWindow(int window) {
        this.window = window;
}
```

```
	public int getThresholdNoOfAppearances() {
		return thresholdNoOfAppearances;
	}

	public void setThresholdNoOfAppearances(int
			thresholdNoOfAppearances) {
		this.thresholdNoOfAppearances =
			thresholdNoOfAppearances;
	}

	public double getThresholdPercentage() {
		return thresholdPercentage;
	}

	public void setThresholdPercentage(double
			thresholdPercentage) {
		this.thresholdPercentage = thresholdPercentage;
	}
}
```

A.6. Η κλάση SentimentAnalysisImp

```
package sentimentanalysis;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import gr.demokritos.iit.jinsect.structs.GraphSimilarity;
import weka.classifiers.Classifier;
import weka.core.Attribute;
import weka.core.DenseInstance;
import weka.core.Instance;
import weka.core.Instances;

public class SentimentAnalysisImp {

	int window;
	boolean preprocess;
	WordGraphsSimilarities values;
	Instances instances;
	Classifier classifier;
	POSRulesHashtable SentimentPOSRulesHashtable;

	public SentimentAnalysisImp() {
		String relationName =
			"Sentiment_of_Similarities_of_WordGraphs";
		ArrayList<Attribute> attributes = new ArrayList<>();
		attributes.add(new Attribute
			("PositiveContainmentSimilarity"));
		attributes.add(new Attribute
			("PositiveNormalizedValueSimilarity"));
		attributes.add(new Attribute
```

```
                ("PositiveValueSimilarity"));
        attributes.add(new Attribute
                ("NegativeContainmentSimilarity"));
        attributes.add(new Attribute
                ("NegativeNormalizedValueSimilarity"));
        attributes.add(new Attribute
                ("NegativeValueSimilarity "));
        attributes.add(new Attribute
                ("PositiveRulesMatch "));
        attributes.add(new Attribute
                ("NegativeRulesMatch "));
        ArrayList<String> sentimentValues = new ArrayList<>();
        sentimentValues.add("0");
        sentimentValues.add("1");
        attributes.add(new Attribute
                ("Sentiment", sentimentValues));
        Instances instances =
                new Instances(relationName, attributes, 0);
        instances.setClassIndex(instances.numAttributes() - 1);
        setInstances(instances);
}

@SuppressWarnings("unchecked")
public int findSentiment(String text)
        throws Exception, IOException, ClassNotFoundException {
        ReviewWordGraph reviewGraph =
                new ReviewWordGraph(window, preprocess);
        reviewGraph.createGraphFromString(text);
        values.graphsSimilaritiesWith(reviewGraph);

        Object patterns[];
        patterns = SentimentPOSRulesHashtable.
                patternMatchesFromString(text);
        int pos_counter = (int) patterns[0];
        int neg_counter = (int) patterns[1];
        ArrayList<String> patterns_list =
                (ArrayList<String>) patterns[2];
        Instance inst =
                getInstance(values, pos_counter, neg_counter);
        int sentiment;
        try {
                sentiment = (int) classifier.
                        classifyInstance(inst);
        } catch (Exception e) {
                throw new Exception(e.getMessage(), e);
        }

        for(String string : patterns_list) {
                POSRulesHashtableValues values;
                if (SentimentPOSRulesHashtable.
                getPOSRulesHashtable(). containsKey (string))
                values = SentimentPOSRulesHashtable.
                        getPOSRulesHashtable().get(string);
```

```
                else
                        values = new POSRulesHashtableValues();
                if(sentiment==1)
                values.setPosCounter
                        (values.getPosCounter()+1);
                else
                        values.setNegCounter
                                (values.getNegCounter()+1);
                SentimentPOSRulesHashtable.
                getPOSRulesHashtable(). put(string,values);
        }

        return sentiment;
}

public POSRulesHashtable getSentimentPOSRulesHashtable() {
        return SentimentPOSRulesHashtable;
}

public void setSentimentPOSRulesHashtable
        (POSRulesHashtable SentimentPOSRulesHashtable) {
        this.SentimentPOSRulesHashtable =
                SentimentPOSRulesHashtable;
}

public int getWindow() {
        return window;
}


public void setWindow(int window) {
        this.window = window;
}

public boolean isPreprocess() {
        return preprocess;
}

public void setPreprocess(boolean preprocess) {
        this.preprocess = preprocess;
}

public WordGraphsSimilarities getValues() {
        return values;
}

public void setValues(WordGraphsSimilarities values) {
        this.values = values;
}

public Classifier getClassifier() {
        return classifier;
}
```

```
	public void setClassifier(Classifier classifier) {
		this.classifier = classifier;
	}

	public void add(Instance instance){
		instances.add(instance);
	}

	public Instance getInstance
		(WordGraphsSimilarities values, int pos, int neg){

		GraphSimilarity posGraphSim =
			values.getPosGraphSimilarities();
		GraphSimilarity negGraphSim =
			values.getNegGraphSimilarities();

		Instance instance = new DenseInstance
			(instances.numAttributes());
		instance.setDataset(instances);
		instance.setValue(0, posGraphSim.ContainmentSimilarity);
		instance.setValue(1, posGraphSim.ValueSimilarity /
			posGraphSim.SizeSimilarity);
		instance.setValue(2, posGraphSim.ValueSimilarity);
		instance.setValue(3, negGraphSim.ContainmentSimilarity);
		instance.setValue(4, negGraphSim.ValueSimilarity /
			negGraphSim.SizeSimilarity);
		instance.setValue(5, negGraphSim.ValueSimilarity);
		instance.setValue(6, pos);
		instance.setValue(7, neg);
		return instance;
	}

	public void storeToFile(String outputFile) throws IOException{
		BufferedWriter writer = new BufferedWriter
			(new FileWriter(outputFile));
		writer.write(instances.toString());
		writer.flush();
		writer.close();
	}

	public Instances getInstances() {
		return instances;
	}

	public void setInstances(Instances instances) {
		this.instances = instances;
	}
}
```

Παράρτημα Β – Πίνακας κανόνων

Στο παράρτημα αυτό παρατίθεται ο πίνακας κανόνων που σχηματίστηκε στην ενότητα 5.3.3, χρησιμοποιώντας ολόκληρο το σύνολο δεδομένων εκπαίδευσης 25.000 κριτικών, με κατώφλι ποσόστωσης 100% και κατώφλι ελάχιστου απαιτούμενου αριθμού εμφανίσεων ενός κανόνα ίσο με 5.

Οι 127 κανόνες έχουν ως εξής:

Rule	**Positive Counter**	**Negative Counter**
NEU_VERB1 NOT2 POS_VERB3 POS_NOUN4 NEU_NOUN5	0	24
NEU_VERB1 NEG_ADJECTIVE2 NEU_NOUN3 POS_VERB4 NEU_ADVERB5 NEU_VERB6	0	23
POS_ADVERB1 NEU_VERB2 NOT3 POS_VERB4	0	23
NEU_NOUN1 NEU_NOUN2 NEU_VERB3 POS_ADJECTIVE4 NEG_ADVERB5	19	0
NEU_NOUN1 NEU_VERB2 POS_ADJECTIVE3 NEG_ADVERB4	18	0
NEU_VERB1 NEG_ADJECTIVE2 NEU_NOUN3 NEU_VERB4 NEU_ADVERB5 NEU_VERB6	0	17
NEU_ADJECTIVE1 NEG_ADJECTIVE2 NEG_ADJECTIVE3	0	13
POS_ADJECTIVE1 NEU_NOUN2 NEU_ADVERB3	12	0
NEU_ADVERB1 NEG_VERB2 NOT3 POS_VERB4	0	11
NEU_NOUN1 NEG_ADJECTIVE2 NEG_NOUN3	0	10
NEU_NOUN1 NEU_NOUN2 NEU_VERB3 NEG_ADJECTIVE4 NEG_ADJECTIVE5	0	9
NEU_ADVERB1 NEU_VERB2 NEU_NOUN3 POS_ADJECTIVE4 NEU_NOUN5	0	9
NEU_ADVERB1 NEU_NOUN2 NEG_NOUN3	0	9
NEU_ADVERB1 NEU_VERB2 POS_ADVERB3 NEG_ADJECTIVE4	0	9
NEU_NOUN1 NEG_ADVERB2 NEU_NOUN3	0	9
NEU_VERB1 POS_ADVERB2 NEU_VERB3 POS_ADJECTIVE4 NEU_NOUN5	0	9
NEU_NOUN1 NEU_VERB2 POS_ADVERB3 POS_ADJECTIVE4 POS_ADJECTIVE5	9	0
NEU_VERB1 POS_ADJECTIVE2 NEU_NOUN3 NEU_ADVERB4 NEU_VERB5	9	0
NEU_ADJECTIVE1 POS_ADVERB2 NEG_ADJECTIVE3	0	8
NEU_ADVERB1 NEG_ADJECTIVE2 NEU_NOUN3 POS_VERB4 NEU_ADVERB5 NEU_VERB6	0	8

NOT1 NEG_ADVERB2 POS_VERB3 NEU_NOUN4	0	8
NEG_VERB1 NEG_VERB2 NEU_NOUN3	0	8
NEU_NOUN1 NEU_NOUN2 NEU_NOUN3 NEU_VERB4 NEG_ADVERB5 NEG_ADJECTIVE6	0	8
NEU_NOUN1 NEG_ADJECTIVE2 NEG_ADJECTIVE3	0	8
NEU_NOUN1 NEU_VERB2 POS_NOUN3 NEU_NOUN4 NEU_NOUN5 NEU_NOUN6	0	8
NEU_NOUN1 NEU_VERB2 POS_ADJECTIVE3 NEU_NOUN4 POS_ADJECTIVE5 NEU_NOUN6	8	0
NEU_VERB1 NEU_NOUN2 POS_ADJECTIVE3 NEU_ADVERB4	0	7
NEU_VERB1 POS_VERB2 NEU_NOUN3 NEU_ADVERB4	0	7
NEU_NOUN1 NEU_NOUN2 NEU_VERB3 NOT4 POS_VERB5 NEU_NOUN6	0	7
NEU_VERB1 NEG_ADJECTIVE2 NEU_ADVERB3	0	7
NEU_VERB1 POS_ADVERB2 POS_ADVERB3 NEG_ADJECTIVE4 NEU_NOUN5	0	7
NEU_VERB1 POS_ADVERB2 NEG_ADJECTIVE3 NEU_NOUN4 POS_VERB5 NEU_ADVERB6 NEU_VERB7	0	7
NEU_ADJECTIVE1 NEU_NOUN2 NEU_NOUN3 NEU_NOUN4 NEU_NOUN5 NEU_NOUN6 NEU_NOUN7	7	0
NEU_VERB1 NEU_ADVERB2 NEG_ADJECTIVE3 NEU_NOUN4 POS_VERB5 NEU_ADVERB6 NEU_VERB7	0	6
NEG_ADJECTIVE1 POS_NOUN2 NEU_NOUN3 NEU_VERB4	0	6
NEU_NOUN1 NEU_VERB2 NOT3 POS_ADJECTIVE4 NEU_ADVERB5	0	6
NEU_ADVERB1 NOT2 NEU_NOUN3	0	6
NOT1 NEU_VERB2 NEU_NOUN3	0	6
NEU_VERB1 NEU_VERB2 NEU_VERB3 NEU_NOUN4 NEU_NOUN5 NEU_NOUN6	0	6
NEG_ADJECTIVE1 NEU_NOUN2 NEU_ADVERB3	0	6
NEU_VERB1 POS_ADVERB2 NEG_ADJECTIVE3 NEU_NOUN4 NEU_ADVERB5 NEU_VERB6	0	6
NEU_VERB1 NOT2 POS_VERB3 POS_VERB4	0	6
POS_VERB1 NEG_NOUN2 NEU_VERB3	0	6
NEU_NOUN1 NEU_NOUN2 NEU_ADJECTIVE3 NEG_NOUN4	0	6
NEG_ADJECTIVE1 NEU_NOUN2 POS_VERB3 NEU_ADVERB4 NEU_VERB5	0	6
NEU_NOUN1 NEU_VERB2 NEG_ADJECTIVE3 NEU_NOUN4 POS_VERB5 NEU_ADVERB6 NEU_VERB7	0	6
NEU_VERB1 NOT2 POS_VERB3 POS_NOUN4 NEU_VERB5	0	6
NEU_ADVERB1 POS_ADVERB2 NEU_ADVERB3 POS_ADJECTIVE4	0	6
POS_ADVERB1 NEG_ADVERB2	0	6

NEU_VERB1 NEU_ADVERB2 NEG_ADJECTIVE3 NEU_NOUN4 NEU_VERB5 NEU_ADVERB6 NEU_VERB7	0	6
POS_NOUN1 POS_VERB2 NEU_NOUN3	0	6
NEU_NOUN1 NEU_VERB2 NOT3 POS_ADVERB4 POS_ADJECTIVE5	0	6
NEU_NOUN1 POS_ADVERB2 NEG_ADJECTIVE3	0	6
NEU_NOUN1 NEU_NOUN2 NEU_NOUN3 NEU_VERB4 POS_ADJECTIVE5 NEG_ADVERB6	6	0
NEU_VERB1 NEU_ADVERB2 POS_ADJECTIVE3 NEU_VERB4	6	0
NEU_VERB1 POS_ADJECTIVE2 NEU_NOUN3 NEU_VERB4 NEU_NOUN5 NEU_NOUN6	6	0
NEU_ADVERB1 POS_ADJECTIVE2 NEG_NOUN3 NEU_VERB4	6	0
NEU_NOUN1 POS_VERB2 POS_ADVERB3 NEU_VERB4	6	0
NEU_NOUN1 NEU_NOUN2 POS_NOUN3 NEU_VERB4 POS_ADVERB5 POS_ADJECTIVE6	6	0
NEU_NOUN1 NEU_NOUN2 NEU_VERB3 POS_ADJECTIVE4 NEU_NOUN5 NEU_NOUN6 NEU_NOUN7 NEU_NOUN8	6	0
NEU_NOUN1 NEU_NOUN2 NEU_VERB3 POS_ADJECTIVE4 NEU_NOUN5 NEU_VERB6 NEU_NOUN7	6	0
POS_VERB1 NEU_NOUN2 NEU_NOUN3 NEU_NOUN4 NEU_NOUN5 NEU_NOUN6	6	0
NEU_ADVERB1 NEU_NOUN2 NEU_VERB3 NEG_ADVERB4 NEG_ADJECTIVE5	0	5
NEU_VERB1 NEG_ADJECTIVE2 NEU_NOUN3 NEU_ADJECTIVE4 NEU_NOUN5	0	5
NEU_VERB1 NEG_ADJECTIVE2 POS_NOUN3 NEU_VERB4 NEU_NOUN5	0	5
NEU_VERB1 NEU_VERB2 POS_VERB3 POS_ADJECTIVE4 NEU_NOUN5	0	5
NEU_VERB1 POS_ADVERB2 NEU_NOUN3 NEU_VERB4 NEU_NOUN5	0	5
NEG_VERB1 NOT2 POS_VERB3 NEU_NOUN4 POS_NOUN5	0	5
NEU_VERB1 NOT2 NEU_ADJECTIVE3 NEG_NOUN4	0	5
POS_ADVERB1 POS_ADVERB2 NEG_ADJECTIVE3	0	5
NEU_VERB1 POS_ADVERB2 NOT3	0	5
NEU_NOUN1 NEG_ADJECTIVE2 NEU_NOUN3 POS_NOUN4	0	5
NEU_VERB1 NEU_ADJECTIVE2 POS_NOUN3 NEU_VERB4	0	5
NEU_NOUN1 NEG_VERB2 NOT3 POS_VERB4 POS_NOUN5	0	5
NEU_ADVERB1 NEU_NOUN2 NEU_VERB3 NOT4 NEU_NOUN5	0	5
NEU_NOUN1 NEU_VERB2 NEG_ADJECTIVE3 NEU_NOUN4 NEU_NOUN5 POS_NOUN6	0	5
NEU_NOUN1 NEG_ADVERB2 POS_ADVERB3 NEU_VERB4	0	5
NEU_VERB1 NEU_NOUN2 POS_VERB3 POS_VERB4	0	5

NEU_NOUN1 NEU_NOUN2 NEU_VERB3 NEU_NOUN4 NEU_NOUN5 NEU_VERB6	0	5
NEG_ADVERB1 POS_VERB2 NEU_ADVERB3	0	5
POS_VERB1 NEG_NOUN2 NEU_VERB3 NEU_VERB4	0	5
POS_NOUN1 NEU_NOUN2 NEU_NOUN3 NEU_VERB4	0	5
NEU_NOUN1 NEU_ADVERB2 NEU_VERB3 NEG_NOUN4	0	5
NEU_VERB1 NEG_VERB2 NOT3 POS_VERB4 NEU_NOUN5	0	5
NEU_VERB1 NEU_ADJECTIVE2 NEU_NOUN3 NEU_VERB4 NEU_ADVERB5 NEU_VERB6	0	5
POS_VERB1 POS_VERB2 NEU_ADVERB3 POS_NOUN4 POS_VERB5 NEU_NOUN6	0	5
NEU_ADJECTIVE1 NEU_NOUN2 NEU_NOUN3 NEU_VERB4 NEG_ADJECTIVE5	0	5
POS_NOUN1 NEU_VERB2 NEU_NOUN3 NEU_NOUN4 NEU_VERB5	0	5
NEU_VERB1 NOT2 POS_VERB3 NEU_VERB4 NEU_VERB5	0	5
NEU_NOUN1 POS_ADJECTIVE2 NEG_NOUN3 NEU_VERB4 NEG_ADJECTIVE5	0	5
NEU_NOUN1 NEU_VERB2 POS_NOUN3 NEG_ADJECTIVE4	0	5
NEU_NOUN1 NEU_VERB2 POS_ADJECTIVE3 NEG_VERB4	0	5
NEU_ADVERB1 NEU_VERB2 NOT3 NEU_VERB4	0	5
NEU_VERB1 POS_ADVERB2 NEG_ADJECTIVE3 NEU_NOUN4 NEU_VERB5 NEU_ADVERB6 NEU_VERB7	0	5
POS_NOUN1 NEU_VERB2 NEG_ADJECTIVE3 NEU_NOUN4 NEU_VERB5 NEG_ADJECTIVE6	0	5
NEG_ADJECTIVE1 POS_ADJECTIVE2 POS_NOUN3	0	5
NEU_VERB1 NEU_VERB2 NEG_VERB3 NEU_NOUN4 NEU_NOUN5	0	5
NEU_NOUN1 NEG_NOUN2 NEU_NOUN3 NEU_VERB4 NEG_ADJECTIVE5	0	5
NEG_ADJECTIVE1 NEU_NOUN2 NEU_VERB3 NEU_ADVERB4 NEU_VERB5	0	5
NEU_NOUN1 NEG_ADJECTIVE2 NEU_NOUN3 NEU_NOUN4 NEU_NOUN5	0	5
POS_VERB1 POS_ADVERB2 POS_ADJECTIVE3	0	5
NEU_NOUN1 NEU_NOUN2 POS_VERB3 NEU_VERB4 POS_ADJECTIVE5	0	5
NEU_NOUN1 NEU_NOUN2 POS_NOUN3 NEU_VERB4 POS_ADVERB5 NEG_ADJECTIVE6	0	5
NEU_VERB1 NEG_VERB2 NEU_VERB3	0	5
NOT1 POS_ADVERB2 POS_VERB3 NEU_NOUN4	0	5
NEU_VERB1 NOT2 NEU_NOUN3 POS_VERB4	0	5
POS_NOUN1 NEU_VERB2 NEU_VERB3 NEU_NOUN4 NEU_NOUN5	5	0
NOT1 POS_VERB2 NEU_ADJECTIVE3	5	0

NEU_NOUN1 NEU_VERB2 NEU_NOUN3 NEU_VERB4 NEU_NOUN5 NEU_NOUN6 NEU_NOUN7	5	0
NEG_NOUN1 NEU_VERB2 NEG_NOUN3 NEU_NOUN4 NEU_VERB5 NEG_NOUN6 NEU_NOUN7 POS_NOUN8 POS_NOUN9 NEU_NOUN10 NEU_VERB11 NEU_VERB12 POS_VERB13 POS_ADJECTIVE14 NEU_NOUN15	5	0
NEU_ADJECTIVE1 POS_NOUN2 POS_ADJECTIVE3 NEU_NOUN4	5	0
NEU_VERB1 POS_ADJECTIVE2 NEU_NOUN3 NEU_NOUN4 NEU_VERB5 NEU_VERB6	5	0
NEU_NOUN1 NEU_NOUN2 NEU_VERB3 POS_ADJECTIVE4 POS_NOUN5 POS_NOUN6	5	0
NEU_NOUN1 NEU_NOUN2 NEU_NOUN3 NEU_NOUN4 NEU_NOUN5 POS_NOUN6	5	0
NEU_VERB1 NEU_NOUN2 NEU_NOUN3 POS_ADVERB4 POS_ADVERB5	5	0
NEU_VERB1 NEU_NOUN2 NEU_NOUN3 NEU_NOUN4 NEU_NOUN5 NEU_NOUN6 NEU_NOUN7 NEU_NOUN8 NEU_NOUN9 NEU_NOUN10	5	0
NEU_ADVERB1 NEU_NOUN2 NEU_NOUN3 NEU_NOUN4 NEU_NOUN5 NEU_NOUN6	5	0
NEU_VERB1 NEG_NOUN2 POS_ADJECTIVE3	5	0
NEU_VERB1 NEU_ADVERB2 NEG_NOUN3 NEU_NOUN4	5	0
NEU_NOUN1 NEU_NOUN2 NEU_VERB3 POS_ADVERB4 POS_ADJECTIVE5 NEU_NOUN6 NEU_NOUN7 NEU_NOUN8	5	0
NEU_VERB1 POS_ADJECTIVE2 POS_ADJECTIVE3 NEU_NOUN4 NEU_NOUN5 NEU_NOUN6	5	0
NEU_NOUN1 NEU_VERB2 POS_ADJECTIVE3 NEG_ADVERB4 POS_ADVERB5	5	0
NEU_VERB1 POS_ADJECTIVE2 NEU_NOUN3 NEU_NOUN4 NEU_ADVERB5 NEU_VERB6	5	0
NEU_NOUN1 NEU_NOUN2 NEU_VERB3 POS_ADJECTIVE4 NEG_ADJECTIVE5 NEU_NOUN6 NEU_NOUN7	5	0
NEU_VERB1 NOT2 POS_NOUN3 NEU_NOUN4	5	0
NEU_VERB1 POS_ADJECTIVE2 NEU_VERB3 POS_NOUN4	5	0
NEU_VERB1 POS_ADJECTIVE2 NEU_NOUN3 NEG_ADVERB4	5	0

Βιβλιογραφία

[1] Aisopos F., Papadakis G., Tserpes K., Varvarigou T., Textual and contextual patterns for sentiment analysis over microblogs, in: Proceedings of the 21st international conference companion on World Wide Web, pp. 453-454. ACM, 2012.

[2] Aisopos F., Papadakis G., Varvarigou T., Sentiment analysis of social media content using n-gram graphs. In Proceedings of the 3rd ACM SIGMM international workshop on Social media, pp. 9-14. ACM, 2011.

[3] Andreevskaia A., Bergler S., Mining WordNet for Fuzzy Sentiment: Sentiment Tag Extraction from WordNet Glosses, in: Proceedings EACL-06, the 11rd Conference of the European Chapter of the Association for Computational Linguistics, January 2006.

[4] Augustyniak L., Kajdanowicz T., Kazienko P., Kulisiewicz M., Tuliglowicz W., An Approach to Sentiment Analysis of Movie Reviews: Lexicon Based vs. Classification, 2014, in: Polycarpou M., de Carvalho A.C.P.L.F., Pan J. S., Woźniak M., Quintian H., Corchado E. (eds), Hybrid Artificial Intelligence Systems. HAIS 2014. Lecture Notes in Computer Science, vol 8480. Springer, Cham.

[5] Baccianella A. E., Sebastiani S., Sebastiani F., SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining, in: Proceedings of LREC, volume 10, pp. 2200-2204, 2010.

[6] Blank A., Why do new meanings occur? A cognitive typology of the motivations for lexical Semantic change, in: Blank A., Koch P., Historical Semantics and Cognition, Berlin/New York: Mouton de Gruyter, pp. 61-90, 1999.

[7] Chapman S., Key Ideas in Linguistics and the Philosophy of Language, Edinburgh University Press, 2009.

[8] Giannakopoulos G., Karkaletsis V., N-gram graphs: Representing documents and document sets in summary system evaluation, in: Proceedings of Text Analysis Conference TAC2009 (To appear), 2009.

[9] Giannakopoulos G., Karkaletsis V., Vouros G., Stamatopoulos P., Summarization system evaluation revisited: N-gram graphs. ACM Trans. Speech Lang. Process. 5. 1-39, 2008.

[10] Godfrey J., Holliman E., Switchboard-1 Release 2 LDC97S62. Web Download. Philadelphia: Linguistic Data Consortium, 1993.

[11] Grzega J., Schöner M., English and general historical lexicology: materials for onomasiology seminars, Onomasiology Online Monographs vol. 1, Katholische Universität Eichstätt-Ingolstadt, Germany, July 2007.

[12] Hatzivassiloglou V., McKeown K. B., Predicting the Semantic Orientation of Adjectives, in: 35th ACL, pp. 174-181, 1997.

[13] Hatzivassiloglou V., McKeown K., Predicting the semantic orientation of adjectives, in: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, pp. 174-181

[14] He Y., Zhou D., Self-training from labeled features for sentiment analysis, Inf Process Manage, 47, pp. 606-616, 2011.

[15] Hu Y., Li W., Document sentiment classification by exploring description model of topical terms, Computer Speech and Language 25 (2011) 386-403.

[16] Isaac M., Twitter to Test Doubling Tweet Length to 280 Characters, New York Times, 26 September 2017 [Online]. Available at: https://www.nytimes.com/2017/09/26/ technology/twitter-280-characters.html. [Accessed: October 2017].

[17] Jurafsky D., Martin J. H., Speech and Language Processing, 3rd ed. (draft), 2017.

[18] Kantola J., Karwowski W., "Knowledge Service Engineering Handbook," CRC Press, Business & Economics, pp. 599, 2016.

[19] Kim S.-M., Hovy E., Determining the sentiment of opinions, in Proceedings of the 20th international conference on Computational Linguistics (COLING-2004), pp. 1367-1373, Geneva, Switzerland, 2004.

[20] Kreutzer J., Witte N., Opinion Mining Using SentiWordNet, Semantic Analysis, Uppsala University. 2013/14.

[21] Kučera H., Francis W. N., Computational analysis of present-day American English, Brown University Press, Providence, 1967.

[22] Kumar P. K., Nandagopalan S., Insights to Problems, Research Trend and Progress in Techniques of Sentiment Analysis, International Journal of Electrical and Computer Engineering (IJECE), Vol. 7, No. 5, October 2017, pp. 2818-2822.

[23] Liberman M., Text on Tap: the ACL/DCI, in: Proceedings of the 1989 DARPA Speech and Natural Language Workshop, Cape Cod, Massachusetts, 1989.

[24] Liu B., Sentiment Analysis and Opinion Mining, Morgan & Claypool Publishers, May 2012.

[25] Maas A. L., Daly R. E., Pham P. T., Huang D., Ng A. Y., Potts C., Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011), 2011.

[26] Malandrakis N., Kazemzadeh A., Potamianos A., Narayanan S., SAIL: A hybrid approach to sentiment analysis, Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), pp. 438-442, 2013.

[27] Medhat W., Hassan A., Korashyb H., Sentiment analysis algorithms and applications: A survey, Ain Shams Eng J (2014), Volume 5, Issue 4, pp. 1093-1113.

[28] Miller G. A., WordNet: a lexical database for english. Communications of the ACM, 38(11):39-41, 1995.

[29] Musto C., Semeraro G., Polignano M., A comparison of Lexicon-based approaches for Sentiment Analysis of microblog posts, Proceedings of the 8th International Workshop on Information Filtering and Retrieval co-located with XIII AI*IA Symposium on Artificial Intelligence (AI*IA 2014), pp. 59-68, 2014.

[30] Myers A., Chris Manning: How computers are learning to understand language, Stanford Engineering, 22 May 2017 [Online]. Available at: https://engineering. stanford.edu/magazine/article/chris-manning-how-computers-are-learning-understand-language. [Accessed October 2017].

[31] Pang B., Lee L., A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume, pp. 271- 278, 2004.

[32] Pang B., Lee L., Opinion Mining and Sentiment Analysis. Found. Trends Inf. Retr. 2, 1-2 (January 2008), 1-135.

[33] Pang B., Lee L., Vaithyanathan S., Thumbs up? Sentiment classification using machine learning techniques. In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, pp. 79-86, 2002.

[34] Raychev V., Nakov P., Language-independent sentiment analysis using subjectivity and positional information, in: Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP), pp. 360-364, 2009.

[35] Rish I., An empirical study of the naive Bayes classifier, in: IJCAI 2001 workshop on empirical methods in artificial intelligence, pp. 41-46, 2001.

[36] Santorini B., Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision), July 1990.

[37] Sasikala P., Mary Immaculate Sheela L., Comparative Study of Sentiment Analysis Techniques in Web, International Journal of Scientific & Engineering Research Volume 8, Issue 5, pp. 125-129, May-2017.

[38] Subasic P., Huettner A., Affect Analysis of Text Using Fuzzy Typing. IEEE-FS, 9:483-496, 2001.

[39] Toutanova K., Klein D., Manning C., Singer Y., Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network, in: Proceedings of HLT-NAACL 2003, pp. 252-259.

[40] Toutanova K., Manning C., Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger, in: Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), Hong Kong, 2000.

[41] Turney P., Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 417-424.

[42] Turney P., Littman M., Unsupervised learning of semantic orientation from a hundred-billion-word corpus, Technical Report ERC-1094 (NRC 44929), National Research Council of Canada, 2002.

[43] Violos J., Tserpes K., Psomakelis E., Psychas K., Varvarigou T., Sentiment Analysis using Word-Graphs, in: Proceedings of the 6th International Conference on Web Intelligence, Mining and Semantics (WIMS '16), Article No. 22, June 2016.

[44] Yu L. C., Wu J. L., Chang P. C., Chu H. S., Using a contextual entropy model to expand emotion words and their intensity for the sentiment classification of stock market news, Knowledge-Based Systems, Volume 41, pp. 89-97, March 2013.

[45] Κιούρτη Π., Ανάλυση συναισθήματος με χρήση υβριδικών n-grams, Διπλωματική Εργασία, Εθνικό Μετσόβιο Πολυτεχνείο, Οκτώβριος 2015.