Εθνικο Μετσοβιο Πολυτεχνειο

Τμημα Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων Πληροφορικης

Εργαστηριο Κατανεμημενων Συστηματων και Διαχειρισης Γνωσης

Ανάλυση Συναισθήματος με χρήση υβριδικών n-grams

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΠΑΝΑΓΙΩΤΑΣ Β. ΚΙΟΥΡΤΗ

Επιβλέπων: Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Οκτώβριος 2015

Εθνικο Μετσοβιο Πολυτεχνειο
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΚΑΤΑΝΕΜΗΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗΣ ΓΝΩΣΗΣ

Ανάλυση Συναισθήματος με χρήση υβριδικών n-grams

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΠΑΝΑΓΙΩΤΑΣ Β. ΚΙΟΥΡΤΗ

Επιβλέπων: Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23η Οκτωβρίου 2015.

.....	.....	.....
Θεοδώρα Βαρβαρίγου Καθηγήτρια Ε.Μ.Π.	Εμμανουήλ Βαρβαρίγος Καθηγητής Ε.Μ.Π.	Βασίλειος Λούμος Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2015.

.....
ΠΑΝΑΓΙΩΤΑ Β. ΚΙΟΥΡΤΗ
Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την καθηγήτρια Θεοδώρα Βαρβαρίγου για την δυνατότητα που μου πρόσφερε να ασχοληθώ με τον τομέα του Sentiment Analysis και να ερευνήσω πάνω σε αυτόν στο Εργαστήριο Κατανεμημένων Συστημάτων και Διαχείρισης Γνώσης.

Επίσης, θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα Βρεττό Μουλό για τις πολύτιμες συμβουλές του, την εμπιστοσύνη που μου έδειξε και τον χρόνο που αφιέρωσε για την εκπόνηση αυτής της διπλωματικής εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω τους ανθρώπους που ήταν δίπλα μου κατά την διάρκεια των σπουδών μου.

Περίληψη

Η ανάλυση συναισθήματος είναι ένας τομέας που αναπτύχθηκε από την ανάγκη αναγνώρισης του συναισθήματος της κοινής γνώμης πάνω σε ένα θέμα μέσα από τον μεγάλο όγκο πληροφορίας ελεύθερης μορφής που διατίθεται στο διαδίκτυο. Οι μέθοδοι ανάλυσης συναισθήματος περιλαμβάνουν μεταξύ άλλων και στατιστικές προσεγγίσεις με την χρήση αλγορίθμων μηχανικής μάθησης. Η εκπαίδευση των αλγορίθμων μηχανικής μάθησης χρησιμοποιώντας χαρακτηριστικά που εξάγονται από τα κείμενα παρέχουν ένα μοντέλο για την πρόβλεψη της πολικότητας νέων κειμένων.

Στην εργασία αυτή μελετάμε την ανάλυση συναισθήματος σε δεδομένα κριτικής ταινιών με χρήση μηχανικής μάθησης και εξετάζουμε πώς η ακρίβεια ταξινόμησης της πολικότητας των κειμένων εξαρτάται από τον τρόπο με τον οποίο τα χαρακτηριστικά που εξάγουμε από τα κείμενα εκμεταλλεύονται τη δομή του. Μελετάμε μία μέθοδο που εξάγει χαρακτηριστικά χρησιμοποιώντας την σύνθεση του κειμένου με διαφορετικό τρόπο από ότι μία πιο κλασική μέθοδος. Συγκρίνουμε τις δυο μεθόδους και τις συνδυάζουμε για την δημιουργία μίας νέας προσέγγισης στον τομέα αυτό. Αξιολογούμε την σχετική επίδοση των τριών μοντέλων και την επίδραση παραμέτρων όπως το πλήθος των δεδομένων, ο αλγόριθμος μάθησης αλλά και η ύπαρξη ή όχι προεπεξεργασίας των κειμένων.

Λέξεις Κλειδιά: Ανάλυση Συναισθήματος, n-gram γράφοι, γράφοι λέξεων, αναπαράσταση bag of words, ταξινόμηση πολικότητας κειμένου, Επιβλεπόμενη Μηχανική Μάθηση, απλοϊκό μοντέλο Bayes

Abstract

The field of sentiment analysis was developed to accommodate the need for recognizing the sentiment of public opinion on a subject that exists in the big unstructured data available on the internet. Some methods of sentiment analysis include statistical approaches using machine learning algorithms. The training of machine learning algorithms using features extracted from the texts provides a model for predicting the polarity of new texts.

We study sentiment analysis in movie reviews through a machine learning approach, and examine how the accuracy of classification depends on the way features exploit the word structure of the text. We consider a model of feature extraction utilizing the composition of the text in a different way than the usual, baseline model. We compare the two models to each other, and additionally to a third, midway approach that combines them. We evaluate their relative performance, and dependence on parameters such as training data size, machine learning algorithm and pre-processing of the input text.

Keywords: Sentiment Analysis, n-gram graph, word graph, bag of words model, polarity classification, Supervised Machine Learning, naive Bayes model

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Ανάλυση Συναισθήματος

Η ανάλυση συναισθήματος είναι ένας τομέας που δημιουργήθηκε λόγω της αύξησης του όγκου αδόμητης πληροφορίας που διατίθεται στο διαδίκτυο από τους χρήστες του. Η ανάγκη επεξεργασίας του συναισθήματος των δεδομένων αφορά τους απλούς χρήστες του διαδικτύου αλλά και τον χώρο της επιστήμης και της βιομηχανίας.

1.1.1 Ορισμός

Η ανάλυση συναισθήματος αποτελεί τη διαδικασία ανίχνευσης της πολικότητας της πληροφορίας που προβάλλεται σε ένα κείμενο ελεύθερης μορφής και στην συνέχεια, η ταξινόμησή της σε θετική, αρνητική ή ουδέτρη. Οι πολικότητες αυτές, συνήθως, αφορούν ένα θέμα και χρησιμοποιούνται για την ανάλυση συναισθήματος των χρηστών πάνω σε αυτό το θέμα. Η πληροφορία του κειμένου όμως, μπορεί να περιλαμβάνει μία ή περισσότερες απόψεις ενός χρήστη σχετικά με κάποια θέματα. Η εξαγωγή απόψεων από το κείμενο πολλές φορές προηγείται, ώστε στην συνέχεια, να κατηγοριοποιηθεί η πολικότητα της κάθε άποψης. Έτσι, μπορεί να έχουμε ανάλυση συναισθήματος σε ολόκληρο έγγραφο, σε παραγράφους, σε προτάσεις, ακόμα και σε χαρακτηριστικά κειμένου. Η διαδικασία εξαγωγής των απόψεων ενός κειμένου ονομάζεται opinion mining. Οι δυο όροι, sentiment analysis και opinion mining, εμφανίστηκαν σχεδόν παράλληλα και χρησιμοποιούνται για να αναφέρουν τον ίδιο σκοπό της ανάλυσης συναισθήματος [23]. Επομένως, η ταξινόμηση του κειμένου σε θετικής, αρνητικής ή ουδέτερης πολικότητας μπορεί να θεωρηθεί ως ένα συγκεκριμένο είδος κατηγοριοποίησης κειμένου, με μεγαλύτερη δυσκολία από ότι η κατηγοριοποίηση κειμένων με βάση το θέμα τους (topic categorization) ([24]).

1.1.2 Εφαρμογές

Η ανάλυση συναισθήματος εφαρμόζεται σε πολλές περιοχές όπου η εξαγωγή συμπεράσματος για την κοινή γνώμη σε ένα θέμα με χειροκίνητο τρόπο από τον άνθρωπο είναι

σχεδόν αδύνατη λόγω του μεγάλου όγκου δεδομένων. Οι περισσότερες από αυτές έχουν να κάνουν με κείμενα ελεύθερης μορφής απλών χρηστών που διατίθονται στο διαδίκτυο. Κάποιες από τις εφαρμογές δίνονται παρακάτω. ([23])

- Η ανάλυση συναισθήματος μπορεί να χρησιμοποιηθεί σε μηχανή αναζήτησης απόψεων ή κριτικών για να συγκεντρώσει την πολικότητα της κοινής γνώμης για ένα θέμα ή προϊόν σε ένα site [14].
- Η ανάλυση συναισθήματος μπορεί να αποτελέσει μέρος της τεχνολογίας άλλων συστημάτων, όπως recommender systems, όπου σε αυτήν την περίπτωση συμβάλλει στο να αποφεύγονται οι συστάσεις σε προϊόντα που δέχονται, κυρίως, αρνητικό feedback.
- Κατά την παρουσίαση online διαφημίσεων η ανάλυση συναισθήματος μπορεί να χρησιμοποιηθεί ώστε να απορρίψουμε ένα μέρος τους ή να προωθήσουμε ένα άλλο ανάλογα με το περιεχόμενο του site στο οποίο εμφανίζονται και το συναίσθημα που προκαλούν στο χρήστη.
- Στον τομέα της εξαγωγής πληροφορίας (information extraction) έχει παρατηρηθεί βελτίωση όταν αγνοούνται πληροφορίες που εξάγονται από υποκειμενικές απόψεις, η αναγνώριση της υποκειμενικότητας των οποίων είναι ένα από τα είδη της ανάλυσης συναισθήματος (subjectivity/objectivity identification).
- Ο πιο σημαντικός τομέας στον οποίο χρησιμοποιείται η ανάλυση συναισθήματος είναι ο τομέας των επιχειρήσεων, όπου χρειάζεται άμεση ανάδραση για ένα προϊον ή υπηρεσία από τους αγοραστές του. Έτσι, η επιχείρηση προσαρμόζει την γραμμή της ανάλογα με το συναίσθημα και τις επιθυμίες της αγοράς, το οποίο είναι σημαντικό για τα κέρδη της.

1.1.3 Προσεγγίσεις

Για την ανάλυση συναισθήματος μπορεί να χρησιμοποιηθούν πολλές μέθοδοι, κάποιες από τις οποίες θα παρουσιαστούν στην ενότητα 1.3. Οι μέθοδοι αυτές κατατάσσονται σε μία από τις 4 κατηγορίες ([5]):

- keyword spotting
- lexical affinity
- statistical methods
- concept-based approaches

1.1.3.1 Keyword spotting

Η κατηγορία αυτή αποτελεί την πιο εύκολη και προσιτή προσέγγιση για την επίλυση προβλημάτων ανάλυσης συναισθήματος. Οι μέθοδοι που κατατάσσονται σε αυτήν χρησιμοποιούν κάποιες λέξεις ενδεικτικές για την αναγνώριση του συναισθήματος που ονομάζονται

affect words. Για παράδειγμα, ελέγχεται η παρουσία λέξεων όπως happy, sad, amused, angry κτλ. Η προσέγγιση αυτή θεωρείται αφελής, επειδή στηρίζεται μόνο σε λέξεις που εκφράζουν ξεκάθαρα ένα συναίσθημα. Η περιοχή της ανάλυσης συναισθήματος έχει να κάνει και με κείμενα όπου το συναίσθημα εκφράζεται μέσα από τα συμφραζόμενα χωρίς την χρήση συναισθηματικών λέξεων (affect words). Για παράδειγμα, υπάρχουν κείμενα που δεν χρησιμοποιούν τις λέξεις αυτές και παρόλα αυτά εκφράζουν έντονα συναισθήματα, όπως η περιγραφή μίας ιστορίας. Επιπλέον, η μέθοδος αυτή, δεν μπορεί να χειριστεί την άρνηση σε προτάσεις όπως 'Today I'm not happy', όπου η λέξη happy θα οδηγήσει την ταξινομήση σε θετικό συναίσθημα.

1.1.3.2 Lexical affinity

Οι μέθοδοι που εντάσσονται στην κατηγορία lexical affinity αναθέτουν τιμές πιθανότητας σε λέξεις για να δηλώσουν την συγγένειά τους με θετικό ή αρνητικό συναίσθημα. Επιπλέον, χρησιμοποιούν λέξεις ενδεικτικές για το συναίσθημα δηλαδή χρησιμοποιούν affect words, όπως στην προηγούμενη ενότητα. Έτσι, αναθέτοντας σε μία λέξη μεγάλη πιθανότητα να σχετίζεται με ένα συναίσθημα, μπορούν να αναγνωρίσουν το συναίσθημα κάποιων προτάσεων. Η ανάθεση πιθανοτήτων γίνεται εκπαιδεύοντας την μέθοδο από γλωσσικά κείμενα. Αυτές οι μέθοδοι ξεπερνούν σε επίδοση τις μεθόδους της κατηγορίας keyword spotting, αλλά υπάρχουν δυο περιπτώσεις στις οποίες αποτυγχάνουν. Πολλές φορές χρησιμοποιούνται φράσεις που ανάλογα με την τοποθέτησή τους σε μία πρόταση δηλώνουν διαφορετικό συναίσθημα ([5]). Σε αυτήν την περίπτωση η ανάθεση πιθανότητας εξαρτάται από τα γλωσσικά κείμενα που έχουν εκπαιδεύσει την μέθοδο. Επομένως, η μέθοδος είναι 'προκατειλημμένη' για την εμφάνιση μίας λέξης με τον ίδιο τρόπο όπως έχει αναγνωρίσει στα γλωσσικά κείμενα εκπαίδευσής της. Επιπλέον, προτάσεις που δηλώνουν άρνηση σε συνδυασμό με λέξεις που επηρεάζουν την πολικότητα (affect words) δυσκολεύουν και σε αυτήν την κατηγορία την επιτυχία των μεθόδων.

1.1.3.3 Statistical methods

Η προσέγγιση των στατιστικών μεθόδων περιλαμβάνει Bayesian συμπερασμό και support vector machines και είναι αρκετά δημοφιλής. Σε αυτήν χρησιμοποιούνται αλγόριθμοι μηχανικής μάθησης (machine learning) σε συνδυασμό με κείμενα που έχουν ταξινομηθεί χειροκίνητα σε κάποιο συναίσθημα, με σκοπό να εκπαιδεύσουν μία μηχανή να αναγνωρίζει το συναίσθημα νέων κειμένων. Η μέθοδος μαθαίνει μέσα από τα χαρακτηριστικά κειμένου που εξάγονται κάθε φορά για το σκοπό αυτό, για τα οποία γνωρίζει την πολικότητά τους. Οι στατιστικές αυτές μέθοδοι λειτουργούν καλά όταν εκπαιδεύονται σε μεγάλα κείμενα, δηλαδή ολόκληρα έγγραφα. Αντίθετα, έχουν μικρή επίδοση σε μικρότερα κείμενα λόγω του περιορισμού των λέξεων που χρησιμοποιούνται.

1.1.3.4 Concept-based approaches

Οι μέθοδοι αυτές βασίζονται σε τρόπους ανίχνευσης των εννοιών των κειμένων χρησιμοποιώντας σημασιολογικά δίκτυα και οντολογίες ιστού για να επιτύχουν τη σημασιολογική

ανάλυση των κειμένων. Για το σκοπό αυτό, αγνοούνται λέξεις κλειδιά ή συντακτική ανάλυση και χρησιμοποιούνται μεγάλες βάσεις σημασιολογικής γνώσης για να ανιχνεύσουν την εννοιολογική πληροφορία που παράγεται από τη χρήση φυσικής γλώσσας. Έτσι, επιτυγχάνεται η ανίχνευση συναισθημάτων σε κείμενα που δεν χρησιμοποιούν affect words και εκφράζουν το συναίσθημα μέσα από το εννοιολογικό περιεχόμενο. Οι βάσεις που χρησιμοποιούνται πρέπει να είναι αντιπροσωπευτικές για την εμβάθυνση στις έννοιες, καθώς καθορίζουν την επιτυχία των μεθόδων αυτών.

1.2 Το πρόβλημα με το οποίο ασχολείται η εργασία αυτή

Στην εργασία αυτή, εφαρμόζουμε ανάλυση συναισθήματος σε κριτικές ταινιών της βάσης IMDb[1]. Κάθε κριτική αποτελεί ένα έγγραφο (document) μίας παραγράφου που αναφέρεται σε μία ταινία. Επομένως, θεωρούμε ότι οι κριτικές που χρησιμοποιούνται εκφράζουν μία άποψη για την οποία επιθυμούμε να ταξινομήσουμε την πολικότητά της και η ανάλυση συναισθήματος γίνεται σε επίπεδο εγγράφου δηλαδή σε ολόκληρο το κείμενο κριτικής (review). Σκοπός μας είναι η μελέτη επίδοσης μία μεθόδου εξαγωγής χαρακτηριστικών κειμένου σε σχέση με μία baseline μέθοδο εξαγωγής χαρακτηριστικών. Τα χαρακτηριστικά αυτά χρησιμοποιούνται ως είσοδο σε έναν αλγόριθμο μηχανικής μάθησης που συμβάλλει στην κατηγοριοποίηση του συναισθήματος του κειμένου, όπως παρουσιάζεται στην ενότητα της μηχανικής μάθησης (ενότητα 2.1). Επομένως, στην εργασία αυτή ασχολούμαστε με μεθόδους που εντάσσονται στην κατηγορία των στατιστικών μεθόδων ανάλυσης συναισθήματος.

1.3 Σχετικές ερευνητικές εργασίες

Στην ενότητα αυτή παρουσιάζονται παλιότερες ερευνητικές εργασίες που σχετίζονται με ανάλυση συναισθήματος σε κριτικές προϊόντων ή υπηρεσιών (όπως ταινίες) από χρήστες.

Στην εργασία που παρουσιάζεται στο [24], χρησιμοποιούνται τρόποι εξαγωγής χαρακτηριστικών κειμένου και μετέπειτα εκπαίδευση μίας μηχανής μέσω αλγορίθμων μηχανικής μάθησης από τα χαρακτηριστικά αυτά, για την αναγνώριση του συναισθήματος. Τα δεδομένα που χρησιμοποιήθηκαν ήταν κριτικές ταινιών της βάσης IMDb. Αρχικά παρουσιάζεται η καλύτερη επίδοση των 'μηχανικών' μοντέλων σχετικά με 'ανθρώπινα' μοντέλα εξαγωγής χαρακτηριστικών. Συγκεκριμένα, χρησιμοποιείται ένα μοντέλο που εξάγει μέσω προγράμματος λέξεις από τα κείμενα για την χρήση τους ως χαρακτηριστικά κειμένου, και συγκρίνεται με την περίπτωση όπου οι λέξεις επιλέχθηκαν χειροκίνητα από ανθρώπους. Μετά την διαπίστωση της ανάγκης χρήσης μηχανικών μοντέλων, μελετάται η προσέγγιση bag-of-features για την δημιουργία του διανύσματος χαρακτηριστικών (feature vector) που χρησιμοποιείται για την εκπαίδευση τριών αλγορίθμων μάθησης. Παρακάτω δίνουμε τις 8 περιπτώσεις χαρακτηριστικών που χρησιμοποιήθηκαν κάθε φορά για την δημιουργία του διάνυσματος.

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

1. unigrams, όπου κάθε χαρακτηριστικό αντιστοιχεί στην συχνότητα εμφάνισης μίας λέξης.

2. unigrams, όπου καθε χαρακτηριστικό δηλώνει την παρουσία ή όχι μίας λέξης.

3. bigrams, όπου κάθε χαρακτηριστικό αντιστοιχεί στην παρουσία ή όχι δυο γειτονικών λέξεων.

4. unigrams και bigrams, όπου ένα χαρακτηριστικό αντιστοιχεί σε ένα ζεύγος που περιέχει τα δυο προηγούμενα χαρατηριστικά.

5. unigrams με Part-Of-Speech tagging, όπου εμπλουτίζονται οι λέξεις με 'ετικέτες' που δηλώνουν τι μέρος του λόγου είναι κάθε φορά στην πρόταση.

6. adjectives, όπου κάθε χαρακτηριστικό είναι ένας επιθετικός προσδιορισμός.

7. top 2633 unigrams, όπου χρησιμοποιούνται μόνο οι 2633 πιο συχνές λέξεις ως χαρακτηριστικά τα οποία αντιστοιχούν σε δήλωση παρουσίας ή όχι της λέξης στο κείμενο.

8. unigrams με position όπου σε κάθε λέξη προστίθεται μία 'ετικέτα' η οποία δηλώνει σε ποιο τέταρτο του κειμένου εμφανίστηκε.

Κάθε μία από τις παραπάνω περιπτώσεις χρησιμοποιήθηκε για να εκπαιδεύσει τους αλγορίθμους μάθησης naive Bayes, maximum entropy classification και support vector machines σε κάθε πείραμα. Και οι 8 περιπτώσεις εξαγωγής χαρακτηριστικών αποδείχθηκαν καλύτερες από την χειροκίνητη εξαγωγή χαρακτηριστικών. Και οι 3 αλγόριθμοι παρουσιάζουν μεγάλη επίδοση με την μεγαλύτερη επίδοση να παρουσιάζεται όταν χρησιμοποιείται η παρουσία λέξεων (unigrams) ως χαρακτηριστικά. Οι συγγραφείς της εργασίας αυτής μελέτησαν πιο θεωρητικά το πρόβλημα πετυχαίνοντας καλύτερα ποσοστά ακρίβειας σε επόμενες ερευνητικές εργασίες ([21], [22]).

Η εργασία που παρουσιάζεται στο [12] ασχολείται με την δημιουργία ενός συστήματος 'περίληψης' των απόψεων για κάποια συγκεκριμένα χαρακτηριστικά ενός προϊόντος από κείμενα κριτικής προϊόντων και η ταξινόμησή τους σε θετικά ή αρνητικά. Σκοπός του συστήματος είναι να ανακαλύψει την άποψη των χρηστών για ένα συγκεκριμένο προϊόν και τα χαρακτηριστικά του. Συγκεκριμένα, το σύστημα χωρίζεται σε δυο μέρη. Στο πρώτο μέρος επιτυγχάνεται η εξαγωγή των χαρακτηριστικών του προϊόντος που σχολιάζονται στις κριτικές των χρηστών. Στο δεύτερο μέρος, ταξινομούνται οι κριτικές που σχετίζονται με κάθε χαρακτηριστικό σε θετικές ή αρνητικές, και έτσι διευκολύνεται η αναζήτηση των απόψεων για ένα χαρακτηριστικό του προϊόντος από έναν νέο χρήστη. Στην εργασία τους μελετάται το πρώτο μέρος, ενώ το μέρος της ταξινόμησης παρουσιάζεται σε επόμενο paper ([11]).

Στο πρώτος μέρος, οι κριτικές αρχικά αποθηκεύονται σε μία βάση δεδομένων. Στην συνέχεια, για κάθε review, χρησιμοποιείται part-of-speech tagging σε κάθε λέξη για την αναγνώριση των ουσιαστικών τα οποία είναι πιθανό να αποτελούν τα χαρακτηριστικά του προϊόντος για τα οποία θέλουμε να εξάγουμε απόψεις. Από το σύνολο αυτό, υπολογίζονται τα πιο συχνά εμφανιζόμενα σε προτάσεις χαρακτηριστικά προϊόντων με χρήση κανόνων εξαγωγής συσχετισμού (association rule mining). Για την εξαγωγή των πιο συχνών χαρακτηριστικών προϊόντος χρησιμοποιούνται οι φράσεις με 3 λέξεις το πολύ, καθώς στην

εργασία τους θεωρείται ότι ένα χαρακτηριστικό περιγράφεται με 3 λέξεις το πολύ. Στην συνέχεια τα χαρακτηριστικά αυτά μειώνονται με μία διαδικασία pruning με σκοπό να αφαιρεθούν τα χαρακτηριστικά που αποτελούν υποσύνολο ενός χαρακτηριστικού (π.χ. life αποτελεί υποσύνολο του χαρακτηριστικού battery life ενός προϊόντος και πρέπει να αφαιρεθεί), καθώς και χαρακτηριστικά τα οποία εμφανίζονται με παραπάνω από 2 ουσιαστικά απομακρυνσμένα μέσα σε μία πρόταση ενός μόνο review. Μετά από την εξαγωγή των πιο συχνών χαρακτηριστικών ανιχνεύονται επιθετικοί προσδιορισμοί γειτονικά των χαρακτηριστικών στις προτάσεις οι οποίοι θεωρούνται ότι εκφράζουν μία θετική ή αρνητική άποψη για αυτό. Στην συνέχεια χρησιμοποιούνται τα επίθετα αυτά για ανίχνευση πιο σπάνιων χαρακτηριστικών σε προτάσεις όπου δεν έχουμε ανιχνεύσει χαρακτηριστικά προϊόντος. Και σε αυτήν την περίπτωση θεωρείται ότι ένας επιθετικός προσδιορισμός που εκφράζει θετική ή αρνητική άποψη βρίσκεται γειτονικά του χαρακτηριστικού. Αν και στα πιο σπάνια χαρακτηριστικά τοποθετούνται και χαρακτηριστικά άσχετα με το προϊόν, αυτό δεν αποτελεί πρόβλημα μιας και το μεγαλύτερο ποσοστό χαρακτηριστικών αποτελούν τα συχνά εμφανιζόμενα. Τέλος, για κάθε λέξη - επίθετο που δηλώνει μία άποψη χρησιμοποιείται το WordNet και μία τεχνική bootstrapping για την αναγνώριση της σημασιολογίας της. Έπειτα, κάθε πρόταση κατηγοριοποιείται με το συναίσθημα που κυριαρχεί από αυτά που δηλώνουν οι λέξεις - επίθετα που περιέχει, το οποίο παρουσιάζεται στην εργασία [11].

Η εργασία [25] έχει τον ίδιο σκοπό με την εργασία [12], που παρουσιάζεται στην προηγούμενη παράγραφο, και ονομάζει το σύστημά της OPINE. Σε αυτήν χρησιμοποιείται μία μη επιβλεπόμενη μέθοδο (OPINE's Feature Assessor) για την εξαγωγή ουσιαστικών που αποτελούν τα μέρη ενός προϊόντος. Η μέθοδος αυτή χρησιμοποιεί αναδρομικές κλήσεις για να ανακαλύψει τα μέρη του προϊόντος και τα χαρακτηριστικά του καθώς και τα 'υπο-μέρη' αυτών και τα χαρακτηριστικά τους. Για την εξαγωγή των φράσεων που εκφράζουν άποψη πάνω στα μέρη ενός προϊόντος χρησιμοποιεί 10 κανόνες εξαγωγής που λαμβάνουν υπόψη την συντακτική δομή των προτάσεων, όπως υπολογίζεται από τον MINIPAR parser. Αυτές οι φράσεις είναι πιθανές φράσεις για τη δήλωση μίας άποψης (θετικής ή αρνητικής). Για την σημασιολογική ερμήνευση των φράσεων χρησιμοποιείται μία καινούρια μέθοδος που ανακαλύπτει σημασιολογικές 'ετικέτες' για κάθε φράση που αφορά ένα χαρακτηριστικό - μέρος του προϊόντος και αναφέρεται σε μία πρόταση. Η διαδικασία τοποθέτησης των ετικέτων είναι μία μη επιβλεπόμενη μέθοδος κατηγοριοποίησης που ονομάζεται relaxation labeling. Η μέθοδος αυτή χρησιμοποιεί τις ετικέτες γειτονικών λέξεων, τις εκ των προτέρων πιθανότητες για την ανάθεση μίας ετικέτας σε μία λέξη ή φράση και μία συνάρτηση που δηλώνει την επίδραση των γειτονικών φράσεων σε αυτήν. Έτσι, η μη επιβλεπόμενη μέθοδο ανακαλύπτει τις πιο συχνές ετικέτες και εξάγει από αυτές τις φράσεις στις οποίες έχει ανατεθεί θετική ή αρνητική ως φράσεις απόψεων. Η μέθοδος αυτή επιτυγχάνει μεγαλύτερη ακρίβεια από την μέθοδο που παρουσιάζεται στην προηγούμενη παράγραφο (εργασία [12]). Η επιτυχία της οφείλεται στην αναγνώριση της σημασιολογικής ετικέτας μίας λέξης που οδηγεί στην εξαγωγή των σωστών απόψεων που εκφράζονται μέσα στα συμφραζόμενα.

Η εργασία [27] προσεγγίζει το πρόβλημα ανάλυσης συναισθήματος σε κριτικές ταινιών της βάσης IMDb με τη χρήση του αλγορίθμου μάθησης multinomial naive Bayes. Αρχικά παρουσιάζει την γνωστή προσέγγιση bag of words που αφορά την εκπαίδευση του αλγορίθμου με χρήση απλών χαρακτηριστικών κειμένου, δηλαδή την ένδειξη εμφάνισης (1) ή

όχι (0) λέξεων ενός λεξιλογίου στο κείμενο. Στην συνέχεια, δοκιμάζεται η τοποθέτηση τιμών-βαρών που σχετίζονται με τη θέση εμφάνισης των λέξεων στο κείμενο για κάθε λέξη του λεξιλογίου που εμφανίζεται στο κείμενο. Συγκεκριμένα, δίνεται μεγαλύτερη βαρύτητα στις λέξεις των τελευταίων προτάσεων καθώς στην εργασία αυτή θεωρείται ότι οι απόψεις εκφράζονται στο τέλος του κειμένου. Έτσι, ξεκινώντας από μία θετική τιμή a για την πρώτη λέξη και καταλήγοντας σε μία θετική τιμή $a + q$ για την τελευταία λέξη του κειμένου χρησιμοποιεί την θέση (position) κάθε λέξης ώστε να υπολογίσει τα βάρη των ενδιάμεσων λέξεων με γραμμική παρεμβολή ($a + q \times \frac{position}{size}$, όπου το $size$ αποτελεί το μέγεθος του κειμένου σε λέξεις). Για την υποστήριξη της υπόθεσης ότι οι απόψεις εκφράζονται στο τέλος της πρότασης χρησιμοποιείται ένα φίλτρο για την αναγνώριση των υποκειμενικών προτάσεων και τοποθέτησή τους με φθίνουσα σειρά σχετικά με την υποκειμενικότητα. Το φίλτρο αυτό περιλαμβάνει την εκπαίδευση ενός naive Bayes classifier με ένα υποκειμενικό αρχείο εκπαίδευσης ώστε να αναγνωρίζει σε νέα δεδομένα τις υποκειμενικές προτάσεις. Με την ταξινόμησή τους σε φθίνουσα σειρά ως προς την υποκειμενικότητα επιτυγχάνεται βελτίωση της ακρίβειας από την αρχική baseline μέθοδο.

Στην εργασία [26] παρουσιάζονται συγκεντρωμένα ερευνητικές εργασίες πάνω σε ταξινόμηση της πολικότητας διαφόρων ειδών από κριτικές (reviews), όπου μπορεί να ανατρέξει κανείς για περισσότερες πληροφορίες. Επιπλέον, παρουσιάζεται η υβριδική ταξινόμηση η οποία ορίζεται ως η ταξινόμηση όπου εφαρμόζονται ταξινομητές ο ένας μετά τον άλλον. Για τον σκοπό αυτό παρουσιάζονται μέθοδοι Rule-Based ταξινόμησης όπως, Rule-Based Classifier (RBC), General Inquirer Based Classifier (GIBC), Statistics Based Classifier (SBC) Induction Rule Based Classifier (IRBC) και ο αλγόριθμος μηχανικής μάθησης SVM οι οποίοι συνδυάζονται για την επίτευξη μεγάλων ποσοστών F1-measure πάνω στα δεδομένα.

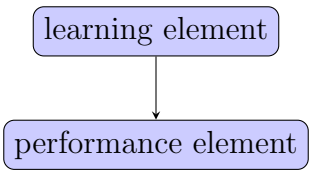
Κεφάλαιο 2

Έννοιες και προσεγγίσεις προβλήματος

Στην εργασία αυτή μελετάμε δυο προσεγγίσεις εξαγωγής χαρακτηριστικών κειμένου (feature extraction). Τα χαρακτηριστικά κειμένου αποτελούν την είσοδο μίας μηχανής για την επιστροφή του συναισθήματος κειμένου, όπως αναλύουμε παρακάτω. Σκοπός μας είναι η μελέτη και σύγκριση της προσέγγισης των n-gram γράφων με μία πιο κλασική μέθοδο εξαγωγής χαρακτηριστικών που χρησιμοποιεί bag of words αναπαράσταση. Οι δυο προσεγγίσεις παρουσιάζονται σε επόμενες ενότητες, ενώ παρακάτω δίνουμε τους βασικούς όρους της μηχανικής μάθησης που χρησιμοποιούνται στην εργασία αυτή.

2.1 Μηχανική Μάθηση

Η διαδικασία της μηχανικής μάθησης αποτελείται από ένα στοιχείο εκτέλεσης (performance element), το οποίο αποφασίζει τι ενέργειες θα κάνει και ένα στοιχείο μάθησης (learning element) που τροποποιεί το στοιχείο εκτέλεσης ώστε να παίρνει καλύτερες αποφάσεις [29] (σχήμα 3.11).



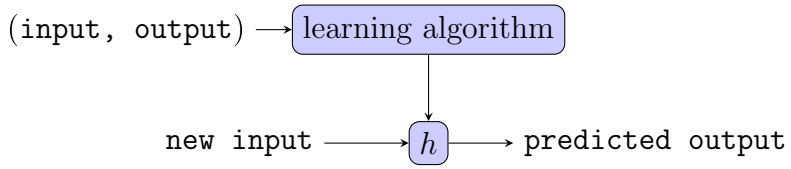
Σχήμα 2.1: Στοιχεία μηχανικής μάθησης

Το στοιχείο μάθησης εκπαιδεύει τις συνιστώσες του στοιχείου εκτέλεσης και η διαδικασία ονομάζεται εκπαίδευση. Η εκπαίδευση επιτυγχάνεται επειδή διατίθεται ανάδραση στο στοιχείο μάθησης με κάποιο τρόπο. Ο τύπος της ανάδρασης προσδιορίζει την περίπτωση μάθησης. Ο τομέας της μηχανικής μάθησης διακρίνει 3 περιπτώσεις μάθησης: *επιβλεπόμενη μάθηση*

(supervised learning), *μη επιβλεπόμενη μάθηση* (unsupervised learning) και *ενισχυτική μάθηση* (reinforcement learning).

2.1.1 Επιβλεπόμενη Μάθηση

Η διαδικασία της επιβλεπόμενης μάθησης περιλαμβάνει την μάθηση μίας συνάρτησης από παραδείγματα εισόδων και εξόδων (input, output). Το σύνολο αυτών ονομάζεται σύνολο εκπαίδευσης (training set). Το στοιχείο εκτέλεσης είναι η συνάρτηση που ονομάζεται συνήθως *hypothesis* και συμβολίζεται με (h).



Σχήμα 2.2: Διαδικασία επιβλεπόμενης μάθησης

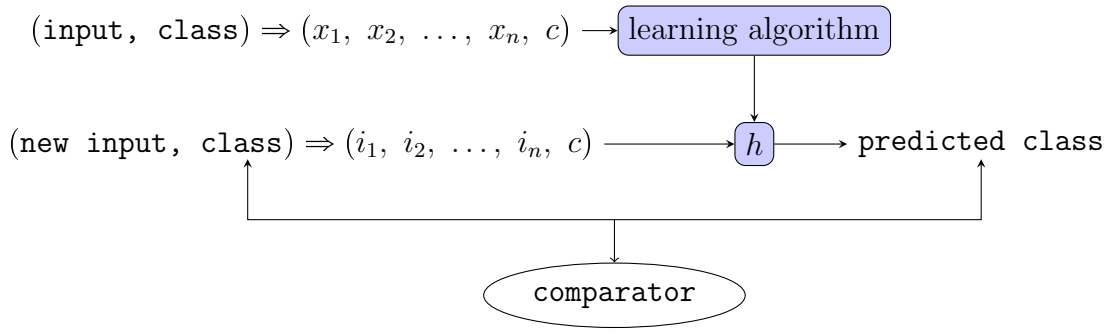
2.1.1.1 Ταξινόμηση (Classification)

Όπως είδαμε, στην επιβλεπόμενη μάθηση δίνουμε στην μηχανή ένα αντικείμενο ή μία κατάσταση (input) και την αναμενόμενη έξοδο που θέλουμε να επιστραφεί (output). Το αντικείμενο ή η κατάσταση περιγράφεται από ένα σύνολο χαρακτηριστικών (features ή attributes). Το σύνολο χαρακτηριστικών είναι ένα διάνυσμα που ονομάζεται διάνυσμα χαρακτηριστικών (feature vector) και αποτελείται συνήθως από αριθμούς [4], όπως φαίνεται στο σχήμα 2.3. Τα χαρακτηριστικά εισόδου μπορούν να παίρνουν είτε διακριτές είτε συνεχείς τιμές. Η έξοδος μπορεί, επίσης, να παίρνει είτε διακριτές είτε συνεχείς τιμές. Η μάθηση μίας συνάρτησης h με διακριτές τιμές εξόδου ονομάζεται μάθηση ταξινόμησης (classification). Οι διακριτές τιμές εξόδου ονομάζονται *κλάσεις*.

Εκπαίδευση μοντέλου Μία είσοδος παρέχεται κατά την εκπαίδευση με το διάνυσμα χαρακτηριστικών μαζί με την κλάση στην οποία ανήκει. Έτσι, έχουμε ένα επιπλέον χαρακτηριστικό στο διάνυσμα, το χαρακτηριστικό κλάσης (class attribute) που αποτελεί την αναμενόμενη έξοδο που επιθυμούμε. Το σύνολο των χαρακτηριστικών εισόδου μαζί με το χαρακτηριστικό κλάσης για μία δεδομένη είσοδο ονομάζεται training instance (ενδεχόμενο εκπαίδευσης). Για παράδειγμα, γνωρίζοντας ότι ένα email ανήκει στην κατηγορία spam υποδεικνύουμε στον αλγόριθμο μάθησης με είσοδο τα χαρακτηριστικά του email και έξοδο την κλάση spam ότι πρέπει να μεταβάλλει τις συνιστώσες της h, ώστε η έξοδος για μία παρόμοια πιθανή νέα είσοδο να είναι η κλάση spam.

Αξιολόγηση μοντέλου Μετά την εκπαίδευση το μοντέλο πρέπει να ελεγχθεί σε νέα δεδομένα στα οποία θα προβλέψει την έξοδό τους. Τα δεδομένα αυτά δίνονται ως

είσοδο με το διάνυσμα χαρακτηριστικών χωρίς όμως να χρησιμοποιείται το χαρακτηριστικό κλάσης. Το σύνολο αυτό των χαρακτηριστικών εισόδου με το χαρακτηριστικό κλάσης για μία δεδομένη είσοδο ονομάζεται test instance (ενδεχόμενο ελέγχου). Το σύνολο όλων των test instances ονομάζεται test set. Η μηχανή επομένως, δεν γνωρίζει την αναμενόμενη έξοδο και μας επιστρέφει μία πρόβλεψη της κλάσης για την νέα είσοδο μέσω της 'εκπαιδευμένης' συνάρτησης h. Το χαρακτηριστικό κλάσης χρησιμοποιείται πλέον για να συγκριθεί με την προβλεπόμενη έξοδο που έδωσε το μοντέλο. Έτσι, μπορούμε να υπολογίσουμε μετρικές για το πόσο καλά προβλέπει το μοντέλο (ενότητα 2.1.1.3).



Σχήμα 2.3: Διαδικασία ταξινόμησης

2.1.1.2 Παλινδρόμηση (Regression)

Η μάθηση μίας συνάρτησης h με συνεχείς τιμές εξόδου (συνεχής συνάρτηση) ονομάζεται παλινδρόμηση (regression). Στην περίπτωση αυτή η έξοδος είναι ένας πραγματικός αριθμός, ενώ η διαδικασία μάθησης παραμένει ίδια όπως στο classification (Ενότητα 2.1.1.1).

2.1.1.3 Μετρικές αξιολόγησης μοντέλων

Ακρίβεια ταξινόμησης Μία συχνά χρησιμοποιούμενη μετρική αξιολόγησης είναι η ακρίβεια ταξινόμησης του μοντέλου (Accuracy). Η ακρίβεια υπολογίζεται ως

$$\texttt{Accuracy} = \frac{\texttt{number of correctly classified test instances}}{\texttt{number of test intances}}$$

2.1.2 Μη Επιβλεπόμενη Μάθηση

Η διαδικασία της μη επιβλεπόμενης μάθησης περιλαμβάνει μάθηση για πρότυπα εισόδου (patterns) χωρίς να παρέχονται τιμές εξόδου. Δηλαδή στο σύστημα παρέχεται ως είσοδος κάποια δεδομένα για τα οποία η μηχανή πρέπει να αναγνωρίσει κάποια δομή. Για παράδειγμα, ένα στοιχείο μάθησης μπορεί να αναπτύξει 'αντίληψη' για κείμενα που αφορούν το ίδιο θέμα. Έτσι, το στοιχείο μάθησης δεν γνωρίζει πληροφορίες για την σωστή ενέργεια που πρέπει να κάνει αλλά μαθαίνει να αναγνωρίζει κάποια κοινά πρότυπα στις εισόδους. Στην περίπτωση αυτή, η έξοδος του μοντέλου είναι μία και δεν αφορά κάθε είσοδο ξεχωριστά. Η έξοδος

μπορεί να αφορά ομαδοποίηση των εισόδων σε clusters ή υπολογισμός της πυκνότητας των δεδομένων (density estimation) κ.α. [10].

2.1.3 Ενισχυτική Μάθηση

Η ενισχυτική μάθηση είναι η πιο γενική περίπτωση των τριών και περιλαμβάνει τις περιπτώσεις όπου το στοιχείο μάθησης δεν έχει την δυνατότητα απευθείας ανάδρασης για την σωστή κίνηση που έπρεπε να είχε κάνει αλλά η ανάδραση δίνεται παρατηρώντας αν η κίνησή του οδήγησε ή οδηγεί στο επιθυμητό αποτέλεσμα.

2.1.4 Επιβλεπόμενη Μάθηση για Ανάλυση Συναισθήματος

Στην εργασία αυτή χρησιμοποιούμε επιβλεπόμενη μάθηση καθώς κατά την μάθηση χρησιμοποιούνται κριτικές ταινιών ως instances για τις οποίες γνωρίζουμε την κλάση στην οποία ανήκουν. Κάθε κλάση αντιπροσωπεύει μία πολικότητα ή συναίσθημα. Επομένως, υποδεικνύουμε στην μηχανή την απόφαση που πρέπει να πάρει για το κείμενο αυτό, δίνοντας ως είσοδο την κλάση μαζί με τα χαρακτηριστικά του κειμένου. Αυτό έχει ως αποτέλεσμα την εκπαίδευση της μηχανής να προβλέπει την σωστή κλάση για κάθε κείμενο με νέα χαρακτηριστικά. Επομένως, η ανάλυση συναισθήματος αποτελεί ένα πρόβλημα ταξινόμησης.

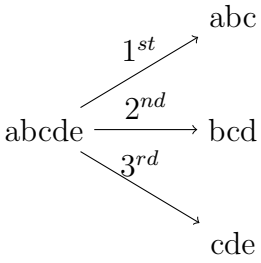
2.2 N-grams

Κατά την παρουσίαση των μεθόδων εξαγωγής χαρακτηριστικών κειμένου χρησιμοποιείται η έννοια του n-gram. Τα n-grams έχουν ευρεία χρήση σε πιθανοτικές μεθόδους της επεξεργασίας φυσικής [20] γλώσσας (Natural Language Processing), όπως είναι η διόρθωση ορθογραφίας (spell correction), η αναγνώριση φωνής (speech recognition) ή η αναγνώριση γλώσσας, η μηχανική μετάφραση (machine translation) αλλά και σε μεθόδους ανάκτησης και αποθήκευσης πληροφορίας (Information Storage and Retrieval), όπως η εξαγωγή περίληψης κειμένου (text summarization) για ευκολία αναζήτησης πληροφορίας, η συμπίεση δεδομένων κ.ά. Επιπλέον, χρησιμοποιούνται στον τομέα της βιοπληροφορικής κατά την ανάκτηση γενετικών ακολουθιών (με τον όρο q-grams) και κατά την αναγνώριση είδους από το οποίο έγινε εξαγωγή αναγνώσματος DNA (DNA read) (με τον όρο k-mers).

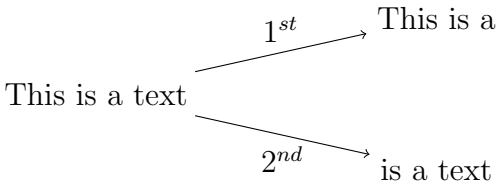
2.2.1 Ορισμός

Ένα n-gram ορίζεται από μία ακολουθία n συνεχόμενων στοιχείων τα οποία εξάγονται από μία ακολουθία χαρακτήρων. Στον τομέα της υπολογιστικής γλωσσολογίας (computational linguistics) το στοιχείο εξαγωγής (extraction unit) μπορεί να είναι ένας χαρακτήρας ή μία λέξη, ανάλογα με την μορφή της αρχικής ακολουθίας και τον σκοπό της εφαρμογής. Στην περίπτωση που εξάγουμε n χαρακτήρες, ονομάζονται n-grams χαρακτήρων (character n-grams) ενώ στην περίπτωση που εξάγουμε n λέξεις, ονομάζονται n-grams λέξεων (word

n-grams). Τα εξαγόμενα n-grams είναι επικαλυπτόμενα, όπως φαίνεται στα σχήματα 2.4 και 2.5 όπου το n ισούται με 3. Η πρώτη περίπτωση αντιστοιχεί σε εξαγωγή χαρακτήρων χωρίς να υπάρχει δυνατότητα εξαγωγής λέξεων, ενώ η δεύτερη αντιστοιχεί σε εξαγωγή λέξεων, όπου υπάρχει και η δυνατότητα εξαγωγής χαρακτήρων.



Σχήμα 2.4: Εξαγωγή 3-grams από την ακολουθία 'abcde'.



Σχήμα 2.5: Εξαγωγή 3-grams από την ακολουθία 'This is a text'.

2.3 Η προσέγγιση των n-gram γράφων

Η μέθοδος των n-gram γράφων δημιουργήθηκε για την αξιολόγηση μοντέλων που εξάγουν αυτόματα περίληψη κειμένου (text summarization) [7], [8]. Η διαδικασία περιλαμβάνει αρχικά τη δημιουργία μίας περίληψης από κείμενα ενός συνόλου. Η περίληψη αποθηκεύεται στην δομή του n-gram γράφου. Κατά την δημιουργία αποθηκεύονται στον γράφο γειτονικά n-grams χαρακτήρων από τα κείμενα του συνόλου. Ο γράφος μειώνει την πληροφορία των αρχικών κειμένων διότι αποθηκεύει μία φορά την εκάστοτε ακολουθία χαρακτήρων που εμφανίζεται με τον ίδιο τρόπο σε ένα ή περισσότερα κείμενα, όπως θα παρουσιαστεί στην συνέχεια. Έτσι, ο γράφος αποτελεί ένα 'μοντέλο' κειμένου που αντιπροσωπεύει το αρχικό σύνολο κειμένων, έχοντας αποθηκεύσει το 'κοινό χαρακτηριστικό' που αυτά μπορεί να έχουν (π.χ. κοινή θεματολογία).

2.3.1 Εφαρμογή στην Ανάλυση Συναισθήματος

Η ιδέα αυτή, εφαρμόστηκε με επιτυχία στην ανάλυση συναισθήματος σε περιεχόμενο μέσων κοινωνικής δικτύωσης (social media) [2], [3]. Στην περίπτωση αυτή, το μοντέλο κειμένου χρησιμοποιείται για να αναπαραστήσει ένα σύνολο κειμένων κριτικής ταινιών που έχουν κοινό 'συναίσθημα' ως κοινό χαρακτηριστικό ({positive, negative, neutral} review). Η δημιουργία των μοντέλων είναι απαραίτητη για την εξαγωγή χαρακτηριστικών κειμένου και αναλύεται παρακάτω.

2.3.2 Αναπαράσταση μοντέλου με έναν n-gram γράφο

Για την δημιουργία του μοντέλου κειμένου, αρχικά αναπαριστούμε το εκάστοτε κείμενο κριτικής του συνόλου με έναν n-gram γράφο. Η αναπαράσταση ενός κειμένου με έναν n-gram γράφο θα παρουσιαστεί με ένα παράδειγμα.

2.3.2.1 Αναπαράσταση κειμένου με έναν n-gram γράφο

Έστω ότι το κείμενο κριτικής αποτελείται από την ακολουθία χαρακτήρων 'Great movie'. Αρχικά επιλέγουμε την τιμή του n να είναι 3 και εξάγουμε όλα τα επικαλυπτόμενα 3-grams χαρακτήρων από το κείμενο.

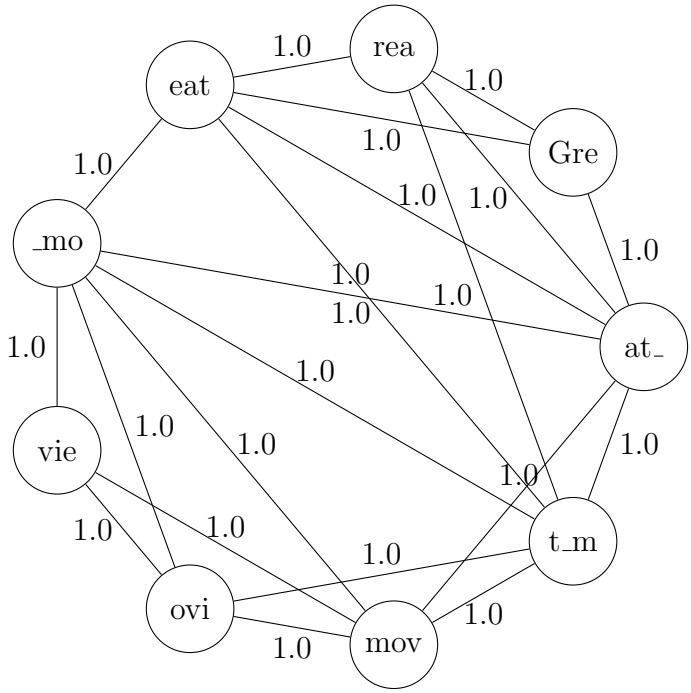
- 'Great_movie':
    1. Gre
    2. rea
    3. eat
    4. at_
    5. t_m
    6. _mo
    7. mov
    8. ovi
    9. vie

Για κάθε μοναδικό n-gram (εδώ 3-gram) δημιουργούμε έναν κόμβο στον γράφο. Η χρήση γράφου έχει κύριο σκοπό την αποθήκευση της γειτνίασης των n-grams, η οποία εμπεριέχει πληροφορία για την σειρά εμφάνισης των χαρακτήρων στο αρχικό κείμενο. Για το σκοπό αυτό επιλέγουμε μία τιμή για ένα μήκος παραθύρου (D_{win}). Το μήκος παραθύρου αποτελεί το πλήθος των γειτονικών στο n-gram χαρακτήρων, για τα n-grams των οποίων, θα θεωρούμε γειτνίαση με αυτό το n-gram. Δηλαδή, αν ένα n-gram βρίσκεται μέσα στο προκαθορισμένο παράθυρο του υπο μελέτη n-gram, τοποθετούμε ακμή μεταξύ των αντίστοιχων κόμβων. Ως βάρος ακμής αποθηκεύουμε το πλήθος εμφανίσεων του ζεύγους των n-grams

μέσα στο προκαθορισμένο παράθυρο (συνήθως δεν ξεπερνούν το ένα με μικρό μήκος παραθύρου). Αν επιλέξουμε παράθυρο 3 χαρακτήρων, τότε η περιοχή 3 χαρακτήρων πριν και μετά το 3-gram 't_m' χρησιμοποιείται για την τοποθέτηση γειτονικών κόμβων στον κόμβο t_m, όπως φαίνεται στο σχήμα 2.6. Παρατηρούμε ότι οι γειτονικοί κόμβοι είναι τα 3-grams: rea, eat, at_, _mo, mov, ovi. Ο τελικός γράφος του κειμένου φαίνεται στο σχήμα 2.7.

G**rea**t_m**ovi**e

Σχήμα 2.6: Παράθυρο 3 χαρακτήρων για το 3-gram 't_m'.

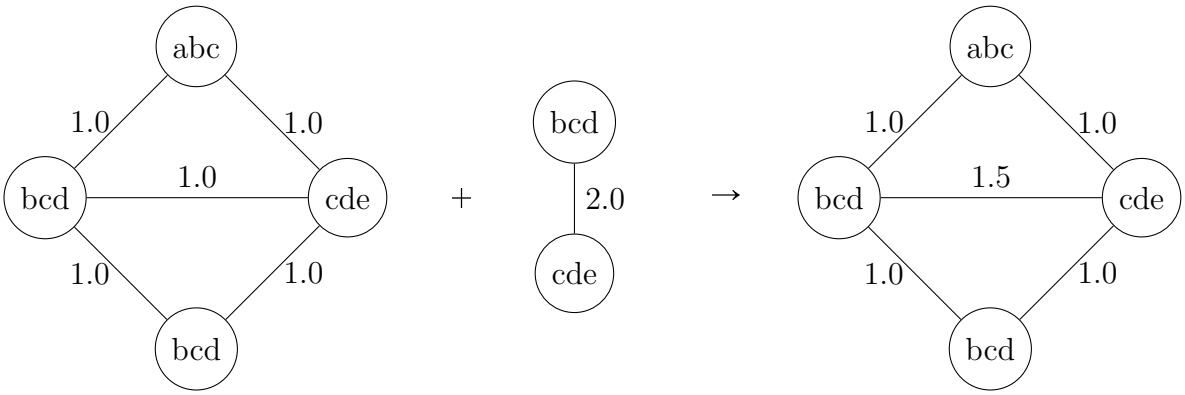


Σχήμα 2.7: 3-gram γράφος ακολουθίας 'Great_movie'.

2.3.2.2 Συγχώνευση γράφων για δημιουργία μοντέλου

Ο γράφος του μοντέλου αρχικοποιείται με τον γράφο του πρώτου κειμένου από το σύνολο. Κάθε επόμενος γράφος κειμένου συγχωνεύεται με τον γράφο μοντέλου. Για παράδειγμα, έστω ότι αρχικοποιείται με τον γράφο της ακολουθίας 'abcde' με $n = 3$ και $D_{win} = 2$ (σχήμα 2.8). Υποθέτουμε ότι ο επόμενος γράφος με τον οποίο θα συγχωνευτεί περιλαμβάνει την ακολουθία 'bcde', επομένως σε αυτόν εμφανίζεται ακμή που συνδέει τους κόμβους bcd, cde

(έστω με βάρος 2.0). Κατά την συγχώνευση τοποθετούμε στον γράφο μοντέλου τα νέα ζεύγη n-grams. Στην περίπτωση που έχουμε να τοποθετήσουμε ήδη υπάρχουσα ακμή μεταξύ δύο κόμβων του γράφου, τότε απλώς ανανεώνουμε το βάρος της ακμής ώστε να αποτελεί το μέσο όρο των δυο βαρών.



Σχήμα 2.8: Συγχώνευση ήδη υπάρχουσας ακμής στο γράφο.

Στην γενική περίπτωση που μία ακμή εμφανίζεται στον γράφο μοντέλου και σε $n-1$ γράφους κειμένων, πρέπει να ληφθεί υπόψη με βάρος τον μέσο όρο των n βαρών. Κατά την n-οστή συγχώνευση, ο γράφος έχει ήδη συγχωνεύσει $n-2$ φορές την ακμή με αποτέλεσμα αυτή να έχει ένα βάρος που αντιστοιχεί σε μέσο όρο $n-1$ προηγούμενων αριθμών. Το βάρος της ακμής πρέπει να ανανεωθεί κατάλληλα ώστε να αποτελεί το μέσο όρο όλων των n αριθμών. Επειδή ο αλγόριθμος δεν αποθηκεύει τα προηγούμενα βάρη για να υπολογίζει κάθε φορά το νέο μέσο όρο, αλλά λαμβάνει τα βάρη στις υπο συγχώνευση ακμές, γνωρίζοντας το πλήθος τους (n), μπορούμε να τον υπολογίσουμε με την παρακάτω παράσταση.

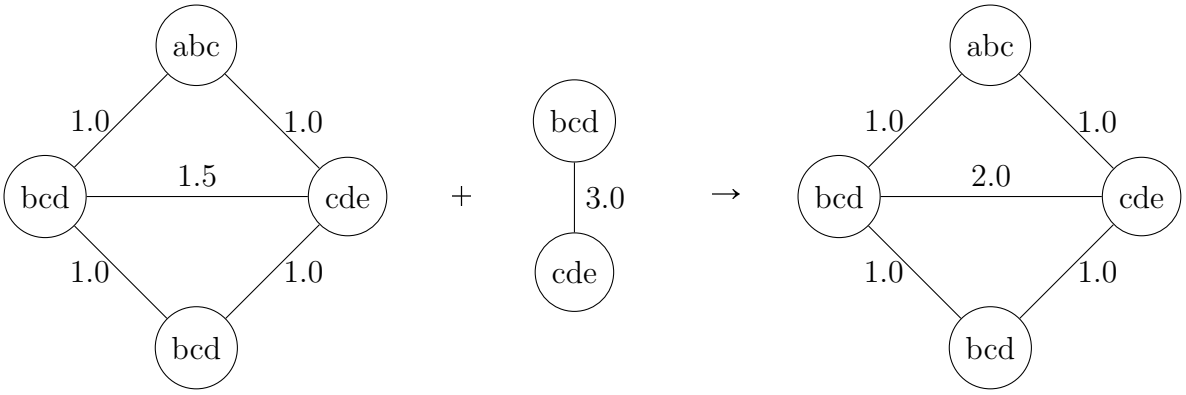
$$new\_average = old\_average + \frac{1}{(n)} \times (new\_weight - old\_average)$$

Η παράσταση αποδεικνύεται εύκολα ότι δίνει τον μέσο όρο n στοιχείων αν θέσουμε

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

Αυτό μπορεί να γίνει κατανοητό αν θεωρήσουμε ότι στον προκύπτον γράφο του σχήματος 2.8 εμφανίζεται για δεύτερη φορά η ακμή {bcd, cde} κατά την συγχώνευση (σχήμα 2.9). Στην περίπτωση αυτή, έχουμε να λάβουμε υπόψη τρία βάρη για το νέο μέσο όρο, γνωρίζοντας το νέο βάρος (3.0) και τον παλιό μέσο όρο (1.5).

Επομένως, για το σύνολο κειμένων που γνωρίζουμε ότι έχουν θετικό συναίσθημα δημιουργούμε ένα μοντέλο κειμένου που ονομάζουμε γράφο θετικής πολικότητας (positive graph), και αντίστοιχα για το σύνολο κειμένων με αρνητικό συναίσθημα, έναν γράφο αρνητικής πολικότητας (negative graph). Οι γράφοι αυτοί, χρησιμοποιούνται στην συνέχεια για εξαγωγή χαρακτηριστικών για την μηχανική μάθηση.



Σχήμα 2.9: Συγχώνευση ήδη υπάρχουσας ακμής για δεύτερη φορά.

2.3.3 Εξαγωγή χαρακτηριστικών (features) με χρήση των γράφων

Στην εργασία αυτή θα επικεντρωθούμε σε *binary* (δυϊκή) ταξινόμηση (δύο κλάσεις) κατά την οποία ταξινομούμε κάθε ενδεχόμενο κατάστασης (instance) ως αληθές (1) ή ψευδές (0). Στην περίπτωσή μας, κάθε instance αντιπροσωπεύει με κάποιο τρόπο ένα κείμενο κριτικής, διαφορετικό από τα κείμενα που χρησιμοποιήθηκαν κατά την δημιουργία των γράφων πολικότητας. Η ταξινόμησή του αφορά την κλάση θετικής πολικότητας (αληθές ή 1) ή την κλάση αρνητικής πολικότητας (ψευδές ή 0).

Κάθε κείμενο αντιπροσωπεύεται από ένα σύνολο χαρακτηριστικών (features) το οποίο αποτελεί ένα instance προς ταξινόμηση. Το σύνολο αυτό εξάγεται από το κείμενο και επιθυμούμε να είναι αντιπροσωπευτικό για την πολικότητά του. Για το σκοπό αυτό χρησιμοποιούμε τα μοντέλα. Αρχικά για κάθε κλάση - πολικότητα δημιουργούμε ένα γράφο πολικότητας, όπως παρουσιάστηκε. Στην συνέχεια δημιουργούμε τον n-gram γράφο του κάθε κειμένου που αποτελεί ενδεχόμενο προς ταξινόμηση. Ο γράφος του κειμένου συγκρίνεται με κάθε γράφο πολικότητας, ώστε να εξάγουμε κάποιες τιμές ομοιότητας με τον καθένα από αυτούς. Επομένως, οι τιμές αυτές μπορούν να αποτελέσουν κατάλληλα χαρακτηριστικά του κειμένου για τον συμπερασμό της πολικότητάς του μέσα από μία μηχανή.

Κατά την διαδικασία σύγκρισης χρησιμοποιήθηκαν οι δείκτες ομοιότητας που αναφέρονται για την σύγκριση γράφων κατά την αξιολόγηση περιλήψεων [8]. Αυτές είναι:

- Coocurrence ή Containment Similarity (CS): είναι μια πραγματική τιμή για την ομοιότητα δύο γράφων και δηλώνει το ποσοστό των ακμών του μικρότερου γράφου που εμφανίζονται και στον μεγαλύτερο γράφο χωρίς να λαμβάνουμε υπόψη το βάρος των ακμών. Μικρότερος γράφος θεωρείται αυτός που έχει λιγότερες ακμές. Δηλαδή, για τον υπολογισμό της τιμής, ελέγχουμε για κάθε ακμή που συνδέει δυο κόμβους στον μικρότερο γράφο, αν υπάρχει ακμή που συνδέει τους ίδιους κόμβους και στον μεγαλύτερο γράφο. Επομένως, αυτή η τιμή βρίσκεται μεταξύ 0 και 1 (δηλώνοντας το αντίστοιχο ποσοστό). Υπολογίζεται από την παρακάτω παράσταση.

$$\frac{\texttt{\# edges of the smallest graph cooccuring in the biggest graph}}{\texttt{\# edges of the smallest graph}}$$

- Size Similarity (SS): είναι μια πραγματική τιμή που δηλώνει την αναλογία των μεγεθών των δύο γράφων και παίρνει τιμές μεταξύ 0 και 1. Υπολογίζεται από την παρακάτω παράσταση.

$$\frac{\texttt{\# edges of the smallest graph}}{\texttt{\# edges of the biggest graph}}$$

- Value Similarity (VS): χρησιμοποιείται για να δηλώσει τον αριθμό των ακμών του μικρότερου γράφου που εμφανίζονται και στον μεγαλύτερο λαμβάνοντας υπόψη και τα βάρη των ακμών τους. Επομένως, αν μία ακμή υπάρχει και στους δυο γράφους με διαφορετικό βάρος τότε λαμβάνεται το μικρότερο βάρος από τα δυο διότι το μικρότερο βάρος αντιστοιχεί σε πλήθος εμφάνισεων του ζεύγους των n-grams που αποθηκεύτηκε και στους δυο. Η παράσταση δίνεται παρακάτω με το w_e^i να αντιστοιχεί στο βάρος της ακμής του ενός γράφου και το w_e^j στο βάρος της ίδιας ακμής του δεύτερου γράφου. Η τιμή της ομοιότητας κυμαίνεται μεταξύ 0 και 1.

$$\frac{\sum \frac{min(w_e^i, w_e^j)}{max(w_e^i, w_e^j)}}{\texttt{\# edges of the biggest graph}}$$

- Normalized Value Similarity (NVS): αποτελεί μία σημαντική τιμή η οποία αποσυνδέει το Value Similarity από το μέγεθος των γράφων. Για το σκοπό αυτό διαιρούμε το VS με το SS. Οι τιμές του κυμαίνονται επίσης μεταξύ 0 και 1.

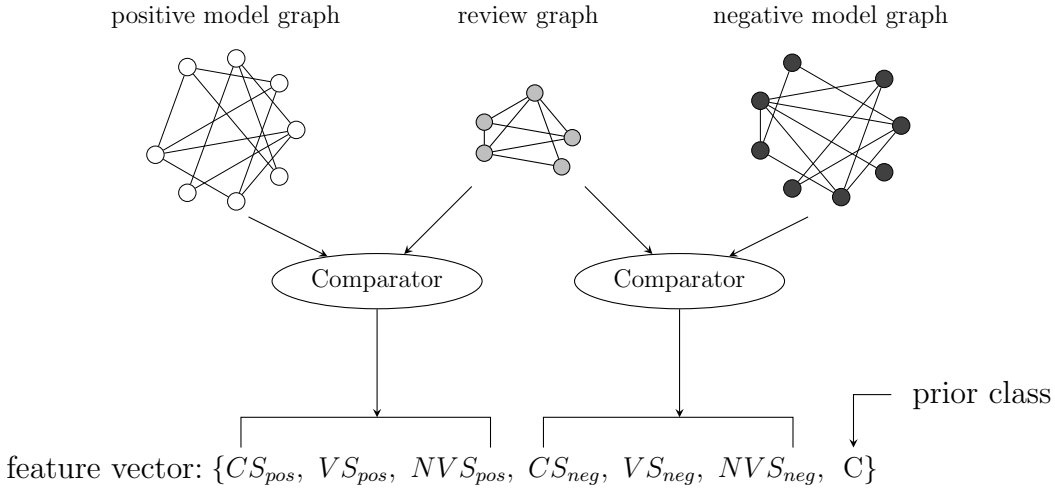
$$\frac{VS}{SS}$$

ή

$$\frac{\sum \frac{min(w_e^i, w_e^j)}{max(w_e^i, w_e^j)}}{\texttt{\# edges of the smallest graph}}$$

Στο σχήμα 2.10 παρουσιάζουμε την διαδικασία εξαγωγής χαρακτηριστικών κειμένου κριτικής. Παρατηρούμε ότι τα χαρακτηριστικά για κάθε ενδεχόμενο είναι τρεις τιμές ομοιότητας με τον κάθε γράφο πολικότητας-κλάσης. Στην περίπτωσή μας, έχουμε τρεις τιμές για την ομοιότητα με τον γράφο θετικής πολικότητας και τρεις τιμές για την ομοιότητα με τον γράφο αρνητικής πολικότητας, συνολικά 6 τιμές στο διάστημα {0, 1} η καθεμία. Μαζί με την τιμή C της κλάσης του κειμένου (0 ή 1) δημιουργούμε για κάθε ενδεχόμενο του συνόλου εκπαίδευσης, ένα διάνυσμα 7 τιμών με τις οποίες εκπαιδεύουμε την μηχανή να προβλέπει την

σωστή κλάση ενός νέου ενδεχομένου. Αντίστοιχα ενεργούμε και για κάθε ενδεχόμενο του συνόλου ελέγχου. Αυτό το πρόβλημα ονομάζεται Binary Polarity Problem. Στην γενική περίπτωση που συμπεριλαμβάνουμε κλάση ουδέτερης πολικότητας, υπολογίζουμε τις επιπλέον τιμές ομοιότητας με τους αντίστοιχο γράφο ουδέτερης πολικότητας και το πρόβλημα ονομάζεται General Polarity Problem.



Σχήμα 2.10: Σύγκριση γράφου με τα μοντέλα για εξαγωγή χαρακτηριστικών.

2.4 Η προσέγγιση της μεθόδου bag of words

Στην εργασία αυτή χρησιμοποιούμε την μέθοδο bag of words για να κατατάξουμε την επίδοση της μεθόδου των n-grams. Η μέθοδος χρησιμοποιεί τα n-grams με στοιχείο εξαγωγής μία λέξη (extraction unit). Επομένως, πρόκειται για n-grams λέξεων (word n-grams) με $n = 1$. Η μέθοδος bag of words είναι πολύ απλή και πετυχημένη στη χρήση. Γι΄ αυτό χησιμοποιείται συχνά σε μεθόδους της Επεξεργασίας Φυσικής Γλώσσας, όπως φιλτράρισμα ανεπιθύμητων μηνυμάτων (spam filtering), δρομολόγηση ηλεκτρονικού ταχυδρομείου (e-mail routing) (ίσως παραπομπή), αναγνώριση γλώσσας, κατηγοριοποίηση κειμένου (document classification), ανάλυση συναισθήματος κ.ά.

2.4.1 Ορισμός

Ο όρος bag of words χρησιμοποιείται για να ορίσει ένα σύνολο λέξεων όπου αυτές αποθηκεύονται αγνοώντας τυχόν σειρά εμφάνισής τους και γραμματική [19], [20]. Οι λέξεις του συνόλου προέρχονται από ένα αρχικό κείμενο και το σύνολο χρησιμοποιείται για να αναπαραστήσει το κείμενο ιδιαίτερα στις περιπτώσεις εφαρμογών που δεν είναι σημαντική οποιαδήποτε πληροφορία της δομής του κειμένου.

2.4.2 Αναπαράσταση κειμένου

Για την bag of words αναπαράσταση ενός κειμένου χρησιμοποιείται ένα διάνυσμα (vector). Κάθε θέση του διανύσματος αντιστοιχεί σε μία λέξη ενός λεξιλογίου (vocabulary set) που έχουμε δημιουργήσει. Η τιμή που αποθηκεύεται σε κάθε θέση του διανύσματος αποτελεί την συχνότητα εμφάνισης της λέξης αυτής στο κείμενο (term frequency) ή απλώς την ένδειξη εμφάνισης της (presence). Η συχνότητα εμφάνισης αντιστοιχεί σε έναν ακέραιο που δηλώνει το πλήθος των φορών που την συναντήσαμε στο κείμενο, ενώ η ένδειξη εμφάνισης είναι μία binary μεταβλητή που δηλώνει αληθές όταν η λέξη εμφανίζεται στο κείμενο, και ψευδές αντιθέτως.

2.4.2.1 Παράδειγμα

Έστω ότι το λεξιλόγιο αποτελείται από τις λέξεις του σχήματος 2.11 και θεωρείται επαρκές για να την αναπαράσταση κειμένων κάποιας εφαρμογής. Στο σχήμα αυτό παρουσιάζουμε την bag of words αναπαράσταση του κειμένου με ακολουθία: 'I thought this was a quiet good movie. It was fun to watch it.', ενώ στο σχήμα 2.12, βλέπουμε την αναπάρασταση του κειμένου 'Great movie.' με χρήση αυτού του λεξιλογίου. Παρατηρούμε και τις δυο περιπτώσεις αναπαράστασης, δηλαδή συχνότητα εμφάνισης και ένδειξη εμφάνισης λέξεων.

Vocabulary set	
0	I
1	thought
2	this
3	was
4	a
5	quiet
6	good
7	movie.
8	It
9	fun
10	to
11	watch
12	it.
13	Great

Term frequency	
0	2
1	1
2	1
3	2
4	1
5	1
6	1
7	2
8	1
9	1
10	1
11	1
12	1
13	0

Presense	
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	0

Σχήμα 2.11: Bag of words αναπαράσταση κειμένου 'I thought this was a quiet good movie. It was fun to watch it.'

Term frequency	
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0
13	1

Presense	
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0
13	1

Σχήμα 2.12: Bag of words αναπαράσταση κειμένου 'Great movie.'

2.4.3 Εξαγωγή χαρακτηριστικών με χρήση της μεθόδου Bag of words

Στην ενότητα αυτή παρουσιάζουμε την χρήση της αναπαράστασης bag of words για την εξαγωγή των features που δίνονται ως είσοδο στον αλγόριθμο μηχανικής μάθησης. Για μία εισαγωγή στους όρους της μηχανικής μάθησης που χρησιμοποιούνται βλέπε ενότητα 2.1.1.

Και στην μέθοδο αυτή επιθυμούμε να χρησιμοποιοήσουμε ένα σύνολο χαρακτηριστικών το οποίο να είναι αντιπροσωπευτικό για την πολικότητα κάθε κειμένου που αποτελεί ένα ενδεχόμενο (instance) προς εκπαίδευση ή κατηγοριοποίηση. Η μέθοδος αυτή χρησιμοποιεί ως διάνυσμα χαρακτηριστικών το διάνυσμα των λέξεων της προσέγγισης bag of words. Η αναπαράσταση κάθε κειμένου επιτυγχάνεται με συχνότητες εμφάνισης των λέξεων ενός λεξιλογίου ή με ένδειξη εμφάνισης τους, το οποίο παρουσιάστηκε στην προηγούμενη ενότητα.

Αρχικά δημιουργούμε το λεξιλόγιο χρησιμοποιώντας τις λέξεις των κειμένων του συνόλου εκπαίδευσης της μηχανής. Η δημιουργία του γίνεται εύκολα με την δυνατότητα που προσφέρει η βιβλιοθήκη Weka[1] στην οποία ζητείται ένα άνω όριο λέξεων που θα αποθηκευτούν σε αυτό. Στην συνέχεια, για κάθε κείμενο του συνόλου εκπαίδευσης ή του συνόλου ελέγχου δημιουργούμε το διάνυσμα λέξεων. Στο διάνυσμα αυτό, προστίθεται ως χαρακτηριστικό κλάσης η εκ των προτέρων πολικότητα του κειμένου (0 ή 1), που έχουμε αναθέσει χειροκίνητα σε αυτό. Το διάνυσμα αυτό αποτελεί το διάνυσμα χαρακτηριστικών που δίνεται ως είσοδο στον αλγόριθμο μάθησης. Στην περίπτωση εκπαίδευσης του, το χαρακτηριστι-

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

κό κλάσης χρησιμοποιείται για την εκπαίδευση της μηχανής, ενώ στην περίπτωση ελέγχου χρησιμοποιείται για την αξιολόγηση της ακρίβειας ταξινόμησης των κειμένων.

2.5 Μοντέλα ταξινομητών

Το πιο κλασικό μοντέλο για ταξινόμηση κειμένων (document classification) ([16], [18]) είναι το μοντέλο naive Bayes που παρουσιάζεται παρακάτω και στο οποίο επικεντρωνόμαστε στην εργασία αυτή.

2.5.1 Απλοϊκό μοντέλο Bayes (naive Bayes model)

Ο όρος naive Bayes χρησιμοποιείται για να δηλώσει μοντέλα αλγορίθμων μηχανικής μάθησης που χρησιμοποιούν τον κανόνα του Bayes κατά την ανάθεση κλάσης σε ένα ενδεχόμενο κατάστασης το οποίο αναπαριστάται από ένα διάνυσμα χαρακτηριστικών (features). Επιπλέον, η λέξη naive χρησιμοποιείται για να δηλώσει το σύνολο αυτών, στους οποίους κάνουμε μία αφελής υπόθεση ανεξαρτησίας για τις μεταβλητές των χαρακτηριστικών (features) του ενδεχομένου. Επειδή στα μοντέλα αυτά η κλάση, που αποτελεί την έξοδο - απόφαση του μοντέλου, παίρνει διακριτές τιμές ορίζοντας ένα πεπερασμένο σύνολο, τα μοντέλα naive Bayes αποτελούν μοντέλα ταξινόμησης (classifiers). Παρακάτω παρουσιάζουμε τον κανόνα του Bayes και την 'αφελή' υπόθεση, δηλαδή τα χαρακτηριστικά ενός naive Bayes classifier.

2.5.1.1 Κανόνας του Bayes (Bayes' rule)

Έχοντας δυο τυχαίες μεταβλητές X_1, X_2 που αντιπροσωπεύουν δυο τμήματα ενός κόσμου, ο κανόνας του Bayes ορίζει την πιθανότητα να συμβεί $X_1 = a$, δεδομένου ότι έχει συμβεί $X_2 = b$, δηλαδή την υπό συνθήκη πιθανότητα $P(a \mid b)$ ως,

$$P(a \mid b) = \frac{P(b \mid a)P(a)}{P(b)}$$

Ο κανόνας προκύπτει εύκολα αν σκεφτούμε ότι ο κανόνας γινομένου των δυο γεγονότων είναι αντιμεταθετικός:

$$P(a \wedge b) = P(b \wedge a)$$

και

$$P(a \wedge b) = P(a \mid b)P(b)$$

$$P(b \wedge a) = P(b \mid a)P(a)$$

2.5.1.2 Μοντέλα Bayes

Στα μοντέλα Bayes, για να αποφασίσουμε ποια κλάση θα αναθέσουμε σε ένα ενδεχόμενο υπολογίζουμε απλώς την πιθανότητα κάθε κλάσης (κατηγορίας) με βάση τα δεδομένα, χρησιμοποιώντας τον κανόνα Bayes. Συγκεκριμένα, υπολογίζουμε την υπο συνθήκη πιθανότητα να ανατεθεί μία κλάση σε ένα ενδεχόμενο, δεδομένου των τιμών των χαρακτηριστικών του ενδεχομένου,

$$P(c \mid x_1,\ x_2,\ \ldots\ x_n)$$

όπου c αποτελεί την κλάση και είναι μία διακριτή τιμή από το πεπερασμένο σύνολο C των δυνατών κλάσεων, και $\{x_1,\ x_2,\ \ldots,\ x_n\}$ είναι το διάνυσμα των χαρακτηριστικών $\mathbf{x}$. Η πιθανότητα αυτή υπολογίζεται από τον κανόνα του Bayes ως:

$$P(c \mid x_1,\ x_2,\ \ldots,\ x_n) = \frac{P(c)\ P(x_1,\ x_2,\ \ldots,\ x_n \mid c)}{P(x_1,\ x_2,\ \ldots,\ x_n)}$$

Δηλαδή έχουμε να υπολογίσουμε για κάθε συνδυασμό τιμών των χαρακτηριστικών, την συνδυασμένη πιθανότητα $P(x_1,\ x_2,\ \ldots,\ x_n \mid c)$. Αυτός ο υπολογισμός είναι αδύνατος όταν τα χαρακτηριστικά παίρνουν συνεχείς τιμές ή/και όταν το πλήθος τους n είναι μεγάλο. Για το σκοπό αυτό, τα μοντέλα naive Bayes κάνουν μία ισχυρή υπόθεση για τις τυχαίες μεταβλητές των χαρακτηριστικών που απλοποιεί τον υπολογισμό, αλλά έχει ως αποτέλεσμα να μην αντιπροσωπεύει προβλήματα του πραγματικού κόσμου και να θεωρείται αφελής. Παρόλα αυτά, παραμένουν γρήγορα και ανταγωνιστικά μοντέλα και έχει μελετηθεί η βελτιστότητά τους [6], [9], [30].

2.5.1.3 Υπόθεση naive Bayes

Για τον υπολογισμό της συνδυασμένης πιθανότητας $P(x_1,\ x_2,\ \ldots,\ x_n \mid c)$, τα μοντέλα naive Bayes υποθέτουν ότι κάθε χαρακτηριστικό (feature), ως τυχαία μεταβλητή, είναι ανεξάρτητο από οποιοδήποτε άλλο χαρακτηριστικό δεδομένης της κλάσης. Η υπόθεση αυτή για την ανεξαρτησία των χαρακτηριστικών οδηγεί στην παρακάτω απλούστευση του υπολογισμού της υπο συνθήκης πιθανότητας κάθε χαρακτηριστικού x_i που περιλαμβάνει το μοντέλο,

$$P(x_i \mid c,\ x_1,\ x_2,\ \ldots\ x_{i-1},\ x_{i+1},\ \ldots,\ x_n) = P(x_i \mid c) \quad \forall\ i\ \in \{1,\ \ldots\ ,\ n\}$$

Επομένως,

$$P(c \mid x_1,\ x_2,\ \ldots,\ x_n) = \frac{P(c)\ P(x_1,\ x_2,\ \ldots,\ x_n \mid c)}{P(x_1,\ x_2,\ \ldots,\ x_n)} \Rightarrow$$

$$P(c \mid x_1,\ x_2,\ \ldots,\ x_n) = \frac{P(c)\ P(x_1 \mid c)\ P(x_2 \mid c,\ x_1)\ \ldots\ P(x_n \mid c,\ x_1,\ x_2,\ \ldots,\ x_{n-1})}{P(x_1,\ x_2,\ \ldots,\ x_n)} \Rightarrow$$

$$P(c \mid x_1,\ x_2,\ \ldots,\ x_n) = \frac{P(c)\ P(x_1 \mid c)\ P(x_2 \mid c)\ \ldots\ P(x_n \mid c)}{P(x_1,\ x_2,\ \ldots,\ x_n)} \Rightarrow$$

$$P(c \mid x_1,\ x_2,\ \ldots,\ x_n) = \frac{P(c)\ \prod_{i=1}^{n} P(x_i \mid c)}{P(x_1,\ x_2,\ \ldots,\ x_n)}$$

Από την τελευταία εξίσωση παρατηρούμε ότι παίρνουμε την υπο συνθήκη πιθανότητα κάθε κλάσης $P(c \mid x_1,\ x_2,\ \ldots,\ x_n)$, ή αλλιώς την εκ των υστέρων πιθανότητα (a posteriori) κάθε κλάσης χρησιμοποιώντας

- την χωρίς συνθήκη ή εκ των προτέρων πιθανότητα της κλάσης (a priori), $P(c)$
- την πιθανοφάνεια των δεδομένων ($\mathbf{x}$), δεδομένης της κλάσης (likelihood), $\prod_{i=1}^{n} P(x_i \mid c)$
- την πιθανότητα εμφάνισης των δεδομένων (evidence), $P(x_1,\ x_2,\ \ldots,\ x_n)$

δηλαδή,

$$posterior = \frac{prior \times likelihood}{evidence}$$

Σε κάθε ενδεχόμενο μας ενδιαφέρει να αναθέσουμε την κλάση με την μεγαλύτερη εκ των υστέρων (a posteriori) πιθανότητα. Επειδή για ένα συγκεκριμένο ενδεχόμενο με διάνυσμα $\{x_1,\ x_2,\ \ldots,\ x_n\}$ η πιθανότητα εμφάνισης των δεδομένων (evidence) είναι σταθερή ποσότητα, μπορούμε να αγνοήσουμε τον παρονομαστή κατά τον υποπλογισμό των εκ των υστέρων πιθανοτήτων των κλάσεων. Αναθέτουμε τότε στην έξοδο, την κλάση με την μέγιστη εκ των υστέρων πιθανοφάνεια (Maximum a posteriori likelihood ή MAP),

$$y = \underset{c \in C}{\operatorname{argmax}}\, P(c) \prod_{i=1}^{n} P(x_i \mid c)$$

Έτσι, έχουμε να υπολογίσουμε τις πιθανότητες $P(x_i \mid c)$, το πλήθος των οποίων είναι γραμμικό ως προς το πλήθος των χαρακτηριστικών n, και τις εκ των προτέρων πιθανότητες $P(c)$.

Οι εκ των προτέρων πιθανότητες $P(c)$ έχουν καθοριστεί κατά την εκπαίδευση του μοντέλου. Συνήθως είναι το ποσοστό των ενδεχομένων του συνόλου εκπαίδευσης που βρίσκονται στην κλάση c, δηλαδή,

$$P(c) = \frac{\texttt{number of training instances in class c}}{\texttt{total number of training instances}}$$

Επιπλέον, για τον υπολογισμό των υπο συνθήκη πιθανοτήτων $P(x_i \mid c)$ πρέπει να γνωρίζουμε την κατανομή των πιθανοτήτων των x_i δεδομένης της κλάσης c. Για το σκοπό αυτό κάνουμε μία υπόθεση για την κατανομή αυτή κατά την εκπαίδευση, όπου και καθορίζονται οι παράμετροί της με βάση τα δεδομένα του συνόλου εκπαίδευσης. Η υπόθεση για την κατανομή των χαρακτηριστικών αποκαλείται event model (μοντέλο γεγονότων) του naive Bayes. Στην συνέχεια χρησιμοποιούμε την κατανομή για τον υπολογισμό των αντίστοιχων πιθανοτήτων κατά τον έλεγχο (testing).

2.5.1.4 Guassian naive Bayes

Αν τα χαρακτηριστικά των ενδεχομένων (instances) παίρνουν συνεχείς τιμές υποθέτουμε κανονική κατανομή (Gaussian) για τις πιθανότητες των x_i δεδομένης της κλάσης c, $P(x_i \mid c)$. Στην περίπτωση αυτή, το μοντέλο ονομάζεται Gaussian naive Bayes classifier. Επειδή η κανονική κατανομή χρησιμοποιείται στην γενική περίπτωση, το μοντέλο ονομάζεται και naive Bayes classifier.

Κατά την εκπαίδευση ακολουθούμε την παρακάτω διαδικασία για τα instances κάθε κλάσης [13]. Αρχικά επιλέγουμε για την κλάση c τα αντίστοιχα instances που έχουν ταξινομηθεί χειροκίνητα σε αυτήν. Στην συνέχεια, από το σύνολο αυτό, για κάθε χαρακτηριστικό υπολογίζουμε τον μέσο όρο των τιμών του, μ_c και την διασπορά σ_c^2 και με αυτά καθορίζουμε την κανονική κατανομή του χαρακτηριστικού για την κλάση c. Τελικά, έχουμε για κάθε χαρακτηριστικό και για κάθε κλάση μία κανονική κατανομή που καθορίζει την κατανομή του χαρακτηριστικού για την κλάση αυτή. Η πιθανοφάνεια επομένως, των χαρακτηριστικών κατά τον έλεγχο (testing) υπολογίζεται από τον τύπο,

$$P(x_i \mid c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp\left(-\frac{(x_i - \mu_c)^2}{2\sigma_c^2}\right)$$

2.5.1.5 Multinomial naive Bayes

Η πολυωνυμική κατανομή χρησιμοποιείται όταν τα χαρακτηριστικά αναπαριστούν συχνότητες $\{x_1,\ x_2,\ \ldots,\ x_n\}$ γεγονότων $\{1,\ 2,\ \ldots,\ n\}$, δηλαδή διακριτές τιμές, όπου οι συχνότητες αυτές έχουν πιθανότητες να εμφανιστούν $\{p_{c_1},\ p_{c_2},\ \ldots,\ p_{c_n}\}$ στην κλάση c. Επομένως, έχουμε ένα πολυώνυμο πιθανοτήτων $\{p_1,\ p_2,\ \ldots,\ p_n\}$ για κάθε κλάση, που αντιπροσωπεύει την εμφάνιση των γεγονότων $\{1,\ 2,\ \ldots,\ n\}$ σε αυτήν την κλάση με συχνότητες $\{x_1,\ x_2,\ \ldots,\ x_n\}$. Ο ταξινομητής ονομάζεται Multinomial naive Bayes classifier και δημιουργήθηκε για την ταξινόμηση κειμένων κατά την οποία έχουμε συχνότητες λέξεων ως χαρακτηριστικά. [παραπομπή] (βλέπε 2.4.3). Το $P(\mathbf{x} \mid c)$ υπολογίζεται τώρα ως:

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

όπου τα παραγοντικά αντιπροσωπεύουν διαφορετικές διατάξεις των λέξεων και είναι σταθερές στον υπολογισμό κάθε κλάσης. Επομένως, μας ενδιαφέρουν τα $P(x_i \mid c)$ που υπολογίζονται από το σύνολο εκπαίδευσης ως η συχνότητα εμφάνισης x_i του γεγονότος i στην κλάση c,

$$P(x_i \mid c) = \frac{x_i}{\sum_{i=1}^{n} x_i}, \quad \texttt{where class}(x_i) = c$$

Από τις δυο σχέσεις παρατηρούμε ότι αν μία λέξη δεν υπάρχει στο training set ή αν τυγχάνει να μην εμφανίζεται ($x_i = 0$) στα instances της κλάσης c, τότε, μηδενίζει την τελική πιθανότητα $P(c \mid x_1,\ x_2,\ \ldots,\ x_n)$ να αναθέσουμε την κλάση αυτή στο instance που περιέχει

αυτήν την λέξη κατά τον έλεγχο. Για την αντιμετώπιση αυτού του προβλήματος προστίθεται μία τιμή σε όλες τις πιθανότητες ώστε να μην μηδενίζεται καμία. Όταν η τιμή ισούται με 1 η διαδικασία ονομάζεται Laplace smoothing και έχουμε:

$$P(x_i \mid c) = \frac{n_{c_i} + 1}{\sum_{i=1}^{n}(n_{c_i} + 1)}$$

2.5.2 Δέντρο Αποφάσεων (Decision Tree)

Ένα δέντρο αποφάσεων αποτελεί έναν αλγόριθμο μάθησης που αποφασίζει για το αποτέλεσμα της εξόδου μέσω μίας ακολουθίας ελέγχων σε μορφή δέντρου. Κάθε κόμβος είναι ένας έλεγχος πάνω στην τιμή ενός χαρακτηριστικού (feature), ενώ οι απόγονοι του κόμβου είναι όσοι και οι δυνατές τιμές του ελέγχου. Τα φύλλα του δέντρου περιλαμβάνουν την τιμή (όταν έχουμε regression) ή κλάση (όταν έχουμε classification) που θα επιστραφεί όταν φτάσουμε σε ένα από αυτά καθώς επεξεργάζοντας τις τιμές των features κάποιας εισόδου με το δέντρο. Επομένως, κάθε έλεγχος έχει στόχο να διαχωρίσει το σύνολο της πληροφορίας των χαρακτηριστικών σε υποσύνολα, καθένα από τα οποία βρίσκεται πιο κοντά σε μία απόφαση για την έξοδο [19], [29].

2.5.2.1 Δημιουργία δέντρου

Η δημιουργία του δέντρου γίνεται κατά την εκπαίδευση. Αρχικά τοποθετούμε έναν κόμβο - ρίζα, ο οποίος περιέχει όλα τα instances. Η επιλογή του χαρακτηριστικού, με βάση τις τιμές του οποίου θα διαιρεθεί ο κόμβος, γίνεται με τέτοιο τρόπο ώστε να ελαχιστοποιηθεί το βάθος του τελικού δέντρου. Επομένως, επιλέγουμε ένα χαρακτηριστικό το οποίο διαιρεί τα instances σε όσο το δυνατόν ξεκάθαρα ως προς την κλάση σύνολα, δηλαδή προχωρά όσο γίνεται περισσότερο στην ταξινόμηση των instances.

2.5.2.2 Επιλογή χαρακτηριστικού

Για να διαπιστώσουμε ποιο χαρακτηριστικό οδηγεί πιο γρήγορα στην ταξινόμηση χρειαζόμαστε ένα κατάλληλο μέτρο που θα εφαρμόσουμε στα χαρακτηριστικά. Η τιμή του μέτρου πρέπει να υποδηλώνει για κάθε χαρακτηριστικό αν μετά την επιλογή του, ελαχιστοποιήθηκε η πληροφορία που πρέπει να ταξινομήσουμε. Για το σκοπό αυτό σκεφτόμαστε ότι κάθε χαρακτηριστικό παρέχει μία ποσότητα πληροφορίας - με τις τιμές που παίρνει - για το ποια κλάση θα επιλεγεί. Επομένως, υπολογίζουμε την εντροπία πληροφορίας που παρέχει κάθε χαρακτηριστικό για την κλάση. Η ελαχιστοποίηση της πληροφορίας μετά την επιλογή ενός χαρακτηριστικού ελέγχεται αθροίζοντας την τιμή της εντροπίας πληροφορίας των υπόλοιπων χαρακτηριστικών για την κλάση. Η τιμή του αθροίσματος αποτελεί μία ένδειξη για την πληροφορία που απομένει για το υπόλοιπο δέντρο και επιλέγουμε το χαρακτηριστικό που προσφέρει τη μικρότερη τιμή για αυτό το άθροισμα.

Εναλλακτικά, μπορούμε να αφαιρέσουμε το άθροισμα εντροπίας των υπόλοιπων χαρακτηριστικών από την αρχική εντροπία, το οποίο μας δίνει την πληροφορία που "κερδίζουμε" με την επιλογή του χαρακτηριστικού. Η ποσότητα αυτή ονομάζεται κέρδος πληροφορίας και

είναι διαφορετική για κάθε χαρακτηριστικό. Σε αυτήν την περίπτωση, επιλέγουμε το χαρακτηριστικό που προσφέρει το μεγαλύτερο κέρδος πληροφορίας. Η διαδικασία παρουσιάζεται παρακάτω έχοντας υποθέσει διακριτές τιμές για τα χαρακτηριστικά. Για ευκολία θεωρούμε ότι έχουμε δυο κλάσεις, την κλάση 0 και την κλαση 1. Για τον υπολογισμό της εντροπίας κάθε χαρακτηριστικού f αρχικά σκεφτόμαστε ότι για κάθε τιμή i του χαρακτηριστικού, παίρνουμε

- με πιθανότητα $P_i(0) = \frac{n_{i_0}}{n}$ την κλάση 0 και
- με πιθανότητα $P_i(1) = \frac{n_{i_1}}{n}$ την κλάση 1.

όπου n_{i_0} το πλήθος εμφάνισης της τιμής i όταν οδηγούμαστε στην κλάση 0 και αντίστοιχα για το n_{i_1}, και n το πλήθος των instances. Οι πιθανότητες αυτές χρησιμοποιούνται για να υπολογίσουμε την εντροπία πληροφορίας για το χαρακτηριστικό κλάσης (class attribute) από αυτήν την τιμή. Η τιμή της εντροπίας αντιπροσωπεύει την ποσότητα της πληροφορίας σε bits. Έτσι, υπολογίζουμε για κάθε τιμή i του χαρακτηριστικού,

$$H_{f_i} = - P_{f_i}(0) \log P_{f_i}(0) \; - \; P_{f_i}(1) \log P_{f_i}(1) \quad \forall \; i$$

Αν η τιμή εμφανίζεται συνολικά n_i φορές ενώ έχουμε n instances, τότε έχουμε πιθανότητα εμφάνισης της τιμής

$$P_i = \frac{n_{f_i}}{n}$$

Συνολικά το χαρακτηριστικό f προσφέρει πληροφορία για το χαρακτηριστικό κλάσης

$$H_f = \sum_i \; P_i \log H_{f_i}$$

Η τιμή H_f αντιπροσωπεύει την ποσότητα της πληροφορίας που παρέχεται από το χαρακτηριστικό για την κλάση. Αν θεωρήσουμε ότι επιλέγουμε αυτό το χαρακτηριστικό η πληροφορία που απομένει για να ταξινομήσουμε στην συνέχεια από τα υπόλοιπα χαρακτηριστικά είναι

$$H_{rest}(f) = \sum_{j \neq f} \; H_j$$

Τελικά έχουμε κέρδος πληροφορίας (information gain) του χαρακτηριστικού f

$$IG_f = H - H_{rest}(f)$$

όπου H η αρχική εντροπία πληροφορίας από όλα τα instances

$$H = - P(0) \log P(0) \; - \; P(1) \log P(1)$$

Ο κόμβος αυτός διαιρείται με βάση τις τιμές του χαρακτηριστικού που μεγιστοποιεί το κέρδος πληροφορίας. Κατά την διαίρεση, δημιούργουμε έναν νέο κόμβο - ρίζα του δέντρου, ο οποίος περιέχει πλέον το χαρακτηριστικό και συνδεέται με δυο κόμβους - απογόνους υπό συνθήκη για την τιμή του χαρακτηριστικού. Οι κόμβοι απόγονοι περιέχουν τα δυο υποσύνολα των instances στα οποία διαιρέθηκε ο αρχικός με βάση την τιμή του χαρακτηριστικού και αντιστοιχούν σε τιμές του χαρακτηριστικού που ικανοποιούν την συνθήκη του μονοπατιού που οδηγεί σε αυτόν τον κόμβο. Η διαίρεση συνεχίζεται με τον ίδιο τρόπο μέχρι να φτάσουμε σε κόμβους με instances που έχουν ταξινομηθεί στην ίδια κλάση.

Κεφάλαιο 3

Συνδυασμός μεθόδων

Στο Κεφάλαιο αυτό παρουσιάζουμε τα αποτέλεσματα της πρώτης σύγκρισης της μεθόδου των n-gram γράφων με την baseline μέθοδο, η οποία αντιστοιχεί στην εκπαίδευση του αλγορίθμου μάθησης multinomial naive Bayes με χαρακτηριστικά στην μορφή bag of words. Στην συνέχεια, σχολιάζουμε τα αποτελέσματα και παρουσιάζουμε τους λόγους που η μέθοδος επιτυγχάνει αυτήν την επίδοση.

3.1 Πρώτη σύγκριση μεθόδων

Στην ενότητα αυτή επιχειρούμε να συγκρίνουμε για πρώτη φορά τις δυο μεθόδους που παρουσιάστηκαν στο δεύτερο Κεφάλαιο. Για τα χαρακτηριστικά της baseline μεθόδου χρησιμοποιήθηκαν τιμές που δηλώνουν συχνότητα εμφάνισης λέξεων (term frequency) καθώς και οι γράφοι αποθηκεύουν βάρη που δηλώνουν τον μέσο όρο της συχνότητας εμφάνισης ενός ζεύγους n-grams. Οι παράμετροι και τα αποτελέσματα δίνονται παρακάτω.

3.1.1 Παράμετροι

Δεδομένα (Εκπαίδευση & Αξιολόγηση) Κατά την σύγκριση των δυο μεθόδων χρησιμοποιήσαμε κριτικές ταινιών της βάσης IMDb[1]. Τα δεδομένα αυτά συγκεντρώθηκαν από το πανεπιστήμιο του Stanford για την μελέτη που παρουσιάζεται στο [17]. Επιχειρήσαμε να συγκρίνουμε τις δυο μεθόδους χρησιμοποιώντας 5000 κριτικές για την εκπαίδευση του εκάστοτε αλγορίθμου μάθησης (training set) και 6000 κριτικές για την αξιολόγησή του (test set). Για τις δυο μεθόδους η εκπαίδευση του ταξινομητή πραγματοποιήθηκε στο ίδιο ακριβώς σύνολο των 5000 κριτικών ταινιών και, αντίστοιχα, αξιολογήθηκε στο ίδιο σύνολο των 6000 κριτικών για να μπορούμε να συγκρίνουμε τις επιδόσεις τους.

Παράμετροι γράφων Σύμφωνα με τη μέθοδο των n-gram γράφων δημιουργήσαμε δυο γράφους πολικότητας (θετικής και αρνητικής), ο καθένας από τους οποίους ενσωμάτωσε (ή συγχώνευσε) 800 κριτικές ταινιών από την βάση IMDb[1]. Η επιλογή των 800 reviews έγινε με σκοπό να έχουμε αρκετά μεγάλους γράφους και επομένως, μία εικόνα για την

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

"καλύτερη" επίδοση που μπορεί να έχει η χρήση γράφων. Επιπλέον, θέτουμε $D_{win} = n$ εφόσον σύμφωνα με μελέτες (βλέπε [1]) αποτελεί την περίπτωση για την οποία έχουμε καλύτερη επίδοση. Στις εργασίες [1] και [3] διαπιστώνεται καλύτερη η περίπτωση όπου το $n = 4$, επομένως την χρησιμοποιούμε για να συγκρίνουμε τις μεθόδους. Δεν χρησιμοποιείται καμία προεπεξεργασία για την αφαίρεση ειδικών χαρακτήρων του κειμένου.

Επιλογή αλγορίθμου μάθησης Οι αλγόριθμοι naive Bayes είναι δημοφιλής για την ταξινόμηση κειμένων, και ιδιαίτερα ο multinomial naive Bayes με χρήση bag of words, ο οποίος όμως, εφαρμόζεται σε διακριτές τιμές χαρακτηριστικών, όπως είδαμε στην ενότητα 2.5.1.5. Αντίθετα οι n-gram γράφοι εξάγουν χαρακτηριστικά που παίρνουν συνεχείς τιμές, επομένως, για να εφαρμόσουμε αυτόν τον αλγόριθμο μάθησης στην μέθοδο των γράφων πρέπει να μετατρέψουμε τις συνεχείς τιμές ομοιότητας που χρησιμοποιεί ως χαρακτηριστικά, σε διακριτές. Η διακριτοποίηση αυτή πρέπει να δημιουργεί χαρακτηριστικά αντιπροσωπευτικά για την πολικότητα του κάθε κειμένου αλλά και κατάλληλα για την εφαρμογή της κατανομής που υποθέτει ο multinomial naive Bayes (συχνότητες εμφάνισης γεγονότων).

Διακριτοποίηση χαρακτηριστικών ομοιότητας n-gram γράφων Για το σκοπό της πρώτης σύγκρισης, με χρήση του αλγορίθμου multinomial naive Bayes μετατρέπουμε τις συνεχείς τιμές των δεικτών ομοιότητας μεταξύ δυο n-gram γράφων σύμφωνα με την προσέγγιση της εργασίας [3]. Σε αυτήν για κάθε instance του συνόλου εκπαίδευσης και του συνόλου ελέγχου με διάνυσμα χαρακτηριστικών τις τιμές ομοιότητας, που έχει παρουσιαστεί στο σχήμα 2.10, ακολουθούμε την διαδικασία:

$$\{CS_{pos},\ NVS_{pos},\ VS_{pos},\ CS_{neg},\ NVS_{neg},\ VS_{neg},\ C\} \Rightarrow$$
$$\{\texttt{dsim}(CS_{pos},\ CS_{neg}),\ \texttt{dsim}(NVS_{pos},\ NVS_{neg}),\ \texttt{dsim}(VS_{pos},\ VS_{neg}),\ C\}$$

όπου η συνάρτηση dsim διακριτοποιεί τις τιμές ομοιότητες με τον παρακάτω τρόπο:

$$\texttt{dsim}(sim_{pos}, sim_{neg}) = \begin{cases} \texttt{positive} & \text{if } sim_{pos} > sim_{neg} \\ \texttt{negative} & \text{if } sim_{pos} < sim_{neg} \\ \texttt{equal} & \text{if } sim_{pos} = sim_{neg} \end{cases}$$

Πλέον έχουμε τρεις διακριτές τιμές ομοιότητας ως χαρακτηριστικά και το χαρακτηριστικό κλάσης C. Κάθε μία από τις τρεις τιμές προκύπτει από την σύγκριση ενός δείκτη ομοιότητας για τον θετικό γράφο με τον ίδιο δείκτη ομοιότητας για τον αρνητικό γράφο.

Επομένως, μπορούμε να ελέγξουμε τις δυο μεθόδους σε ίδιους classifiers όπως, naive Bayes, Decision Tree, multinomial naive Bayes, αλλά επικεντρωνόμαστε στον naive Bayes ο οποίος εφαρμόζεται και στις δυο περιπτώσεις χαρακτηριστικών, χωρίς να μεταβάλλουμε την αρχική δομή τους. Η χρήση naive Bayes αντί για την χρήση multinomial naive Bayes επιφέρει μία μείωση της ακρίβειας της μεθόδου bag of words, η οποία όμως, δεν επηρεάζει την γενικά καλή επίδοσή της για την σύγκριση των μεθόδων. Στο κεφάλαιο 4, κατά την τελική σύγκριση χρησιμοποιούμε για την εκάστοτε μέθοδο τον αλγόριθμο μάθησης για τον οποίο αυτή έχει καλύτερη επίδοση.

Αν η μέθοδος είναι κατάλληλη για το συγκεκριμένο πρόβλημα, περιμένουμε να έχει ποσοστά ακρίβειας συγκρίσιμα με τα ποσοστά της μεθόδου bag of words (επαναλαμβάνουμε ότι για την μέθοδο bag of words χρησιμοποιήθηκαν ως χαρακτηριστικά συχνότητες εμφάνισης λέξεων). Η ακρίβεια για τις δυο μεθόδους φαίνεται στον πίνακα 3.1.

3.1.2 Αποτελέσματα

# Experiment	Classifier	Method	
		N-gram graphs (800 reviews)	Bag of words
1	Naive Bayes	**56.8%**	74.6%
2		**55.6%**	73.5%
3		**59.3%**	74.9%
1	J48 (tree)	72.9%	70.5%
2		72.4%	70.4%
3		72.3%	71.4%
1	Naive Bayes Multinomial	55.6%	81.2%
2		**51.7%**	80.9%
3		**73.6%**	81.4%

Πίνακας 3.1: Ακρίβεια των δυο μεθόδων για 3 διαφορετικά πειράματα με 800 reviews σε κάθε γράφο πολικότητας.

Στον πίνακα 3.1 συγκεντρώσαμε τα αποτελέσματα της ακρίβειας των δυο μεθόδων κατά την χρήση τριών ταξινομητών. Παρατηρούμε ότι η μέθοδος των n-gram γράφων δεν πλησιάζει τα ποσοστά της μεθόδου bag of words. Η διαφορά τους βρίσκεται περίπου στο 17%. Αυτή η διαφορά δεν μπορεί να γίνει αποδεκτή αν σκεφτεί κανείς ότι η μέθοδος bag of words είναι πολύ πιο γρήγορη, καθώς εξάγει κατευθείαν τις λέξεις από το κείμενο και ο χρόνος εκτέλεσής της καθορίζεται, κυρίως, από το χρόνο εκπαίδευσης και αξιολόγησης του ταξινομητή. Αντίθετα, η μέθοδος των n-gram γράφων καταναλώνει τον περισσότερο χρόνο κατά την δημιουργία των γράφων και στην συνέχεια κατά την σύγκρισή τους με τους γράφους των reviews για την εξαγωγή των χαρακτηριστικών και την δημιουργία των instances εκπαίδευσης και ελέγχου. Επιπλέον, οι γράφοι καταναλώνουν πολύ περισσότερο χώρο, καθώς είναι φορτωμενοι στην μνήμη κατά την σύγκριση. Για να έχουμε καλύτερη εικόνα της διαφοράς των δυο μεθόδων, διεξάγουμε ένα πείραμα με περισσότερα reviews στους γράφους. Συγκεκριμένα ενσωματώθηκαν 2000 reviews σε κάθε γράφο πολικότητας και τα αποτελέσματα φαίνονται στον πίνακα 3.2.

Και σε αυτήν την περίπτωση, όπου οι γράφοι ενσωμάτωσαν πολύ μεγαλύτερο πλήθος από reviews παρατηρούμε ότι η επίδοσή τους στο πρόβλημα αυτό δεν πλησιάζει την baseline μέθοδο, με διαφορά γύρω στο 14% με εξαίρεση την περίπτωση του πρώτου πειράματος όπου η διαφορά τους βρίσκεται στο 4%. Επομένως, μελετάμε τους λόγους για τους οποίους η μέθοδος αυτή δεν έχει καλή επίδοση στην ενότητα 3.2. Επιπλέον παρατηρούμε ότι και στους δυο πίνακες ο αλγόριθμος του δέντρου αποφάσεων με τους n-gram γράφους παρουσιάζει καλύτερη επίδοση από ότι με την μέθοδο bag of words με κόστος, όμως, κατά μέσο όρο 189

# Experiment	Classifier	Method	
		N-gram graphs (2000 reviews)	Bag of words
1	Naive Bayes	**69.7%**	73.7%
2		**60.7%**	75.4%
3		**61%**	74.9%
1	J48 (tree)	72%	69.7%
2		71%	69.9%
3		72.2%	70.6%
1	Naive Bayes Multinomial	**73.4%**	82.2%
2		**67.2%**	82.2%
3		71.2%	81.3%

Πίνακας 3.2: Ακρίβεια των δυο μεθόδων για 3 διαφορετικά πειράματα με 2000 reviews σε κάθε γράφο πολικότητας.

λεπτών για την δημιουργία των γράφων, την δημιουργία του ταξινομητή και την αξιολόγησή του. Τέλος, παρατηρούμε ότι η διακριτοποίηση των τιμών δεν έχει καλά αποτελέσματα καθώς όπως βλέπουμε κυρίως στον πίνακα 3.1 έχουμε μεγάλη διακύμανση ανάμεσα σε δυο πειράματα με ίδιες παραμέτρους και διαφορετικούς γράφους. Αυτό οφείλεται στο ότι έχουμε τρεις διακριτές τιμές (`positive`, `negative`, `equal`) και δυο κλάσεις πολικότητας, επομένως, υπάρχει μεγάλη πιθανότητα να έχουμε για τα ίδιες τιμές χαρακτηριστικών διαφορετική κλάση πολικότητας. Αυτό δεν οδηγεί σε σωστή εκπαίδευση του ταξινομητή.

3.2 Μελέτη επίδοσης n-gram γράφων

Οι λόγοι που οι n-gram γράφοι δεν έχουν καλή επίδοση σε σχέση με την baseline μέθοδο (και με χρήση naive Bayes μοντέλου) μπορεί να είναι οι εξής:

- Αρχικά σκεφτόμαστε ότι η χρήση λέξεων είναι αντιπροσωπευτική για την πρόβλεψη του συναισθήματος του κειμένου, καθώς η ύπαρξη μίας λέξης μεμονωμένα μπορεί να είναι δείκτης της πολικότητας του κειμένου, όπως γίνεται στην μέθοδο bag of words. Επομένως, κατά την εξαγωγή n-grams χαρακτήρων, είναι πιθανό δυο λέξεις διαφορετικής πολικότητας να έχουν τα ίδια n-grams και η διαίρεσή τους σε αυτά να χάνει τη δυνατότητα ένδειξης της πολικότητας. Για παράδειγμα, οι λέξεις happy και unhappy έχουν περίπου τα ίδια 3-grams:

  1. **hap, app, ppy**
  2. unh, nha, **hap, app, ppy**

  ή και 4-grams:

  1. **happ, appy**
  2. unha, nhap, **happ, appy**

Σε αυτήν την περίπτωση η λέξη happy εμφανίζεται και στους δυο γράφους πολικότητας. Κατά την σύγκριση του γράφου ενός review που περιέχει την λέξη happy με τους γράφους πολικότητας, τα n-grams της λέξης αυτής συνεισφέρουν το ίδιο στον δείκτη ομοιότητας Containment Similarity με τους δύο γράφους. Επιπλέον, η τιμή των δυο ομοιοτήτων που λαμβάνουν υπόψη τα βάρη των ακμών θα επηρεαστούν από το γεγονός ότι τα n-grams υπάρχουν και στους δυο γράφους πολικότητας. Στην γενική περίπτωση, οι λέξεις διαφορετικής πολικότητας με κοινά n-grams οδηγούν στο να μην εξάγονται αντιπροσωπευτικές τιμές ομοιότητας με τους γράφους. Επομένως, η εξαγωγή n-grams δημιουργεί θόρυβο για για το περιεχόμενο του κειμένου.

- Μία λέξη που εμφανίζεται συχνά σε ένα κείμενο χωρίς να επηρεάζει την πολικότητά του (όπως η λέξη the η οποία αποτελεί συχνά εμφανιζόμενη λέξη σε stop words list) είναι πιθανό να εμφανίζεται και στους δυο γράφους με τα αντίστοιχα n-grams. Οι λέξεις αυτές επηρεάζουν την πραγματική ομοιότητα ενός review με τους γράφους θετικής ή αρνητικής πολικότητας. Επομένως, οι γράφοι μπορεί να μην δημιουργούν πολύ διαφορετικές τιμές ομοιότητας εξαιτίας τέτοιων λέξεων.

3.3 Προτεινόμενες λύσεις

Στην προηγούμενη ενότητα παρουσιάσαμε τους πιθανούς λόγους για τους οποίους οι n-gram γράφοι έχουν την επίδοση που παρατηρήσαμε στους πίνακες 3.1, 3.2 σχετικά με την baseline μέθοδο. Παρακάτω δίνουμε δυο λύσεις για τα προβλήματα αυτά.

- Στο παραπάνω παράδειγμα παρατηρούμε ότι όταν αυξάνουμε το n μειώνεται το πλήθος των κοινών n-grams που μοιράζονται οι δυο γράφοι. Αν θέσουμε το n ίσο με 5 τότε οι δυο γράφοι μοιράζονται μόνο ένα 5-gram, την λέξη happy. Επομένως, μία λύση είναι να υπολογίσουμε το μέσο μήκος λέξης των κειμένων και να θέτουμε το n ίσο με αυτό. Μία καλύτερη προσέγγιση είναι να θεωρήσουμε ότι έχουμε ένα μεταβλητό μήκος n το οποίο να ισούται με το μήκος της λέξης κάθε φορά, κατά την εξαγωγή n-grams. Έτσι, οι λέξεις happy και unhappy τοποθετούνται πλέον στους δυο γράφους δημιουργώντας ένα κόμβο η καθεμία χωρίς να μοιράζονται οι δυο γράφοι αυτούς τους κόμβους. Για να πετύχουμε αυτή την δυνατότητα πρέπει να αφαιρέσουμε από τους γράφους την παράμετρο n που σταθεροποιεί το μήκος των n-grams. Επομένως, αλλάζουμε την μέθοδο ώστε να τοποθετεί κάθε φορά μία λέξη σε έναν κόμβο του γράφου. Έτσι, εκμεταλλευόμαστε την παρουσία των λέξεων, όπως στην μέθοδο bag of words. Στην γενική περίπτωση, ίδιοι κόμβοι μεταξύ των γράφων πλέον αφορούν μόνο λέξεις που εμφανίζονται σε κείμενα και των δυο πολικοτήτων, όπως η λέξη the.

- Όπως παρουσιάσαμε στην ενότητα 3.2 αλλά και στην προηγούμενη παράγραφο, λέξεις ουδέτερης πολικότητας είναι πιθανό να εμφανίζονται και στους δυο γράφους. Επομένως, πρέπει να 'πολώσουμε' τους δυο γράφους προς την κατεύθυνση του συναισθήματος που αναπαριστούν. Για να πολώσουμε τον κάθε γράφο, μπορούμε να αφαιρέσουμε κοινές λέξεις που εμφανίζονται και στους δυο, αφαιρώντας ακμές που συνδέουν ίδιους κόμβους και στους δυο γράφους, καθώς αυτές παίρνουν κύριο μέρος στην

εξαγωγή δεικτών ομοιότητας. Για το σκοπό αυτό υπολογίζουμε τον κοινό υπογράφο των δυο γράφων, και στην συνέχεια τον αφαιρούμε από κάθε γράφο πολικότητας. Η αφαίρεση του κοινού υπογράφου είχε ήδη υλοποιηθεί στις βιβλιοθήκες των γράφων που χρησιμοποιούμε[1]. Αρχικά βρίσκουμε τον κοινό υπογράφο, ελέγχοντας για κάθε ακμή που συνδέει δυο κόμβους στον γράφο θετικής πολικότητας, αν υπάρχει ακμή που συνδέει τους ίδιους κόμβους και στον γράφο αρνητικής πολικότητας. Στην συνέχεια, αφαιρούμε τις ακμές αυτές από τους δυο γράφους που αποτελούν τον κοινό υπογράφο, χωρίς να ελέγχουμε την τιμή του βάρους που είχαν.

3.4 Παρουσίαση νέας μεθόδου

Και οι δυο παρατηρήσεις χρησιμοποιήθηκαν για την βελτίωση της ακρίβειας των γράφων. Η μέθοδος προέκυψε από τον συνδυασμό της μεθόδου bag of words και της μεθόδου των character n-gram γράφων. Οι αλλαγές ενσωματώθηκαν στον κώδικα της μέθοδου των n-gram γράφων και η μέθοδος που προκύπτει παρουσιάζετε παρακάτω.

3.4.1 Αναπαράσταση μοντέλου με έναν γράφο λέξεων

Η μέθοδος των n-gram γράφων με εξαγωγή λέξεων αποκαλείται στην εργασία αυτή μέθοδος γράφων λέξεων (word graphs). Η αναπαράσταση του μοντέλου - κειμένου πλέον γίνεται με έναν γράφο λέξεων, ο οποίος δημιουργείται και πάλι σταδιακά με την συγχώνευση του γράφου λέξεων κάθε κειμένου κριτικής ταινιών.

3.4.1.1 Αναπαράσταση κειμένου με έναν γράφο λέξεων

Σε κάθε βήμα εξάγουμε από το κείμενο μία λέξη αντί για n χαρακτήρες. Επομένως, έχουμε 1-grams λέξεων ή word 1-grams, δηλαδή έχουμε πάντα $n = 1$ εφόσον εξάγουμε μία λέξη ανά βήμα. Για παράδειγμα, έστω ότι το κείμενο κριτικής αποτελείται από την ακολουθία: 'I thought this was a quiet good movie.' Οι εξαγόμενες λέξεις είναι:

1. I
2. thought
3. this
4. was
5. a
6. quiet
7. good

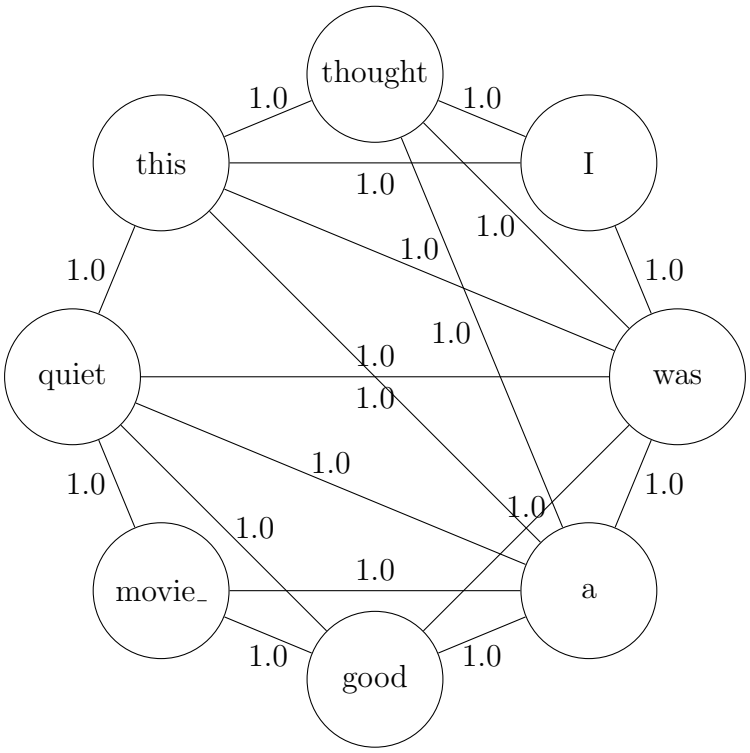
[1] http://sourceforge.net/projects/jinsect/

8. movie_

Για κάθε μοναδική λέξη δημιουργούμε, όπως και στην περίπτωση των n-grams χαρακτήρων, ένα κόμβο στον γράφο. Η διαδικασία τοποθέτησης ακμών προϋποθέτει την ύπαρξη ενός παραθύρου, το οποίο πλέον δηλώνει το πλήθος των γειτονικών λέξεων για κάθε λέξη, με τις οποίες αυτή θα γειτνιάζει στον γράφο. Για παράδειγμα, η λέξη 'was', για μήκος παραθύρου ίσο με 3, γειτνιάζει με τις λέξεις I, thought, this, a, quiet και good όπως φαίνεται στο σχήμα 3.1. Τα βάρη στις ακμές των κόμβων αντιπροσωπεύουν και πάλι το πλήθος εμφανίσεων του ζεύγους των λέξεων μέσα στο παράθυρο αυτό. Ο γράφος αυτής της ακολουθίας δίνεται στο σχήμα 3.2.

I thought this was **a quiet good** movie_

Σχήμα 3.1: Παράθυρο 3 λέξεων για την λέξη 'was'.



Σχήμα 3.2: Γράφος λέξεων παραθύρου 3 της ακολουθίας 'I thought this was a quiet good movie.'

3.4.1.2 Συγχώνευση γράφων για δημιουργία μοντέλου

Οι γράφοι των κειμένων κριτικής ταινιών με όμοια πολικότητα συγχωνεύονται με τον ίδιο τρόπο που παρουσιάστηκε στην ενότητα 2.3.2.2. Σε αυτήν την περίπτωση οι γράφοι περιλαμβάνουν μία λέξη σε κάθε κόμβο, οι οποίοι όμως, δεν επηρεάζουν την διαδικασία της συγχώνευσης που έχει ήδη παρουσιαστεί. Η συγχώνευση οδηγεί στην δημιουργία ενός γράφου μοντέλου, που αντιπροσωπεύει το σύνολο των κειμένων αυτών με κοινή πολικότητα-συναίσθημα.

3.4.2 Εξαγωγή χαρακτηριστικών (features) με χρήση των γράφων

Για κάθε πολικότητα, που αποτελεί μία κλάση κατηγοριοποίησης, δημιουργούμε και πάλι τον αντίστοιχο γράφο - μοντέλο, ο οποίος είναι τώρα ένας γράφος λέξεων. Οι γράφοι πολικότητας χρησιμεύουν για την σύγκριση με τους γράφους λέξεων των κειμένων με σκοπό την εξαγωγή 6 χαρακτηριστικών για το καθένας, που θα αποτελέσουν instances του συνόλου εκπαίδευσης και του συνόλου ελέγχου, όπως παρουσιάστηκε στην ενότητα 2.3.3 (σχήμα 2.10).

3.5 Υλοποίηση

Στην ενότητα αυτή παρουσιάζουμε την αρχιτεκτονική της υλοποίησης στην οποία στηρίχτηκαν τα πειράματα για την σύγκριση των τριών μεθόδων. Για την διαδικασία της μηχανικής μάθησης χρησιμοποιήθηκε η βιβλιοθήκη Weka[1] η οποία ενσωματώνει υλοποιήσεις πολλών ταξινομητών καθώς και μεθόδους αξιολόγησής τους. Επιπλέον, προσφέρει δυνατότητα δημιουργίας κατάλληλων αρχείων για την εκπαίδευση των ταξινομητών από τα χαρακτηριστικά των κειμένων.

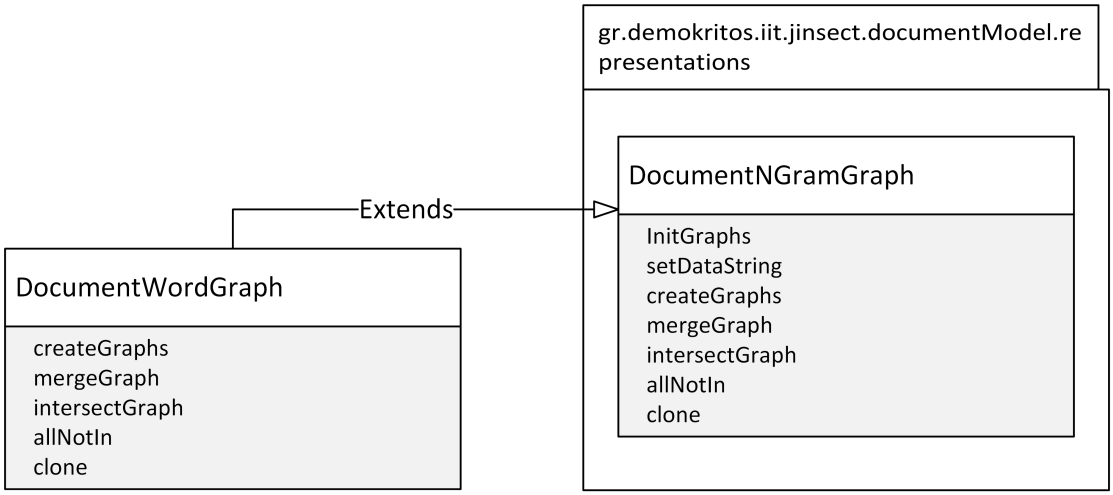
3.5.1 Υλοποίηση μεθόδων

Υλοποίηση μεθόδου n-gram γράφων Οι n-gram γράφοι έχουν υλοποιηθεί και προσφέρονται από την βιβλιοθήκη JInsect[2]. Κατά κύριο λόγο, χρησιμοποιούμε τις μεθόδους της κλάσης DocumentNGramGraph. Η βιβλιοθήκη JInsect χρησιμοποιεί την βιβλιοθήκη OpenJGraph της java για τον χειρισμό των γράφων.

Υλοποίηση μεθόδου γράφων λέξεων Για την υλοποίηση της προσέγγισης των γράφων λέξεων, χρησιμοποιούμε την ήδη υπάρχουσα κλάση της βιβλιοθήκης JInsect που δημιουργεί n-gram γράφους. Επεκτείνοντας την κλάση αυτή σε γλώσσα προγραμματισμού java (extend), μεταβάλλαμε τις μεθόδους των γράφων ώστε να εξάγουν και να χρησιμοποιούν λέξεις κάθε φορά αντί για n χαρακτήρες. Οι μέθοδοι που επεκτείναμε φαίνονται στο σχήμα 3.3. Ο αντίστοιχος κώδικας δίνεται στο παράρτημα.

[1] http://www.cs.waikato.ac.nz/~ml/weka/
[2] http://sourceforge.net/projects/jinsect/

Υλοποίηση μεθόδου bag of words Το μοντέλο bag of words υλοποιήθηκε με τη βοήθεια της βιβλιοθήκης Weka. Η βιβλιοθήκη προσφέρει την δυνατότητα χρήσης ενός φίλτρου που μετατρέπει ένα κείμενο αποθηκευμένο σε ένα String σε ένα διάνυσμα λέξεων (StringToWordVector filter). Η διαδικασία επιλογής λέξεων για το διάνυσμα γίνεται από τα δεδομένα εκπαίδευσης και το πλήθος των λέξεων που θα επιλεγούν καθορίζεται από μία παράμετρος η οποία έχει default τιμή τις 1000 λέξεις. Το διάνυσμα αυτό, χρησιμοποιείται μαζί με το χαρακτηριστικό κλάσης για την εκπαίδευση και τον έλεγχο του εκάστοτε ταξινομητή (ενότητα 2.4.3).



Σχήμα 3.3: Επέκταση την κλάσης των n-gram γράφων.

3.5.2 Υλοποίηση προγράμματος σύγκρισης τριών μεθόδων

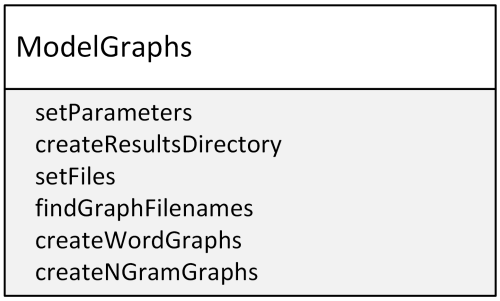
Για την σύγκριση των μεθόδων δημιουργούμε τρία στάδια, όπου για κάθε στάδιο υλοποιήθηκε μία κλάση, όπως αναλύεται στις επόμενες ενότητες.

1. Δημιουργία και αποθήκευση γράφων.
2. Δημιουργία και αποθήκευση αρχείων επκαίδευσης και ελέγχου (training και test set).
3. Δημιουργία ταξινομητή και αξιολόγησή του.

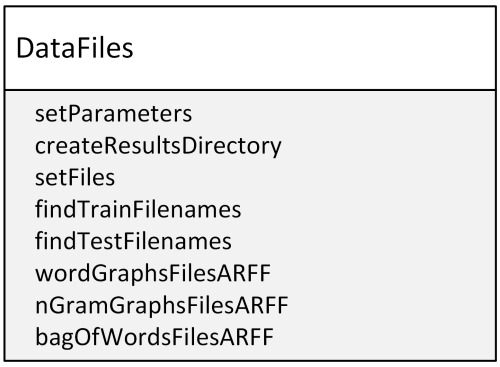
3.5.2.1 Πρώτο στάδιο

Στο πρώτο στάδιο δημιουργούνται και αποθηκεύονται οι n-gram γράφοι και οι γράφοι λέξεων που αποτελούν τα μοντέλα. Έτσι, έχουμε έναν n-gram γράφο και έναν γράφο λέξεων

για κάθε πολικότητα. Οι γράφοι των δυο μεθόδων δημιουργούνται με τρόπο ώστε να ενσωματώνουν τους γράφους των ίδιων κειμένων κριτικής. Η αποθήκευσή τους χρησιμεύει στην ανάκτησή τους για διαφορετικά πειράματα. Για το στάδιο αυτό δημιουργήθηκε μία κλάση που ονομάστηκε ModelGraphs. Το στάδιο αυτό μπορεί να θεωρηθεί στάδιο 'προεπεξεργασίας' καθώς δεν αποτελεί μέρος της διαδικασίας της μηχανικής μάθησης και δεν ασχολείται καθόλου με αυτό η μέθοδος bag of words. Οι μέθοδοι της κλάσης παρουσιάζονται στο σχήμα 3.4, ενώ ο κώδικας δίνεται στο παράρτημα.



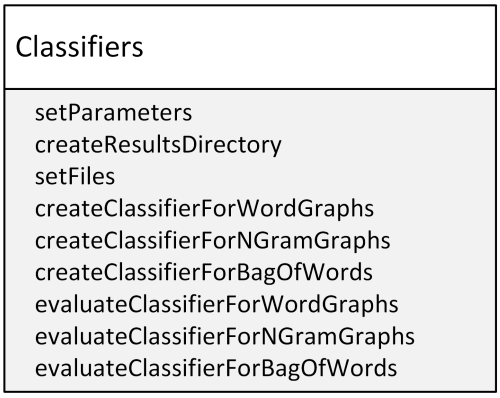
Σχήμα 3.4: Στάδιο 1ο.



Σχήμα 3.5: Στάδιο 2ο.

3.5.2.2 Δεύτερο στάδιο

Το δεύτερο στάδιο χρησιμοποιείται για την δημιουργία των αρχείων εκπαίδευσης και ελέγχου. Σε αυτό υλοποιείται η σύγκριση μεταξύ των γράφων πολικότητας του πρώτου σταδίου με τους γράφους των υποψήφιων για εκπαίδευση ή έλεγχο κειμένων κριτικής. Τα αρχεία που δημιουργούνται περιλαμβάνουν τα διανύσματα χαρακτηριστικών (instances) των κειμένων αυτών και είναι στην μορφή Attribute Relation File Format (ARFF). Στο στάδιο αυτό δημιουργούνται και τα αρχεία εκπαίδευσης και ελέγχου της μεθόδου bag of words.



Σχήμα 3.6: Στάδιο 3ο.

Η κλάση που αντιστοιχεί σε αυτό το στάδιο ονομάστηκε DataFiles και οι μέθοδοί της παρουσιάζονται στο σχήμα 3.5, ενώ ο κώδικας δίνεται στο παράρτημα.

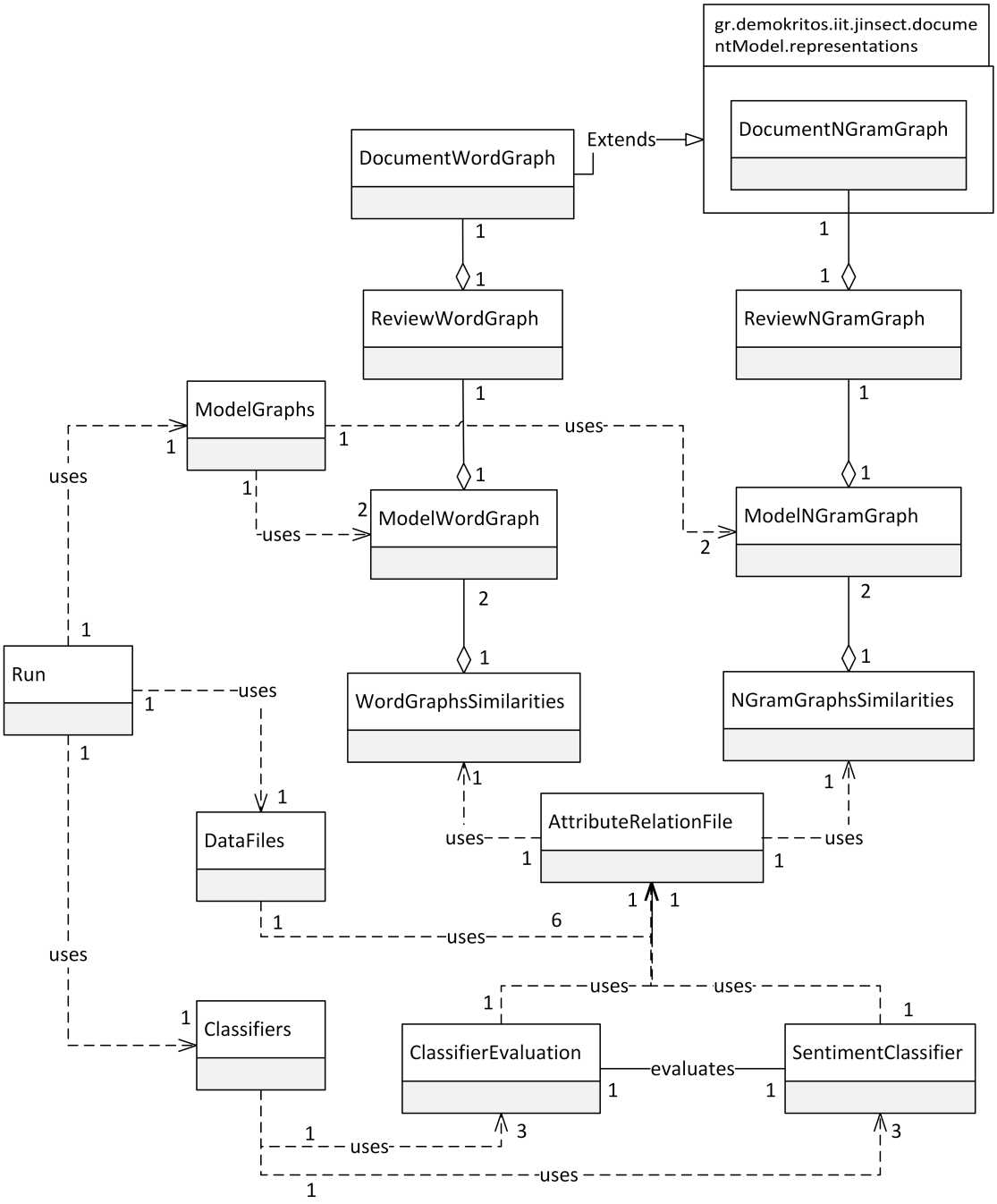
3.5.2.3 Τρίτο στάδιο

Στο τρίτο στάδιο δημιουργούμε και αξιολογούμε τον classifier που καθορίζεται από τον χρήστη με τα δεδομένα των αρχείων που δίνονται κάθε φορά και έχουν δημιουργηθεί στο δεύτερο στάδιο. Το τρίτο στάδιο υλοποιήθηκε από την κλάση Classifiers. Οι μέθοδοι της κλάσης φαίνονται στο σχήμα 3.6 και ο κώδικας στο παράρτημα.

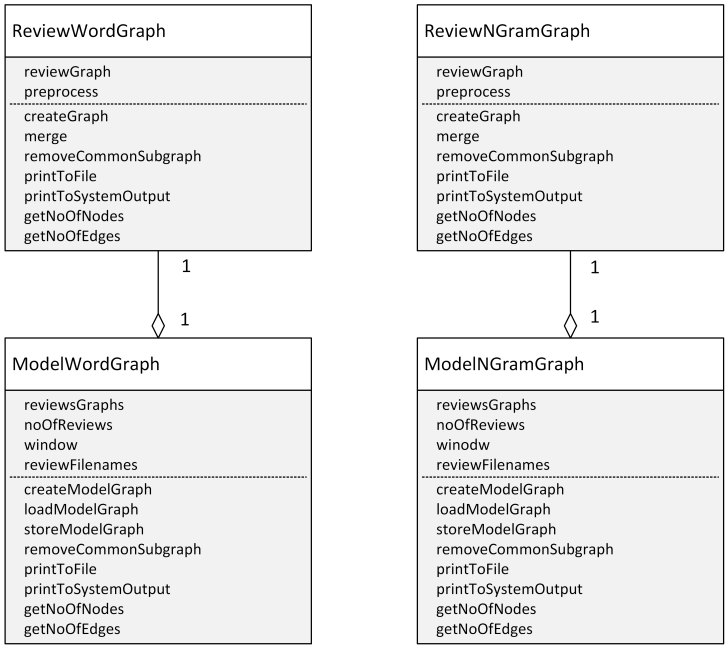
3.5.2.4 Σύνδεση κομματιών

Στο σχήμα 3.7 παρουσιάζουμε το uml διάγραμμα της υλοποίησης όπου δεν περιλαμβάνονται οι μέθοδοι και τα πεδία των java κλάσεων. Οι πληροφορίες για κάθε java κλάση παρουσιάζονται σε επόμενα σχήματα. Όπως φαίνεται και στο διάγραμμα, για κάθε μέθοδο γράφων δημιουργούμε μία κλάση για την αναπαράσταση του γράφου - μοντέλου στο οποίο αποθηκεύεται ο γράφος πολικότητας. Για την περίπτωση των n-gram γράφων δημιουργήσαμε την κλάση ModelNGramGraph με την οποία δημιουργείται ένας γράφος και αποθηκεύεται σε ένα binary αρχείο. Αντίστοιχα για την περίπτωση της μεθόδου των γράφων λέξεων δημιουργήσαμε την κλάση ModelWordGraph.

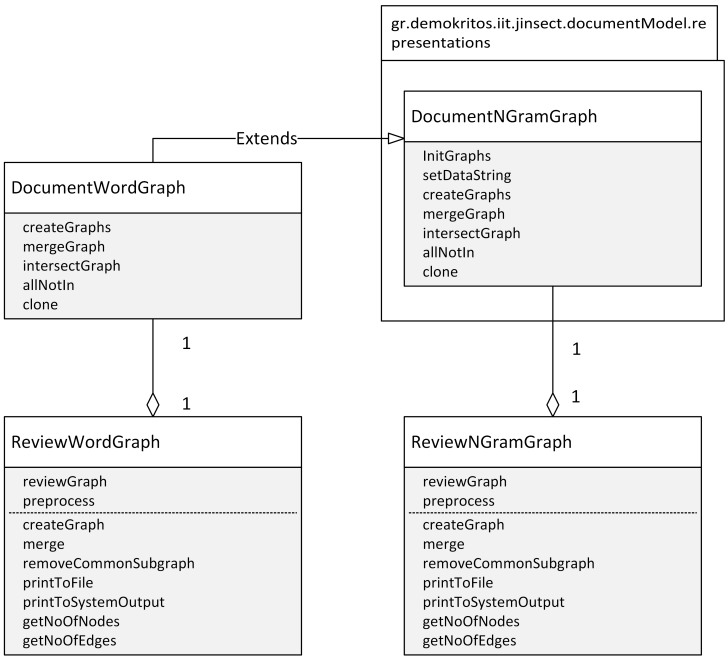
Η δημιουργία του μοντέλου προϋποθέτει την δημιουργία του γράφου κάθε κειμένου που θα συγχωνευτεί σε αυτό (ενότητες 2.3.2, 3.4.1). Για το σκοπό αυτό, δημιουργήσαμε κλάσεις για την αναπαράσταση ενός κειμένου κριτικής με έναν γράφο. Η κλάση που αφορά την αναπαράσταση του κειμένου με έναν n-gram γράφο είναι η ReviewNGramGraph, ενώ η κλάση που αφορά την αναπαράστασή του με έναν γράφο λέξεων είναι η ReviewWordGraph. Οι κλάσεις αυτές χρησιμοποιούνται και για την δημιουργία του γράφου κειμένου που θα συγκριθεί με τους γράφους πολικότητας ώστε να δημιουργήσει ένα instance του συνόλου εκπαίδευσης ή του συνόλου ελέγχου. Έτσι, ένα αντικείμενο της κλάσης ModelWordGraph συνδέεται με ένα αντικείμενο της κλάσης ReviewWordGraph, όπως φαίνεται



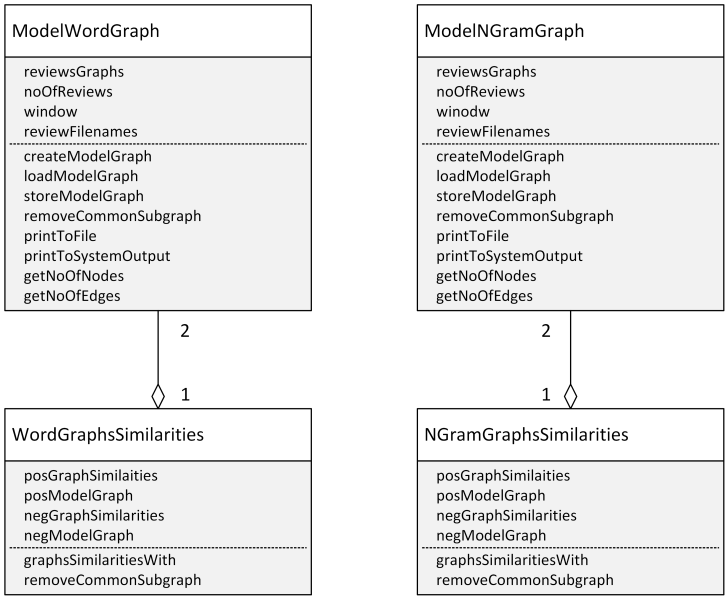
Σχήμα 3.7: Uml διάγραμμα υλοποίησης.



Σχήμα 3.8: Κλάσεις αναπαράστασης μοντέλου με έναν γράφο.



Σχήμα 3.9: Κλάσεις αναπαράστασης κειμένου με έναν γράφο.



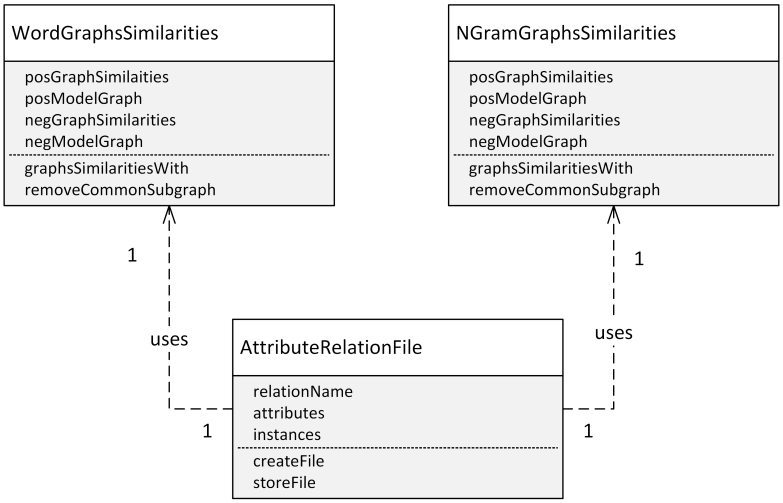
Σχήμα 3.10: Κλάσεις σύγκρισης γράφων.

στο σχήμα 3.8. Αντίστοιχα για την κλάση ModelNGramGraph. Επιπλέον, ένα αντικείμενο των κλάσεων ReviewNGramGraph, ReviewWordGraph ενσωματώνει ένα αντικείμενο των κλάσεων DocumentWordGraph, DocumentNGramGraph αντίστοιχα, καθώς αυτές οι κλάσεις ενσωματώνουν το πραγματικό αντικείμενο του γράφου (σχήμα 3.9).

Για την σύγκριση μεταξύ n-gram γράφων υλοποιήθηκε η κλάση NGramGraphsSimilarities ενώ αντίστοιχα για την σύγκριση μεταξύ γράφων λέξεων υλοποιήθηκε η κλάση WordGraphsSimilarities. Οι κλάσεις αυτές ενσωματώνουν την κλήση της μεθόδου της κλάσης NGramCachedGraphComparator η οποία υλοποιήθηκε στην βιβλιοθήκη JInsect. Η σύγκριση γράφων περιγράφεται στις ενότητες 2.3.3 και 3.4.2. Οι κλάσεις αυτές χρησιμοποιούν τους αντίστοιχους γράφους πολικότητας για την διαδικασία της σύγκρισης, όπως φαίνεται στο σχήμα 3.10. Οι γράφοι πολικότητας είναι δυο, εφόσον ασχολούμαστε με δυο κλάσεις πολικότητας, την θετική και την αρνητική.

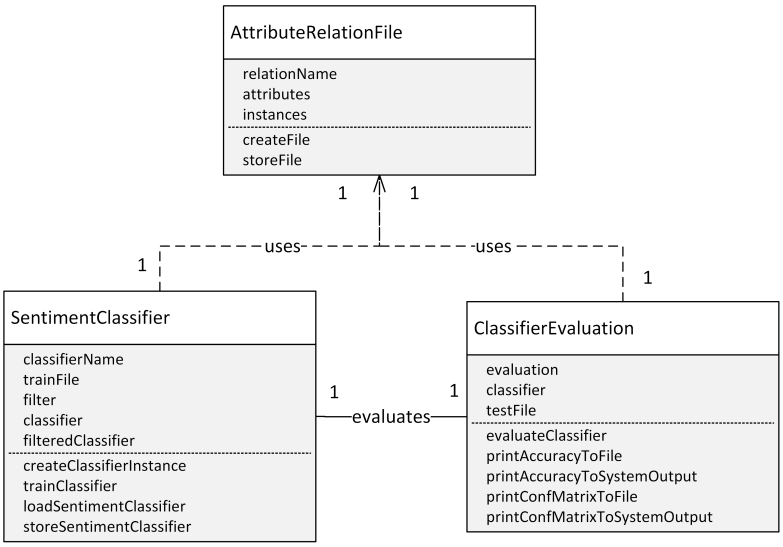
Για την δημιουργία των αρχείων εκπαίδευσης και ελέγχου χρησιμοποιείται η κλάση AttributeRelationFile. Η κλάση χρησιμοποιεί τις τιμές ομοιότητες μεταξύ των γράφων για την αποθήκευση των χαρακτηριστικών στα αρχεία. Επιπλέον, ενσωματώνει την δημιουργία των αρχείων για την bag of words προσέγγιση. Η σύνδεση των κλάσεων παρουσιάζεται στο σχήμα 3.11.

Για την δημιουργία ενός instance ενός συγκεκριμένου ταξινομητή χρησιμοποιείται η κλάση SentimentClassifier. Η κλάση αυτή χρησιμοποιεί το αρχείο εκπαίδευσης που δημιουργείται από την κλάση AttributeRelationFile για την εκπαίδευση του ταξινομητή. Η αξιολόγησή του επιτυγχάνεται με την κλάση ClassifierEvaluation και με χρήση του αρχείου ελέγχου που δημιουργήθηκε, επίσης, από την κλάση AttributeRelationFile. Οι μέθοδοι των



Σχήμα 3.11: Κλάσεις δημιουργίας αρχείων εκπαίδευσης και ελέγχου.

κλάσεων παρουσιάζονται στο σχήμα 3.12. Ο αντίστοιχος κώδικας δίνεται στο παράρτημα.



Σχήμα 3.12: Κλάσεις δημιουργίας ταξινομητών και αξιολόγησής του.

Κεφάλαιο 4

Αξιολόγηση μεθόδων σε δεδομένα κριτικής ταινιών

Στο πλαίσιο αυτής της εργασίας για την εξαγωγή συμπερασμάτων συγκρίθηκαν οι παρακάτω τρεις μέθοδοι εξαγωγής χαρακτηριστικών κειμένου σε κοινά δεδομένα.

- word graph (Ενότητα 3.4)
- n-gram graph (Ενότητα 2.3)
- bag of words με χρήση συχνότητας εμφάνισης λέξεων (Ενότητα 2.4.3)

Η χρήση συχνοτήτων εμφάνισης λέξεων ως χαρακτηριστικά στην μέθοδο bag of words οφείλεται στο ότι και οι γράφοι χρησιμοποιούν βάρη στις ακμές για να δηλώσουν το μέσο όρο συχνότητας εμφάνισης ζεύγους λέξεων ή n-grams. Αρχικά παρουσιάζουμε τις παραμέτρους του προβλήματος σύγκρισης. Στην συνέχεια, επιλέγουμε 'βέλτιστες' τιμές για αυτές, ώστε να επιτευχθεί η παράλληλη σύγκριση. Τα αποτελέσματα και τα πειράματα δίνονται σε επόμενες ενότητες.

4.1 Δεδομένα

Θα χρησιμοποιήσουμε ξανά τα δεδομένα από κριτικές ταινιών της βάσης IMDb[1], όπως στην πρώτη σύγκριση (Ενότητα 3.1.1). Τα δεδομένα αυτά προσφέρονται από το εργαστήριο του Stanford σε δυο σύνολα όπου το ένα αφορά την διαδικασία της εκπαίδευσης και το άλλο την διαδικασία αξιολόγησης - ελέγχου. Κάθε σύνολο περιλαμβάνει 12500 positive και 12500 negative reviews. Επομένως, συνολικά έχουμε διαθέσιμα 50000 reviews. Καθε review είναι αποθηκευμένο σε ένα αρχείο το όνομα του οποίου περιλαμβάνει το id του review και το rating που έδωσε ο χρήστης για την ταινία που σχολιάζει σε αυτό, δηλαδή είναι στη μορφή id_rating.txt. Τα αρχεία που περιλαμβάνουν positive reviews έχουν ratings που κυμαίνονται από 7 έως 10. Αντίστοιχα, τα αρχεία που περιλαμβάνουν negative reviews έχουν ratings που κυμαίνονται από 0 έως 4.

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

4.2 Παράμετροι προγράμματος σύγκρισης

Προκειμένου να χρησιμοποιήσουμε τις μεθόδους σε ένα σύνολο δεδομένων, πρέπει να θέσουμε τιμές στις εγγενείς παραμέτρους του προγράμματος σύγκρισης. Η προσέγγιση των n-gram γράφων επιβάλλει την επιλογή τριών παραμέτρων για την δημιουργία των γράφων, η οποία προηγείται της εκπαίδευσης του αλγορίθμου. Αυτές είναι:

1. `graph reviews`: πλήθος των reviews, το κείμενο των οποίων θα ενσωματώσει ο γράφος (βλέπε ενότητα 2.3.2.2).
2. n: πλήθος των χαρακτήρων που εξάγονται κάθε φορά από το κείμενο κριτικής.
3. D_{win}: μήκος παραθύρου (βλέπε ενότητα 2.3.2).

Αντίστοιχα για τους γράφους λέξεων έχουμε:

4. `graph reviews`: πλήθος των reviews, το κείμενο των οποίων θα ενσωματώσει ο γράφος (βλέπε ενότητα 3.4.1.2).
5. D_{win}: μήκος παραθύρου (βλέπε ενότητα 3.4.1).

Εδώ έχουμε απαλείψει την παράμετρο n, όπως παρουσιάστηκε στην ενότητα της μεθόδου, 3.4.1. Για τους γράφους χρησιμοποιούμε επιπλέον τις παραμέτρους:

6. `remove`: δηλώνει την αφαίρεση του κοινού υπογράφου των δυο γράφων πολικότητας όταν τίθεται σε true,
7. `preprocess`: αφορά την αφαίρεση ειδικών χαρακτήρων του κειμένου προτού αυτό ενσωματωθεί στον γράφο, όταν τίθεται σε true.

Όσον αφορά την μηχανική μάθηση που καλύπτεται στα στάδια 2 και 3 (Ενότητα 3.5.2.2) έχουμε τις παραμέτρους:

8. `training reviews`: αντιστοιχεί στο πλήθος των reviews που τοποθετούμε στο training set.
9. `test reviews`: αντιστοιχεί στο πλήθος των reviews που τοποθετούμε στο test set.
10. `positive rate`: καθορίζει το ποσοστό των reviews των δυο παραπάνω συνόλων που θα ανήκουν στην κλάση positive.
11. `classifier`: τίθεται ίσο με το όνομα του weka classifier ο οποίος θα χρησιμοποιηθεί.

Στις παραμέτρους του προγράμματος συμπεριλαμβάνουμε και παραμέτρους με τις οποίες καθορίζουμε το διάστημα στο οποίο θα κυμαίνονται τα ratings των reviews που χρησιμοποιούμε για τους γράφους, το σύνολο εκπαίδευσης ή το σύνολο ελέγχου. Αυτές είναι:

12. `minPosRating`: καθορίζει το μικρότερο θετικό rating που έχουν τα reviews που λαμβάνονται υπόψη.

13. `maxPosRating`: καθορίζει το μεγαλύτερο θετικό rating που έχουν τα reviews που λαμβάνονται υπόψη.

14. `minNegRating`: καθορίζει το μικρότερο αρνητικό rating που έχουν τα reviews που λαμβάνονται υπόψη.

15. `maxNegRating`: καθορίζει το μεγαλύτερο αρνητικό rating που έχουν τα reviews που λαμβάνονται υπόψη.

4.2.1 Παραδοχές

Για την απλοποίηση των πειραμάτων θεωρούμε πάντα τα εξής:

- Το πλήθος των `graph reviews` είναι ίδιο και για τις δυο μεθόδους των γράφων.
- Η παράμετρος n των n-gram γράφων τίθεται πάντα ίση με το μήκος παραθύρου D_{win}, όπως συνηθίζεται στη σχετική βιβλιογραφία (βλέπε [1]). Επομένως, για τους n-gram γράφους καθορίζουμε μόνο την παράμετρο n και στην συνέχεια θέτουμε το D_{win} ίσο με n.
- Με βάση την προηγούμενη σημείωση η παράμετρος D_{win} πλέον αφορά μόνο το μήκος παραθύρου των γράφων λέξεων στην υλοποίηση αυτή.

4.3 Σταθεροποίηση τιμών παραμέτρων

Κατά την τελική σύγκριση των τριών μεθόδων επιλέγουμε τιμές για τις παραμέτρους ώστε να έχουμε σχεδόν την ίδια ακρίβεια από πείραμα σε πείραμα. Για το σκοπό αυτό προηγήθηκε διεξαγωγή απλών πειραμάτων, που παρουσιάζονται στην ενότητα αυτή, κατά τα οποία μεταβάλλουμε τις τιμές βασικών παραμέτρων με σκοπό να μελετήσουμε την μεταβολή της ακρίβειας που επιτυγχάνει κάθε μέθοδος. Σε κάθε πείραμα μεταβάλλουμε μία βασική παράμετρο, ενώ επιλέγουμε γενικά αποδεκτές τιμές για τις υπόλοιπες. Στο τέλος, επιλέγουμε μία βέλτιστη τιμή για την υπο μελέτη παράμετρο. Παρακάτω δίνουμε τις τιμές των υπόλοιπων παραμέτρων που χρησιμοποιήθηκαν σε όλα τα πειράματα έυρεσης βέλτιστων τιμών για τις βασικές παραμέτρους:

1. `minPosRating`: 7
2. `maxPosRating`: 10
3. `minNegRating`: 0
4. `maxNegRating`: 4
5. `positive rate`: 50
6. `remove`: `true`

7. `shuffle`: `true`

8. `preprocess`: `false`

9. `classifier`: `weka.classifiers.bayes.NaiveBayes`

4.3.1 Σταθεροποίηση του πλήθους των reviews για το training και το test set

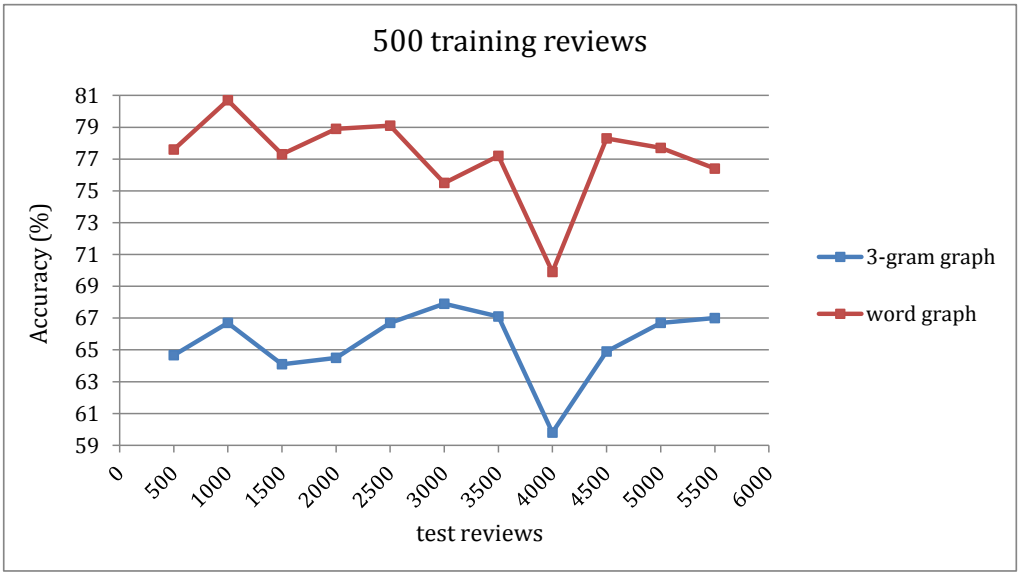
Στην ενότητα αυτή μελετάμε την μεταβολή της ακρίβειας σχετικά με την μεταβολή του πλήθους των training και test instances. Κάθε instance αντιστοιχεί σε ένα review, επομένως, αυξάνοντας το πλήθος των reviews αυξάνουμε τα instances του συνόλου. Για την σταθεροποίηση της παραμέτρου αυτής, διεξήχθηκαν πειράματα με τις παρακάτω τιμές:

1. $n = D_{win} = 3$ για τους n-gram γράφους (βλέπε ενότητα 4.2.1).

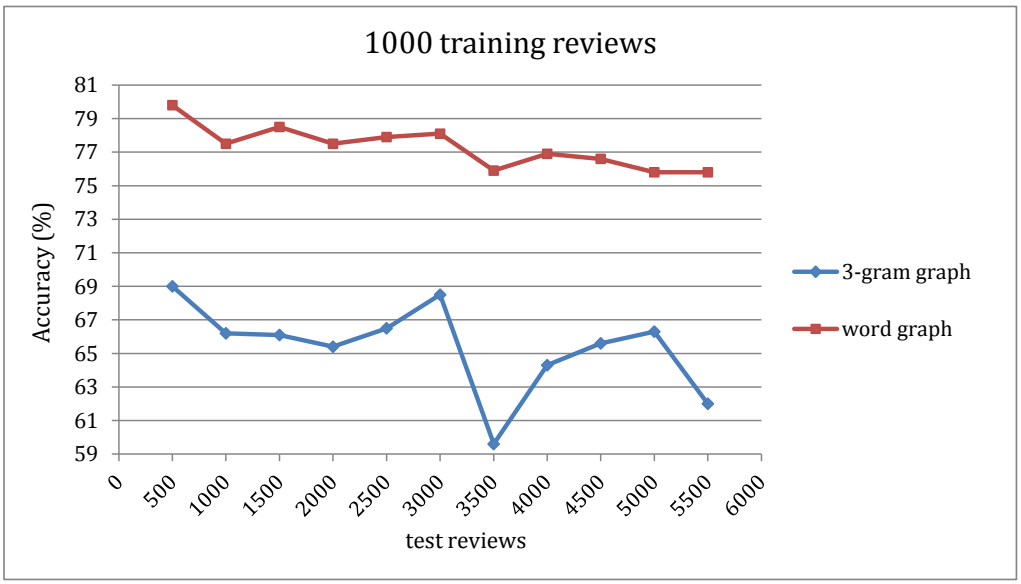
2. $D_{win} = 3$ για τους γράφους λέξεων.

3. `graph reviews`: 2000.

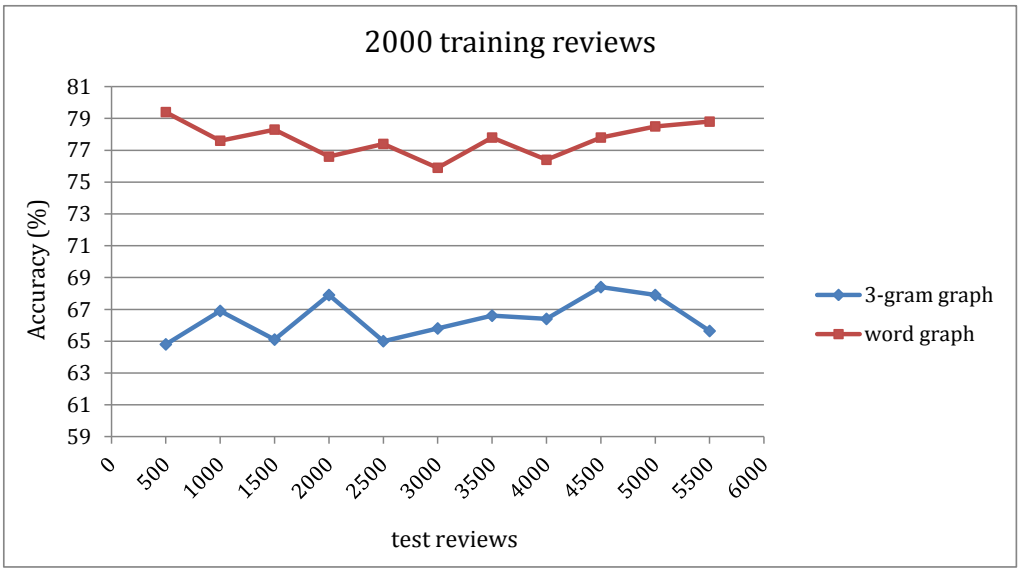
Στα πειράματα μεταβάλλαμε το πλήθος των test instances από 500 έως 5500 με βήμα 500. Επιπλέον, θέτουμε αρχικά το πλήθος των training instances σε 500 και στην συνέχεια το μεταβάλλουμε από 1000 έως 5000 με βήμα 1000. Τα αποτελέσματα κάθε τιμής ακρίβειας συγκεντρώθηκαν σε ένα διάγραμμα για κάθε τιμή του πλήθους των training instances και φαίνονται στα σχήματα 4.1 έως 4.6.



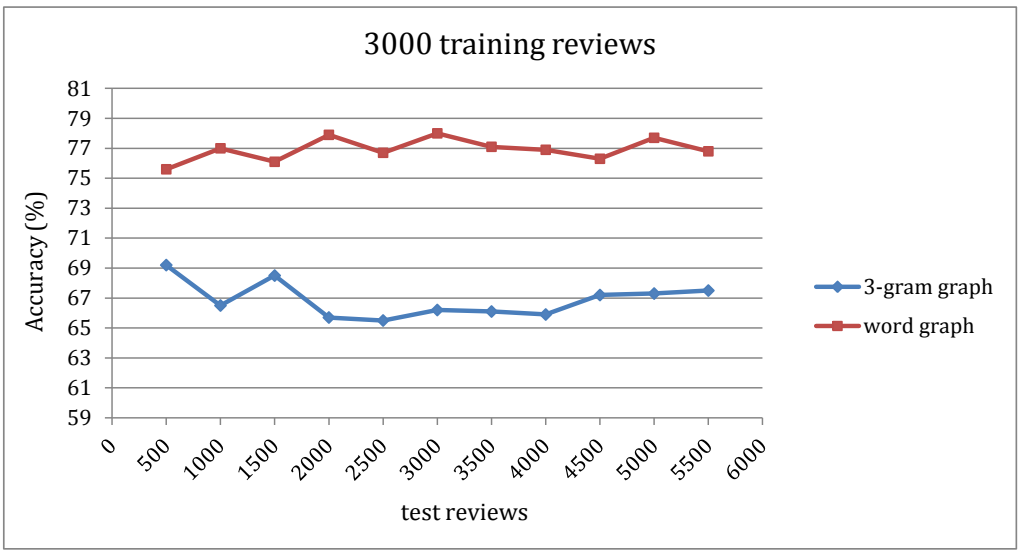
Σχήμα 4.1: Μεταβολή της ακρίβειας καθώς αυξάνονται τα test instances για 500 reviews στο training set.



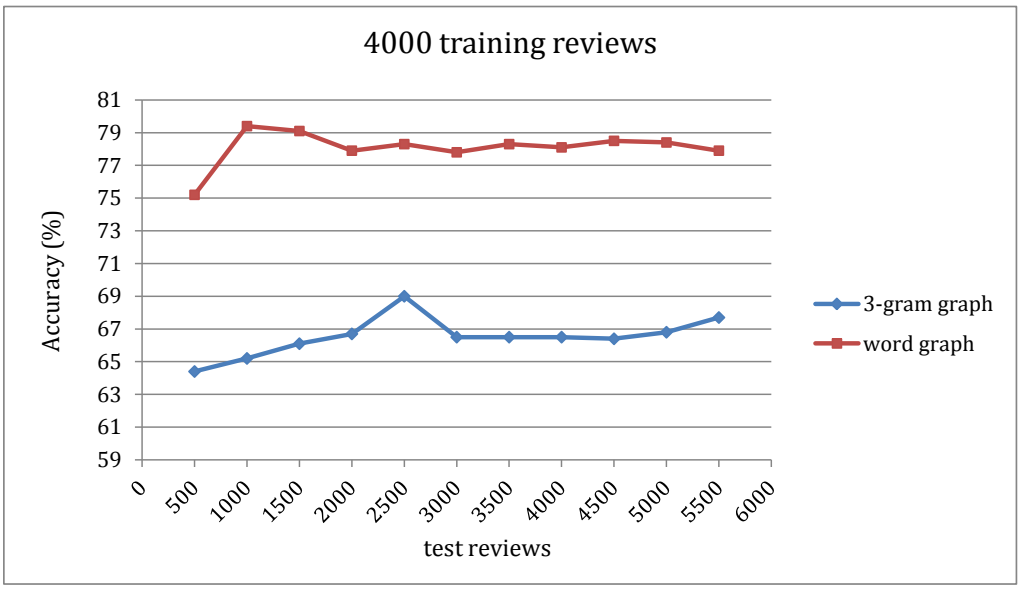
Σχήμα 4.2: Μεταβολή της ακρίβειας καθώς αυξάνονται τα test instances για 1000 reviews στο training set.



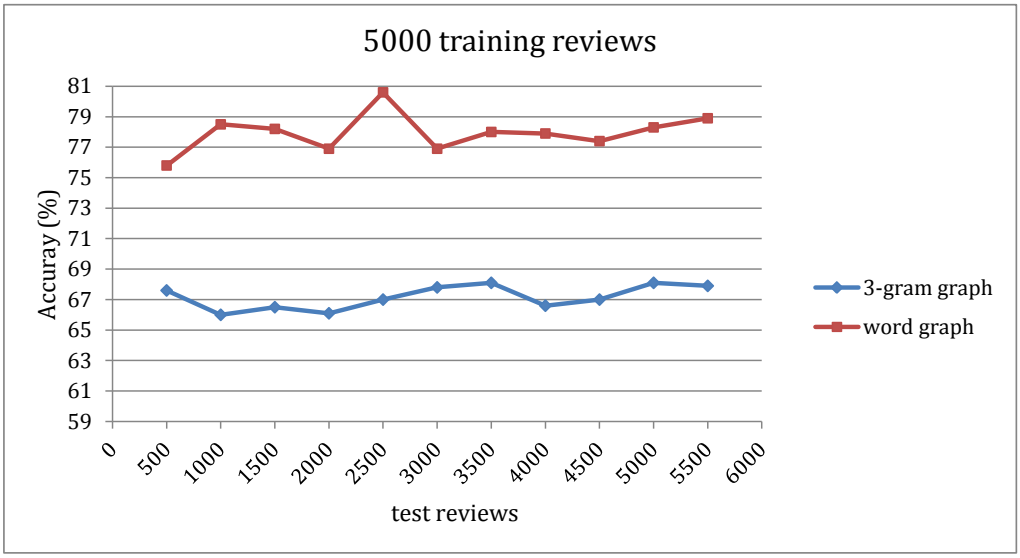
Σχήμα 4.3: Μεταβολή της ακρίβειας καθώς αυξάνονται τα test instances για 2000 reviews στο training set.



Σχήμα 4.4: Μεταβολή της ακρίβειας καθώς αυξάνονται τα test instances για 3000 reviews στο training set.



Σχήμα 4.5: Μεταβολή της ακρίβειας καθώς αυξάνονται τα test instances για 4000 reviews στο training set.



Σχήμα 4.6: Μεταβολή της ακρίβειας καθώς αυξάνονται τα test instances για 5000 reviews στο training set.

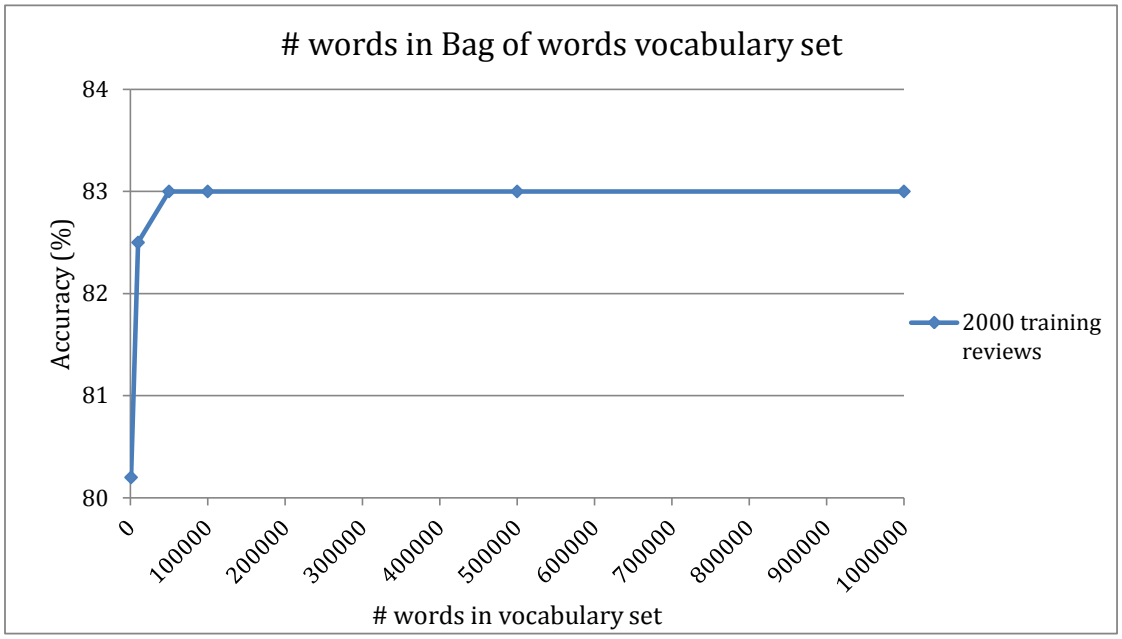
Από τα διαγράμματα που προέκυψαν παρατηρούμε ότι όσο αυξάνουμε το μέγεθος του training και του test set έχουμε μικρότερη διακύμανση στην τιμή της ακρίβειας κάθε μεθόδου. Οι περιπτώσεις επιλογής 500 ή 1000 reviews στο σύνολο εκπαίδευσης φαίνεται να μην επιφέρουν σύγκλιση της τιμής της ακρίβειας γύρω από μία τιμή (σχήματα 4.1, 4.2). Προκειμένου να έχουμε αξιόπιστη εικόνα της επίδοσης των μεθόδων πρέπει να επιλέξουμε τουλάχιστον 2000 training reviews. Για λόγους οικονομίας πόρων και χρόνου επιλέγουμε ακριβώς 2000 reviews για το σύνολο εκπαίδευσης των επόμενων πειραμάτων.

Αντίστοιχα για το σύνολο ελέγχου (test set) παρατηρούμε ότι η διακύμανση μειώνεται ιδιαίτερα όταν χρησιμοποιούμε περισσότερα από 2500 test reviews (σχήματα 4.3, 4.4, 4.5, 4.6). Η διακύμανση όμως, είναι πολύ μικρή και μπορεί να οφείλεται στα δεδομένα. Επομένως, για λόγους οικονομίας χρόνου επιλέγουμε 1000 reviews για το σύνολο ελέγχου των επόμενων πειραμάτων.

4.3.2 Σταθεροποίηση πλήθους λέξεων στο vocabulary set

Το πλήθος των λέξεων που αποθηκεύονται στο λεξιλόγιο της bag of words προσέγγισης καθορίζεται από ένα άνω όριο λέξεων που δίνουμε ως είσοδο. Ο λέξεις του λεξιλογίου προέρχονται από το training set. Για την εύρεση ενός ικανοποιητικού ορίου λέξεων σταθεροποιούμε το πλήθος των training reviews σε 2000, όπως υπολογίστηκε στην προηγούμενη ενότητα. Έτσι, μπορούμε να υπολογίσουμε μία 'βέλτιστη' τιμή για το μέγεθος του λεξιλογίου που δημιουργείται από τις λέξεις των 2000 reviews του training set. Επιπλέον,

θέτουμε 1000 reviews στο test set, και μεταβάλλουμε το άνω όριο του πλήθους των λέξεων από 1000 έως 1000000 για να μελετήσουμε την μεταβολή της ακρίβειας. Στην περίπτωση αυτή, χρησιμοποιούμε τον γνωστό για την προσέγγιση bag of words ταξινομητή multinomial naive Bayes. Το πείραμα αυτό αφορά μόνο την bag of words προσέγγιση. Το αποτέλεσμα παρουσιάζεται στο σχήμα 4.7.



Σχήμα 4.7: Μεταβολή της ακρίβειας σχετικά με το μέγεθος του λεξιλογίου της bag of words προσέγγισης.

Παρατηρούμε ότι όσο αυξάνουμε το άνω όριο του πλήθους των λέξεων τόσο αυξάνεται η ακρίβεια έως ότου επέλθει κορεσμός. Αυτό οφείλεται στο ότι η μέθοδος προσπαθεί να δημιουργήσει ένα διάνυσμα με μέγεθος κοντά στο άνω όριο του πλήθους λέξεων που θέσαμε και οι λέξεις που χρησιμοποιούνται προέρχονται από τα 2000 training reviews. Επομένως, οι διακριτές λέξεις που εξάγονται για το διάνυσμα δεν μπορούν να ξεπεράσουν συνολικά μία τιμή. Επομένως, μπορούμε να θέσουμε το άνω όριο του πλήθους των λέξεων σε 50000 ή 100000 όταν χρησιμοποιούμε 2000 training reviews.

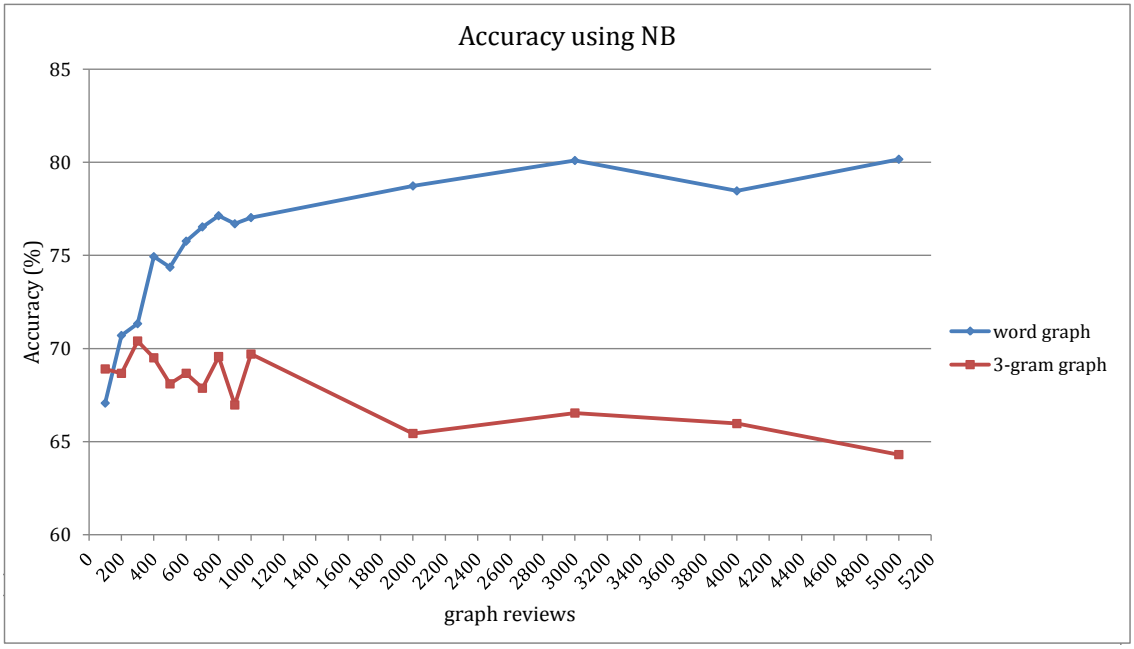
4.3.3 Σταθεροποίηση πλήθους reviews για τους γράφους

Το πλήθος των reviews που ενσωματώνει ο κάθε γράφος πολικότητας αποτελεί μία σημαντική παράμετρος, καθώς ο γράφος αντιπροσωπεύει κείμενα της αντίστοιχης πολικότητας και από αυτόν εξάγονται τιμές ομοιότητας με τους γράφους των reviews που θα αποτελέσουν τα instances εκπαίδευσης και ελέγχου. Επομένως, επιθυμούμε την ενσωμάτωση μεγάλου πλήθους reviews, ώστε ο γράφος να 'αναγνωρίζει' τα reviews της ίδιας πολικότητας με αυτόν.

Για την εύρεση μίας καλής τιμής για την παράμετρο αυτή, χρησιμοποιήσαμε τις παρακάτω τιμές για τις παραμέτρους.

1. $D_{win} = n = 3$ για τους n-gram γράφους.
2. $D_{win} = 3$ για τους γράφους λέξεων.
3. `training reviews`: 2000.
4. `test reviews`: 1000.

Το πλήθος των graph reviews μεταβάλλεται από 100 έως 1000 με βήμα 100 και στην συνέχεια από 1000 έως 5000 με βήμα 1000. Κάθε πείραμα με συγκεκριμένο πλήθος graph reviews (π.χ. 5000) διεξήχθηκε τρεις φορές με σκοπό να υπολογίσουμε το μέσο όρο της ακρίβειας και να αποφύγουμε τυχόν διακυμάνσεις που οφείλονται σε θόρυβο των δεδομένων. Το αποτέλεσμα των πειραμάτων συγκεντρώθηκε στο διάγραμμα 4.8.



Σχήμα 4.8: Μεταβολή της ακρίβειας καθώς αυξάνονται τα reviews σε κάθε γράφο πολικότητας (NB: naive Bayes).

Το διάγραμμα αυτό δηλώνει ξεκάθαρα την υπεροχή της χρήσης λέξεων σχετικά με την χρήση 3-grams χαρακτήρων στους γράφους. Σε επόμενη ενότητα παρατηρούμε την επίδοση για διαφορετικά μήκη παραθύρου καθώς και για χρήση 4-grams χαρακτήρων στους γράφους, οι οποίοι συνηθίζουν να επιτυγχάνουν μεγαλύτερη ακρίβεια από τους 3-gram γράφους.

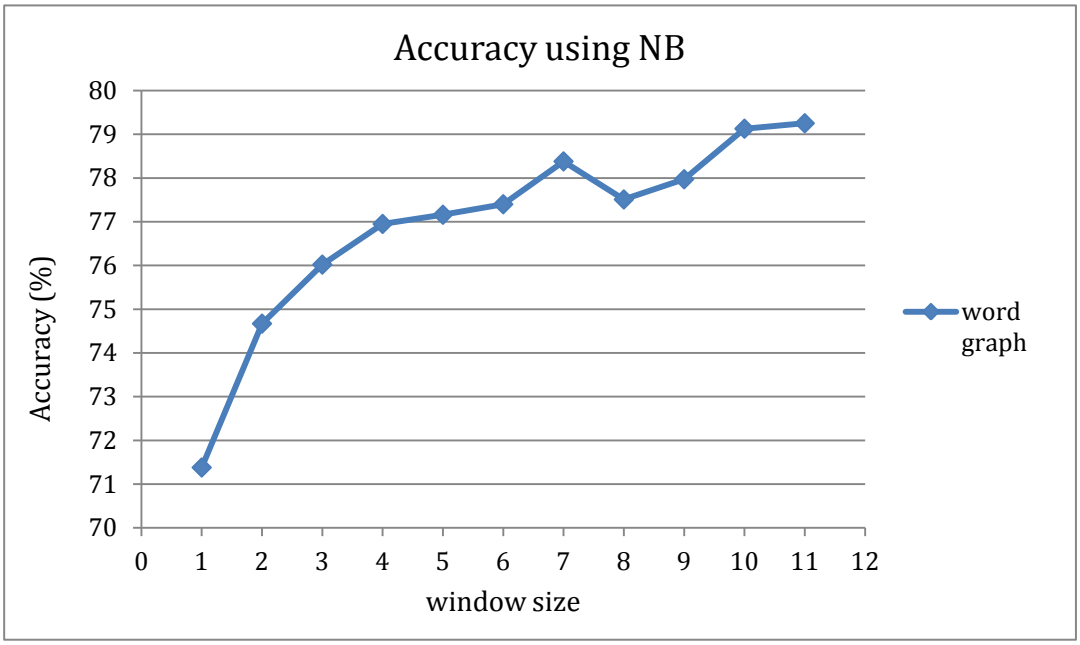
Σκοπός μας είναι η επιλογή μίας τιμής γύρω από την οποία δεν υπάρχει μεγάλη διακύμανση της ακρίβειας των μεθόδων. Παρατηρώντας το διάγραμμα διαπιστώνουμε ότι όλες

οι τιμές πάνω από 1000 reviews είναι ικανοποιητικές για την σύγκρισή τους, καθώς φαίνεται να σταθεροποιείται η ακρίβεια γύρω από μία τιμή. Λόγω της μεγάλης κατανάλωσης χρόνου για τα επόμενα πειράματα επιλέγουμε 800 reviews για την ενσωμάτωση στον κάθε γράφο πολικότητας, καθώς αποτελούν μία περίπτωση όπου η ακρίβεια είναι κοντά στην τιμή που επιτυγχάνεται με περισσότερα reviews. Στο τέλος του κεφαλαίου διεξάγουμε πειράματα με περισσότερα graph reviews ώστε να μελετήσουμε την σχετική εξέλιξη της ακρίβειας των δυο μεθόδων έχοντας σταθεροποιήσει τις υπόλοιπες παραμέτρους σε βέλτιστες τιμές.

4.3.4 Σταθεροποίηση μήκους παραθύρου των γράφων λέξεων.

Το μήκος παραθύρου είναι μία τιμή που δηλώνει το παράθυρο από το οποίο εξάγουμε γειτονικές λέξεις κάθε φορά (Ενότητα 3.4.1). Για την επιλογή μίας καλής τιμής για αυτήν την παράμετρο διεξάγουμε το παρακάτω πείραμα με τις τιμές παραμέτρων που προέκυψαν από τα συμπεράσματα των προηγούμενων ενοτήτων:

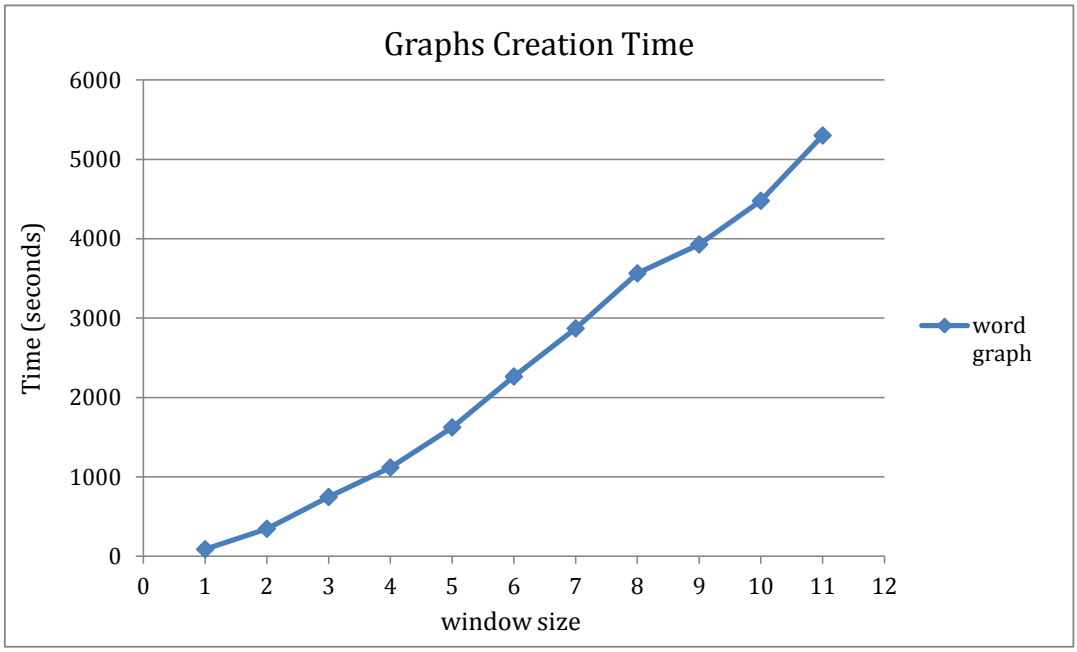
1. `graph reviews`: 800
2. `training reviews`: 2000
3. `test reviews`: 1000



Σχήμα 4.9: Μεταβολή της ακρίβειας καθώς αυξάνενται το μήκος παραθύρου (NB: naive Bayes).

Στα πειράματα η τιμή του μήκους παραθύρου μεταβάλλεται από 1 έως 11. Για κάθε συγκεκριμένη τιμή του μήκους παραθύρου διεξήχθηκαν 10 διαφορετικά πειράματα και υπολογίσαμε το μέσο όρο της ακρίβειας για να αποφύγουμε τυχόν διακυμάνσεις. Το αποτέλεσμα παρουσιάζεται στο διάγραμμα 4.9.

Παρατηρούμε ότι η ακρίβεια αυξάνεται αρχικά και παραμένει σχεδόν σταθερή μετά από την τιμή 4. Επομένως, διαπιστώνουμε ότι η αποθήκευση της γειτνίασης των λέξεων είναι σημαντική για την επίδοση των γράφων με χρήση του αλγορίθμου naive Bayes. Αξίζει να σημειωθεί ότι όσο αυξάνουμε τις γειτονικές λέξεις που αποθηκεύονται αυξάνεται και ο χρόνος δημιουργίας του γράφων, όπως φαίνεται στο διάγραμμα 4.10.



Σχήμα 4.10: Χρόνοι δημιουργίας γράφων για κάθε μήκος παραθύρου.

Παρατηρώντας το διάγραμμα της ακρίβειας διαπιστώνουμε ότι η επιλογή μεγαλύτερου παραθύρου δεν επιφέρει χειρότερη ακρίβεια. Αυτό οφείλεται στο ότι η αποθήκευση περισσότερων γειτονικών λέξεων δεν επηρεάζει αρνητικά τις ομοιότητες που προέκυπταν μεταξύ γράφων με μικρότερο μήκος παραθύρου. Οι επιπλέον ακμές που τοποθετούνται επιφέρουν αν όχι μεγαλύτερη ακρίβεια, τουλάχιστον την ακρίβεια που θα προέκυπτε για μικρότερο μήκος παραθύρου, δηλαδή για λιγότερες ακμές. Η τυχόν μικρή διακύμανση οφείλεται στην χρήση διαφορετικών δεδομένων. Όμως, ο χρόνος δημιουργίας των γράφων αυξάνεται γραμμικά με την αύξηση των γειτονικών λέξεων που τοποθετούνται. Επομένως, για την επιλογή μίας τιμής για το μήκος παραθύρου χρησιμοποιούμε το συμπέρασμα της ερευνητικής εργασίας [1], όπου το μήκος παραθύρου για τους n-gram γράφους τίθεται όσο το n και επιφέρει την μέγιστη επίδοση. Κατά την δημιουργία των γράφων λέξεων απαλείψαμε την παράμετρο n των n-gram γράφων που καθορίζει το πλήθος των χαρακτήρων που εξάγονται και έχουμε λέξεις με μεταβλητό πλήθος χαρακτήρων. Για το λόγο αυτό, στην περίπτωση των γράφων

λέξεων επιλέγουμε μήκος παραθύρου που αντιστοιχεί στο μέσο μήκος μίας αγγλικής λέξης, δηλαδή 6. Αυτή η τιμή επιφέρει ακρίβεια κοντά στις τιμές ακρίβειας που επιτυγχάνουμε με μεγαλύτερο παράθυρο. Επιπλέον, είναι μία τιμή όπου ο χρόνος δημιουργίας των γράφων είναι σχεδόν μισή ώρα.

4.4 Σύγκριση

Στο σημείο αυτό χρησιμοποιούμε τις βέλτιστες τιμές των παραμέτρων που επιλέξαμε στην προηγούμενη ενότητα για την σύγκριση των τριών μεθόδων. Κατά την σύγκριση επικεντρωθήκαμε στον naive Bayes ταξινομητή.

4.4.1 Χρήση naive Bayes

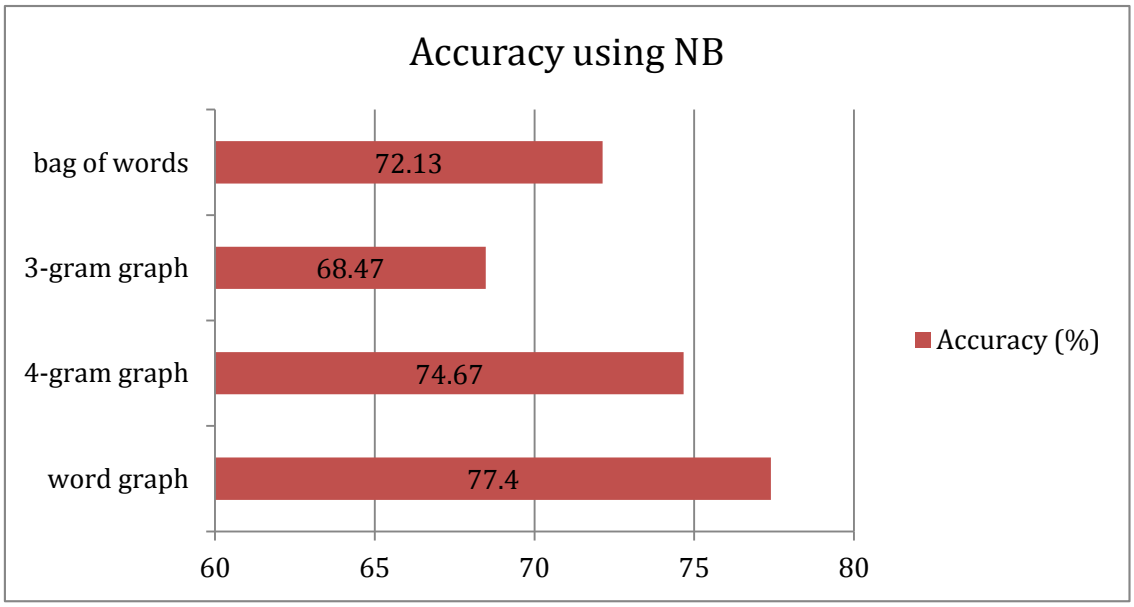
Στην ενότητα αυτή, θα μελετήσουμε την επίδοση με χρήση του naive Bayes ταξινομητή (ενότητα 2.5.1.4). Δηλαδή θέτουμε:

- `classifier`: `weka.classifiers.bayes.NaiveBayes`

Για ευκολία παρουσιάζουμε τις υπόλοιπες παραμέτρους:

1. $D_{win} = 6$ (γράφοι λέξεων)
2. `graph reviews`: 800
3. `training reviews`: 2000
4. `test reviews`: 1000
5. $D_{win} = n = 4$ (n-gram γράφοι)
6. `remove`: `true`
7. `shuffle`: `true`
8. `preprocess`: `false`

Θέτοντας αυτές τις παραμέτρους τρέξαμε τρία πειράματα και υπολογίσαμε το μέσο όρο της ακρίβειας. Στην συνέχεια τρέξαμε τρία πειράματα με $n = 3$ για τους n-gram γράφους, καθώς αποτελεί συχνά χρησιμοποιούμενη τιμή μαζί με την τιμή $n = 4$, και υπολογίσαμε το μέσο όρο της ακρίβειάς τους.



Σχήμα 4.11: Ακρίβεια που επιτυγχάνει κάθε μέθοδος με χρήση naive Bayes.

4.4.1.1 Ακρίβεια

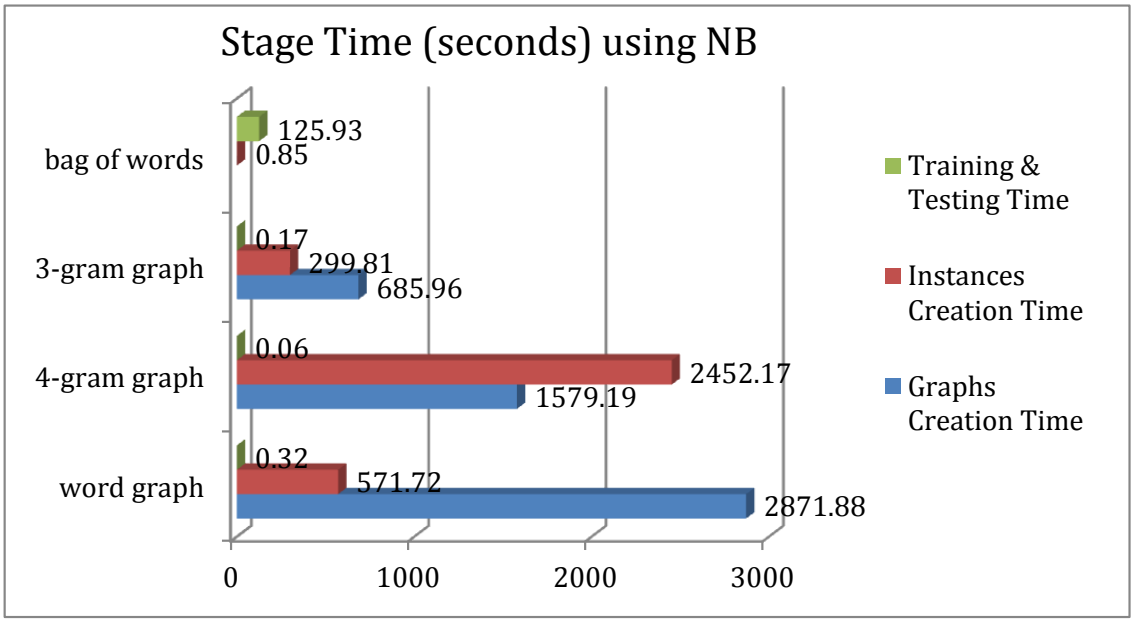
Στο διάγραμμα που φαίνεται στο σχήμα 4.11 συγκεντρώσαμε τον μέσο όρο της ακρίβειας των πειραμάτων. Παρατηρούμε ότι η μέθοδος των γράφων λέξεων επιτυγχάνει την μεγαλύτερη ακρίβεια. Αρχικά, η βελτίωση των n- gram γράφων σχετικά με την πρώτη σύγκριση που παρουσιάστηκε στην ενότητα 3.1.1 οφείλεται στην αφαίρεση του κοινού υπογράφου μεταξύ των δυο γράφων πολικότητας. Η επίδραση του κοινού υπογράφου μελετάται στην ενότητα 4.6. Η περαιτέρω βελτίωση της μεθόδου των n-gram γράφων επιτυγχάνεται με την χρήση λέξεων το οποίο αντιστοιχεί στην μέθοδο word graph. Αυτό οφείλεται στο ότι αποφεύγουμε πλέον την εμφάνιση ίδιων n-grams στους γράφους πολικότητας από λέξεις διαφορετικής πολικότητας, σημαντικές για την εξαγωγή συναισθήματος. Η μέθοδος bag of words επιτυγχάνει μικρότερη ακρίβεια με χρήση του naive Bayes από ότι με την χρήση του multinomial naive bayes που είδαμε στην πρώτη σύγκριση τους στο Κεφάλαιο 3, γιατί είναι διαφορετική η υπόθεση για την κατανομή των δεδομένων των χαρακτηριστικών, όπως αναφέραμε στην ενότητα 2.5.1.2 για τους naive Bayes ταξινομητές. Για την καλύτερη κατανόηση της επίδοσης κάθε μεθόδου συγκρίνουμε σε επόμενες ενότητες και την κατανάλωση πόρων της κάθε μίας.

4.4.1.2 Χρόνοι εκτέλεσης

Είναι σημαντικό να αξιολογήσουμε τις μεθόδους και με βάση το χρόνο που καταναλώνουν για την επίτευξη αυτής της επίδοσης. Στο διάγραμμα 4.12 παρουσιάζουμε τους χρόνους κάθε σταδίου για κάθε μέθοδο, που αντιστοιχούν στον μέσο όρο των χρόνων των τριών

πειραμάτων. Υπενθυμίζουμε τα στάδια της ενότητας 3.5.2.

1. Πρώτο στάδιο: Δημιουργία γράφων (Στα διαγράμματα: Graphs Creation).
2. Δεύτερο στάδιο: Δημιουργία αρχείων με training και test instances (Στα διαγράμματα: Instances Creation).
3. Τρίτο στάδιο: Δημιουργία ταξινομητή και αξιολόγησή του (Στα διαγράμματα: Training & Testing).

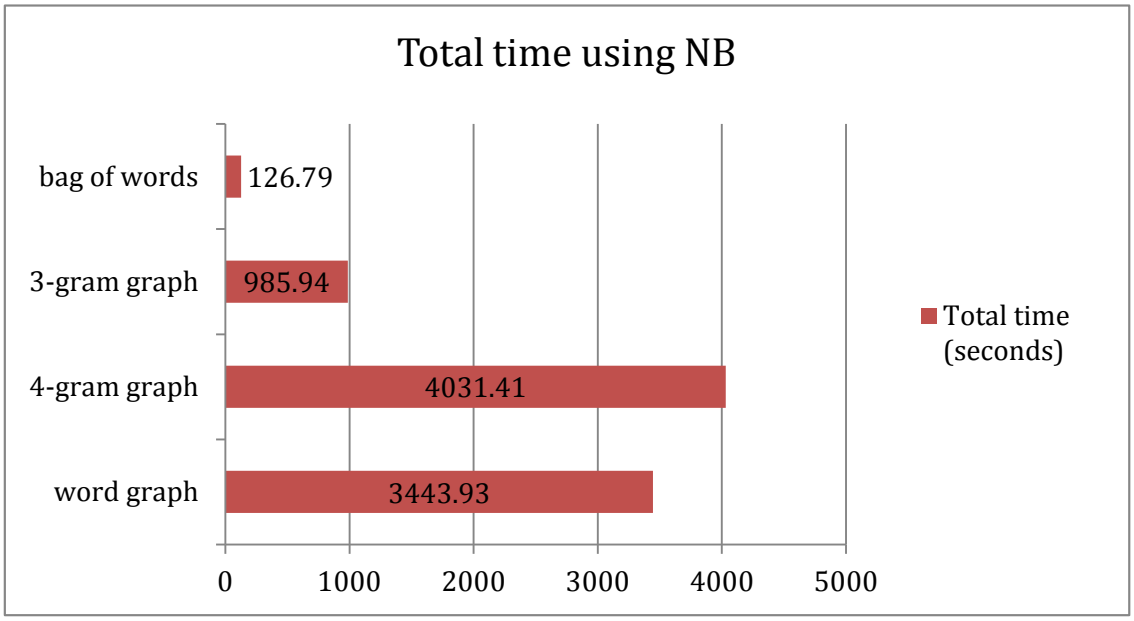


Σχήμα 4.12: Χρόνοι 3 σταδίων για κάθε μέθοδο.

Ο χρόνος εκπαίδευσης με 2000 reviews μαζί με τον χρόνο ταξινόμησης των 1000 reviews είναι λιγότερο από ένα δευτερόλεπτο για τους γράφους ενώ είναι δυο λεπτά για τη μέθοδο bag of words. Αυτό οφείλεται στο ότι η μέθοδος bag of words χρησιμοποιεί ένα μεγάλο πλήθος από features που αντιστοιχούν σε λέξεις. Αντίθετα, και οι τρεις μέθοδοι γράφων χρησιμοποιούν μόνο 6 features που αντιστοιχούν σε τιμές ομοιότητας γράφων. Επομένως, η χρήση των μεθόδων γράφων για την κατηγοριοποίηση του συναισθήματος ενός κειμένου είναι πολύ γρήγορη.

Από την άλλη, η μέθοδος bag of words δεν περιλαμβάνει τη διαδικασία δημιουργίας γράφων η οποία μπορεί να θεωρηθεί διαδικασία προεπεξεργασίας, καθώς προηγείται της διαδικασίας της μηχανικής μάθησης. Οι μέθοδοι των γράφων επομένως, αντισταθμίζουν το μικρό χρόνο ταξινόμησης των reviews με κατανάλωση μεγάλου χρόνου για την δημιουργία των γράφων πριν το στάδιο της μηχανικής μάθησης. Οι χρόνοι δημιουργίας των γράφων παρουσιάζονται επίσης στο διάγραμμα 4.12. Οι χρόνοι αυτοί μπορεί να μην ληφθούν υπόψη

αν σκεφτεί κανείς ότι οι γράφοι δημιουργούνται μία φορά και στην συνέχεια φορτώνονται στη μνήμη για την χρήση σε κάποια εφαρμογή ταξινόμησης συναισθήματος κειμένου. Όμως, είναι σημαντικά μεγαλύτερος ο συνολικός χρόνος για τις μεθόδους των γράφων ο οποίος παρουσιάζεται στο διάγραμμα 4.13. Σε αυτό, βλέπουμε ότι η μέθοδος γράφων λέξεων είναι πιο γρήγορη από τη μέθοδο των 4-gram γράφων. Αυτό οφείλεται στην πιο γρήγορη εξαγωγή ομοιοτήτων μεταξύ γράφων λέξεων από ότι μεταξύ 4-gram γράφων. Από την άλλη παρατηρούμε ότι ο χρόνος δημιουργίας των γράφων λέξεων είναι μεγαλύτερος από τον χρόνο δημιουργίας των n-gram γράφων.

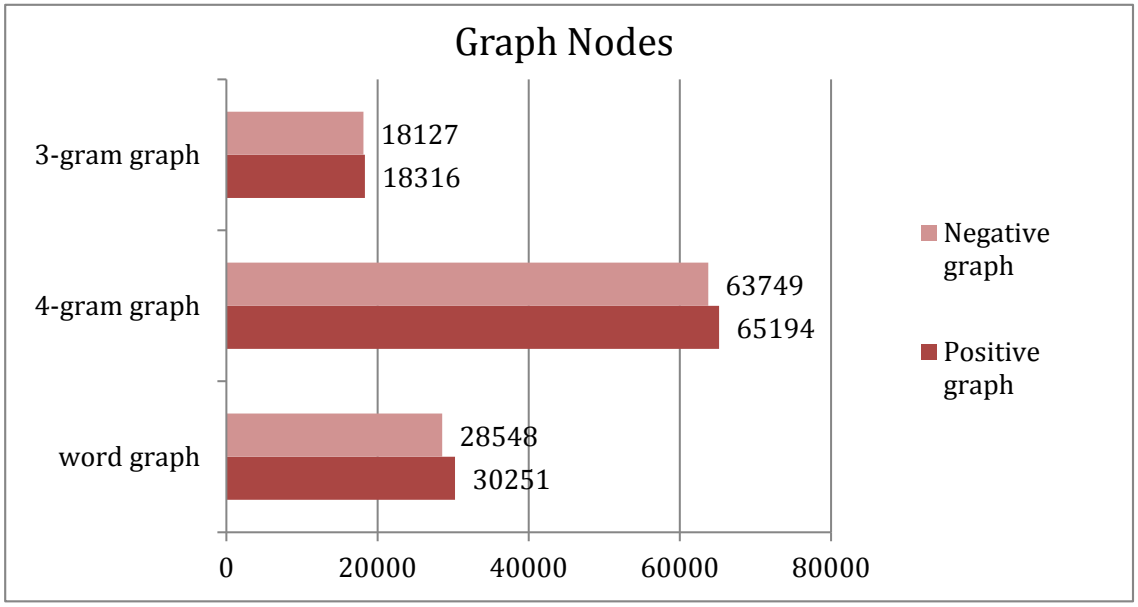


Σχήμα 4.13: Συνολικός χρόνος για κάθε μέθοδο.

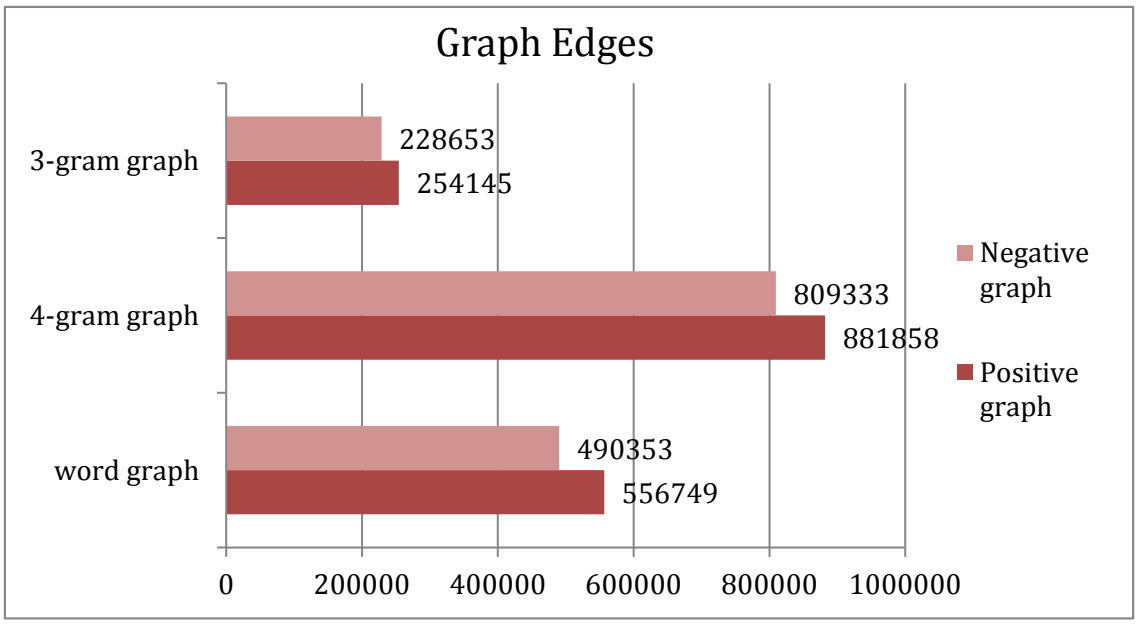
4.4.1.3 Μεγέθη γράφων

Στην ενότητα αυτή συγκρίνουμε την κατανάλωση χώρου των τριών μεθόδων των γράφων. Για το σκοπό αυτό εξάγουμε το πλήθος των κόμβων και των ακμών που περιέχει κάθε γράφος πολικότητας για αυτές τις μεθόδους. Τα αποτελέσματα συγκεντρώνονται στα σχήματα 4.14 και 4.15.

Και στα δυο διαγράμματα παρατηρούμε ότι η μέθοδος των γράφων λέξεων καταναλώνει λιγότερο 'χώρο' από ότι η μέθοδος των 4-gram γράφων. Αυτό οφείλεται στο ότι οι κοινές λέξεις μεταξύ των reviews συγχωνεύονται σε ένα κόμβο και έτσι μειώνουν τους κόμβους που χρησιμοποιούν τα 4-grams. Επιβεβαιώνουμε με αυτόν τον τρόπο το θόρυβο που δημιουργεί η εξαγωγή 4-grams. Αντίστοιχα, και οι ακμές που εμφανίζονται στους γράφους λέξεων είναι πολύ λιγότερες, εφόσον έχουμε πολύ λιγότερους κόμβους και έχουμε καταφέρει να συλλάβουμε το περιεχόμενο του κειμένου, καθώς ένα ζευγάρι λέξεων μπορεί να υπάρχει ήδη



Σχήμα 4.14: Σύγκριση κόμβων γράφων πολικότητας με 800 reviews ο καθένας.



Σχήμα 4.15: Σύγκριση ακμών γράφων πολικότητας με 800 reviews ο καθένας.

στον γράφο με αποτέλεσμα να χρειάζεται ενημέρωση του βάρους της ακμής και όχι τοποθέτηση νέας ακμής. Η μέθοδος των 3-gram γράφων καταναλώνει πολύ λιγότερο χώρο λόγω της μικρής τιμής n η οποία συνεπάγεται πολύ λιγότερο συνδυασμό n-grams που μπορούν να εμφανιστούν στο γράφο. Με την χρήση των 3-grams χάνεται όμως, η δυνατότητα σύλληψης του περιεχομένου και οδηγούμαστε σε μικρότερη ακρίβεια παρόλο το μεγάλο πλήθος των reviews.

4.4.1.4 Περισσότερα reviews στους γράφους πολικότητας

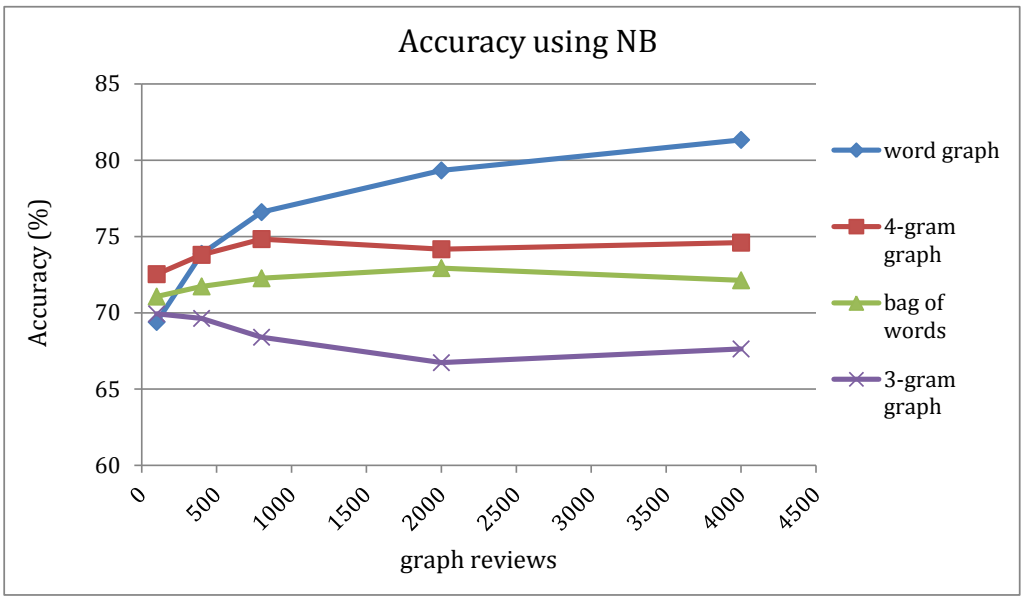
Στην ενότητα αυτή μελετάμε την επίδοση της μεθόδου των γράφων λέξεων μεταβάλλοντας πλέον το πλήθος των reviews το κείμενο των οποίων αποθηκεύουν οι γράφοι πολικότητας (graph reviews). Ο λόγος που επιθυμούμε να αυξήσουμε το πλήθος των reviews των γράφων είναι επειδή όταν σταθεροποιήσαμε την τιμή τους σε 800 reviews στην Ενότητα 4.3.3 είχαμε παρατηρήσαμε μεγαλύτερη ακρίβεια για μεγαλύτερο πλήθος reviews, αλλά επιλέξαμε τα 800 reviews με σκοπό να διευκολύνουμε την διεξαγωγή των επόμενων πειραμάτων της εργασίας αυτής. Έχει ήδη αναφερθεί ότι η χρήση πολλών reviews στους γράφους δεν θα πρέπει να μας ενδιαφέρει, καθώς αυτοί δημιουργούνται μία φορά για την χρησιμοποίησή τους σε κάποια εφαρμογή. Η ενημέρωση των γράφων με επιπλέον δεδομένα - reviews εξαρτάται από την εφαρμογή και δεν αποτελεί μέρος της διαδικασίας ταξινόμησης κάθε κειμένου αλλά μπορεί να συμβαίνει ανά διαστήματα όταν το απαιτεί η εφαρμογή. Τα πειράματα που διεξήχθηκαν ήταν στην μορφή:

1. $D_{win} = 6$ (γράφοι λέξεων)
2. $D_{win} = n = 4$ (n-gram γράφοι)
3. `training reviews`: 2000
4. `test reviews`: 1000
5. `classifier`: `weka.classifiers.bayes.NaiveBayes`
6. `remove`: `true`
7. `shuffle`: `true`
8. `preprocess`: `false`

όπου το πλήθος των graph reviews παίρνει τις τιμές 100, 400, 800, 2000 και 4000 reviews. Για κάθε πλήθος των graph reviews διεξήχθηκαν τρία πειράματα από τα οποία υπολογίστηκε ο μέσος όρος της ακρίβειας. Τα αποτελέσματα δίνονται στο σχήμα 4.16.

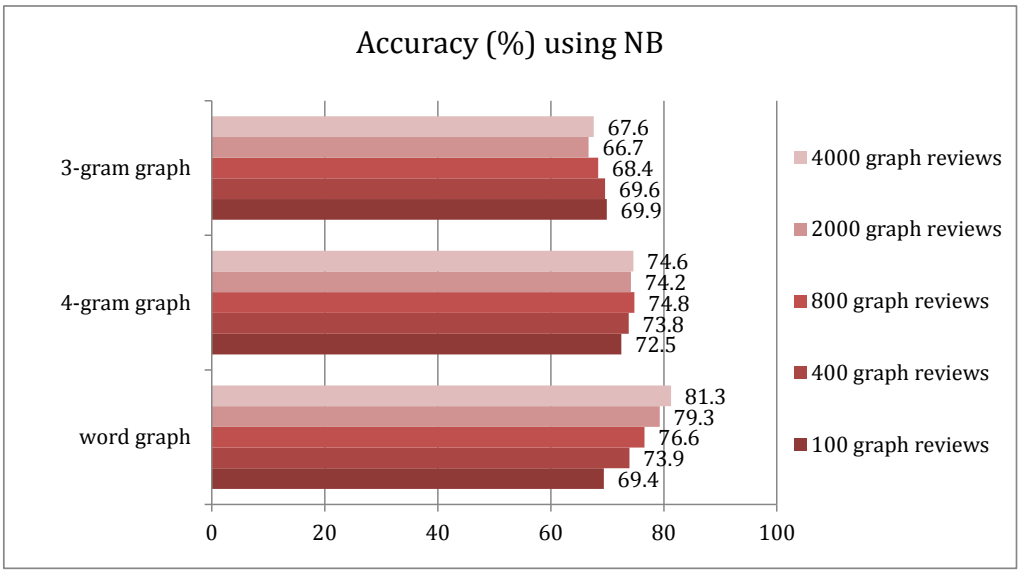
Παρατηρούμε ότι η μέθοδος των γράφων λέξεων βελτιώνει την επίδοσή της όσο αυξάνεται το πλήθος των reviews που ενσωματώνουν οι γράφοι πολικότητας. Συγκεκριμένα η ακρίβεια που επιτυγχάνουν οι μέθοδοι έχει μεγαλύτερη διαφορά από αυτήν που παρουσίαζε για 800 graph reviews. Η σταδιακή βελτίωση της μεθόδου οφείλεται στην ενσωμάτωση περισσότερων reviews, επομένως, και περισσότερων λέξεων στους γράφους μαζί με τις γειτονικές, το οποίο έχει ως αποτελέσμα την σύλληψη του τρόπου με τον οποίο εκφράζονται

για ένα θέμα οι χρήστες. Η ιδέα αυτή μπορεί να θεωρηθεί πολύ κοντινή στην διαδικασία της μεθόδου bag of words η οποία μέσω της μηχανικής μάθησης 'μαθαίνει' από την χρήση περισσότερων reviews. Αντίθετα η μέθοδος των 4- gram γράφων φαίνεται να διακυμαινέται γύρω από μία τιμή ακρίβειας χωρίς να 'μαθαίνει' από την ενσωμάτωση περισσότερων reviews, ενώ η μέθοδος των 3-gram γράφων φαίνεται να χειροτερεύει. Να σημειωθεί ότι η μέθοδος bag of words τοποθετήθηκε στο διάγραμμα για την παρουσίαση της ακρίβειας που πετυχαίνει, καθώς δεν μεταβάλλει την ακρίβεια της με την αλλαγή των reviews των γράφων αφού δεν τους χρησιμοποιεί. Η ακρίβειά της μεταβάλλεται λόγω της χρήσης διαφορετικών training και test συνόλων. Στο διάγραμμα 4.17 παρουσιάζουμε για διευκόλυνση τις ακριβείς τιμές του μέσου όρου της ακρίβειας κάθε μεθόδου (κάθε τιμή αντιστοιχεί σε μέσο όρο τριών πειραμάτων).

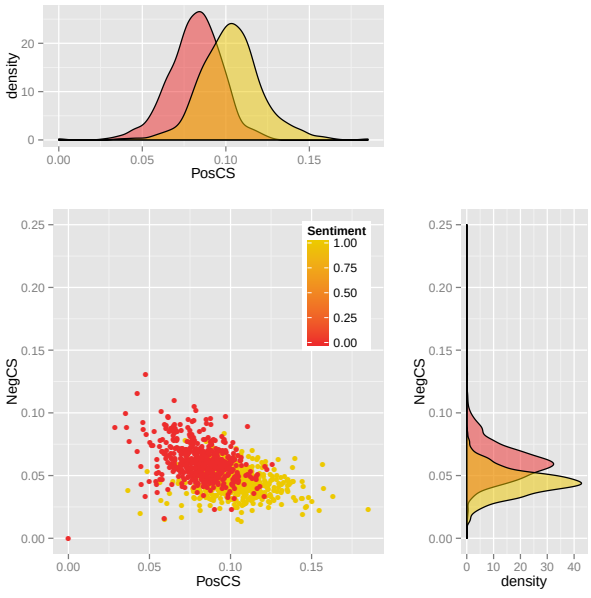


Σχήμα 4.16: Σύγκριση μεθόδων για διαφορετικό πλήθος reviews κάθε γράφου πολικότητας με χρήση naive Bayes. **Σημείωση:** Η μέθοδος bag of words δεν δημιουργεί γράφους και τοποθετήθηκε για λόγους πληρότητας.

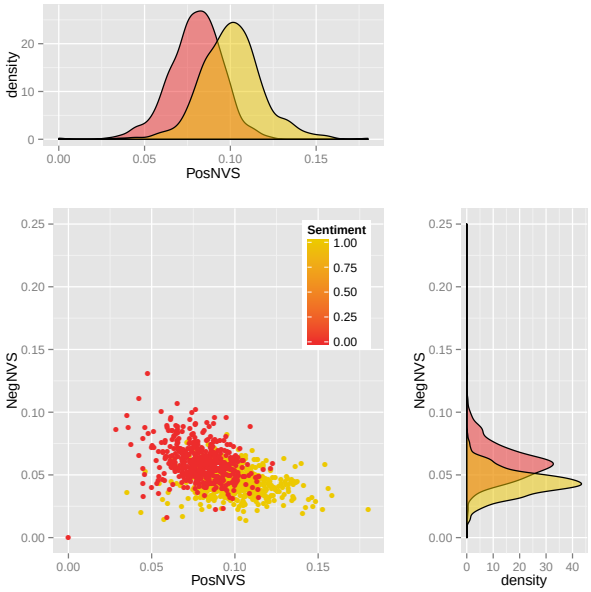
Για να μελετήσουμε περισσότερο την χρήση του naive Bayes στους γράφους ερευνήσαμε την κατανομή των χαρακτηριστικών (features) δεδομένης της κλάσης, όπου όπως έχουμε αναφέρει ο κλασικός naive Bayes ταξινομητής (Ενότητα 2.5.1.4) υποθέτει ότι είναι κανονική. Αρχικά υπολογίσαμε την κατανομή των χαρακτηριστικά των test instances των γράφων λέξεων με χρήση γλώσσας R. Παρουσιάζουμε την κατανομή κάθε ενός από τα 6 χαρακτηριστικά, PosCS, NegCS, PosNVS, NegNVS, PosVS, NegVS δεδομένης της κλάσης πολικότητας, όπου υπενθυμίζουμε ότι η αρνητική κλάση αντιστοιχεί σε 0 και η θετική κλάση σε 1. Η πρόθεση Pos αντιστοιχεί στον δείκτη ομοιότητας με τον γράφο θετικής πολικότητας, και αντίστοιχα η πρόθεση Neg με τον γράφο αρνητικής πολικότητας. Τα αποτελέσματα φαίνονται στα σχήματα 4.18 έως 4.20.



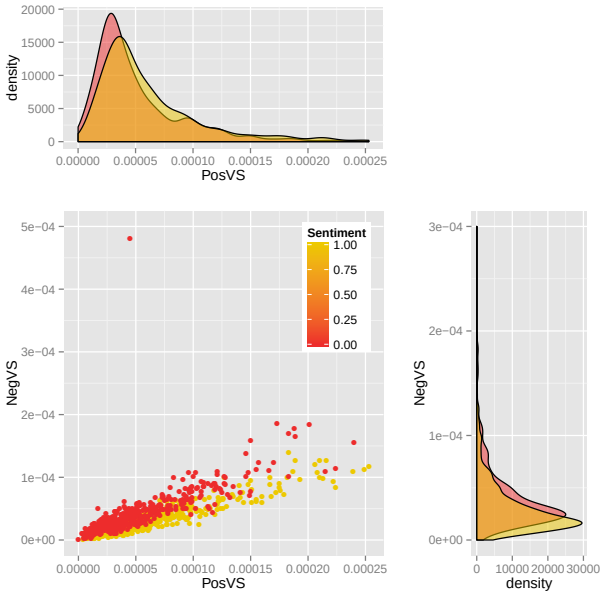
Σχήμα 4.17: Τιμές ακρίβειας μεθόδων για διαφορετικό πλήθος reviews κάθε γράφου πολικότητας.



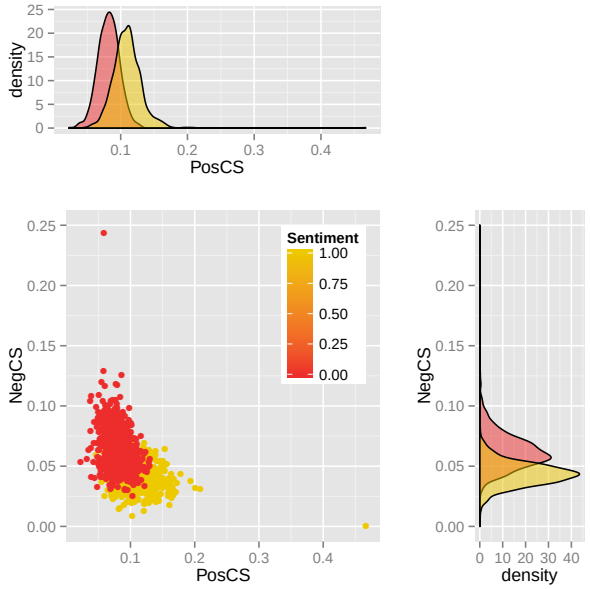
Σχήμα 4.18: Κατανομή των τυχαίων μεταβλητών Containment Similarity δεδομένης της κλάσης από τα δεδομένα των test instances.



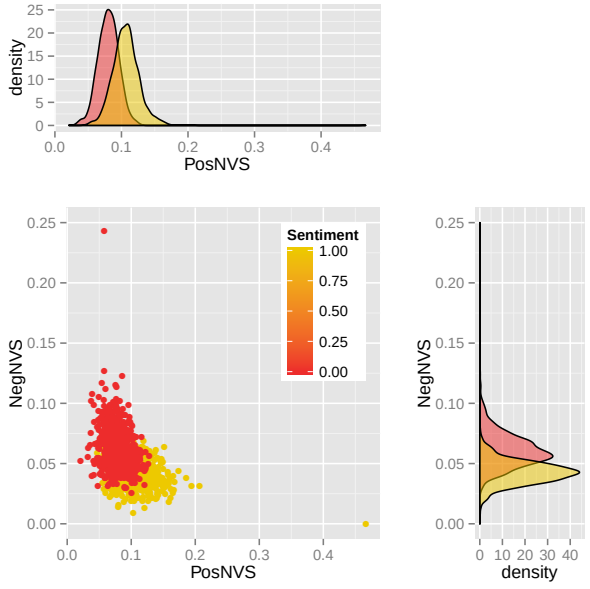
Σχήμα 4.19: Κατανομή των τυχαίων μεταβλητών Normalized Value Similarity δεδομένης της κλάσης από τα δεδομένα των test instances.



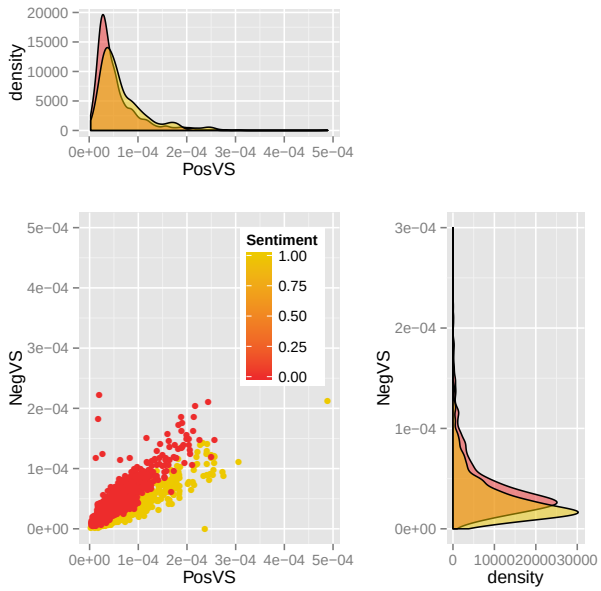
Σχήμα 4.20: Κατανομή των τυχαίων μεταβλητών Value Similarity δεδομένης της κλάσης από τα δεδομένα των test instances.



Σχήμα 4.21: Κατανομή των τυχαίων μεταβλητών Containment Similarity δεδομένης της κλάσης από τα δεδομένα των training instances.



Σχήμα 4.22: Κατανομή των τυχαίων μεταβλητών Normalized Value Similarity δεδομένης της κλάσης από τα δεδομένα των training instances.



Σχήμα 4.23: Κατανομή των τυχαίων μεταβλητών Value Similarity δεδομένης της κλάσης από τα δεδομένα των training instances.

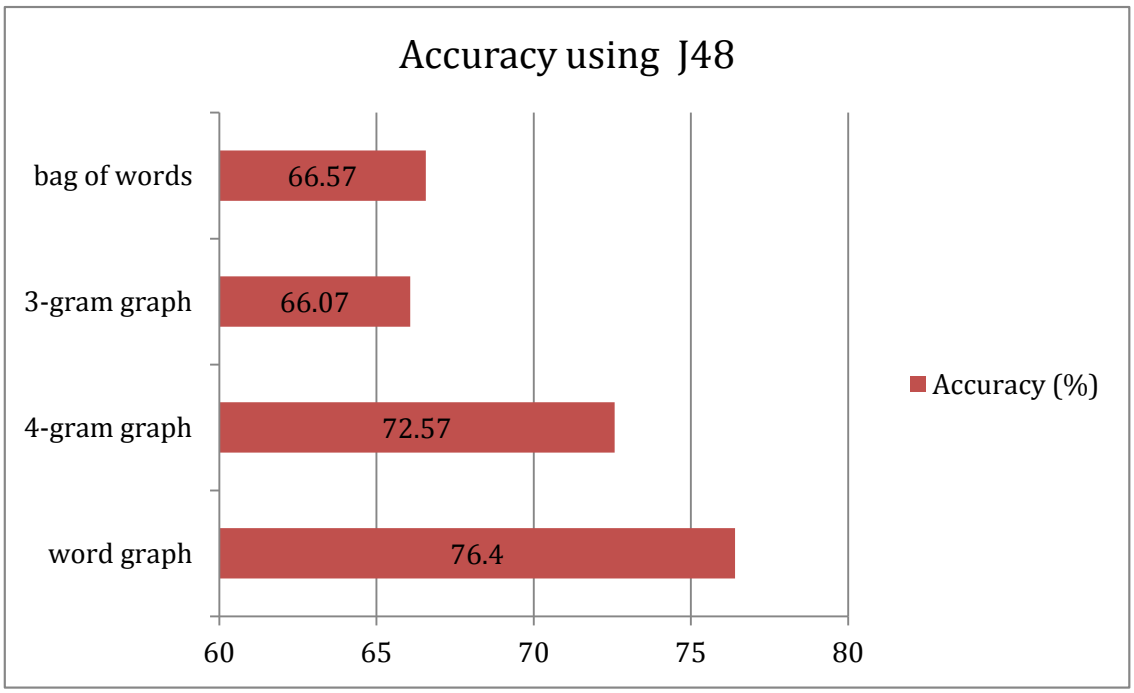
Παρατηρούμε ότι η κατανομή των χαρακτηριστικών δεδομένης της κλάσης είναι κανονική, με εξαίρεση την κατανομή του VS similarity η οποία όμως, δεν δημιουργεί μεγάλο πρόβλημα με την υπόθεση κανονικής κατανομής του naive Bayes. Για το λόγο αυτό η εφαρμογή του naive Bayes είναι καλή επιλογή με την προϋπόθεση ότι η κατανομή των χαρακτηριστικών στο training set να είναι και αυτή κανονική. Εφαρμόζοντας την ίδια διαδικασία για τα χαρακτηριστικά των training instances παρατηρήσαμε επίσης, κανονική κατανομή όπως φαίνεται στα σχήματα 4.21 έως 4.22, με εξαίρεση και πάλι της κατανομής του VS similarity. Επομένως, ο αλγόριθμος naive Bayes εκπαιδεύτηκε βρίσκοντας την κατανομή για κάθε χαρακτηριστικό δεδομένης της κλάσης από τα training instances και στην συνέχεια την εφάρμοσε για την κατηγοριοποίηση των 2000 test instances, όπου πέτυχε 81.3% ακρίβεια. Επιπλέον, από τα διαγράμματα (π.χ. το διάγραμμα 4.21) αξίζει να σημειωθεί ότι τα κείμενα που ανήκουν στην θετική κλάση (1), παρουσιάζουν μεγαλύτερα similarties με το θετικό γράφο (π.χ. PosCS), και αντίστοιχα για τα κείμενα που ανήκουν στην αρνητική κλάση (0), όπου παρουσιάζουν μεγαλύτερα similarities με τον αρνητικό γράφο (π.χ. NegCS). Παρόλα αυτά υπάρχει επικάλυψη μεταξύ των συνόλων των τιμών των ομοιότητων για τα θετικά και για τα αρνητικά reviews.

4.4.2 Χρήση άλλων classifiers

Στην ενότητα αυτή συγκρίνουμε τις μεθόδους χρησιμοποιώντας το δέντρο αποφάσεων (J48) και τον multinomial naive Bayes. Στο διάγραμμα 4.24 συγκεντρώσαμε τα αποτε-

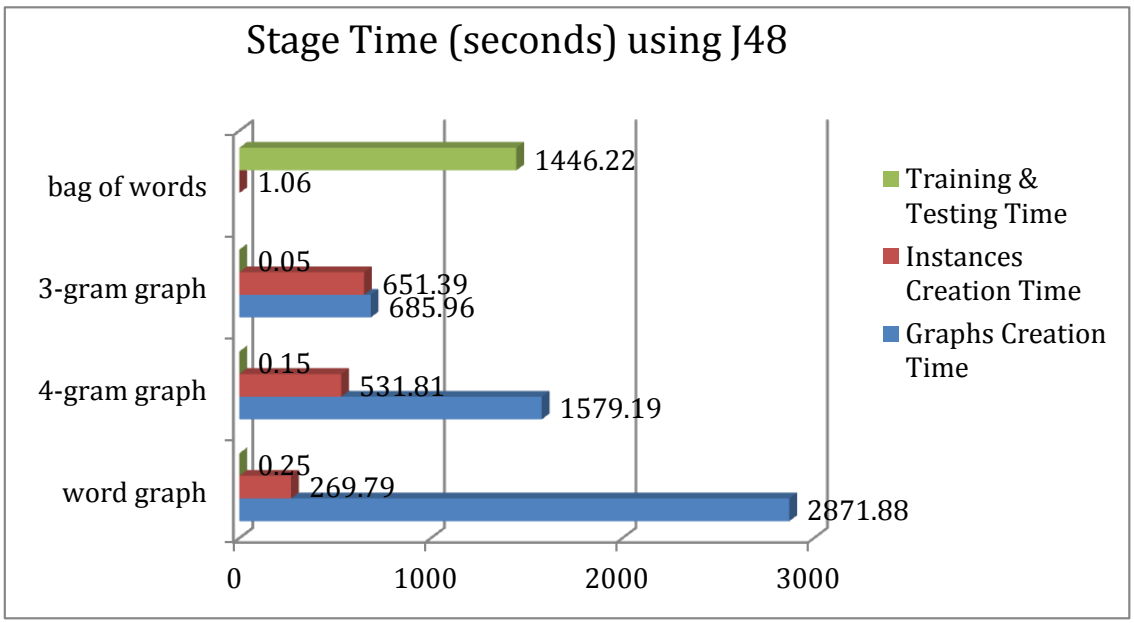
λέσματα των τρίων εκτελέσεων του παρακάτω πειράματος:

1. $D_{win} = 6$ (γράφοι λέξεων)
2. $D_{win} = n = 4$ (n-gram γράφοι)
3. `graph reviews`: 800
4. `training reviews`: 2000
5. `test reviews`: 1000
6. `classifier`: `weka.classifiers.trees.J48`
7. `remove`: `true`
8. `shuffle`: `true`
9. `preprocess`: `false`

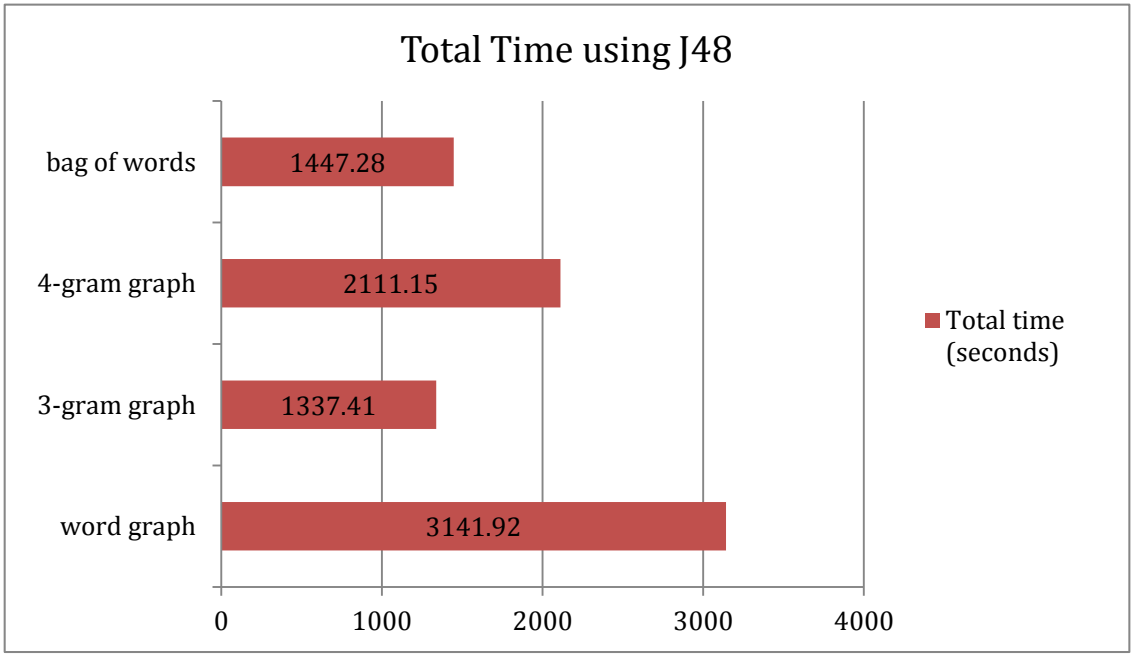


Σχήμα 4.24: Ακρίβεια που επιτυγχάνει κάθε μέθοδος με χρήση δέντρου αποφάσεων (J48).

Και σε αυτήν την περίπτωση παρατηρούμε ότι η μέθοδος των γράφων λέξεων πετυχαίνει τη μεγαλύτερη ακρίβεια. Αποδεικνύεται έτσι, ότι η μέθοδος των γράφων λέξεων οδήγησε σε



Σχήμα 4.25: Χρόνος κάθε σταδίου που καταναλώνει κάθε μέθοδος.



Σχήμα 4.26: Συνολικός χρόνος για κάθε μέθοδο.

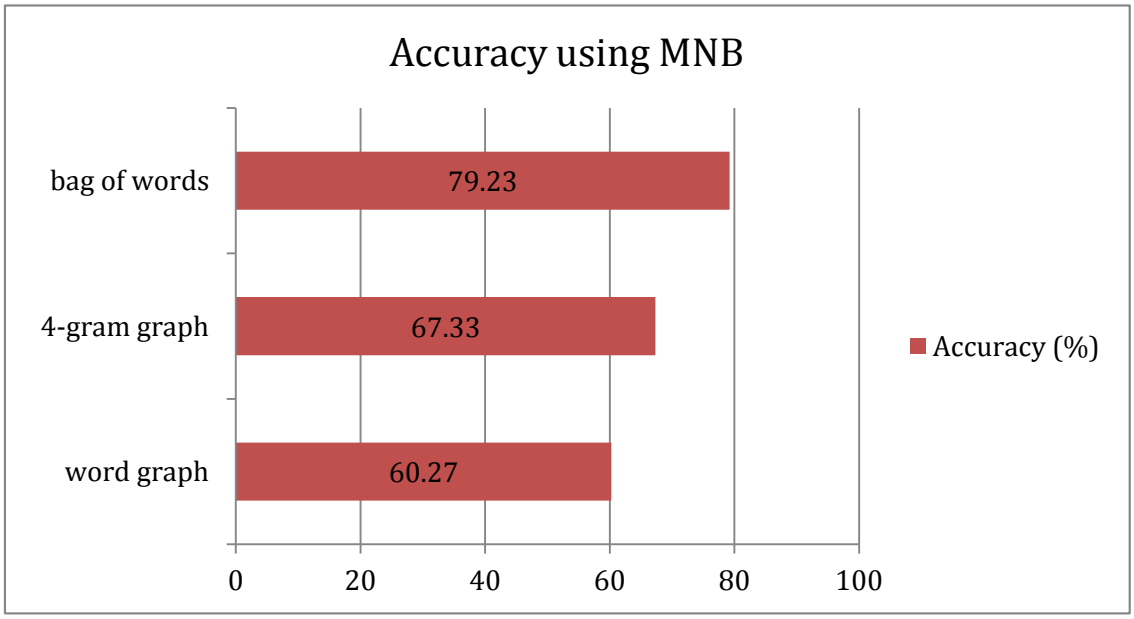
περισσότερο διαχωρίσιμα σύνολα, το οποίο επιθυμεί ο αλγόριθμος του δέντρου αποφάσεων (Ενότητα 2.5.2). Επομένως, συμπεραίνουμε και πάλι ότι η μέθοδος των n-grams δημιουργεί θόρυβο καθώς 'σπάει' τις λέξεις σε n-grams.

Στο σχήμα 4.25 δίνουμε τους χρόνους κάθε σταδίου για κάθε μέθοδο. Παρατηρούμε μεγαλύτερη κατανάλωση χρόνου από την μέθοδο bag of words, όπως επιβεβαιώνει το σχήμα συνολικού χρόνου εκτέλεσης 4.26. Για τον λόγο αυτό και επειδή πετυχαίνει μικρότερη ακρίβεια από τον naive Bayes, δεν προτείνεται η χρήση του δέντρου αποφάσεων για την σύγκριση των τριών μεθόδων.

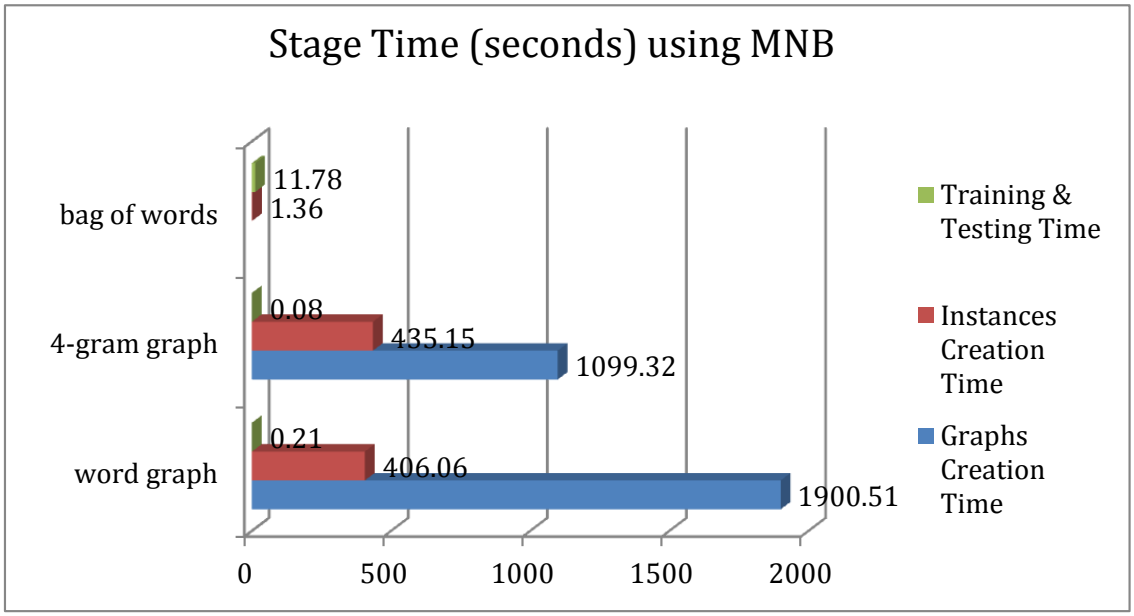
Για να μελετήσουμε την επίδοση με τον αλγόριθμο multinomial naive Bayes, διεξήχθηκαν τρεις εκτελέσεις του παρακάτω πειράματος:

1. $D_{win} = 6$ (γράφοι λέξεων)
2. $D_{win} = n = 4$ (n-gram γράφοι)
3. `graph reviews`: 800
4. `training reviews`: 2000
5. `test reviews`: 1000
6. `classifier`: `weka.classifiers.bayes.NaiveBayesMultinomial`
7. `remove`: `true`
8. `shuffle`: `true`
9. `preprocess`: `false`

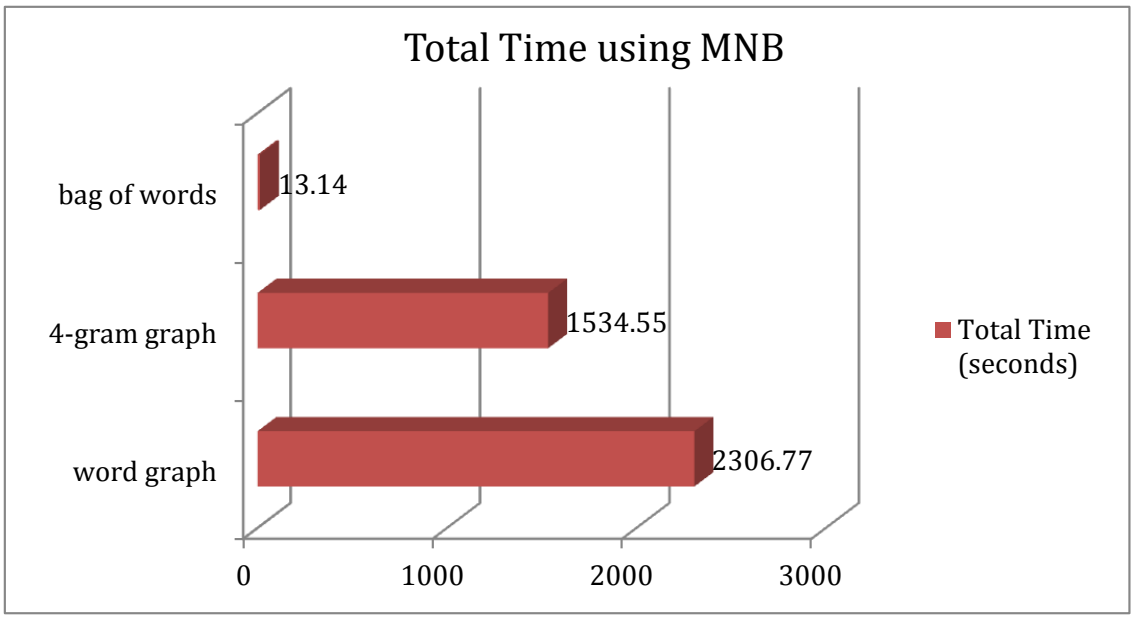
Στο σχήμα 4.27 δίνουμε τα αποτελέσματα του μέρου όρου της ακρίβειας των τριών πειραμάτων. Για την εφαρμογή του multinomial naive Bayes χρησιμοποιήσαμε την διακριτοποίηση των τιμών χαρακτηριστικών που παρουσιάστηκε στην ενότητα της πρώτης σύγκρισης (Ενότητα 3.1.1). Ο αλγόριθμος multinomial naive Bayes χρησιμοποιεί μία υπόθεση για την κατανομή των τιμών των χαρακτηριστικών η οποία δεν αντιπροσωπεύει την κατανομή των διακριτών τιμών που προκύπτουν μετά την εφαρμογή της διακριτοποίησης στις συνεχείς τιμές των ομοιοτήτων. Για το σκοπό αυτό, δεν αποδεικνύεται κατάλληλη η χρήση του με αυτό το είδος διακριτοποίησης για τις μεθόδους των γράφων. Ο αλγόριθμος δημιουργήθηκε για την χρήση του με bag of words αναπαράσταση, όπου παρατηρούμε μεγάλη επίδοση και έτσι, η baseline μέθοδος παραμένει ανταγωνιστικό μοντέλο. Στο σχήμα 4.28 δίνουμε τους χρόνους κάθε σταδίου για κάθε μέθοδο και στο σχήμα 4.29 το συνολικό χρόνο για κάθε μέθοδο. Όπως βλέπουμε η μέθοδος bag of words χρησιμοποιεί ελάχιστο χρόνο εκπαίδευσης για να επιτύχει ακρίβεια σχεδόν 80%.



Σχήμα 4.27: Ακρίβεια που επιτυγχάνει κάθε μέθοδος με χρήση multinomial naive Bayes.



Σχήμα 4.28: Χρόνος κάθε σταδίου που καταναλώνει κάθε μέθοδος.



Σχήμα 4.29: Συνολικός χρόνος για κάθε μέθοδο.

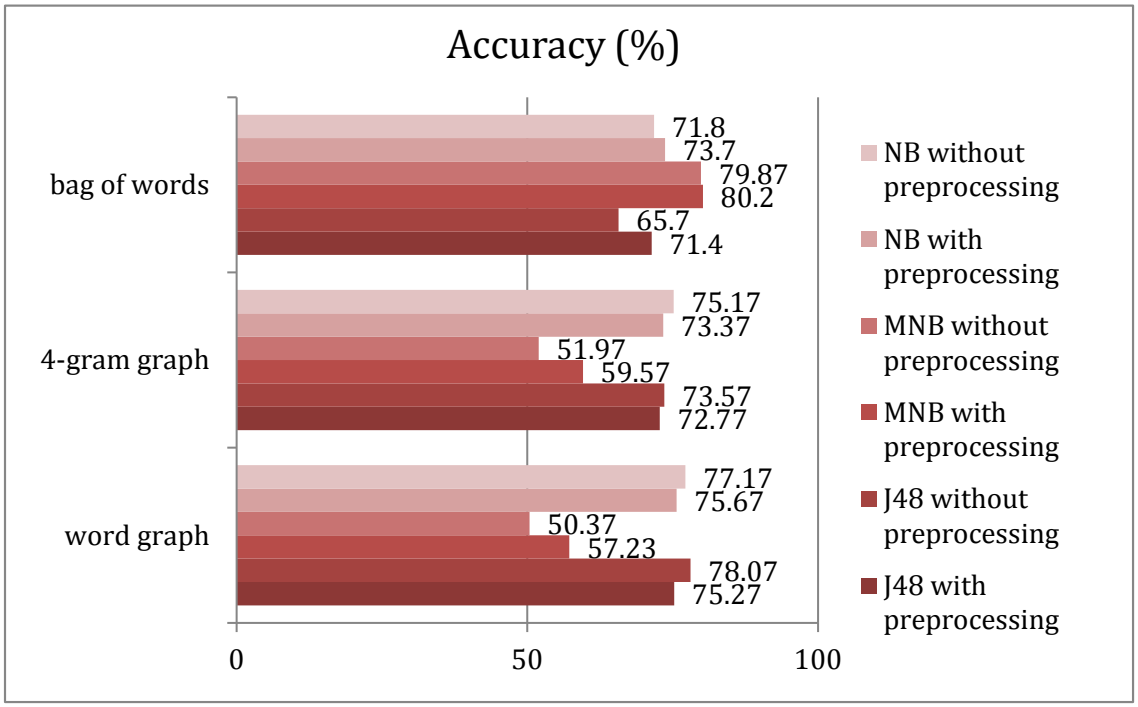
4.5 Προεπεξεργασία δεδομένων

Στην ενότητα αυτή μελετάμε πως επηρεάζει η προεπεξεργασία του κείμενου την ακρίβεια των τριών μεθόδων. Η διαδικασία αυτή περιλαμβάνει αφαίρεση ειδικών χαρακτήρων του κειμένου και εφαρμόζεται πριν την χρήση του από τις μεθόδους. Για την μελέτη αυτή χρησιμοποιήσαμε τις τιμές:

1. $D_{win} = 6$ (γράφοι λέξεων)
2. $D_{win} = n = 4$ (n-gram γράφοι)
3. `graph reviews`: 800
4. `training reviews`: 2000
5. `test reviews`: 1000
6. `remove`: `true`
7. `shuffle`: `true`

όπου ο ταξινομητής τέθηκε σε `weka.classifiers.bayes.NaiveBayesMultinomial`, `weka.classifiers.trees.J48` και `weka.classifiers.bayes.NaiveBayes` και η παράμετρος `preprocess` σε `true` και `false`. Για κάθε συνδυασμό των τιμών αυτών τρέξαμε τρία πειράματα και υπολογίσαμε το μέσο όρο της ακρίβειας. Τα αποτέλεσματα δίνονται στο διάγραμμα 4.30.

Παρατηρούμε ότι η αφαίρεση ειδικών χαρακτήρων δεν βελτιώνει την επίδοση των γράφων. Αυτό μπορεί να οφείλεται στην σπουδαιότητα της χρήσης σημείων στίξης που δίνουν έμφαση ([24]) αλλά και εκφράζουν συναίσθημα, σε συνδυασμό με το ότι οι μέθοδοι των γράφων βασίζονται στην ιδέα της αποθήκευσης της γειτνίασης χαρακτήρων ώστε να συλλάβουν το περιεχόμενο του κειμένου. Αντίθετα, η επίδοση της μεθόδου bag of words βελτιώνεται. Αυτό μπορεί να οφείλεται στην άμεση επίδραση της αφαίρεσης των ειδικών χαρακτήρων στα χαρακτηριστικά της μεθόδου bag of words. Συγκεκριμένα οι λέξεις αποτελούν τα χαρακτηριστικά και επομένως, η αφαίρεση τυχόν ειδικών χαρακτήρων που μπορεί να έχει μία λέξη οδήγησε στην αφαίρεση 'θορύβου' που πιθανόν επέφερε για την μέθοδο κατά τον υπολογισμό της συχνοτήτας εμφάνισής της.



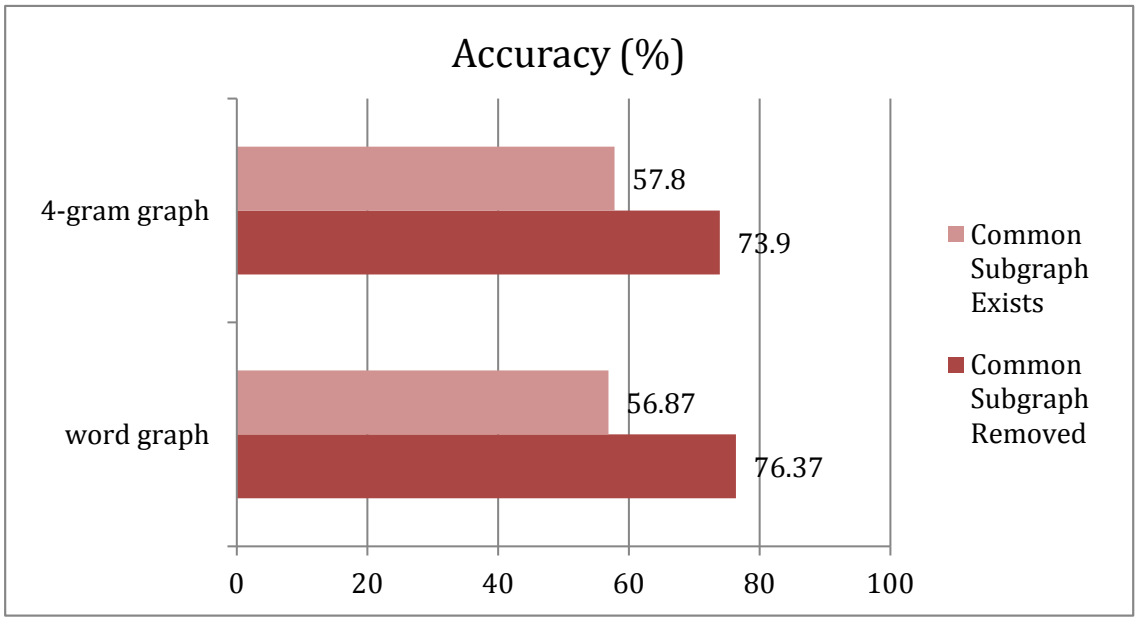
Σχήμα 4.30: Ακρίβεια για κάθε μέθοδο με και χωρίς προεπεξεργασία δεδομένων, όπου NB: naive Bayes, J48: Δέντρο Αποφάσεων, MNB: multinomial naive Bayes.

4.6 Κοινός υπογράφος

Στην ενότητα αυτή μελετάμε την επιρροή του κοινού υπογράφου μεταξύ των γράφων πολικότητας στην ακρίβεια των μεθόδων. Ο κοινός υπογράφος αφαιρείται με σκοπό την 'πόλωση' των γράφων πολικότητας προς την αντίστοιχη πολικότητα που αντιπροσωπεύει ο καθένας (Ενότητα 3.3). Για το σκοπό αυτό συγκρίθηκε η ακρίβεια των μεθόδων όταν αφαιρέσαμε τον κοινό υπογράφο σχετικά με την περίπτωση κατά την οποία χρησιμοποιήσαμε τους γράφους όπως δημιουργήθηκαν. Τα πειράματα είχαν τις εξής τιμές στις παραμέτρους:

1. $D_{win} = 6$ (γράφοι λέξεων)
2. $D_{win} = n = 4$ (n-gram γράφοι)
3. `graph reviews`: 800
4. `training reviews`: 2000
5. `test reviews`: 1000
6. `classifier`: `weka.classifiers.bayes.NaiveBayes`
7. `shuffle`: `true`
8. `preprocess`: `false`

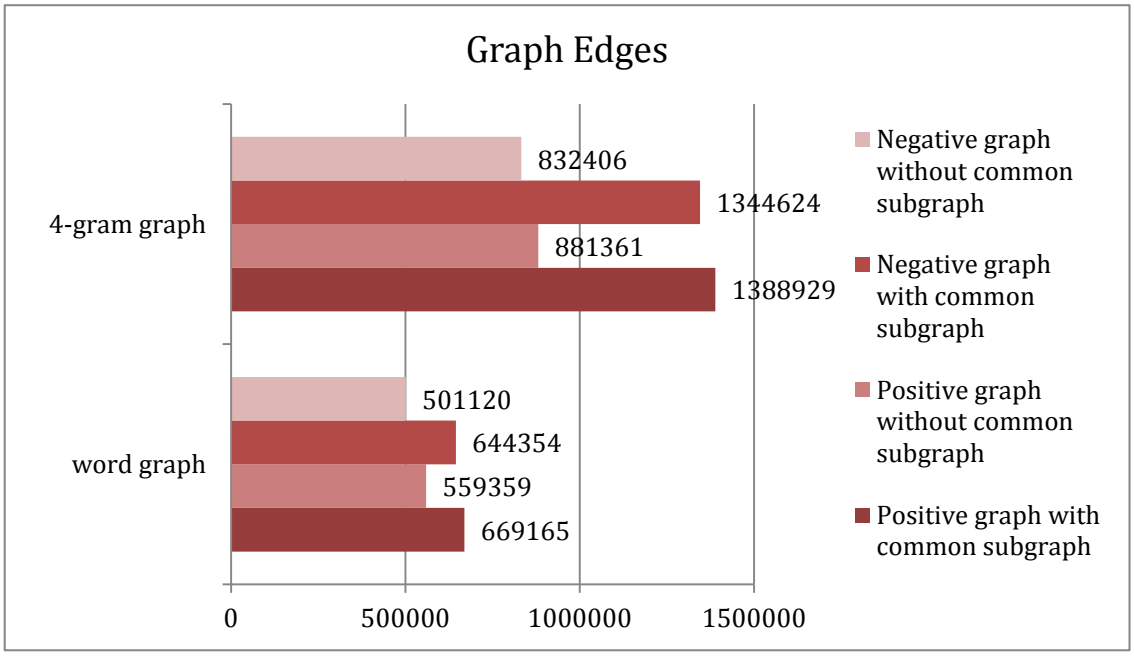
όπου η παράμετρος `remove` τίθεται σε `true` για τη αφαίρεση του κοινού υπογράφου και σε `false` για την χρησιμοποίηση των γράφων με τον κοινό υπογράφο. Για την διεξαγωγή των δυο πειραμάτων με και χωρίς τον υπογράφο σε ίδια δεδομένα θέσαμε το seed με το οποίο επιλέγονται δεδομένα στην ίδια τιμή και για τα δυο πειράματα. Η διαδικασία των δυο πειραμάτων επαναλήφθηκε δυο φορές με σκοπό να πάρουμε το μέσο όρο της ακρίβειας. Στο διάγραμμα 4.31 παρουσιάζουμε τον μέσο όρο της ακρίβειας των τριών πειραμάτων για κάθε περίπτωση.



Σχήμα 4.31: Ακρίβεια που πετυχαίνει κάθε μέθοδος.

Παρατηρούμε ότι η παραμονή του κοινού υπογράφου προκαλεί την κατακόρυφη μείωση της ακρίβειας. Ο naive Bayes χρησιμοποιεί τη μέγιστη πιθανοφάνεια ως κριτήριο για την

επιλογή της κλάσης. Στην περίπτωση που οι ομοιότητες που εξάγονται για τον γράφο ενός review με τους δυο γράφους πολικότητας είναι παρόμοιες, η πιθανοφάνεια που θα επιλεγεί είναι πιθανόν να είναι η αντίθετη από την σωστή. Η εξαγωγή παρόμοιων τιμών ομοιότητας μπορεί να συμβεί όταν οι γράφοι πολικότητας μοιράζονται κοινές λέξεις ή n-grams στους κόμβους τους, δηλαδή όταν έχουμε ύπαρξη κοινού υπογράφου. Έτσι, η ακρίβεια που επιτυγχάνεται είναι πολύ μικρή και δεν είναι αντιπροσωπευτική για την επίδοση των μεθόδων των γραφών. Επομένως, είναι απαραίτητη η αφαιρεση του κοινού υπογράφου για την δημιουργία του ταξινομητή. Όπως έχει αναφερθεί η αφαίρεση του κοινού υπογράφου επιτυγχάνεται με την αφαίρεση ακμών, το πλήθος των οποίων παρουσιάζεται για τις δυο πρειπτώσεις στο σχήμα 4.32.



Σχήμα 4.32: Ακμές γράφων πολικότητας με ή χωρίς τον κοινό υπογράφο τους.

Για την καλύτερη κατανόηση του περιεχομένου που αποθηκεύουμε στους γράφους πολικότητας χρησιμοποιήθηκαν τα reviews του πειράματος για την δημιουργία δυο word clouds, ένα με τις λέξεις των reviews που ενσωμάτωσε ο θετικός γράφος και ένα με τις λέξεις των reviews που ενσωμάτωσε ο αρνητικός γράφος. Συγκεκριμένα με χρήση γλώσσας R έγινε αποθήκευση των κειμένων και φιλτράρισμα των ειδικών χαρακτήρων, καθώς και αφαίρεση stop words που περιλαμβάνουν λέξεις όπως the, ώστε να διακρίνουμε λέξεις πιο καθοριστικές για το συναίσθημα. Το αποτέλεσμα φαίνεται στο σχήμα 4.33 όπου έχουν χρησιμοποιηθεί οι πιο συχνές λέξεις που εμφανίζονται στο 99.7% των κειμένων σε κάθε περίπτωση. Το μέγεθος μίας λέξης αντιπροσωπεύει τη συχνότητα εμφάνισής της στα κείμενα.

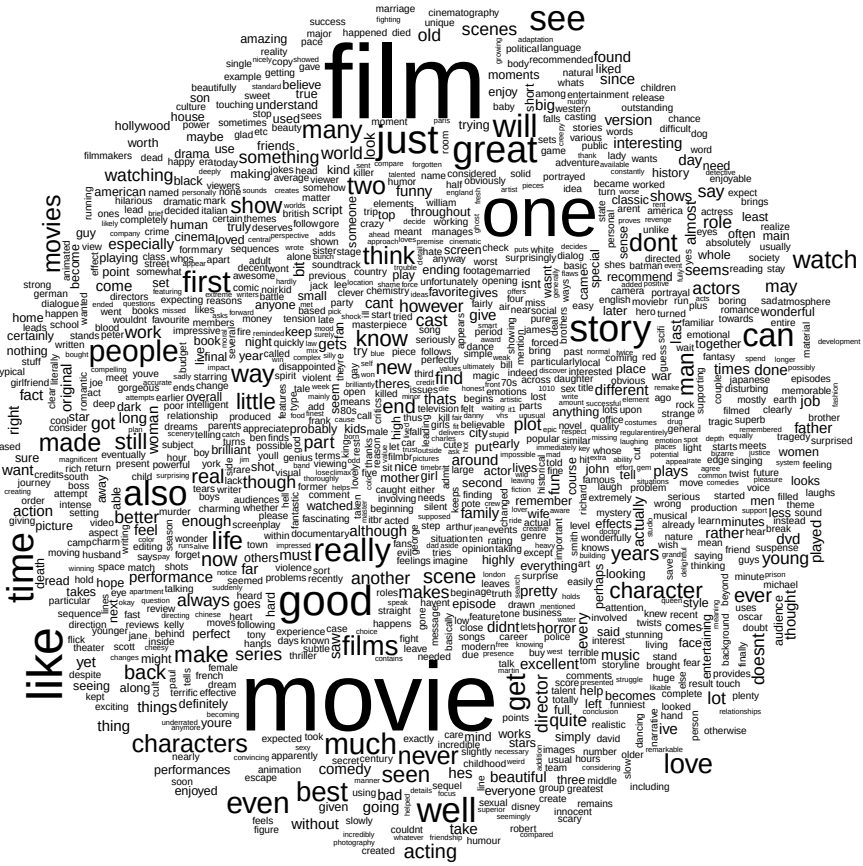
Παρατηρούμε ότι τη μεγαλύτερη συχνότητα εμφάνισης έχουν οι λέξεις movie, film, movies και films, όπως ήταν αναμενόμενο. Για το λόγο αυτό αφαιρούμε τις λέξεις αυτές και

το αποτέλεσμα φαίνεται στο σχήμα 4.34. Το cloud για τις λέξεις των αρνητικών reviews είναι μικρότερο. Αυτό φάνηκε και κατά την παρουσίαση του πλήθους των κόμβων του θετικού και του αρνητικού γράφου που παρουσιάστηκε στην ενότητα για το χώρο κατανάλωσης των γράφων (4.4.1.3, σχήμα 4.14). Επιπλέον, παρατηρούμε την εμφάνιση λέξεων όπως like, love, great, good, best, excellent στο θετικό cloud και bad, boring, terrible, poor, awful, stupid, worst στο αρνητικό cloud. Το αρνητικό cloud περιλαμβάνει πολλές λέξεις που χρησιμοποιούνται για την εκδήλωση άρνησης, όπως doesn't, don't, didn't, isn't, can't και αυτό εξηγεί την εμφάνιση λέξεων θετικού συναισθήματος, όπως good, love, well, σε αυτό. Δηλαδή, είναι πιθανή η χρήση άρνησης με λέξεις θετικής πολικότητας, καθώς συνηθίζεται για την έκφραση αρνητικού συναισθήματος. Η χρήση άρνησης εξηγεί και την εμφάνιση λιγότερων λέξεων - κόμβων στον αρνητικό γράφο, σε αντίθεση με τον θετικό γράφο όπου φαίνεται να δηλώνει την χρήση περισσότερων λέξεων κατά την περιγραφή ενός θετικού συναισθήματος. Τέλος, παρατηρούμε την ύπαρξη κοινών λέξεων στα δυο clouds το οποίο επηρεάζει τις ομοιότητες που θα εξαχθούν με τους αντίστοιχους γράφους και επιβεβαιώνει την ανάγκη αφαίρεσης του κοινού υπογράφου μέσω αφαίρεσης ακμών (και όχι λέξεων - κόμβων).

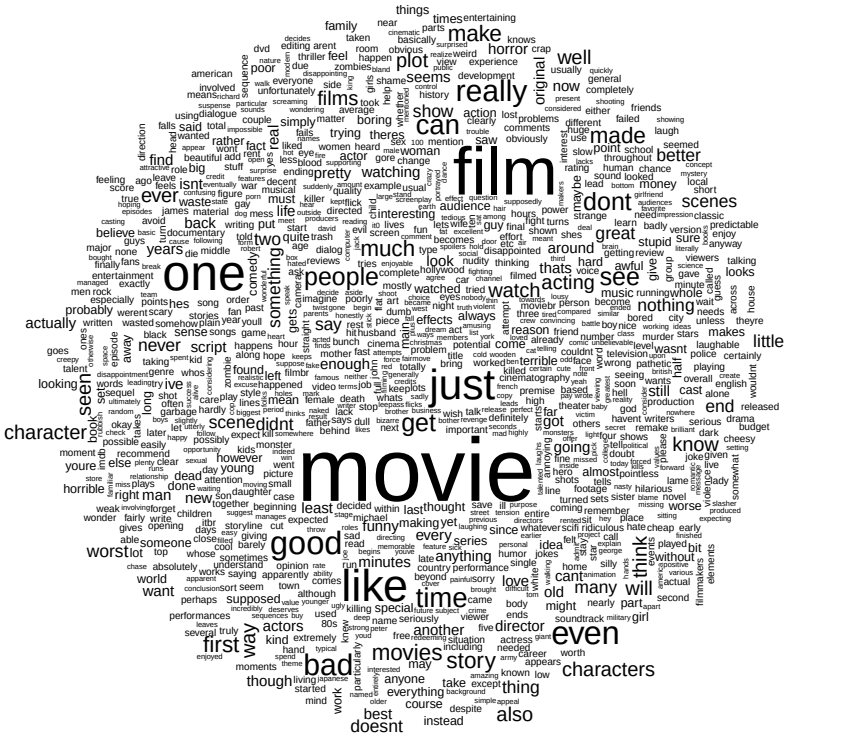
4.7 Σύγκριση δυο καλύτερων μοντέλων

Στην ενότητα αυτή χρησιμοποιούμε τα αποτελέσματα των προηγούμενων ενοτήτων για την σύγκριση των δυο καλύτερων μοντέλων και τελικά την εξαγωγή συμπεράσματος. Όπως, είναι φυσικό η μέθοδος bag of words παραμένει ανταγωνιστική και ως προς τον χρόνο που καταναλώνει και ως προς την ακρίβεια που επιτυγχάνει. Από τις μεθόδους των γράφων καλύτερη αποδείχθηκε η μέθοδος των γράφων λέξεων και ως προς τον χώρο που καταναλώνει και ως προς την ακρίβεια που επιτυγχάνει. Επομένως, χρησιμοποιούμε τις παραμέτρους που επιτυγχάνουν την μεγαλύτερη ακρίβεια για κάθε μία από τις δυο μεθόδους.

- Bag of words:
    1. `training reviews`: 2000
    2. `test reviews`: 1000
    3. `shuffle`: `true`
    4. `preprocess`: `true`
    5. `classifier`: `weka.classifiers.bayes.NaiveBayesMultinomial`
- Γράφοι λέξεων:
    1. $D_{win} = 6$
    2. `graph reviews`: 4000
    3. `training reviews`: 2000
    4. `test reviews`: 1000



(α′) Word cloud των λέξεων του γράφου θετικής πολικότητας



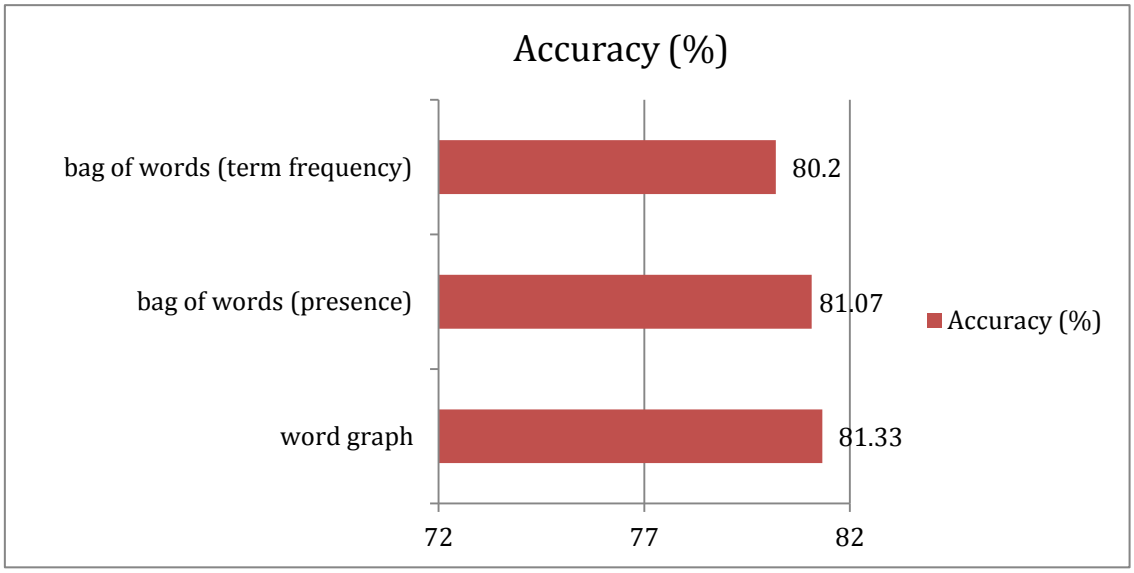
(β′) Word cloud των λέξεων του γράφου αρνητικής πολικότητας

Σχήμα 4.33: Word clouds των λέξεων που τοποθετούμε στους γράφους πολικότητας από θετικά και αρνητικά reviews

(α′) Word cloud των λέξεων του γράφου θετικής πολικότητας, έχοντας αφαιρέσει τις λέξεις “movie”, “movies”, “film”, “film”, “one”, “time”

(β′) Word cloud των λέξεων του γράφου αρνητικής πολικότητας, έχοντας αφαιρέσει τις λέξεις “movie”, “movies”, “film”, “film”, “one”, “time”

Σχήμα 4.34: Word clouds των λέξεων που τοποθετούμε στους γράφους πολικότητας από θετικά και αρνητικά reviews



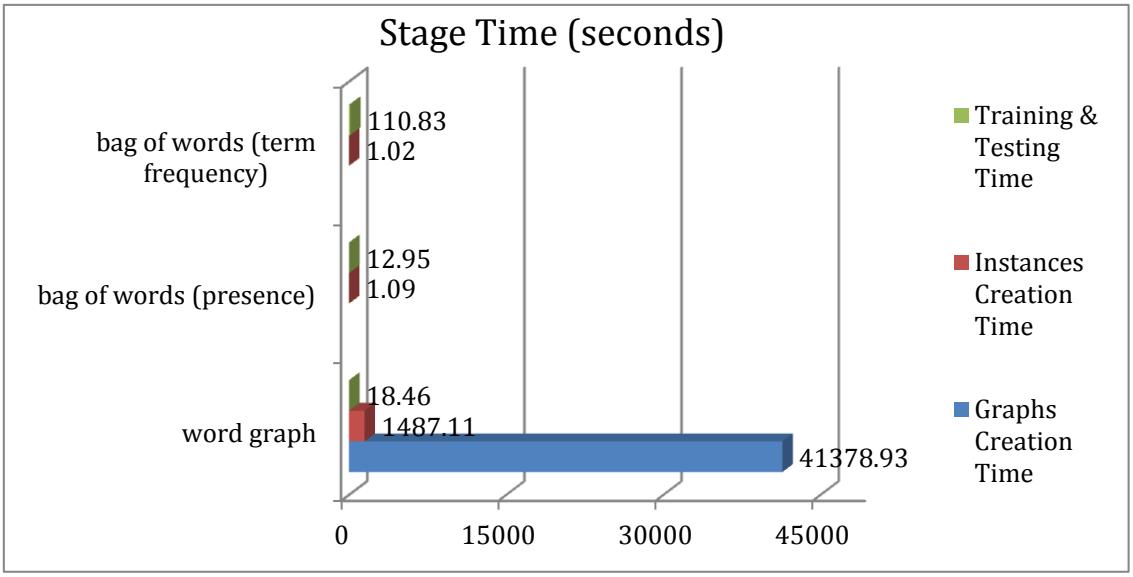
Σχήμα 4.35: Ακρίβεια για τις δυο καλύτερες μεθόδους.

5. `remove`: `true`
6. `shuffle`: `true`
7. `preprocess`: `false`
8. `classifier`: `weka.classifiers.bayes.NaiveBayes`

Στον κάθε γράφο πολικότητας ενσωματώσαμε 4000 reviews, καθώς για αυτό το σύνολο παρατηρήσαμε υψηλή ακρίβεια. Τα αποτελέσματα παρουσιάζονται στο διάγραμμα 4.35. Επιπλέον, για την διεξαγωγή του ίδιου πειράματος με διαφορετική μέθοδο, θέσαμε το σεεδ με το οποίο επιλέγονται reviews στην ίδια τιμή, ώστε να είναι δυνατή η παράλληλη σύγκριση των μεθόδων.

Παρατηρούμε ότι η μέθοδος των γράφων ξεπερνάει σε ακρίβεια την μέθοδο bag of words με την χρήση συχνότητας εμφάνισης λέξεων και με την χρήση παρουσίας λέξεων ως χαρακτηριστικά. Η διαφορά, όμως, είναι πολύ μικρή και μπορεί να μεταβληθεί με χρήση μεγαλύτερου training set, καθώς γνωρίζουμε από την σχετική βιβλιογραφία ότι η μέθοδος bag of words επιτυγχάνει 83% ακρίβεια όταν χρησιμοποιούνται ως χαρακτηριστικά μεταβλητές που δηλώνουν παρουσία λέξεων. Επομένως, θεωρούμε τις δυο μεθόδους ισάξιες.

Επιπλέον, στο διάγραμμα 4.36 δίνουμε τους χρόνους του κάθε σταδίου. Η μέθοδος των γράφων καταναλώνει πολύ μεγάλο χρόνο για την δημιουργία των γράφων πολικότητας, όπως ήδη έχουμε αναφέρει. Αυτό το στάδιο μπορεί και πάλι να μην ληφθεί υπόψη αν σκεφτούμε ότι οι γράφοι δημιουργούνται μία φορά και στην συνέχεια φορτώνονται στην μνήμη. Αντίθετα η μέθοδος bag of words καταναλώνει πολύ λιγότερο χρόνο. Επιπλέον, οι χρόνοι επαίδευσης και ελέγχου περιλαμβάνουν 2000 reviews και 1000 reviews αντίστοιχα, επομένως, ο χρόνος για την ταξινόμηση ενός μόνο review θα είναι ελάχιστος και από τις δυο μεθόδους. Επομένως, οι μέθοδοι μπορούν να θεωρηθούν ισάξιες και ως προς το χρόνο ταξινόμησης.



Σχήμα 4.36: Χρόνος κάθε σταδίου για τις δυο καλύτερες μεθόδους.

4.8 Συμπέρασμα

Στην εργασία αυτή μελετήσαμε την εφαρμογή μίας προσέγγισης ανάλυσης συναισθήματος σε κείμενα κριτικής ταινιών που είναι μεγαλύτερα από τα κείμενα των tweets με τα οποία έχει ήδη μελετηθεί στη σχετική βιβλιογραφία. Με μία αλλαγή της μεθόδου αυτής ώστε να χρησιμοποιεί λέξεις πετύχαμε ακρίβεια 81%. Έτσι, συμπεραίνουμε τη σπουδαιότητα της χρήσης λέξεων για την σύλληψη του περιεχομένου με σκοπό την εξαγωγή του συναισθήματος του κειμένου. Η αρχική μέθοδος των n-gram γράφων δημιουργήθηκε με σκοπό να χρησιμοποιείται σε πολύγλωσσα κείμενα ως language independent tool. Στην περίπτωσή μας, επεκτείναμε την μέθοδο για να την βελτιώσουμε από το θόρυβο που προκαλούν τα n-grams αλλά και επειδή είχαμε στην διάθεσή μας αγγλικά κείμενα. Ωστόσο, η χρήση λέξεων δεν συνεπάγεται άρση του χαρακτηριστικού language independence, καθώς η μέθοδος υποθέτει μόνο την ύπαρξη κενών στο κείμενο με βάση τα οποία εξάγει χωριστές οντότητες, οι οποίες συνήθως είναι λέξεις. Από την άλλη πλευρά, μπορούμε να εφαρμόσουμε την μέθοδο σε αγγλικά και να επεκτείνουμε για προσαρμογή σε τυχόν διαφορετικής φιλοσοφίας γλώσσα, όπως και στην εφαρμογή σημασιολογικών μεθόδων. Τέλος, να σημειωθεί ότι οι n-gram γράφοι έχουν υλοποιηθεί και για την χρήση n-grams λέξεων. Παρόλα αυτά δεν χρησιμοποιήσαμε αυτήν την προσέγγιση για την χρήση λέξεων καθώς έχει αποδειχθεί γενικότερα καλύτερη η χρήση n-grams χαρακτήρων στην εργασία [8] στην οποία στηρίχτηκε η μελέτη της εργασίας αυτής. Έτσι, προτιμήθηκε η ενσωμάτωση μίας μικρής αλλαγής στον κώδικα των character n-gram γράφων για την αποφυγή αφαίρεσης κάποιων χαρακτήρων που εφαρμόζει η μέθοδος word n-grams και για την επίτευξη παράλληλης σύγκρισης.

4.9 Μελλοντικές Εργασίες

Η μελέτη της μεθόδου των γράφων λέξεων μπορεί να επεκταθεί ενσωματώνοντας την κλάση ουδέτερης πολικότητας στο πρόβλημα. Η κλάση ουδέτερης πολικότητας θεωρείται σημαντική για την ανάλυση συναισθήματος ([15]). Επιπλέον, μπορούμε να βελτιώσουμε την επίδοση της μεθόδου με χρήση multinomial naive Bayes μελετώντας έναν καλύτερο τρόπο διακριτοποίησης των συνεχών τιμών των χαρακτηριστικών. Στην συνέχεια, μπορούμε να χρησιμοποιήσουμε τρόπους για την αντιμετώπιση του προβλήματος που δημιουργεί η υπόθεση ανεξαρτησίας των χαρακτηριστικών που περιλαμβάνει ο αλγόριθμος του naive Bayes (Ενότητα 2.5.1.3) ([28]). Επιπλέον, μπορούμε να μελετήσουμε την ενσωμάτωση reviews στους γράφους με ratings 9, 10 για τα θετικά και 0, 1 για τα αρνητικά, δηλαδή με κείμενα που να περιέχουν πιθανόν πιο έντονη έκφραση της πολικότητάς τους.

Βιβλιογραφία

[1] Fotis Aisopos, George Papadakis, Konstantinos Tserpes, and Theodora Varvarigou. Content vs. context for sentiment analysis: a comparative analysis over microblogs. In *Proceedings of the 23rd ACM conference on Hypertext and social media*, pages 187–196. ACM, 2012.

[2] Fotis Aisopos, George Papadakis, Konstantinos Tserpes, and Theodora Varvarigou. Textual and contextual patterns for sentiment analysis over microblogs. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 453–454. ACM, 2012.

[3] Fotis Aisopos, George Papadakis, and Theodora Varvarigou. Sentiment analysis of social media content using n-gram graphs. In *Proceedings of the 3rd ACM SIGMM international workshop on Social media*, pages 9–14. ACM, 2011.

[4] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[5] Erik Cambria, Bjorn Schuller, Yunqing Xia, and Catherine Havasi. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, (2):15–21, 2013.

[6] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[7] George Giannakopoulos and Vangelis Karkaletsis. N-gram graphs: Representing documents and document sets in summary system evaluation. In *Proceedings of Text Analysis Conference TAC2009 (To appear)*, 2009.

[8] George Giannakopoulos, Vangelis Karkaletsis, George Vouros, and Panagiotis Stamatopoulos. Summarization system evaluation revisited: N-gram graphs. *ACM Transactions on Speech and Language Processing (TSLP)*, 5(3):5, 2008.

[9] David J Hand and Keming Yu. Idiot's bayes-not so stupid after all? *International statistical review*, 69(3):385–398, 2001.

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *Unsupervised learning*. Springer, 2009.

[11] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[12] Minqing Hu and Bing Liu. Mining opinion features in customer reviews. In *AAAI*, volume 4, pages 755–760, 2004.

[13] George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

[14] Hyun Duk Kim, Kavita Ganesan, Parikshit Sondhi, and ChengXiang Zhai. Comprehensive review of opinion summarization. 2011.

[15] Moshe Koppel and Jonathan Schler. The importance of neutral examples for learning sentiment. *Computational Intelligence*, 22(2):100–109, 2006.

[16] David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Machine learning: ECML-98*, pages 4–15. Springer, 1998.

[17] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[18] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[19] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[20] James H Martin and Daniel Jurafsky. Speech and language processing. *International Edition*, 2000.

[21] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

[22] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics, 2005.

[23] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

[24] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[25] Ana-Maria Popescu and Orena Etzioni. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer, 2007.

[26] Rudy Prabowo and Mike Thelwall. Sentiment analysis: A combined approach. *Journal of Informetrics*, 3(2):143–157, 2009.

[27] Veselin Raychev and Preslav Nakov. Language-independent sentiment analysis using subjectivity and positional information. In *RANLP*, pages 360–364, 2009.

[28] Jason D Rennie, Lawrence Shih, Jaime Teevan, David R Karger, et al. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, volume 3, pages 616–623. Washington DC), 2003.

[29] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[30] Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.

Παράρτημα Α΄

Παράρτημα

```
package sentimentanalysis;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintWriter;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;

import gr.demokritos.iit.jinsect.utils;


public class ReviewWordGraph implements Serializable {

        private static final long serialVersionUID = 1L;
        private DocumentWordGraph reviewGraph;
        private boolean preprocess;

        public ReviewWordGraph(int window, boolean preprocess) {
                reviewGraph = new DocumentWordGraph(window);
                this.preprocess = preprocess;
        }

        public void createGraph(String reviewFile) throws IOException {
                BufferedReader reader = new BufferedReader(new
                   FileReader(reviewFile));

                if (preprocess) {
                        String imdbReview =
                           removeSpecialCharacters(reader.readLine());
                        reviewGraph.setDataString(imdbReview);
                } else
                        reviewGraph.setDataString(reader.readLine());
                reader.close();
        }

        private String removeSpecialCharacters(String review) {

                String pattern = "\\w|\\s";
                char[] reviewCharacters = review.toCharArray();
                char[] filteredReview = new char[review.length()];
                int size = 0;
                for (int index = 0; index < review.length(); index++)
```

```
                if (String.valueOf(reviewCharacters[index]).matches(pattern))
                        filteredReview[size++] = reviewCharacters[index];

        char[] returnReview = new char[size];
        for (int i = 0; i < size; i++)
                returnReview[i] = filteredReview[i];

        String imdbReview = String.valueOf(returnReview);
        return imdbReview.toLowerCase();
    }

    public void merge(ReviewWordGraph g, double weight) {
        reviewGraph.merge(g.getGraph(), weight);
    }

    public DocumentWordGraph[] removeCommonSubgraph(DocumentWordGraph g2) {

        DocumentWordGraph newGraphs[] = new DocumentWordGraph[2];
    newGraphs[0] = (DocumentWordGraph) reviewGraph.clone();
    newGraphs[1] = (DocumentWordGraph) g2.clone();

        DocumentWordGraph commonSubgraph = newGraphs[0].intersectGraph(
                        newGraphs[1]);

    newGraphs[0] = newGraphs[0].allNotIn(commonSubgraph);
    newGraphs[1] = newGraphs[1].allNotIn(commonSubgraph);
    commonSubgraph = newGraphs[0].intersectGraph(newGraphs[1]);

    return newGraphs;
    }

    public void printToFile(String filepath)
                    throws FileNotFoundException, UnsupportedEncodingException {
        PrintWriter writer = new PrintWriter(filepath, "UTF-8");
        writer.println(utils.graphToDot(reviewGraph.getGraphLevel(0), true));
        writer.close();
    }

    public void printToSystemOutput() {
        System.out.println(utils.graphToDot(reviewGraph.getGraphLevel(0),
                        true));
    }

    private void writeObject(ObjectOutputStream out) throws IOException {
        out.writeObject(reviewGraph);
    }

    private void readObject(ObjectInputStream in)
                    throws IOException, ClassNotFoundException {
        reviewGraph = (DocumentWordGraph) in.readObject();
    }

    public DocumentWordGraph getGraph() {
        return reviewGraph;
    }

    public int getNoOfNodes() {
        return reviewGraph.getGraphLevel(0).getVerticesCount();
    }

```

```
104         public int getNoOfEdges() {
105                 return reviewGraph.getGraphLevel(0).getEdgesCount();
106         }
107 
108         public void setGraph(DocumentWordGraph newGraph) {
109                 reviewGraph = newGraph;
110         }
111 
112         public boolean isPreprocess() {
113                 return preprocess;
114         }
115 
116         public void setPreprocess(boolean preprocess) {
117                 this.preprocess = preprocess;
118         }
119 }
```

```java
package sentimentanalysis;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintWriter;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;

import gr.demokritos.iit.jinsect.utils;
import gr.demokritos.iit.jinsect.documentModel.representations.DocumentNGramGraph;


public class ReviewNGramGraph implements Serializable {

        private static final long serialVersionUID = 1L;
        private DocumentNGramGraph reviewGraph;
        private boolean preprocess;

        public ReviewNGramGraph(int nSize, boolean preprocess) {
                reviewGraph = new DocumentNGramGraph(nSize, nSize, nSize);
                this.preprocess = preprocess;
        }

        public void createGraph(String reviewFile) throws IOException {
                BufferedReader reader = new BufferedReader(new
                   FileReader(reviewFile));

                if (preprocess) {
                        String imdbReview =
                           removeSpecialCharacters(reader.readLine());
                        reviewGraph.setDataString(imdbReview);
                } else
                        reviewGraph.setDataString(reader.readLine());
                reader.close();
        }

        private String removeSpecialCharacters(String review) {

                String pattern = "\\w|\\s";
                char[] reviewCharacters = review.toCharArray();
                char[] filteredReview = new char[review.length()];
                int size = 0;
                for (int index = 0; index < review.length(); index++)
                        if (String.valueOf(reviewCharacters[index]).matches(pattern))
                                filteredReview[size++] = reviewCharacters[index];

                char[] returnReview = new char[size];
                for (int i = 0; i < size; i++)
                        returnReview[i] = filteredReview[i];

                String imdbReview = String.valueOf(returnReview);
                return imdbReview.toLowerCase();
        }

        public void merge(ReviewNGramGraph g, double weight) {
                reviewGraph.merge(g.getGraph(), weight);
        }
```

```
59
60          public DocumentNGramGraph[] removeCommonSubgraph(DocumentNGramGraph g2) {
61
62                  DocumentNGramGraph newGraphs[] = new DocumentNGramGraph[2];
63          newGraphs[0] = (DocumentNGramGraph) reviewGraph.clone();
64          newGraphs[1] = (DocumentNGramGraph) g2.clone();
65
66                  DocumentNGramGraph commonSubgraph = newGraphs[0].intersectGraph(
67                                  newGraphs[1]);
68
69          newGraphs[0] = newGraphs[0].allNotIn(commonSubgraph);
70          newGraphs[1] = newGraphs[1].allNotIn(commonSubgraph);
71          commonSubgraph = newGraphs[0].intersectGraph(newGraphs[1]);
72
73          return newGraphs;
74          }
75
76          public void printToFile(String filepath)
77                          throws FileNotFoundException, UnsupportedEncodingException {
78                  PrintWriter writer = new PrintWriter(filepath, "UTF-8");
79                  writer.println(utils.graphToDot(reviewGraph.getGraphLevel(0), true));
80                  writer.close();
81          }
82
83          public void printToSystemOutput() {
84                  System.out.println(utils.graphToDot(reviewGraph.getGraphLevel(0),
85                                  true));
86          }
87
88          public DocumentNGramGraph getGraph() {
89                  return reviewGraph;
90          }
91
92          public int getNoOfNodes() {
93                  return reviewGraph.getGraphLevel(0).getVerticesCount();
94          }
95
96          public int getNoOfEdges() {
97                  return reviewGraph.getGraphLevel(0).getEdgesCount();
98          }
99
100         public void setGraph(DocumentNGramGraph newGraph) {
101                 reviewGraph = newGraph;
102         }
103
104         public boolean isPreprocess() {
105                 return preprocess;
106         }
107
108         public void setPreprocess(boolean preprocess) {
109                 this.preprocess = preprocess;
110         }
111
112         private void writeObject(ObjectOutputStream out) throws IOException {
113                 out.writeObject(reviewGraph);
114         }
115
116         private void readObject(ObjectInputStream in)
117                         throws IOException, ClassNotFoundException {
118                 reviewGraph = (DocumentNGramGraph) in.readObject();
```

```
119             }
120 }
```

```java
package sentimentanalysis;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;

public class ModelWordGraph {

        private ReviewWordGraph reviewsGraph;
        private int noReviews;
        private int window;
        private ArrayList<String> reviewFilenames;

        public ModelWordGraph() { }

        public ModelWordGraph(int noReviews, int window) {
                this.noReviews = noReviews;
                this.window = window;
                reviewsGraph = null;
                reviewFilenames = null;
        }

    public void createModelGraph(String reviewFilepath,
                ArrayList<String> reviewFilenames, boolean preprocess)
                throws IOException {

                ReviewWordGraph reviewGraph = new ReviewWordGraph(window,
                   preprocess);
                reviewsGraph = new ReviewWordGraph(window, preprocess);
                this.reviewFilenames = new ArrayList<String>();

                int mergedReviews = 0;
                for (String s: reviewFilenames) {
                        reviewGraph.createGraph(reviewFilepath.concat(s));
                        reviewsGraph.merge(reviewGraph, 1 / (1 + mergedReviews));
                        this.reviewFilenames.add(s);
                        if (++mergedReviews >= noReviews)
                                break;
                }
        }

        @SuppressWarnings("unchecked")
        public void loadModelGraph(String inputGraphFile)
                        throws ClassNotFoundException, IOException {

                FileInputStream input = new FileInputStream(inputGraphFile);
                ObjectInputStream inputGraphStream = new ObjectInputStream(input);

                noReviews = inputGraphStream.readInt();
                window = inputGraphStream.readInt();
                reviewsGraph = ((ReviewWordGraph) inputGraphStream.readObject());
                reviewFilenames = (ArrayList<String>) inputGraphStream.readObject();
                inputGraphStream.close();
        }

```

```
60          public void storeModelGraph(String outputGraphFile) throws IOException {
61                  FileOutputStream output = new FileOutputStream(outputGraphFile);
62                  ObjectOutputStream outputGraphStream = new
                       ObjectOutputStream(output);
63
64                  outputGraphStream.writeInt(noReviews);
65                  outputGraphStream.writeInt(window);
66                  outputGraphStream.writeObject(reviewsGraph);
67                  outputGraphStream.writeObject(reviewFilenames);
68                  outputGraphStream.close();
69          }
70
71          public ReviewWordGraph[] removeCommonSubgraph(ReviewWordGraph graph2) {
72                  DocumentWordGraph[] wordGraphs =
73                                  reviewsGraph.removeCommonSubgraph(graph2.getGraph());
74                  reviewsGraph.setGraph(wordGraphs[0]);
75                  graph2.setGraph(wordGraphs[1]);
76
77                  ReviewWordGraph[] newGraphs = new ReviewWordGraph[2];
78                  newGraphs[0] = reviewsGraph;
79                  newGraphs[1] = graph2;
80                  return newGraphs;
81          }
82
83          public void printToFile(String filepath)
84                          throws FileNotFoundException, UnsupportedEncodingException {
85                  reviewsGraph.printToFile(filepath);
86          }
87
88          public void printToSystemOutput() {
89                  reviewsGraph.printToSystemOutput();
90          }
91
92          public ReviewWordGraph getReviewsGraph() {
93                  return reviewsGraph;
94          }
95
96          public int getNoOfNodes() {
97                  return reviewsGraph.getNoOfNodes();
98          }
99
100         public int getNoOfEdges() {
101                 return reviewsGraph.getNoOfEdges();
102         }
103
104         public int getWindow() {
105                 return window;
106         }
107
108         public int getNoReviews() {
109                 return noReviews;
110         }
111
112         public void setReviewsGraph(ReviewWordGraph newReviewsGraph) {
113                 reviewsGraph = newReviewsGraph;
114         }
115
116         public ArrayList<String> getReviewFilenames() {
117                 return reviewFilenames;
118         }
```

```
119 }
```

```java
package sentimentanalysis;

import gr.demokritos.iit.jinsect.documentModel.representations.DocumentNGramGraph;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;

public class ModelNGramGraph {

        private ReviewNGramGraph reviewsGraph;
        private int noReviews;
        private int nSize;
        private ArrayList<String> reviewFilenames;

        public ModelNGramGraph() { }

        public ModelNGramGraph(int noReviews, int nSize) {
                this.noReviews = noReviews;
                this.nSize = nSize;
                reviewsGraph = null;
                reviewFilenames = null;
        }

        public void createModelGraph(String reviewFilepath,
                        ArrayList<String> reviewFilenames, boolean preprocess)
                        throws IOException {

                ReviewNGramGraph reviewGraph = new ReviewNGramGraph(nSize,
                   preprocess);
                reviewsGraph = new ReviewNGramGraph(nSize, preprocess);
                this.reviewFilenames = new ArrayList<String>();

                int mergedReviews = 0;
                for (String s: reviewFilenames) {
                        reviewGraph.createGraph(reviewFilepath.concat(s));
                        reviewsGraph.merge(reviewGraph, 1 / (1 + mergedReviews));
                        this.reviewFilenames.add(s);
                        if (++mergedReviews >= noReviews)
                                break;
                }
        }

        @SuppressWarnings("unchecked")
        public void loadModelGraph(String inputGraphFile)
                        throws IOException, ClassNotFoundException {

                FileInputStream input = new FileInputStream(inputGraphFile);
                ObjectInputStream inputGraphStream = new ObjectInputStream(input);

                noReviews = inputGraphStream.readInt();
                nSize = inputGraphStream.readInt();
                reviewsGraph = ((ReviewNGramGraph) inputGraphStream.readObject());
                reviewFilenames = (ArrayList<String>) inputGraphStream.readObject();
                inputGraphStream.close();
```

```
60          }
61
62          public void storeModelGraph(String outputGraphFile) throws IOException {
63                  FileOutputStream output = new FileOutputStream(outputGraphFile);
64                  ObjectOutputStream outputGraphStream = new
                       ObjectOutputStream(output);
65
66                  outputGraphStream.writeInt(noReviews);
67                  outputGraphStream.writeInt(nSize);
68                  outputGraphStream.writeObject(reviewsGraph);
69                  outputGraphStream.writeObject(reviewFilenames);
70                  outputGraphStream.close();
71          }
72
73          public ReviewNGramGraph[] removeCommonSubgraph(ReviewNGramGraph graph2) {
74
75                  DocumentNGramGraph[] wordGraphs =
76                                  reviewsGraph.removeCommonSubgraph(graph2.getGraph());
77                  reviewsGraph.setGraph(wordGraphs[0]);
78                  graph2.setGraph(wordGraphs[1]);
79
80                  ReviewNGramGraph[] newGraphs = new ReviewNGramGraph[2];
81                  newGraphs[0] = reviewsGraph;
82                  newGraphs[1] = graph2;
83                  return newGraphs;
84          }
85
86          public void printToFile(String filepath)
87                          throws FileNotFoundException, UnsupportedEncodingException {
88                  reviewsGraph.printToFile(filepath);
89          }
90
91          public void printToSystemOutput() {
92                  reviewsGraph.printToSystemOutput();
93          }
94
95          public ReviewNGramGraph getReviewsGraph() {
96                  return reviewsGraph;
97          }
98
99          public int getNoOfNodes() {
100                 return reviewsGraph.getNoOfNodes();
101         }
102
103         public int getNoOfEdges() {
104                 return reviewsGraph.getNoOfEdges();
105         }
106
107         public int getNSize() {
108                 return nSize;
109         }
110
111         public int getNoReviews() {
112                 return noReviews;
113         }
114
115         public void setReviewsGraph(ReviewNGramGraph newReviewsGraph) {
116                 reviewsGraph = newReviewsGraph;
117         }
118
```

```
119         public ArrayList<String> getReviewFilenames() {
120                 return reviewFilenames;
121         }
122 }
```

```
package sentimentanalysis;

import gr.demokritos.iit.jinsect.documentModel.representations.DocumentNGramGraph;
import gr.demokritos.iit.jinsect.structs.*;

import java.util.*;

import salvo.jesus.graph.*;

public class DocumentWordGraph extends DocumentNGramGraph {

        private static final long serialVersionUID = 1L;

        public DocumentWordGraph(int windowSize) {
                super(windowSize, windowSize, windowSize);
        }

        public void createGraphs() {

                String sUsableString = (new StringBuilder()).
                                append(DataString).toString();

                if(TextPreprocessor != null)
                        sUsableString = TextPreprocessor.preprocess(sUsableString);

                String[] extractedWords = sUsableString.split("\\s+");
                int length = extractedWords.length;
                String sCurNGram = null;
                HashMap<String, Double> hTokenAppearence = new HashMap<String,
                   Double>();

                for (int iCur = 0; iCur < length; iCur++) {
                        sCurNGram = extractedWords[iCur];

                        if(hTokenAppearence.containsKey(sCurNGram))
                                hTokenAppearence.put(sCurNGram,
                                        Double.valueOf(
                                        hTokenAppearence.get(sCurNGram).doubleValue()
                                        + 1.0D));
                        else
                                hTokenAppearence.put(sCurNGram,
                                   Double.valueOf(1.0D));
                }

                Vector<String> PrecedingNeighbours = new Vector<String>();
                UniqueVertexGraph gGraph = getGraphLevelByNGramSize(MinSize);
                sCurNGram = "";

                for (int iCur = 0; iCur < length; iCur++) {
                        sCurNGram = extractedWords[iCur];

                        if(WordEvaluator != null &&
                           !WordEvaluator.evaluateWord(sCurNGram))
                                continue;

                        String aFinalNeighbours[];
                        if(Normalizer != null)
                                aFinalNeighbours = (String[])
                                   Normalizer.normalize(null,
                                                PrecedingNeighbours.toArray());
```

```
57                          else {
58                                  aFinalNeighbours = new
                                       String[PrecedingNeighbours.size()];
59                                  PrecedingNeighbours.toArray(aFinalNeighbours);
60                          }
61
62                          createEdgesConnecting(gGraph, sCurNGram,
63                                          Arrays.asList(aFinalNeighbours),
                                                hTokenAppearence);
64
65                          PrecedingNeighbours.add(sCurNGram);
66
67                          if(PrecedingNeighbours.size() > CorrelationWindow)
68                                  PrecedingNeighbours.removeElementAt(0);
69                  }
70
71                  int iNeighboursLen = PrecedingNeighbours.size();
72                  if(iNeighboursLen < CorrelationWindow && iNeighboursLen > 0)
73                          createEdgesConnecting(gGraph, sCurNGram,
74                                          PrecedingNeighbours, hTokenAppearence);
75          }
76
77          public void mergeGraph(DocumentNGramGraph dgOtherGraph,
78                          double fWeightPercent) {
79
80                  if(dgOtherGraph == this)
81                          return;
82
83                  for(int iCurLvl = MinSize; iCurLvl <= MaxSize; iCurLvl++) {
84                          UniqueVertexGraph gGraph = getGraphLevelByNGramSize(MinSize);
85                          UniqueVertexGraph gOtherGraph =
86                                          dgOtherGraph.getGraphLevelByNGramSize(MinSize);
87                          if(gOtherGraph == null)
88                                  return;
89
90                          Iterator<?> iIter = gOtherGraph.getEdgeSet().iterator();
91                          ArrayList<String> lOtherNodes = new ArrayList<String>();
92                          String sHead;
93                          double dWeight;
94                          for(; iIter.hasNext();
95                                          createWeightedEdgesConnecting(gGraph, sHead,
96                                                          lOtherNodes, dWeight,
                                                                dWeight,
97                                                          fWeightPercent)) {
98
99                                  WeightedEdge weCurItem = (WeightedEdge)iIter.next();
100                                 sHead = weCurItem.getVertexA().getLabel();
101                                 String sTail = weCurItem.getVertexB().getLabel();
102                                 dWeight = weCurItem.getWeight();
103                                 lOtherNodes.clear();
104                                 lOtherNodes.add(sTail);
105                         }
106                 }
107         }
108
109         public DocumentWordGraph intersectGraph(DocumentWordGraph dgOtherGraph) {
110
111                 DocumentWordGraph gRes = new DocumentWordGraph(CorrelationWindow);
112                 EdgeCachedLocator ecl = new EdgeCachedLocator(1000);
113 label0:
```

```
114                 for(int iCurLvl = MinSize; iCurLvl <= MaxSize; iCurLvl++) {
115                         UniqueVertexGraph gGraph = getGraphLevelByNGramSize(iCurLvl);
116                         UniqueVertexGraph gOtherGraph =
117                                         dgOtherGraph.getGraphLevelByNGramSize(iCurLvl);
118                         UniqueVertexGraph gNewGraph =
119                                         gRes.getGraphLevelByNGramSize(iCurLvl);
120
121                         if(gOtherGraph == null)
122                                 continue;
123
124                         Iterator<?> iIter = gOtherGraph.getEdgeSet().iterator();
125                         do {
126                                 WeightedEdge weCurItem;
127                                 String sHead;
128                                 String sTail;
129                                 WeightedEdge eEdge;
130                                 do {
131                                         if(!iIter.hasNext())
132                                                 continue label0;
133                                         weCurItem = (WeightedEdge)iIter.next();
134                                         sHead = weCurItem.getVertexA().getLabel();
135                                         sTail = weCurItem.getVertexB().getLabel();
136                                         eEdge =
                                                (WeightedEdge)ecl.locateEdgeInGraph(gGraph,
137                                                         weCurItem.getVertexA(),
                                                            weCurItem.getVertexB());
138                                 } while(eEdge == null);
139
140                                 try {
141                                         List<String> l = new ArrayList<String>();
142                                         l.add(sTail);
143                                         double dTargetWeight = 0.5D *
                                                (eEdge.getWeight() +
144                                                         weCurItem.getWeight());
145                                         createWeightedEdgesConnecting(gNewGraph,
                                                sHead, l,
146                                                         dTargetWeight,
                                                            dTargetWeight, 1.0D);
147                                 } catch(Exception e) {
148                                         e.printStackTrace();
149                                 }
150                         } while(true);
151                 }
152
153                 return gRes;
154         }
155
156         public DocumentWordGraph allNotIn(DocumentWordGraph dgOtherGraph) {
157
158                 EdgeCachedLocator eclLocator =
159                                 new EdgeCachedLocator(Math.max(length(),
160                                                 dgOtherGraph.length()));
161                 DocumentWordGraph dgClone = (DocumentWordGraph)clone();
162 label0:
163                 for(int iCurLvl = MinSize; iCurLvl <= MaxSize; iCurLvl++) {
164
165                         UniqueVertexGraph gCloneLevel =
166                                         dgClone.getGraphLevelByNGramSize(iCurLvl);
167                         UniqueVertexGraph gOtherGraphLevel =
168                                         dgOtherGraph.getGraphLevelByNGramSize(iCurLvl);
```

```
169
170                         if(gOtherGraphLevel == null)
171                                 continue;
172
173                         Iterator<Object> iIter =
174                                         Arrays.asList(
175                                                         gCloneLevel.getEdgeSet().toArray())
176                         do {
177                                 WeightedEdge weCurItem;
178                                 Edge eEdge;
179                                 do {
180                                         if(!iIter.hasNext())
181                                                 continue label0;
182                                         weCurItem = (WeightedEdge)iIter.next();
183                                         eEdge =
184                                                 eclLocator.locateDirectedEdgeInGraph(
185                                                                 gOtherGraphLevel,
186                                                                 weCurItem.getVertexA(),
187                                                                 weCurItem.getVertexB());
188                                 } while(eEdge == null);
189
190                                 try {
191                                         gCloneLevel.removeEdge(weCurItem);
192                                         eclLocator.resetCache();
193                                 } catch(Exception ex) {
194                                         ex.printStackTrace();
195                                 }
196                         } while(true);
197                 }
198
199                 return dgClone;
200         }
201
202         @SuppressWarnings("unchecked")
203         public Object clone() {
204                 DocumentWordGraph gRes = new DocumentWordGraph(CorrelationWindow);
205                 gRes.DataString = DataString;
206                 gRes.DegradedEdges.putAll((HashMap<?, ?>)DegradedEdges.clone());
207                 gRes.NGramGraphArray = new UniqueVertexGraph[NGramGraphArray.length];
208                 int iCnt = 0;
209                 UniqueVertexGraph auniquevertexgraph[] = NGramGraphArray;
210
211         int i = auniquevertexgraph.length;
212         for(int j = 0; j < i; j++) {
213                 UniqueVertexGraph uCur = auniquevertexgraph[j];
214                 gRes.NGramGraphArray[iCnt++] = (UniqueVertexGraph)uCur.clone();
215         }
216
217         gRes.Normalizer = Normalizer;
218         gRes.TextPreprocessor = TextPreprocessor;
219         gRes.WordEvaluator = WordEvaluator;
220         return gRes;
221     }
222 }
```

```java
package sentimentanalysis;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
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.Properties;

public class ModelGraphs {

	private int noOfNGramGraphsReviews;
	private int noOfWordGraphsReviews;

	private int nSize;
	private int window;

	private boolean remove;
	private boolean shuffle;
	private boolean preprocess;

	private long seed;
	private int minPosRating;
	private int maxPosRating;
	private int minNegRating;
	private int maxNegRating;

	private String posGraphFilepath;
	private String negGraphFilepath;

	private int graphRangeId;
	private File resultsDirectory;
	private File infoFile;

	private String posNGramGraphBinaryFile;
	private String negNGramGraphBinaryFile;
	private String posWordGraphBinaryFile;
	private String negWordGraphBinaryFile;

	private ArrayList<String> posGraphFilenames;
	private ArrayList<String> negGraphFilenames;

	public static void main(String[] args) throws IOException {
		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 stage1 = Boolean.parseBoolean(args[1]);
		boolean allStages = Boolean.parseBoolean(args[2]);
```

```
61
62              System.out.println("Running First Stage of Sentiment "
63                              + "Classification: Creating Graphs\n");
64
65              long start = System.currentTimeMillis();
66
67              ModelGraphs object = new ModelGraphs();
68
69              object.setParameters(properties, true);
70              object.createResultsDirectory();
71              object.setFiles(allStages);
72
73              object.findGraphFilenames(object.getNoOfWordGraphsReviews(), stage1);
74
75              long wordGraphs = System.currentTimeMillis();
76
77              object.createWordGraphs(allStages, stage1);
78
79              long nGramGraphs = System.currentTimeMillis();
80
81              object.createNGramGraphs(allStages, stage1);
82
83              long end = System.currentTimeMillis();
84
85              NumberFormat formatter = new DecimalFormat("#0.00000");
86
87              FileWriter output = new FileWriter(object.getInfoFile(), true);
88
89              output.write("Time for word graphs: ");
90              output.write(formatter.format((nGramGraphs - wordGraphs) / 1000d)
91                              + " seconds\n");
92              output.write("Time for n-gram graphs: ");
93              output.write(formatter.format((end - nGramGraphs) / 1000d)
94                              + " seconds\n\n");
95              output.write("windowSize is" + object.getWindow() + "\n\n");
96              output.write("\nExecution time is ");
97              output.write(formatter.format((end - start) / 1000d) + " seconds\n");
98              output.close();
99
100             System.out.println("\nwindowSize is" + object.getWindow() + "\n\n");
101             System.out.println("\nExecution time is ");
102             System.out.println(formatter.format((end - start) / 1000d) + "
                    seconds");
103
104             return;
105     }
106
107     public void setParameters(Properties properties,
108                     boolean onlyThisStage) {
109
110             System.out.println("Setting parameters...");
111             setNoOfWordGraphsReviews(Integer.parseInt(
112                             properties.getProperty("noOfGraphReviews")));
113             setNoOfNGramGraphsReviews(Integer.parseInt(
114                             properties.getProperty("noOfGraphReviews")));
115
116             setnSize(Integer.parseInt(properties.getProperty("nSize")));
117             setWindow(Integer.parseInt(properties.getProperty("windowSize")));
118
119             setMinPosRating(Integer.parseInt(
```

```
                        properties.getProperty("minPositiveRating")));
                setMaxPosRating(Integer.parseInt(
                        properties.getProperty("maxPositiveRating")));
                setMinNegRating(Integer.parseInt(
                        properties.getProperty("minNegativeRating")));
                setMaxNegRating(Integer.parseInt(
                        properties.getProperty("maxNegativeRating")));

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

                setPosGraphFilepath(properties.getProperty("positiveGraphFilepath"));
                setNegGraphFilepath(properties.getProperty("negativeGraphFilepath"));
        }

        public void createResultsDirectory() {
                System.out.println("Creating directory for results...");
                String preprocessing = "NP", removing = "NR", results;
                if (isRemove())
                        removing = "R";
                if (isPreprocess())
                        preprocessing = "P";

                results = getWindow() + "_" + removing + "_" + preprocessing + "_";
                results += getnSize() + "_" + removing + "_" + preprocessing +
                        "_Graphs";

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

        public void setFiles(boolean allStages) {
                System.out.println("Setting the output filenames...");

                graphRangeId = 0;
                String filePrefix = getResultsDirectory() + "/";
                String fileSuffix = "_" + getNoOfWordGraphsReviews() +
                   "_posWordGraph";

                if (allStages) {
                        setPosNGramGraphBinaryFile(filePrefix +
                           "BinaryPositiveNGramGraph"
                                + getNoOfNGramGraphsReviews());
                        setNegNGramGraphBinaryFile(filePrefix +
                           "BinaryNegativeNGramGraph"
                                + getNoOfNGramGraphsReviews());
                        setPosWordGraphBinaryFile(filePrefix +
                           "BinaryPositiveWordGraph"
                                + getNoOfWordGraphsReviews());
                        setNegWordGraphBinaryFile(filePrefix +
                           "BinaryNegativeWordGraph"
                                + getNoOfWordGraphsReviews());
                        setInfoFile(new File(filePrefix + "Info.txt"));
                } else {
```

```
175                     while (new File(filePrefix + graphRangeId +
                            fileSuffix).isFile())
176                             graphRangeId++;
177
178                     setPosWordGraphBinaryFile(filePrefix + graphRangeId +
                            fileSuffix);
179
180                     fileSuffix = "_" + getNoOfWordGraphsReviews() +
                            "_negWordGraph";
181                     setNegWordGraphBinaryFile(filePrefix + graphRangeId +
                            fileSuffix);
182
183                     fileSuffix = "_" + getNoOfNGramGraphsReviews() +
                            "_posNGramGraph";
184                     setPosNGramGraphBinaryFile(filePrefix + graphRangeId +
                            fileSuffix);
185
186                     fileSuffix = "_" + getNoOfNGramGraphsReviews() +
                            "_negNGramGraph";
187                     setNegNGramGraphBinaryFile(filePrefix + graphRangeId +
                            fileSuffix);
188
189                     fileSuffix = "_" + getNoOfWordGraphsReviews();
190                     fileSuffix += "_" + getNoOfNGramGraphsReviews() +
                            "_Info.txt";
191                     setInfoFile(new File(filePrefix + graphRangeId +
                            fileSuffix));
192             }
193     }
194
195     public void findGraphFilenames(int noOfReviews, boolean stage)
196                     throws IOException {
197
198             System.out.println("Selecting reviews for graphs...");
199
200             FilenamePattern pattern;
201             pattern = new FilenamePattern(getMinPosRating(), getMaxPosRating());
202             setPosGraphFilenames(pattern.findFilenames(getPosGraphFilepath(),
203                             noOfReviews, isShuffle(), getSeed()));
204
205             pattern = new FilenamePattern(getMinNegRating(), getMaxNegRating());
206             setNegGraphFilenames(pattern.findFilenames(getNegGraphFilepath(),
207                             noOfReviews, isShuffle(), getSeed()));
208
209             if (stage) {
210                     writeFilenames(getPosGraphFilenames(),
                            "FilenamesForPosGraph_" +
211                                     graphRangeId + "_" +
                                        getNoOfWordGraphsReviews() + ".txt");
212                     writeFilenames(getNegGraphFilenames(),
                            "FilenamesForNegGraph_" +
213                                     graphRangeId + "_" +
                                        getNoOfWordGraphsReviews() + ".txt");
214             } else {
215                     writeFilenames(getPosGraphFilenames(),
                            "FilenamesForPosGraph.txt");
216                     writeFilenames(getNegGraphFilenames(),
                            "FilenamesForNegGraph.txt");
217             }
218     }
```

```
219
220         public void writeFilenames(ArrayList<String> filenames, String outputFile)
221                 throws IOException {
222
223             FileWriter writer = new FileWriter(
224                     new File(getResultsDirectory() + "/" + outputFile),
                        true);
225
226             for (String s: filenames)
227                 writer.write(s +  "\n");
228
229             writer.close();
230         }
231
232         public void createWordGraphs(boolean allStages, boolean stage1)
233                 throws FileNotFoundException, IOException {
234
235             System.out.println("Creating positive and negative model "
236                     + "word graphs...");
237
238             ModelWordGraph posGraph = createWordGraph(getPosGraphFilepath(),
239                     getPosGraphFilenames());
240             ModelWordGraph negGraph = createWordGraph(getNegGraphFilepath(),
241                     getNegGraphFilenames());
242
243             if (isRemove()) {
244                 WordGraphsSimilarities values =
245                         new WordGraphsSimilarities(posGraph,
                            negGraph);
246                 values.removeCommonSubGraphFromSentimentGraphs(isRemove());
247                 posGraph = values.getPosModelGraph();
248                 negGraph = values.getNegModelGraph();
249             }
250
251             storeWordGraphs(posGraph, negGraph);
252
253             if (allStages) {
254                 String filePrefix = getResultsDirectory() + "/";
255                 String infoPos = "PositiveWordGraph";
256                 String infoNeg = "NegativeWordGraph";
257                 String suffix = getNoOfWordGraphsReviews() + ".txt";
258
259                 posGraph.printToFile(filePrefix + infoPos + suffix);
260                 negGraph.printToFile(filePrefix + infoNeg + suffix);
261             } else if (stage1) {
262                 String filePrefix = getResultsDirectory() + "/";
263                 String infoPos = "PositiveWordGraph_" + graphRangeId + "_";
264                 String infoNeg = "NegativeWordGraph_" + graphRangeId + "_";
265                 String suffix = getNoOfWordGraphsReviews() + ".txt";
266
267                 posGraph.printToFile(filePrefix + infoPos + suffix);
268                 negGraph.printToFile(filePrefix + infoNeg + suffix);
269             }
270         }
271
272         public ModelWordGraph createWordGraph(String reviewsFilepath,
273                 ArrayList<String> reviewFilenames)
274                         throws FileNotFoundException, IOException {
275
276             ModelWordGraph graph;
```

```
277
278             graph = new ModelWordGraph(getNoOfWordGraphsReviews(), getWindow());
279             graph.createModelGraph(reviewsFilepath, reviewFilenames,
280                             isPreprocess());
281
282             return graph;
283         }
284
285         public void storeWordGraphs(ModelWordGraph posGraph,
286                         ModelWordGraph negGraph)
287                                         throws IOException {
288
289             FileWriter output =     new FileWriter(getInfoFile(), true);
290
291             output.write("Nodes of pos wordGraph: " + posGraph.getNoOfNodes());
292             output.write("\nEdges of pos wordGraph: " + posGraph.getNoOfEdges());
293             output.write("\nNodes of neg wordGraph: " + negGraph.getNoOfNodes());
294             output.write("\nEdges of neg wordGraph: " + negGraph.getNoOfEdges());
295             output.write("\n");
296             output.close();
297
298             posGraph.storeModelGraph(getPosWordGraphBinaryFile());
299             negGraph.storeModelGraph(getNegWordGraphBinaryFile());
300         }
301
302         public void createNGramGraphs(boolean allStages, boolean stage1)
303                         throws FileNotFoundException, IOException {
304
305             System.out.println("Creating positive and negative model "
306                             + "n-gram graphs...");
307
308             ModelNGramGraph posGraph = createNGramGraph(getPosGraphFilepath(),
309                             getPosGraphFilenames());
310             ModelNGramGraph negGraph = createNGramGraph(getNegGraphFilepath(),
311                             getNegGraphFilenames());
312
313             if (isRemove()) {
314                     NGramGraphsSimilarities values =
315                                     new NGramGraphsSimilarities(posGraph,
                                           negGraph);
316                     values.removeCommonSubGraphFromSentimentGraphs(isRemove());
317                     posGraph = values.getPosModelGraph();
318                     negGraph = values.getNegModelGraph();
319             }
320
321             storeNGramGraphs(posGraph, negGraph);
322
323             String filePrefix = getResultsDirectory() + "/";
324             String fileSuffix = getNoOfNGramGraphsReviews() + ".txt";
325
326             if (allStages) {
327                     String infoPos = "PositiveNGramGraph";
328                     String infoNeg = "NegativeNGramGraph";
329                     posGraph.printToFile(filePrefix + infoPos + fileSuffix);
330                     negGraph.printToFile(filePrefix + infoNeg + fileSuffix);
331             } else if (stage1) {
332                     String infoPos = "PositiveNGramGraph_" + graphRangeId + "_";
333                     String infoNeg = "NegativeNGramGraph_" + graphRangeId + "_";
334                     posGraph.printToFile(filePrefix + infoPos + fileSuffix);
335                     negGraph.printToFile(filePrefix + infoNeg + fileSuffix);
```

```
            }
        }

        public ModelNGramGraph createNGramGraph(String reviewsFilepath,
                ArrayList<String> reviewFilenames)
                                throws FileNotFoundException, IOException {

            ModelNGramGraph graph;

            graph = new ModelNGramGraph(getNoOfNGramGraphsReviews(), getnSize());
            graph.createModelGraph(reviewsFilepath, reviewFilenames,
                            isPreprocess());

            return graph;
        }

        public void storeNGramGraphs(ModelNGramGraph posGraph,
                ModelNGramGraph negGraph)
                                throws IOException {

            FileWriter output =     new FileWriter(getInfoFile(), true);

            output.write("Nodes of pos nGramGraph: " + posGraph.getNoOfNodes());
            output.write("\nEdges of pos nGramGraph: " +
               posGraph.getNoOfEdges());
            output.write("\nNodes of neg nGramGraph: " +
               negGraph.getNoOfNodes());
            output.write("\nEdges of neg nGramGraph: " +
               negGraph.getNoOfEdges());
            output.write("\n");
            output.close();

            posGraph.storeModelGraph(getPosNGramGraphBinaryFile());
            negGraph.storeModelGraph(getNegNGramGraphBinaryFile());
        }

        public int getNoOfNGramGraphsReviews() {
            return noOfNGramGraphsReviews;
        }

        public void setNoOfNGramGraphsReviews(int noOfNGramGraphsReviews) {
            this.noOfNGramGraphsReviews = noOfNGramGraphsReviews;
        }

        public int getNoOfWordGraphsReviews() {
            return noOfWordGraphsReviews;
        }

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

        public int getnSize() {
            return nSize;
        }

        public void setnSize(int nSize) {
            this.nSize = nSize;
        }

```

```
393         public int getWindow() {
394                 return window;
395         }
396 
397         public void setWindow(int window) {
398                 this.window = window;
399         }
400 
401         public boolean isRemove() {
402                 return remove;
403         }
404 
405         public void setRemove(boolean remove) {
406                 this.remove = remove;
407         }
408 
409         public boolean isShuffle() {
410                 return shuffle;
411         }
412 
413         public void setShuffle(boolean shuffle) {
414                 this.shuffle = shuffle;
415         }
416 
417         public boolean isPreprocess() {
418                 return preprocess;
419         }
420 
421         public void setPreprocess(boolean preprocess) {
422                 this.preprocess = preprocess;
423         }
424 
425         public int getMinPosRating() {
426                 return minPosRating;
427         }
428 
429         public void setMinPosRating(int minPosRating) {
430                 this.minPosRating = minPosRating;
431         }
432 
433         public int getMaxPosRating() {
434                 return maxPosRating;
435         }
436 
437         public void setMaxPosRating(int maxPosRating) {
438                 this.maxPosRating = maxPosRating;
439         }
440 
441         public int getMinNegRating() {
442                 return minNegRating;
443         }
444 
445         public void setMinNegRating(int minNegRating) {
446                 this.minNegRating = minNegRating;
447         }
448 
449         public int getMaxNegRating() {
450                 return maxNegRating;
451         }
452 
```

```
453         public void setMaxNegRating(int maxNegRating) {
454                 this.maxNegRating = maxNegRating;
455         }
456
457         public String getPosGraphFilepath() {
458                 return posGraphFilepath;
459         }
460
461         public void setPosGraphFilepath(String posGraphFilepath) {
462                 this.posGraphFilepath = posGraphFilepath;
463         }
464
465         public String getNegGraphFilepath() {
466                 return negGraphFilepath;
467         }
468
469         public void setNegGraphFilepath(String negGraphFilepath) {
470                 this.negGraphFilepath = negGraphFilepath;
471         }
472
473         public File getResultsDirectory() {
474                 return resultsDirectory;
475         }
476
477         public void setResultsDirectory(File resultsDirectory) {
478                 this.resultsDirectory = resultsDirectory;
479         }
480
481         public String getPosNGramGraphBinaryFile() {
482                 return posNGramGraphBinaryFile;
483         }
484
485         public void setPosNGramGraphBinaryFile(String posNGramGraphBinaryFile) {
486                 this.posNGramGraphBinaryFile = posNGramGraphBinaryFile;
487         }
488
489         public String getNegNGramGraphBinaryFile() {
490                 return negNGramGraphBinaryFile;
491         }
492
493         public void setNegNGramGraphBinaryFile(String negNGramGraphBinaryFile) {
494                 this.negNGramGraphBinaryFile = negNGramGraphBinaryFile;
495         }
496
497         public String getPosWordGraphBinaryFile() {
498                 return posWordGraphBinaryFile;
499         }
500
501         public void setPosWordGraphBinaryFile(String posWordGraphBinaryFile) {
502                 this.posWordGraphBinaryFile = posWordGraphBinaryFile;
503         }
504
505         public String getNegWordGraphBinaryFile() {
506                 return negWordGraphBinaryFile;
507         }
508
509         public void setNegWordGraphBinaryFile(String negWordGraphBinaryFile) {
510                 this.negWordGraphBinaryFile = negWordGraphBinaryFile;
511         }
512
```

```
513         public ArrayList<String> getPosGraphFilenames() {
514                 return posGraphFilenames;
515         }
516
517         public void setPosGraphFilenames(ArrayList<String> posGraphFilenames) {
518                 this.posGraphFilenames = posGraphFilenames;
519         }
520
521         public ArrayList<String> getNegGraphFilenames() {
522                 return negGraphFilenames;
523         }
524
525         public void setNegGraphFilenames(ArrayList<String> negGraphFilenames) {
526                 this.negGraphFilenames = negGraphFilenames;
527         }
528
529         public int getGraphRangeId() {
530                 return graphRangeId;
531         }
532
533         public void setGraphRangeId(int id) {
534                 this.graphRangeId = id;
535         }
536
537         public File getInfoFile() {
538                 return infoFile;
539         }
540
541         public void setInfoFile(File infoFile) {
542                 this.infoFile = infoFile;
543         }
544
545         public long getSeed() {
546                 return seed;
547         }
548
549         public void setSeed(long seed) {
550                 this.seed = seed;
551         }
552 }
```

```java
package sentimentanalysis;

import java.io.IOException;

import
   gr.demokritos.iit.jinsect.documentModel.comparators.NGramCachedGraphComparator;
import gr.demokritos.iit.jinsect.structs.GraphSimilarity;


public class WordGraphsSimilarities {

        private GraphSimilarity posGraphSimilarities;
        private ModelWordGraph posModelGraph;

        private GraphSimilarity negGraphSimilarities;
        private ModelWordGraph negModelGraph;

        public WordGraphsSimilarities(String posGraphFile, String negGraphFile)
                        throws ClassNotFoundException, IOException {
                posModelGraph = new ModelWordGraph();
                posModelGraph.loadModelGraph(posGraphFile);
                negModelGraph = new ModelWordGraph();
                negModelGraph.loadModelGraph(negGraphFile);
        }

        public WordGraphsSimilarities(ModelWordGraph posModelGraph,
                        ModelWordGraph negModelGraph) {
                this.posModelGraph = posModelGraph;
                this.negModelGraph = negModelGraph;
        }

        public void graphsSimilaritiesWith(ReviewWordGraph reviewGraph) {

                NGramCachedGraphComparator reviewGraphComparator =
                                new NGramCachedGraphComparator();

                posGraphSimilarities = reviewGraphComparator.getSimilarityBetween(
                                reviewGraph.getGraph(),
                                posModelGraph.getReviewsGraph().getGraph());

                negGraphSimilarities = reviewGraphComparator.getSimilarityBetween(
                                reviewGraph.getGraph(),
                                negModelGraph.getReviewsGraph().getGraph());
        }

        public void removeCommonSubGraphFromSentimentGraphs(boolean remove) {
                if (remove == false)
                        return;
                ReviewWordGraph[] newModels =
                                posModelGraph.removeCommonSubgraph(
                                                negModelGraph.getReviewsGraph());
                posModelGraph.setReviewsGraph(newModels[0]);
                negModelGraph.setReviewsGraph(newModels[1]);
        }

        public GraphSimilarity getPosGraphSimilarities() {
                return posGraphSimilarities;
        }

        public GraphSimilarity getNegGraphSimilarities() {
```

```
60                  return negGraphSimilarities;
61          }
62
63          public ModelWordGraph getPosModelGraph() {
64                  return posModelGraph;
65          }
66
67          public ModelWordGraph getNegModelGraph() {
68                  return negModelGraph;
69          }
70  }
```

```
package sentimentanalysis;

import java.io.IOException;

import
   gr.demokritos.iit.jinsect.documentModel.comparators.NGramCachedGraphComparator;
import gr.demokritos.iit.jinsect.structs.GraphSimilarity;


public class NGramGraphsSimilarities {

        private GraphSimilarity posGraphSimilarities;
        private ModelNGramGraph posModelGraph;

        private GraphSimilarity negGraphSimilarities;
        private ModelNGramGraph negModelGraph;

        public NGramGraphsSimilarities(String posGraphFile, String negGraphFile)
                        throws ClassNotFoundException, IOException {
                posModelGraph = new ModelNGramGraph();
                posModelGraph.loadModelGraph(posGraphFile);
                negModelGraph = new ModelNGramGraph();
                negModelGraph.loadModelGraph(negGraphFile);
        }

        public NGramGraphsSimilarities(ModelNGramGraph posModelGraph,
                        ModelNGramGraph negModelGraph) {
                this.posModelGraph = posModelGraph;
                this.negModelGraph = negModelGraph;
        }

        public void graphsSimilaritiesWith(ReviewNGramGraph reviewGraph) {
                NGramCachedGraphComparator reviewGraphComparator =
                                new NGramCachedGraphComparator();

                posGraphSimilarities = reviewGraphComparator.getSimilarityBetween(
                                reviewGraph.getGraph(),
                                posModelGraph.getReviewsGraph().getGraph());
                negGraphSimilarities = reviewGraphComparator.getSimilarityBetween(
                                reviewGraph.getGraph(),
                                negModelGraph.getReviewsGraph().getGraph());
        }

        public void removeCommonSubGraphFromSentimentGraphs(boolean remove)
                        throws IOException {
                if (remove == false)
                        return;
                ReviewNGramGraph[] newModels =
                                posModelGraph.removeCommonSubgraph(
                                                negModelGraph.getReviewsGraph());
                posModelGraph.setReviewsGraph(newModels[0]);
                negModelGraph.setReviewsGraph(newModels[1]);
        }

        public GraphSimilarity getPosGraphSimilarities() {
                return posGraphSimilarities;
        }

        public GraphSimilarity getNegGraphSimilarities() {
                return negGraphSimilarities;
```

```
60          }
61
62          public ModelNGramGraph getPosModelGraph() {
63                  return posModelGraph;
64          }
65
66          public ModelNGramGraph getNegModelGraph() {
67                  return negModelGraph;
68          }
69  }
```

```
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, String posFilepath,
                        String negFilepath, ArrayList<String> posReviewFilenames,
                        ArrayList<String> negReviewFilenames) throws IOException {

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

        public void createFile(NGramGraphsSimilarities values, String posFilepath,
                        String negFilepath, ArrayList<String> posReviewFilenames,
                        ArrayList<String> negReviewFilenames) throws IOException {

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

        public void createFile(String posFilepath, String negFilepath,
                        ArrayList<String> posReviewFilenames,
                        ArrayList<String> negReviewFilenames) throws IOException {

                createTextAttribute();
                addHeader();
                addData(posFilepath, negFilepath, posReviewFilenames,
                                negReviewFilenames);
        }

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

```

```
61          public void createFile(String file, int levels) throws IOException {
62                  createAttributes();
63                  addHeader();
64                  addData(file, levels);
65          }
66
67          private void createAttributes() {
68                  attributes = new ArrayList<Attribute>();
69                  attributes.add(new Attribute("PositiveContainmentSimilarity"));
70                  attributes.add(new Attribute("PositiveNormalizedValueSimilarity"));
71                  attributes.add(new Attribute("PositiveValueSimilarity"));
72                  attributes.add(new Attribute("NegativeContainmentSimilarity"));
73                  attributes.add(new Attribute("NegativeNormalizedValueSimilarity"));
74                  attributes.add(new Attribute("NegativeValueSimilarity"));
75                  ArrayList<String> sentimentValues = new ArrayList<String>();
76                  sentimentValues.add("0");
77                  sentimentValues.add("1");
78                  attributes.add(new Attribute("Sentiment", sentimentValues));
79          }
80
81          private void createTextAttribute() {
82                  attributes = new ArrayList<Attribute>();
83                  attributes.add(new Attribute("TextReview", (ArrayList<String>)
                       null));
84                  ArrayList<String> sentimentValues = new ArrayList<String>();
85                  sentimentValues.add("0");
86                  sentimentValues.add("1");
87                  attributes.add(new Attribute("ReviewSentiment", sentimentValues));
88          }
89
90          private void createRelativeAttributes() {
91                  attributes = new ArrayList<Attribute>();
92                  attributes.add(new Attribute("RelativeContainmentSimilarity"));
93                  attributes.add(new Attribute("RelativeNormalizedValueSimilarity"));
94                  attributes.add(new Attribute("RelativeValueSimilarity"));
95                  ArrayList<String> sentimentValues = new ArrayList<String>();
96                  sentimentValues.add("0");
97                  sentimentValues.add("1");
98                  attributes.add(new Attribute("Sentiment", sentimentValues));
99          }
100
101         private void addHeader() {
102                 instances = new Instances(relationName, attributes, 0);
103         }
104
105         private void addData(WordGraphsSimilarities values, String posFilepath,
106                         String negFilepath, ArrayList<String> posReviewFilenames,
107                         ArrayList<String> negReviewFilenames) throws IOException {
108
109                 addInstances(values.getPosModelGraph(), posFilepath,
110                                 posReviewFilenames, values, 1);
111                 addInstances(values.getNegModelGraph(), negFilepath,
112                                 negReviewFilenames, values, 0);
113         }
114
115         private void addData(NGramGraphsSimilarities values, String posFilepath,
116                         String negFilepath, ArrayList<String> posReviewFilenames,
117                         ArrayList<String> negReviewFilenames) throws IOException {
118
119                 addInstances(values.getPosModelGraph(), posFilepath,
```

```
                        posReviewFilenames, values, 1);
                addInstances(values.getNegModelGraph(), negFilepath,
                        negReviewFilenames, values, 0);
        }

        private void addData(String posFilepath, String negFilepath,
                        ArrayList<String> posReviewFilenames,
                        ArrayList<String> negReviewFilenames) throws IOException {

                addInstances(posFilepath, posReviewFilenames, 1);
                addInstances(negFilepath, negReviewFilenames, 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 addData(String file, int levels) throws IOException {
                BufferedReader reader = new BufferedReader(new FileReader(file));
                for (int line = 0; line < 11; line++)
                        reader.readLine();

                double[] max = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};

                String line = reader.readLine();
                while(line != null) {
                        String[] values = line.split(",");
                        double[] simValues = new double[6];
                        for (int index = 0; index < values.length - 1; index++)
                                simValues[index] = Double.parseDouble(values[index]);

                        for (int i = 0; i < simValues.length; i++)
                                if (simValues[i] > max[i])
                                        max[i] = simValues[i];
                        line = reader.readLine();
                }

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

                line = reader.readLine();
```

```
180                 while(line != null) {
181                         String[] values = line.split(",");
182                         double[] simValues = new double[values.length];
183                         for (int index = 0; index < values.length - 1; index++)
184                                 simValues[index] = Double.parseDouble(values[index]);
185                         simValues[values.length - 1] =
186                                         Integer.parseInt(values[values.length - 1]);
187 
188                         addInstance(simValues, max, levels);
189                         line = reader.readLine();
190                 }
191                 reader.close();
192         }
193 
194         private void addInstances(ModelWordGraph graph, String reviewsFilepath,
195                         ArrayList<String> reviewFilenames, WordGraphsSimilarities
                            values,
196                         int sentiment) throws IOException {
197 
198                 ReviewWordGraph reviewGraph = new ReviewWordGraph(graph.getWindow(),
199                                 graph.getReviewsGraph().isPreprocess());
200 
201                 for (String s: reviewFilenames) {
202                         reviewGraph.createGraph(reviewsFilepath.concat(s));
203                         values.graphsSimilaritiesWith(reviewGraph);
204                         addInstance(values.getPosGraphSimilarities(),
205                                         values.getNegGraphSimilarities(), sentiment);
206                 }
207         }
208 
209         private void addInstances(ModelNGramGraph graph, String reviewsFilepath,
210                         ArrayList<String> reviewFilenames, NGramGraphsSimilarities
                            values,
211                         int sentiment) throws IOException {
212 
213                 ReviewNGramGraph reviewGraph = new ReviewNGramGraph(graph.getNSize(),
214                                 graph.getReviewsGraph().isPreprocess());
215 
216                 for (String s: reviewFilenames) {
217                         reviewGraph.createGraph(reviewsFilepath.concat(s));
218                         values.graphsSimilaritiesWith(reviewGraph);
219                         addInstance(values.getPosGraphSimilarities(),
220                                         values.getNegGraphSimilarities(), sentiment);
221                 }
222         }
223 
224         private void addInstances(String reviewsFilepath,
225                         ArrayList<String> reviewFilenames,
226                         int sentiment) throws IOException {
227 
228                 String reviewFile = null;
229                 BufferedReader reader = null;
230 
231                 for (String s: reviewFilenames) {
232                         reviewFile = reviewsFilepath.concat(s);
233                         reader = new BufferedReader(new FileReader(reviewFile));
234                         addInstance(reader.readLine(), sentiment);
235                 }
236                 reader.close();
237         }
```

```
238
239         private void addInstance(GraphSimilarity posGraphSim,
240                 GraphSimilarity negGraphSim, int sentiment) {
241
242             double[] instance = new double[instances.numAttributes()];
243
244             instance[0] = posGraphSim.ContainmentSimilarity;
245             instance[1] = posGraphSim.ValueSimilarity/posGraphSim.SizeSimilarity;
246             instance[2] = posGraphSim.ValueSimilarity;
247
248             instance[3] = negGraphSim.ContainmentSimilarity;
249             instance[4] = negGraphSim.ValueSimilarity/negGraphSim.SizeSimilarity;
250             instance[5] = negGraphSim.ValueSimilarity;
251
252             instance[6] = sentiment;
253
254             instances.add(new DenseInstance(1.0, instance));
255         }
256
257         private void addInstance(String text, int sentiment) {
258
259             double[] instance = new double[instances.numAttributes()];
260             instance[0] = instances.attribute(0).addStringValue(text);
261             instance[1] = sentiment;
262             instances.add(new DenseInstance(1.0, instance));
263         }
264
265         private void addInstance(double[] simValues) {
266             double[] instance = new double[instances.numAttributes()];
267
268             instance[0] = dsim(simValues[0], simValues[3]);
269             instance[1] = dsim(simValues[1], simValues[4]);
270             instance[2] = dsim(simValues[2], simValues[5]);
271
272             instance[3] = simValues[6];
273
274             instances.add(new DenseInstance(1.0, instance));
275         }
276
277         private void addInstance(double[] simValues, double[] max, int levels) {
278             double[] instance = new double[instances.numAttributes()];
279
280             for (int i = 0; i < simValues.length - 1; i++)
281                 instance[i] = Math.ceil((simValues[i] / max[i]) * levels);
282
283             instance[instances.numAttributes() - 1] = simValues[simValues.length
                - 1];
284
285             instances.add(new DenseInstance(1.0, instance));
286         }
287
288         private int dsim(double posSim, double negSim) {
289             int equal = 0;
290             int positive = 1;
291             int negative = 2;
292
293             if (posSim < negSim) return negative;
294             else if (posSim > negSim) return positive;
295             else return equal;
296         }
```

```
297
298         public void storeToFile(String outputFile) throws IOException {
299                 BufferedWriter writer = new BufferedWriter(new
                        FileWriter(outputFile));
300                 writer.write(instances.toString());
301                 writer.flush();
302                 writer.close();
303         }
304
305         public Instances getInstances() {
306                 return instances;
307         }
308 }
```

```
package sentimentanalysis;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
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.Properties;

public class DataFiles {

        private int noTrainReviews;
        private int noTestReviews;
        private int noOfNGramGraphsReviews;
        private int noOfWordGraphsReviews;

        private String nGramGraphsTrainFile;
        private String wordGraphsTrainFile;
        private String bagOfWordsTrainFile;

        private String nGramGraphsTestFile;
        private String wordGraphsTestFile;
        private String bagOfWordsTestFile;

        private String posTrainFilepath;
        private String negTrainFilepath;

        private String posTestFilepath;
        private String negTestFilepath;

        private ArrayList<String> posGraphFilenames;
        private ArrayList<String> negGraphFilenames;
        private ArrayList<String> posTrainFilenames;
        private ArrayList<String> negTrainFilenames;
        private ArrayList<String> posTestFilenames;
        private ArrayList<String> negTestFilenames;

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

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

        private String posNGramGraphBinaryFile;
        private String negNGramGraphBinaryFile;
        private String posWordGraphBinaryFile;
        private String negWordGraphBinaryFile;

        private int id;

        private File resultsDirectory;

        public static void main(String[] args)
```

```
61                  throws IOException, ClassNotFoundException {
62
63          if (args.length != 3)
64                  return;
65
66          InputStream reader = new FileInputStream(args[0]);
67          Properties properties = new Properties();
68
69          if (args[0].contains(".xml"))
70                  properties.loadFromXML(reader);
71          else
72                  properties.load(reader);
73
74          boolean stage2 = Boolean.parseBoolean(args[1]);
75          boolean allStages = Boolean.parseBoolean(args[2]);
76
77          long start = System.currentTimeMillis();
78
79          System.out.println("Running Second Stage of Sentiment "
80                          + "Classification: Creating training and testing
                                files for"
81                          + " each approach\n");
82
83          DataFiles object = new DataFiles();
84
85          object.setParameters(properties, true);
86          object.createResultsDirectory();
87          object.setFiles(allStages);
88
89          object.findTrainFilenames(stage2);
90          object.findTestFilenames(stage2);
91
92          long wordGraphs = System.currentTimeMillis();
93
94          object.wordGraphsFilesARFF();
95          long nGramGraphs = System.currentTimeMillis();
96          object.nGramGraphsFilesARFF();
97          long bagOfWords = System.currentTimeMillis();
98          object.bagOfWordsFilesARFF();
99
100         long end = System.currentTimeMillis();
101
102         NumberFormat formatter = new DecimalFormat("#0.00000");
103         String info = object.getNoOfWordGraphsReviews() + "_"
104                         + object.getNoOfNGramGraphsReviews() + "_"
105                         + object.getNoTrainReviews() + "_"
106                         + object.getNoTestReviews();
107
108         FileWriter output =
109                         new FileWriter(new File(object.getResultsDirectory()
110                                         + "/" + object.getId() + "_" + info
                                                + "_Info.txt"));
111
112         output.write("Time for files of word graphs: ");
113         output.write(formatter.format((nGramGraphs - wordGraphs) / 1000d)
114                         + " seconds\n");
115         output.write("Time for files of n-gram graphs: ");
116         output.write(formatter.format((bagOfWords - nGramGraphs) / 1000d)
117                         + " seconds\n");
118         output.write("Time for files of bagOfWords: ");
```

```
119                 output.write(formatter.format((end - bagOfWords) / 1000d)
120                                 + " seconds\n\n");
121                 output.write("\nExecution time is ");
122                 output.write(formatter.format((end - start) / 1000d) + " seconds");
123                 output.close();
124
125                 System.out.print("\nExecution time is " +
126                                 formatter.format((end - start) / 1000d) + "
                                    seconds");
127
128                 return;
129         }
130
131         public void setParameters(Properties properties, boolean onlyThisStage)
132                         throws ClassNotFoundException, IOException {
133
134                 setNoTrainReviews(Integer.parseInt(
135                                 properties.getProperty("noOfTrainReviews")));
136                 setNoTestReviews(Integer.parseInt(
137                                 properties.getProperty("noOfTestReviews")));
138
139                 setPosTrainFilepath(properties.getProperty("positiveTrainFilepath"));
140                 setNegTrainFilepath(properties.getProperty("negativeTrainFilepath"));
141                 setPosTestFilepath(properties.getProperty("positiveTestFilepath"));
142                 setNegTestFilepath(properties.getProperty("negativeTestFilepath"));
143                 setPosRate(Double.parseDouble(
144                                 properties.getProperty("positiveRate"))/100);
145
146                 if (properties.getProperty("seed") != null)
147                         setSeed(Long.parseLong(properties.getProperty("seed")));
148                 else
149                         setSeed(-1);
150
151                 setShuffle(Boolean.parseBoolean(
152                                 properties.getProperty("shuffle")));
153
154                 if (onlyThisStage) {
155                         System.out.println("Setting parameters...");
156
157                         setMinPosRating(Integer.parseInt(
158                                         properties.getProperty("minPositiveRating")));
159
160                         setMaxPosRating(Integer.parseInt(
161                                         properties.getProperty("maxPositiveRating")));
162
163                         setMinNegRating(Integer.parseInt(
164                                         properties.getProperty("minNegativeRating")));
165
166                         setMaxNegRating(Integer.parseInt(
167                                         properties.getProperty("maxNegativeRating")));
168
169                         setPosWordGraphBinaryFile(
170                                         properties.getProperty("posWordGraphFilepath"));
171
172                         setNegWordGraphBinaryFile(
173                                         properties.getProperty("negWordGraphFilepath"));
174
175                         setPosWordGraphFilenames(getPosWordGraphBinaryFile());
176                         setNegWordGraphFilenames(getNegWordGraphBinaryFile());
177
```

```
                setPosNGramGraphBinaryFile(
                                properties.getProperty("posNGramGraphFilepath"));

                setNegNGramGraphBinaryFile(
                                properties.getProperty("negNGramGraphFilepath"));

                setPosNGramGraphFilenames(getPosNGramGraphBinaryFile());
                setNegNGramGraphFilenames(getNegNGramGraphBinaryFile());
            }
        }

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

            String[] resultsFolderInfo = getPosWordGraphBinaryFile().split("/");

            String[] foldername =
                            resultsFolderInfo[resultsFolderInfo.length -
                               2].split("_");

            String[] filename =
                            resultsFolderInfo[resultsFolderInfo.length -
                               1].split("_");

            setNoOfWordGraphsReviews(Integer.parseInt(filename[1]));
            String window = foldername[0];
            String remove = foldername[1];
            String preprocess = foldername[2];
            String nSize = foldername[3];

            String results = filename[0] + "_" + window + "_" + remove
                            + "_" + preprocess + "_";

            resultsFolderInfo = getPosNGramGraphBinaryFile().split("/");
            filename = resultsFolderInfo[resultsFolderInfo.length -
               1].split("_");

            setNoOfNGramGraphsReviews(Integer.parseInt(filename[1]));

            results += filename[0] + "_" + nSize + "_" + remove
                            + "_" + preprocess + "_" + "Train&TestFiles";

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

        public void setFiles(boolean allStages) {

            String filePrefix = getResultsDirectory() + "/";
            String fileInfo = "_" + getNoOfWordGraphsReviews() + "_"
                            + getNoOfNGramGraphsReviews() + "_";

            String fileTrainSuffix = getNoTrainReviews() +
               "_TrainBagOfWords.arff";
            String fileTestSuffix = getNoTestReviews() + "_TestBagOfWords.arff";
            id = 0;

            if (allStages) {
                    setBagOfWordsTrainFile(filePrefix + "Train" +
                       getNoTrainReviews()
```

```
                                        + "BagOfWords.arff");
                        setBagOfWordsTestFile(filePrefix + "Test" +
                            getNoTestReviews()
                                        + "BagOfWords.arff");
                        setWordGraphsTrainFile(filePrefix + "Train" +
                            getNoTrainReviews()
                                        + "WordGraph.arff");
                        setWordGraphsTestFile(filePrefix + "Test" +
                            getNoTestReviews()
                                        + "WordGraph.arff");
                        setnGramGraphsTrainFile(filePrefix + "Train" +
                            getNoTrainReviews()
                                        + "NGramGraph.arff");
                        setnGramGraphsTestFile(filePrefix + "Test" +
                            getNoTestReviews()
                                        + "NGramGraph.arff");
                } else {
                        System.out.println("Setting the output filenames...");
                        File filepath = getResultsDirectory();
                        File[] allFiles = filepath.listFiles();
                        if (allFiles != null)
                                id = findIdOfLastFile(allFiles);

                        setBagOfWordsTrainFile(filePrefix + id + fileInfo +
                            fileTrainSuffix);
                        setBagOfWordsTestFile(filePrefix + id + fileInfo +
                            fileTestSuffix);

                        fileTrainSuffix = getNoTrainReviews() +
                            "_TrainWordGraph.arff";
                        fileTestSuffix = getNoTestReviews() + "_TestWordGraph.arff";
                        setWordGraphsTrainFile(filePrefix + id + fileInfo +
                            fileTrainSuffix);
                        setWordGraphsTestFile(filePrefix + id + fileInfo +
                            fileTestSuffix);

                        fileTrainSuffix = getNoTrainReviews() +
                            "_TrainNGramGraph.arff";
                        fileTestSuffix = getNoTestReviews() + "_TestNGramGraph.arff";
                        setnGramGraphsTrainFile(filePrefix + id + fileInfo +
                            fileTrainSuffix);
                        setnGramGraphsTestFile(filePrefix + id + fileInfo +
                            fileTestSuffix);
                }
        }

        private int findIdOfLastFile(File[] allFiles) {
                long lastMod = Long.MIN_VALUE;
                File last = null;
                for (File file : allFiles)
                        if (file.lastModified() > lastMod) {
                                last = file;
                                lastMod = file.lastModified();
                        }

                if (last != null)
                        return Integer.parseInt(last.getName().split("_")[0]) + 1;
                else
                        return 0;
        }
```

```
280
281         public void findTrainFilenames(boolean stage2) throws IOException {
282                 System.out.println("Selecting reviews for the training corpus...");
283
284                 FilenamePattern filePattern =
285                                 new FilenamePattern(getMinPosRating(),
                                        getMaxPosRating());
286
287                 setPosTrainFilenames(
288                                 filePattern.findFilenames(getPosTrainFilepath(),
289                                                 (int) (getNoTrainReviews() *
                                                        getPosRate()),
290                                                 getPosGraphFilenames(), isShuffle(),
                                                        getSeed()));
291
292                 filePattern =
293                                 new FilenamePattern(getMinNegRating(),
                                        getMaxNegRating());
294
295                 setNegTrainFilenames(
296                                 filePattern.findFilenames(getNegTrainFilepath(),
297                                                 (int) (getNoTrainReviews() * (1 -
                                                        getPosRate())),
298                                                 getNegGraphFilenames(), isShuffle(),
                                                        getSeed()));
299
300                 if (stage2) {
301                         writeFilenames(getPosTrainFilenames(), "FilenamesForTrain_"
                                + id
302                                         + "_" + getNoOfWordGraphsReviews() + "_"
303                                         + getNoOfNGramGraphsReviews() + "_"
304                                         + getNoTrainReviews() + ".txt");
305                         writeFilenames(getNegTrainFilenames(), "FilenamesForTrain_"
                                + id
306                                         + "_" + getNoOfWordGraphsReviews()
307                                         + "_" + getNoOfNGramGraphsReviews()
308                                         + "_" + getNoTrainReviews() + ".txt");
309                 } else {
310                         writeFilenames(getPosTrainFilenames(),
                                "FilenamesForTrain.txt");
311                         writeFilenames(getNegTrainFilenames(),
                                "FilenamesForTrain.txt");
312                 }
313         }
314
315         public void findTestFilenames(boolean stage2) throws IOException {
316                 System.out.println("Selecting reviews for the testing corpus...");
317
318                 FilenamePattern filePattern =
319                                 new FilenamePattern(getMinPosRating(),
                                        getMaxPosRating());
320                 setPosTestFilenames(
321                                 filePattern.findFilenames(getPosTestFilepath(),
322                                                 (int) (getNoTestReviews() * posRate),
323                                                 isShuffle(), getSeed()));
324
325                 filePattern =
326                                 new FilenamePattern(getMinNegRating(),
                                        getMaxNegRating());
327                 setNegTestFilenames(
```

```
328                             filePattern.findFilenames(getNegTestFilepath(),
329                                     (int) (getNoTestReviews() * (1 -
                                            posRate)),
330                                     isShuffle(), getSeed()));
331
332             if (stage2) {
333                     writeFilenames(getPosTestFilenames(), "FilenamesForTest_" +
                            id
334                                     + "_" + getNoOfWordGraphsReviews()
335                                     + "_" + getNoOfNGramGraphsReviews()
336                                     + "_" + getNoTestReviews() + ".txt");
337                     writeFilenames(getNegTestFilenames(), "FilenamesForTest_" +
                            id
338                                     + "_" + getNoOfWordGraphsReviews()
339                                     + "_" + getNoOfNGramGraphsReviews()
340                                     + "_" + getNoTestReviews() + ".txt");
341             } else {
342                     writeFilenames(getPosTestFilenames(),
                            "FilenamesForTest.txt");
343                     writeFilenames(getNegTestFilenames(),
                            "FilenamesForTest.txt");
344             }
345     }
346
347     public void writeFilenames(ArrayList<String> filenames, String outputFile)
348                     throws IOException {
349
350             FileWriter writer = new FileWriter(
351                             new File(getResultsDirectory() + "/" + outputFile),
                                    true);
352
353             for (String s: filenames)
354                     writer.write(s +  "\n");
355
356             writer.close();
357     }
358
359     public void wordGraphsFilesARFF()
360                     throws ClassNotFoundException, IOException {
361
362             System.out.println("Creating training and testing files by comparing
                    "
363                             + "reviews to model word graphs...");
364             WordGraphsSimilarities values =
365                             new
                                    WordGraphsSimilarities(getPosWordGraphBinaryFile(),
366                                             getNegWordGraphBinaryFile());
367
368             AttributeRelationFile train = createTrainFileARFF(values);
369             train.storeToFile(getWordGraphsTrainFile());
370             train = null;
371
372             AttributeRelationFile test = createTestFileARFF(values);
373             test.storeToFile(getWordGraphsTestFile());
374     }
375
376     public AttributeRelationFile createTrainFileARFF(WordGraphsSimilarities
            values)
377                     throws FileNotFoundException, IOException {
378
```

```
379             AttributeRelationFile file =
380                     new
                            AttributeRelationFile("Sentiment_of_Similarities_of_"
381                                     + "ImdbReviewWordGraphs");
382
383             file.createFile(values, getPosTrainFilepath(), getNegTrainFilepath(),
384                     getPosTrainFilenames(), getNegTrainFilenames());
385
386             return file;
387         }
388
389         public AttributeRelationFile createTestFileARFF(WordGraphsSimilarities
                values)
390                 throws FileNotFoundException, IOException {
391
392             AttributeRelationFile file =
393                     new
                            AttributeRelationFile("Sentiment_of_Similarities_of_"
394                                     + "ImdbReviewWordGraphs");
395
396             file.createFile(values, getPosTestFilepath(), getNegTestFilepath(),
397                     getPosTestFilenames(), getNegTestFilenames());
398
399             return file;
400         }
401
402         public void nGramGraphsFilesARFF()
403                 throws ClassNotFoundException, IOException {
404
405             System.out.println("Creating training and testing files by comparing
                    "
406                     + "reviews to model n-gram graphs...");
407             NGramGraphsSimilarities values =
408                     new
                            NGramGraphsSimilarities(getPosNGramGraphBinaryFile(),
409                                     getNegNGramGraphBinaryFile());
410
411             AttributeRelationFile train = createTrainFileARFF(values);
412             train.storeToFile(getnGramGraphsTrainFile());
413             train = null;
414
415             AttributeRelationFile test = createTestFileARFF(values);
416             test.storeToFile(getnGramGraphsTestFile());
417         }
418
419         public AttributeRelationFile createTrainFileARFF(NGramGraphsSimilarities
                values)
420                 throws FileNotFoundException, IOException {
421
422             AttributeRelationFile file =
423                     new
                            AttributeRelationFile("Sentiment_of_Similarities_of_"
424                                     + "ImdbReviewNGramGraphs");
425             file.createFile(values, getPosTrainFilepath(), getNegTrainFilepath(),
426                     getPosTrainFilenames(), getNegTrainFilenames());
427
428             return file;
429         }
430
```

```
431         public AttributeRelationFile createTestFileARFF(NGramGraphsSimilarities
                values)
432                         throws FileNotFoundException, IOException {
433
434                 AttributeRelationFile file =
435                                 new
                                        AttributeRelationFile("Sentiment_of_Similarities_of_"
436                                                         + "ImdbReviewNGramGraphs");
437                 file.createFile(values, getPosTestFilepath(), getNegTestFilepath(),
438                                 getPosTestFilenames(), getNegTestFilenames());
439
440                 return file;
441         }
442
443         public void bagOfWordsFilesARFF()
444                         throws FileNotFoundException, IOException {
445
446                 System.out.println("Creating training and testing files with bag of "
447                                 + "words approach...");
448
449                 AttributeRelationFile train = bagOfWordsTrainFileARFF();
450                 train.storeToFile(getBagOfWordsTrainFile());
451                 train = null;
452
453                 AttributeRelationFile test = bagOfWordsTestFileARFF();
454                 test.storeToFile(getBagOfWordsTestFile());
455         }
456
457         public AttributeRelationFile bagOfWordsTrainFileARFF()
458                         throws FileNotFoundException, IOException {
459
460                 AttributeRelationFile file =
461                                 new
                                        AttributeRelationFile("Sentiment_of_ImdbReviewsText");
462
463                 file.createFile(getPosTrainFilepath(), getNegTrainFilepath(),
464                                 getPosTrainFilenames(), getNegTrainFilenames());
465
466                 return file;
467         }
468
469         public AttributeRelationFile bagOfWordsTestFileARFF()
470                         throws FileNotFoundException, IOException {
471
472                 AttributeRelationFile file =
473                                 new
                                        AttributeRelationFile("Sentiment_of_ImdbReviewsText");
474
475                 file.createFile(getPosTestFilepath(), getNegTestFilepath(),
476                                 getPosTestFilenames(), getNegTestFilenames());
477
478                 return file;
479         }
480
481         public int getNoTrainReviews() {
482                 return noTrainReviews;
483         }
484
485         public void setNoTrainReviews(int noTrainReviews) {
486                 this.noTrainReviews = noTrainReviews;
```

```
487         }
488
489         public int getNoTestReviews() {
490                 return noTestReviews;
491         }
492
493         public void setNoTestReviews(int noTestReviews) {
494                 this.noTestReviews = noTestReviews;
495         }
496
497         public String getPosTrainFilepath() {
498                 return posTrainFilepath;
499         }
500
501         public void setPosTrainFilepath(String posTrainFilepath) {
502                 this.posTrainFilepath = posTrainFilepath;
503         }
504
505         public String getNegTrainFilepath() {
506                 return negTrainFilepath;
507         }
508
509         public void setNegTrainFilepath(String negTrainFilepath) {
510                 this.negTrainFilepath = negTrainFilepath;
511         }
512
513         public String getPosTestFilepath() {
514                 return posTestFilepath;
515         }
516
517         public void setPosTestFilepath(String posTestFilepath) {
518                 this.posTestFilepath = posTestFilepath;
519         }
520
521         public String getNegTestFilepath() {
522                 return negTestFilepath;
523         }
524
525         public void setNegTestFilepath(String negTestFilepath) {
526                 this.negTestFilepath = negTestFilepath;
527         }
528
529         public double getPosRate() {
530                 return posRate;
531         }
532
533         public void setPosRate(double posRate) {
534                 this.posRate = posRate;
535         }
536
537         public boolean isShuffle() {
538                 return shuffle;
539         }
540
541         public void setShuffle(boolean shuffle) {
542                 this.shuffle = shuffle;
543         }
544
545         public int getMinPosRating() {
546                 return minPosRating;
```

```
547         }
548
549         public void setMinPosRating(int minPosRating) {
550                 this.minPosRating = minPosRating;
551         }
552
553         public int getMaxPosRating() {
554                 return maxPosRating;
555         }
556
557         public void setMaxPosRating(int maxPosRating) {
558                 this.maxPosRating = maxPosRating;
559         }
560
561         public int getMinNegRating() {
562                 return minNegRating;
563         }
564
565         public void setMinNegRating(int minNegRating) {
566                 this.minNegRating = minNegRating;
567         }
568
569         public int getMaxNegRating() {
570                 return maxNegRating;
571         }
572
573         public void setMaxNegRating(int maxNegRating) {
574                 this.maxNegRating = maxNegRating;
575         }
576
577         public String getPosNGramGraphBinaryFile() {
578                 return posNGramGraphBinaryFile;
579         }
580
581         public void setPosNGramGraphBinaryFile(String posNGramGraphBinaryFile) {
582                 this.posNGramGraphBinaryFile = posNGramGraphBinaryFile;
583         }
584
585         public String getNegNGramGraphBinaryFile() {
586                 return negNGramGraphBinaryFile;
587         }
588
589         public void setNegNGramGraphBinaryFile(String negNGramGraphBinaryFile) {
590                 this.negNGramGraphBinaryFile = negNGramGraphBinaryFile;
591         }
592
593         public String getPosWordGraphBinaryFile() {
594                 return posWordGraphBinaryFile;
595         }
596
597         public void setPosWordGraphBinaryFile(String posWordGraphBinaryFile) {
598                 this.posWordGraphBinaryFile = posWordGraphBinaryFile;
599         }
600
601         public String getNegWordGraphBinaryFile() {
602                 return negWordGraphBinaryFile;
603         }
604
605         public void setNegWordGraphBinaryFile(String negWordGraphBinaryFile) {
606                 this.negWordGraphBinaryFile = negWordGraphBinaryFile;
```

```
607         }
608
609         public void setPosWordGraphFilenames(String posWordGraphFilepath)
610                 throws ClassNotFoundException, IOException {
611
612             ModelWordGraph model = new ModelWordGraph();
613             model.loadModelGraph(posWordGraphFilepath);
614
615             setPosGraphFilenames(model.getReviewFilenames());
616         }
617
618         public void setNegWordGraphFilenames(String negWordGraphFilepath)
619                 throws ClassNotFoundException, IOException {
620
621             ModelWordGraph model = new ModelWordGraph();
622             model.loadModelGraph(negWordGraphFilepath);
623
624             setNegGraphFilenames(model.getReviewFilenames());
625         }
626
627         public void setPosNGramGraphFilenames(String posNGramGraphFilepath)
628                 throws ClassNotFoundException, IOException {
629
630             ModelNGramGraph model = new ModelNGramGraph();
631             model.loadModelGraph(posNGramGraphFilepath);
632
633             setPosGraphFilenames(model.getReviewFilenames());
634         }
635
636         public void setNegNGramGraphFilenames(String negNGramGraphFilepath)
637                 throws ClassNotFoundException, IOException {
638
639             ModelNGramGraph model = new ModelNGramGraph();
640             model.loadModelGraph(negNGramGraphFilepath);
641
642             setNegGraphFilenames(model.getReviewFilenames());
643         }
644
645         public ArrayList<String> getPosGraphFilenames() {
646             return posGraphFilenames;
647         }
648
649         public void setPosGraphFilenames(ArrayList<String> posGraphFilenames) {
650             this.posGraphFilenames = posGraphFilenames;
651         }
652
653         public ArrayList<String> getNegGraphFilenames() {
654             return negGraphFilenames;
655         }
656
657         public void setNegGraphFilenames(ArrayList<String> negGraphFilenames) {
658             this.negGraphFilenames = negGraphFilenames;
659         }
660
661         public int getId() {
662             return id;
663         }
664
665         public void setId(int id) {
666             this.id = id;
```

```
667         }
668
669         public int getNoOfNGramGraphsReviews() {
670                 return noOfNGramGraphsReviews;
671         }
672
673         public void setNoOfNGramGraphsReviews(int noOfNGramGraphsReviews) {
674                 this.noOfNGramGraphsReviews = noOfNGramGraphsReviews;
675         }
676
677         public int getNoOfWordGraphsReviews() {
678                 return noOfWordGraphsReviews;
679         }
680
681         public void setNoOfWordGraphsReviews(int noOfWordGraphsReviews) {
682                 this.noOfWordGraphsReviews = noOfWordGraphsReviews;
683         }
684
685         public File getResultsDirectory() {
686                 return resultsDirectory;
687         }
688
689         public void setResultsDirectory(File resultsDirectory) {
690                 this.resultsDirectory = resultsDirectory;
691         }
692
693         public String getnGramGraphsTrainFile() {
694                 return nGramGraphsTrainFile;
695         }
696
697         public void setnGramGraphsTrainFile(String nGramGraphsTrainFile) {
698                 this.nGramGraphsTrainFile = nGramGraphsTrainFile;
699         }
700
701         public String getWordGraphsTrainFile() {
702                 return wordGraphsTrainFile;
703         }
704
705         public void setWordGraphsTrainFile(String wordGraphsTrainFile) {
706                 this.wordGraphsTrainFile = wordGraphsTrainFile;
707         }
708
709         public String getBagOfWordsTrainFile() {
710                 return bagOfWordsTrainFile;
711         }
712
713         public void setBagOfWordsTrainFile(String bagOfWordsTrainFile) {
714                 this.bagOfWordsTrainFile = bagOfWordsTrainFile;
715         }
716
717         public String getnGramGraphsTestFile() {
718                 return nGramGraphsTestFile;
719         }
720
721         public void setnGramGraphsTestFile(String nGramGraphsTestFile) {
722                 this.nGramGraphsTestFile = nGramGraphsTestFile;
723         }
724
725         public String getWordGraphsTestFile() {
726                 return wordGraphsTestFile;
```

```
727             }
728
729             public void setWordGraphsTestFile(String wordGraphsTestFile) {
730                     this.wordGraphsTestFile = wordGraphsTestFile;
731             }
732
733             public String getBagOfWordsTestFile() {
734                     return bagOfWordsTestFile;
735             }
736
737             public void setBagOfWordsTestFile(String bagOfWordsTestFile) {
738                     this.bagOfWordsTestFile = bagOfWordsTestFile;
739             }
740
741             public ArrayList<String> getPosTrainFilenames() {
742                     return posTrainFilenames;
743             }
744
745             public void setPosTrainFilenames(ArrayList<String> posTrainFilenames) {
746                     this.posTrainFilenames = posTrainFilenames;
747             }
748
749             public ArrayList<String> getNegTrainFilenames() {
750                     return negTrainFilenames;
751             }
752
753             public void setNegTrainFilenames(ArrayList<String> negTrainFilenames) {
754                     this.negTrainFilenames = negTrainFilenames;
755             }
756
757             public ArrayList<String> getPosTestFilenames() {
758                     return posTestFilenames;
759             }
760
761             public void setPosTestFilenames(ArrayList<String> posTestFilenames) {
762                     this.posTestFilenames = posTestFilenames;
763             }
764
765             public ArrayList<String> getNegTestFilenames() {
766                     return negTestFilenames;
767             }
768
769             public void setNegTestFilenames(ArrayList<String> negTestFilenames) {
770                     this.negTestFilenames = negTestFilenames;
771             }
772
773             public long getSeed() {
774                     return seed;
775             }
776
777             public void setSeed(long seed) {
778                     this.seed = seed;
779             }
780     }
```

```
package sentimentanalysis;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

import weka.classifiers.*;
import weka.classifiers.meta.FilteredClassifier;
import weka.core.Instances;
import weka.core.tokenizers.WordTokenizer;
import weka.filters.unsupervised.attribute.StringToWordVector;

public class SentimentClassifier {

        private String classifierName;
        private String trainFile;
        private String filter;
        private Classifier classifier;
        private FilteredClassifier filteredClassifier;

        public SentimentClassifier() { }

        public SentimentClassifier(String classifierName, String trainFile) {
                this.classifierName = classifierName;
                this.trainFile = trainFile;
                filter = null;
        }

        public SentimentClassifier(String classifierName, String trainFile,
                        String filter) {
                this.classifierName = classifierName;
                this.trainFile = trainFile;
                this.filter = filter;
        }

        public void createClassifierInstance()
                        throws InstantiationException, IllegalAccessException,
                        ClassNotFoundException {
                classifier = (Classifier)
                   Class.forName(classifierName).newInstance();
        }

        public void trainClassifier() throws Exception {
                if (filter == null) {
                        Instances trainInstances = null;
                        if (classifierName.equals(
                                        "weka.classifiers.bayes.NaiveBayesMultinomial"))
                                           {
                                trainInstances =
                                   discretizeSimilarityValues(trainFile);
                        } else {
                                BufferedReader reader =
                                                new BufferedReader(new
                                                   FileReader(trainFile));
                                trainInstances = new Instances(reader);
                        }
```

```
57                      trainInstances.setClassIndex(trainInstances.numAttributes()
                            - 1);
58                      classifier.buildClassifier(trainInstances);
59              }
60      }
61
62      public void trainClassifier(boolean preprocess) throws Exception {
63              if (filter == "StringToWordVector") {
64                      BufferedReader reader = new BufferedReader(
65                                      new FileReader(trainFile));
66                      Instances trainInstances = new Instances(reader);
67                      trainInstances.setClassIndex(trainInstances.numAttributes()
                            - 1);
68                      stringToWordVector(preprocess);
69                      filteredClassifier.buildClassifier(trainInstances);
70              }
71      }
72
73      private void stringToWordVector(boolean preprocess) {
74              filteredClassifier = new FilteredClassifier();
75              filteredClassifier.setClassifier(this.classifier);
76
77              StringToWordVector filter = new StringToWordVector();
78              filter.setAttributeIndices("1,2");
79              filter.setWordsToKeep(100000);
80              filter.setLowerCaseTokens(true);
81
82              if (!preprocess) {
83                      WordTokenizer tokenizer = new WordTokenizer();
84                      tokenizer.setDelimiters(" ");
85                      filter.setTokenizer(tokenizer);
86                      filter.setLowerCaseTokens(false);
87              }
88              filter.setOutputWordCounts(true);
89              filteredClassifier.setFilter(filter);
90      }
91
92      public Instances discretizeSimilarityValues(String file)
93                      throws IOException {
94
95              AttributeRelationFile discreteValuesFile =
96                              new AttributeRelationFile("Sentiment_of_Discrete_"
97                                              +
                                                    "Similarities_of_ImdbReviewGraphs");
98
99              discreteValuesFile.createFile(file);
100             String foldername = new File(file).getParent();
101             String filename = new File(file).getName();
102             String prefix = filename.split("\\.")[0];
103             String newFile = foldername + "/" + prefix + "_discretized.arff";
104             discreteValuesFile.storeToFile(newFile);
105             if (prefix.contains("Train"))
106                     trainFile = newFile;
107             return discreteValuesFile.getInstances();
108     }
109
110     public void storeSentimentClassifier(String outputClassifierFile)
111                     throws IOException {
112
113             FileOutputStream output = new FileOutputStream(outputClassifierFile);
```

```
                ObjectOutputStream outputClassifierStream =
                                new ObjectOutputStream(output);

                outputClassifierStream.writeObject(classifierName);
                outputClassifierStream.writeObject(trainFile);
                outputClassifierStream.writeObject(filter);
                outputClassifierStream.writeObject(classifier);
                outputClassifierStream.writeObject(filteredClassifier);
                outputClassifierStream.close();
        }

        public void loadSentimentClassifier(String inputClassifierFile)
                        throws IOException, ClassNotFoundException {

                FileInputStream input = new FileInputStream(inputClassifierFile);
                ObjectInputStream inputClassifierStream = new
                   ObjectInputStream(input);

                classifierName = (String) inputClassifierStream.readObject();
                trainFile = (String) inputClassifierStream.readObject();
                filter = (String) inputClassifierStream.readObject();
                classifier = (Classifier) inputClassifierStream.readObject();
                filteredClassifier =
                                (FilteredClassifier)
                                   inputClassifierStream.readObject();
                inputClassifierStream.close();
        }

        public String getClassifierName() {
                return classifierName;
        }

        public String getTrainFile() {
                return trainFile;
        }

        public String getFilter() {
                return filter;
        }

        public Classifier getClassifierInstance() {
                return classifier;
        }

        public FilteredClassifier getFilteredClassifierInstance() {
                return filteredClassifier;
        }
}
```

```
package sentimentanalysis;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

import weka.classifiers.Evaluation;
import weka.core.Instances;

public class ClassifierEvaluation {

        private Evaluation evaluation;
        private SentimentClassifier classifier;
        private String testFile;

        public ClassifierEvaluation() { }

        public ClassifierEvaluation(SentimentClassifier classifier,
                        String testFile) {
                this.classifier = classifier;
                this.testFile = testFile;
        }

        public ClassifierEvaluation(String classifierFile, String testFile)
                        throws ClassNotFoundException, IOException {

                classifier = new SentimentClassifier();
                classifier.loadSentimentClassifier(classifierFile);
                this.testFile = testFile;
        }

        public void evaluateClassifier() throws Exception {

                BufferedReader reader = new BufferedReader(
                                new FileReader(classifier.getTrainFile()));
                Instances trainInstances = new Instances(reader);
                trainInstances.setClassIndex(trainInstances.numAttributes() - 1);
                evaluation = new Evaluation(trainInstances);

                Instances testInstances = null;
                if (!testFile.contains("Bag") &&
                        (classifier.getClassifierName()).equals(
                                        "weka.classifiers.bayes.NaiveBayesMultinomial"))
                                           {
                                testInstances =
                                   classifier.discretizeSimilarityValues(testFile);
                } else {
                        reader = new BufferedReader(new FileReader(testFile));
                        testInstances = new Instances(reader);
                }

                testInstances.setClassIndex(testInstances.numAttributes() - 1);

                if (classifier.getFilter() == null)
                        evaluation.evaluateModel(classifier.getClassifierInstance(),
                                        testInstances);
                else
                        evaluation.evaluateModel(
```

```
                                        classifier.getFilteredClassifierInstance(),
                                            testInstances);
        }

        public void printAccuracyToSystemOutput() {
                System.out.println(evaluation.toSummaryString("\nResults\n==========="
                                + "===\n\n", false));
        }

        public void printAccuracyToFile(String filepath) throws IOException {
                FileWriter writer = new FileWriter(filepath);
                writer.write(evaluation.toSummaryString("\nResults\n==========\n\n",
                                false));
                writer.close();
        }

        public void printAccuracyToFile(String filepath, String method)
                        throws IOException {
                FileWriter writer = new FileWriter(filepath, true);
                writer.write(evaluation.toSummaryString("\nResults " + method +
                    "\n=="
                                + "========\n\n", false));
                writer.write("\n");
                writer.close();
        }

        public void printConfMatrixToSystemOutput() throws Exception {
                System.out.println(evaluation.toMatrixString("\nResults\n==========="
                                + "===\n\n"));
        }

        public void printConfMatrixToFile(String filepath) throws Exception {
                FileWriter writer = new FileWriter(filepath, true);
                writer.write(evaluation.toMatrixString("\nResults\n==========\n\n"));
                writer.write("\n");
                writer.close();
        }

        public void printConfMatrixToFile(String filepath, String method)
                        throws Exception {
                FileWriter writer = new FileWriter(filepath, true);
                writer.write(evaluation.toMatrixString("\nResults " + method + "\n=="
                                + "========\n\n"));
                writer.write("\n");
                writer.close();
        }
}
```

```
package sentimentanalysis;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.InputStream;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.Properties;

public class Classifiers {

        private String nGramGraphsTrainFile;
        private String wordGraphsTrainFile;
        private String bagOfWordsTrainFile;

        private String nGramGraphsTestFile;
        private String wordGraphsTestFile;
        private String bagOfWordsTestFile;

        private String wekaClassifierName;
        private String classifierName;

        private int noTrainReviews;
        private int noTestReviews;

        private int noOfNGramGraphsReviews;
        private int noOfWordGraphsReviews;

        private boolean preprocess;

        private int id;

        private String nGramGraphsClassifierBinaryFile;
        private String wordGraphsClassifierBinaryFile;
        private String bagOfWordsClassifierBinaryFile;

        private File resultsDirectory;
        private String resultsFile;

        public static void main(String[] args) throws Exception {
                if (args.length != 2)
                        return;

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

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

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

                long start = System.currentTimeMillis();

                System.out.println("Running Third Stage of Sentiment "
                                + "Classification: Creating classifiers and
                                    evaluation for"
                                + " each approach\n");
```

```
60
61              Classifiers object = new Classifiers();
62
63              object.setParameters(properties, true);
64              object.createResultsDirectory();
65              object.setFiles(allStages);
66
67              System.out.println("Creating classifiers for each pair of training"
68                              + " and testing set...");
69
70              long wordGraphs = System.currentTimeMillis();
71              object.createClassifierForWordGraphs();
72              long nGramGraphs = System.currentTimeMillis();
73              object.createClassifierForNGramGraphs();
74              long bagOfWords = System.currentTimeMillis();
75              object.createClassifierForBagOfWords();
76
77              System.out.println("Evaluating each classifier...");
78              long wordGraphs2 = System.currentTimeMillis();
79              object.evaluateClassifierForWordGraphs();
80              long nGramGraphs2 = System.currentTimeMillis();
81              object.evaluateClassifierForNGramGraphs();
82              long bagOfWords2 = System.currentTimeMillis();
83              object.evaluateClassifierForBagOfWords();
84
85              long end = System.currentTimeMillis();
86
87              NumberFormat formatter = new DecimalFormat("#0.00000");
88
89              FileWriter output = new FileWriter(new File(
90                              object.getResultsFile()), true);
91
92              output.write("Time for classifier of word graphs: ");
93              output.write(formatter.format((nGramGraphs - wordGraphs) / 1000d)
94                              + " seconds\n");
95              output.write("Time for classifier of n-gram graphs: ");
96              output.write(formatter.format((bagOfWords - nGramGraphs) / 1000d)
97                              + " seconds\n");
98              output.write("Time for classifier of bagOfWords: ");
99              output.write(formatter.format((wordGraphs2 - bagOfWords) / 1000d)
100                             + " seconds\n\n");
101
102             output.write("Time for evaluating classifier of word graphs: ");
103             output.write(formatter.format((nGramGraphs2 - wordGraphs2) / 1000d)
104                             + " seconds\n");
105             output.write("Time for evaluating classifier of n-gram graphs: ");
106             output.write(formatter.format((bagOfWords2 - nGramGraphs2) / 1000d)
107                             + " seconds\n");
108             output.write("Time for evaluating classifier of bagOfWords: ");
109             output.write(formatter.format((end - bagOfWords2) / 1000d)
110                             + " seconds\n");
111
112             output.write("\nExecution time is ");
113             output.write(formatter.format((end - start) / 1000d) + " seconds");
114             output.close();
115
116             System.out.print("\nExecution time is " +
117                             formatter.format((end - start) / 1000d) + "
                                    seconds");
118
```

```
119         }
120
121         public void setParameters(Properties props, boolean onlyThisStage) {
122                 setWekaClassifierName(props.getProperty("classifierName"));
123                 String[] classifier = getWekaClassifierName().split("\\.");
124                 String classifierName = classifier[classifier.length - 1];
125                 setClassifierName(classifierName);
126
127                 if (onlyThisStage) {
128                         System.out.println("Setting parameters...");
129                         setBagOfWordsTrainFile(props.getProperty("bagOfWordsTrainFile"));
130                         setnGramGraphsTrainFile(
131                                         props.getProperty("nGramGraphsTrainFile"));
132                         setWordGraphsTrainFile(props.getProperty("wordGraphsTrainFile"));
133
134                         setBagOfWordsTestFile(props.getProperty("bagOfWordsTestFile"));
135                         setnGramGraphsTestFile(props.getProperty("nGramGraphsTestFile"));
136                         setWordGraphsTestFile(props.getProperty("wordGraphsTestFile"));
137                 }
138         }
139
140         public void createResultsDirectory() {
141                 System.out.println("Creating directory for results...");
142                 String results;
143
144                 String[] folderInfo = getnGramGraphsTestFile().split("/");
145                 String[] filename = folderInfo[folderInfo.length - 1].split("_");
146                 setNoTestReviews(Integer.parseInt(filename[3]));
147
148                 folderInfo = getnGramGraphsTestFile().split("/");
149                 String[] foldername = folderInfo[folderInfo.length - 2].split("_");
150                 filename = folderInfo[folderInfo.length - 1].split("_");
151
152                 setId(Integer.parseInt(filename[0]));
153                 setNoOfWordGraphsReviews(Integer.parseInt(filename[1]));
154                 setNoOfNGramGraphsReviews(Integer.parseInt(filename[2]));
155                 setNoTrainReviews(Integer.parseInt(filename[3]));
156                 String preprocess = foldername[3];
157                 if (preprocess.equals("NP"))
158                         setPreprocess(false);
159                 else
160                         setPreprocess(true);
161
162                 results = foldername[0] + "_" + foldername[1] + "_"
163                                 + foldername[2] + "_" + foldername[3] + "_"
164                                 + foldername[4] + "_" + foldername[5] + "_"
165                                 + foldername[6] + "_" + foldername[7] + "_" +
                                        classifierName;
166
167                 setResultsDirectory(new File(results));
168                 getResultsDirectory().mkdir();
169         }
170
171         public void setFiles(boolean allStages) {
172                 String filePrefix = getResultsDirectory() + "/Binary"
173                                 + getClassifierName();
174
175                 if (allStages) {
176                         setBagOfWordsClassifierBinaryFile(filePrefix
177                                         + "ClassifierBagOfWords");
```

```
                setWordGraphsClassifierBinaryFile(filePrefix
                        + "ClassifierWordGraphs");
                setnGramGraphsClassifierBinaryFile(filePrefix
                        + "ClassifierNGramGraphs");
                setResultsFile(getResultsDirectory() + "/" + "Results.txt");
            } else {
                System.out.println("Setting the output filenames...");
                String fileSuffix = id + "_" +
                        getNoOfWordGraphsReviews() + "_" +
                        getNoOfNGramGraphsReviews() + "_" +
                        getNoTrainReviews() + "_" +
                           getNoTestReviews();

                setBagOfWordsClassifierBinaryFile(filePrefix +
                   "ClassifierBagOfWords_"

                setWordGraphsClassifierBinaryFile(filePrefix +
                   "ClassifierWordGraphs_"

                setnGramGraphsClassifierBinaryFile(filePrefix +
                   "ClassifierNGramGraphs_"

                setResultsFile(getResultsDirectory() + "/" + fileSuffix


            }
        }

        public void createClassifierForBagOfWords() throws Exception {

            SentimentClassifier sentimentClassifier =
                        new SentimentClassifier(getWekaClassifierName(),
                                getBagOfWordsTrainFile(),
                                   "StringToWordVector");

            sentimentClassifier.createClassifierInstance();
            sentimentClassifier.trainClassifier(isPreprocess());
            sentimentClassifier.storeSentimentClassifier(
                                getBagOfWordsClassifierBinaryFile());
        }

        public void createClassifierForNGramGraphs() throws Exception {

            SentimentClassifier sentimentClassifier =
                        new SentimentClassifier(getWekaClassifierName(),
                                getnGramGraphsTrainFile());

            sentimentClassifier.createClassifierInstance();
            sentimentClassifier.trainClassifier();
            sentimentClassifier.storeSentimentClassifier(
                                getnGramGraphsClassifierBinaryFile());
        }

        public void createClassifierForWordGraphs() throws Exception {

            SentimentClassifier sentimentClassifier =
                        new SentimentClassifier(getWekaClassifierName(),
```

```
                                        getWordGraphsTrainFile());

                sentimentClassifier.createClassifierInstance();
                sentimentClassifier.trainClassifier();
                sentimentClassifier.storeSentimentClassifier(
                                getWordGraphsClassifierBinaryFile());
        }

        public void evaluateClassifierForWordGraphs() throws Exception {
                ClassifierEvaluation evaluation =
                                new ClassifierEvaluation(
                                        getWordGraphsClassifierBinaryFile(),
                                        getWordGraphsTestFile());

                evaluation.evaluateClassifier();
                evaluation.printAccuracyToFile(getResultsFile(), "WordGraphs");
        }

        public void evaluateClassifierForNGramGraphs() throws Exception {
                ClassifierEvaluation evaluation =
                                new ClassifierEvaluation(
                                        getnGramGraphsClassifierBinaryFile(),
                                        getnGramGraphsTestFile());

                evaluation.evaluateClassifier();
                evaluation.printAccuracyToFile(getResultsFile(), "NGramGraphs");
        }

        public void evaluateClassifierForBagOfWords() throws Exception {
                ClassifierEvaluation evaluation =
                                new ClassifierEvaluation(
                                        getBagOfWordsClassifierBinaryFile(),
                                        getBagOfWordsTestFile());

                evaluation.evaluateClassifier();
                evaluation.printAccuracyToFile(getResultsFile(), "BagOfWords");
        }

        public String getnGramGraphsTrainFile() {
                return nGramGraphsTrainFile;
        }

        public void setnGramGraphsTrainFile(String nGramGraphsTrainFile) {
                this.nGramGraphsTrainFile = nGramGraphsTrainFile;
        }

        public String getWordGraphsTrainFile() {
                return wordGraphsTrainFile;
        }

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

        public String getBagOfWordsTrainFile() {
                return bagOfWordsTrainFile;
        }

        public void setBagOfWordsTrainFile(String bagOfWordsTrainFile) {
                this.bagOfWordsTrainFile = bagOfWordsTrainFile;
```

```
289         }
290
291         public String getnGramGraphsTestFile() {
292                 return nGramGraphsTestFile;
293         }
294
295         public void setnGramGraphsTestFile(String nGramGraphsTestFile) {
296                 this.nGramGraphsTestFile = nGramGraphsTestFile;
297         }
298
299         public String getWordGraphsTestFile() {
300                 return wordGraphsTestFile;
301         }
302
303         public void setWordGraphsTestFile(String wordGraphsTestFile) {
304                 this.wordGraphsTestFile = wordGraphsTestFile;
305         }
306
307         public String getBagOfWordsTestFile() {
308                 return bagOfWordsTestFile;
309         }
310
311         public void setBagOfWordsTestFile(String bagOfWordsTestFile) {
312                 this.bagOfWordsTestFile = bagOfWordsTestFile;
313         }
314
315         public String getWekaClassifierName() {
316                 return wekaClassifierName;
317         }
318
319         public void setWekaClassifierName(String wekaClassifierName) {
320                 this.wekaClassifierName = wekaClassifierName;
321         }
322
323         public String getClassifierName() {
324                 return classifierName;
325         }
326
327         public void setClassifierName(String classifierName) {
328                 this.classifierName = classifierName;
329         }
330
331         public int getNoTrainReviews() {
332                 return noTrainReviews;
333         }
334
335         public void setNoTrainReviews(int noTrainReviews) {
336                 this.noTrainReviews = noTrainReviews;
337         }
338
339         public int getNoTestReviews() {
340                 return noTestReviews;
341         }
342
343         public void setNoTestReviews(int noTestReviews) {
344                 this.noTestReviews = noTestReviews;
345         }
346
347         public int getNoOfNGramGraphsReviews() {
348                 return noOfNGramGraphsReviews;
```

```
349         }
350
351         public void setNoOfNGramGraphsReviews(int noOfNGramGraphsReviews) {
352                 this.noOfNGramGraphsReviews = noOfNGramGraphsReviews;
353         }
354
355         public int getNoOfWordGraphsReviews() {
356                 return noOfWordGraphsReviews;
357         }
358
359         public void setNoOfWordGraphsReviews(int noOfWordGraphsReviews) {
360                 this.noOfWordGraphsReviews = noOfWordGraphsReviews;
361         }
362
363         public void setId(int id) {
364                 this.id = id;
365         }
366
367         public File getResultsDirectory() {
368                 return resultsDirectory;
369         }
370
371         public void setResultsDirectory(File resultsDirectory) {
372                 this.resultsDirectory = resultsDirectory;
373         }
374
375         public boolean isPreprocess() {
376                 return preprocess;
377         }
378
379         public void setPreprocess(boolean preprocess) {
380                 this.preprocess = preprocess;
381         }
382
383         public String getnGramGraphsClassifierBinaryFile() {
384                 return nGramGraphsClassifierBinaryFile;
385         }
386
387         public void setnGramGraphsClassifierBinaryFile(
388                         String nGramGraphsClassifierBinaryFile) {
389                 this.nGramGraphsClassifierBinaryFile =
                    nGramGraphsClassifierBinaryFile;
390         }
391
392         public String getWordGraphsClassifierBinaryFile() {
393                 return wordGraphsClassifierBinaryFile;
394         }
395
396         public void setWordGraphsClassifierBinaryFile(
397                         String wordGraphsClassifierBinaryFile) {
398                 this.wordGraphsClassifierBinaryFile = wordGraphsClassifierBinaryFile;
399         }
400
401         public String getBagOfWordsClassifierBinaryFile() {
402                 return bagOfWordsClassifierBinaryFile;
403         }
404
405         public void setBagOfWordsClassifierBinaryFile(
406                         String bagOfWordsClassifierBinaryFile) {
407                 this.bagOfWordsClassifierBinaryFile = bagOfWordsClassifierBinaryFile;
```

```
408             }
409
410             public String getResultsFile() {
411                     return resultsFile;
412             }
413
414             public void setResultsFile(String resultsFile) {
415                     this.resultsFile = resultsFile;
416             }
417
418             public int getId() {
419                     return id;
420             }
421     }
```

```java
package sentimentanalysis;

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

public class Run {

        private File resultsDirectory;

        public static boolean allStages = false;
        public static boolean stage1 = false;
        public static boolean stage2 = false;
        public static boolean stage3 = false;

        public static void main(String[] args) throws Exception {

                if (args.length != 1)
                        return;

                Run run =  new Run();

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

                String nSize = properties.getProperty("nSize");
                String noTrainReviews = properties.getProperty("noOfTrainReviews");
                String classifier = properties.getProperty("classifierName");

                if (nSize != null && noTrainReviews != null && classifier != null)
                        run.allStages(properties);

                else if (noTrainReviews == null && classifier == null)
                        run.stage1(args);

                else if (nSize == null && classifier == null)
                        run.stage2(args);

                else if (nSize == null && noTrainReviews == null)
                        run.stage3(args);
        }

        public void createResultsDirectory(ModelGraphs graphs, DataFiles files,
                        Classifiers classes) {
                System.out.println("Creating directory for results...");

                Date date = new Date();
                Calendar calendar = Calendar.getInstance();
                calendar.setTime(date);
```

```
61                int day = calendar.get(Calendar.DAY_OF_MONTH);
62                int month = calendar.get(Calendar.MONTH) + 1;
63                int year = calendar.get(Calendar.YEAR);
64                int hours = calendar.get(Calendar.HOUR_OF_DAY);
65                int minutes = calendar.get(Calendar.MINUTE);
66                int seconds = calendar.get(Calendar.SECOND);
67
68                String removing = "NR", preprocessing = "NP", shuffling = "NS",
69                                results;
70                if (graphs.isRemove())
71                        removing = "R";
72                if (graphs.isPreprocess())
73                        preprocessing = "P";
74                if (graphs.isShuffle())
75                        shuffling = "S";
76
77                results = graphs.getNoOfWordGraphsReviews() + "_" +
                     graphs.getWindow();
78                results +=  "_" + removing + "_" + preprocessing + "_" + shuffling;
79                results +=  "_" + graphs.getNoOfNGramGraphsReviews() + "_";
80                results += graphs.getnSize() + "_" + removing + "_" + preprocessing;
81                results += "_" + shuffling + "_" + files.getNoTrainReviews() + "_";
82                results += files.getNoTestReviews() + "_"+
                     classes.getClassifierName();
83                results += "_" + day + "" + month + "" + year;
84                results += "_" + hours + "." + minutes + "." + seconds;
85
86                setResultsDirectory(new File(results));
87                getResultsDirectory().mkdir();
88        }
89
90        public void stage1(String[] args) throws IOException {
91
92                stage1 = true;
93                String[] arg = new String[3];
94                arg[0] = args[0];
95                arg[1] = String.valueOf(stage1);
96                arg[2] = String.valueOf(allStages);
97
98                ModelGraphs.main(arg);
99        }
100
101       public void stage2(String[] args) throws ClassNotFoundException, IOException
              {
102               stage2 = true;
103               String[] arg = new String[3];
104               arg[0] = args[0];
105               arg[1] = String.valueOf(stage2);
106               arg[2] = String.valueOf(allStages);
107
108               DataFiles.main(arg);
109       }
110
111       public void stage3(String[] args) throws Exception {
112               stage3 = true;
113               String[] arg = new String[2];
114               arg[0] = args[0];
115               arg[1] = String.valueOf(allStages);
116
117               Classifiers.main(arg);
```

```
118         }
119
120         public void allStages(Properties properties) throws Exception {
121                 allStages = true;
122
123                 System.out.println("Running all Stages of Sentiment "
124                                 + "Classification\n");
125
126                 long start = System.currentTimeMillis();
127
128                 ModelGraphs graphs = new ModelGraphs();
129                 DataFiles files = new DataFiles();
130                 Classifiers classifiers = new Classifiers();
131
132                 setParameters(properties, graphs, files, classifiers);
133                 setFiles(graphs, files, classifiers);
134
135                 graphsStage(graphs, classifiers.getResultsFile());
136                 filesStage(graphs, files, classifiers.getResultsFile());
137                 classifiersStage(files, classifiers);
138
139                 long end = System.currentTimeMillis();
140
141                 NumberFormat formatter = new DecimalFormat("#0.00000");
142                 FileWriter output = new FileWriter(new File(
143                                 classifiers.getResultsFile()), true);
144
145                 output.write("\nExecution time is " +
146                                 formatter.format((end - start) / 1000d) + "
                                    seconds");
147                 output.close();
148                 System.out.print("\nExecution time is " +
149                                 formatter.format((end - start) / 1000d) + "
                                    seconds");
150         }
151
152         private void setParameters(Properties properties, ModelGraphs graphs,
153                         DataFiles files, Classifiers classifiers)
154                                         throws ClassNotFoundException, IOException {
155
156                 graphs.setParameters(properties, false);
157                 files.setParameters(properties, false);
158                 classifiers.setParameters(properties, false);
159
160                 createResultsDirectory(graphs, files, classifiers);
161
162                 graphs.setResultsDirectory(getResultsDirectory());
163                 files.setResultsDirectory(getResultsDirectory());
164                 classifiers.setResultsDirectory(getResultsDirectory());
165         }
166
167         private void setFiles(ModelGraphs graphs, DataFiles files,
168                         Classifiers classifiers) {
169                 graphs.setFiles(allStages);
170                 files.setFiles(allStages);
171                 classifiers.setFiles(allStages);
172         }
173
174         private void graphsStage(ModelGraphs graphs, String file)
175                         throws IOException {
```

```
176
177             graphs.findGraphFilenames(graphs.getNoOfWordGraphsReviews(), false);
178
179             long wordGraphs = System.currentTimeMillis();
180             graphs.createWordGraphs(allStages, stage1);
181
182             long nGramGraphs = System.currentTimeMillis();
183             graphs.createNGramGraphs(allStages, stage1);
184             long end = System.currentTimeMillis();
185
186             NumberFormat formatter = new DecimalFormat("#0.00000");
187             FileWriter output = new FileWriter(new File(file), true);
188             output.write("Time for word graphs: ");
189             output.write(formatter.format((nGramGraphs - wordGraphs) / 1000d)
190                             + " seconds\n");
191             output.write("Time for n-gram graphs: ");
192             output.write(formatter.format((end - nGramGraphs) / 1000d)
193                             + " seconds\n");
194             output.close();
195         }
196
197         private void filesStage(ModelGraphs graphs, DataFiles files, String file)
198                 throws IOException, ClassNotFoundException {
199
200             files.setPosWordGraphBinaryFile(graphs.getPosWordGraphBinaryFile());
201             files.setNegWordGraphBinaryFile(graphs.getNegWordGraphBinaryFile());
202             files.setPosNGramGraphBinaryFile(graphs.getPosNGramGraphBinaryFile());
203             files.setNegNGramGraphBinaryFile(graphs.getNegNGramGraphBinaryFile());
204             files.setPosGraphFilenames(graphs.getPosGraphFilenames());
205             files.setNegGraphFilenames(graphs.getNegGraphFilenames());
206             files.setMinPosRating(graphs.getMinPosRating());
207             files.setMaxPosRating(graphs.getMaxPosRating());
208             files.setMinNegRating(graphs.getMinNegRating());
209             files.setMaxNegRating(graphs.getMaxNegRating());
210
211             files.findTrainFilenames(false);
212             files.findTestFilenames(false);
213             long wordGraphs = System.currentTimeMillis();
214             files.wordGraphsFilesARFF();
215             long nGramGraphs = System.currentTimeMillis();
216             files.nGramGraphsFilesARFF();
217             long bagOfWords = System.currentTimeMillis();
218             files.bagOfWordsFilesARFF();
219             long end = System.currentTimeMillis();
220             NumberFormat formatter = new DecimalFormat("#0.00000");
221             FileWriter output = new FileWriter(new File(file), true);
222             output.write("Time for files of word graphs: ");
223             output.write(formatter.format((nGramGraphs - wordGraphs) / 1000d)
224                             + " seconds\n");
225             output.write("Time for files of n-gram graphs: ");
226             output.write(formatter.format((bagOfWords - nGramGraphs) / 1000d)
227                             + " seconds\n");
228             output.write("Time for files of bagOfWords: ");
229             output.write(formatter.format((end - bagOfWords) / 1000d)
230                             + " seconds\n");
231             output.close();
232         }
233
234         private void classifiersStage(DataFiles files, Classifiers classifiers)
235                 throws Exception {
```

```
                classifiers.setBagOfWordsTrainFile(files.getBagOfWordsTrainFile());
                classifiers.setBagOfWordsTestFile(files.getBagOfWordsTestFile());
                classifiers.setWordGraphsTrainFile(files.getWordGraphsTrainFile());
                classifiers.setWordGraphsTestFile(files.getWordGraphsTestFile());
                classifiers.setnGramGraphsTrainFile(files.getnGramGraphsTrainFile());
                classifiers.setnGramGraphsTestFile(files.getnGramGraphsTestFile());

                System.out.println("Creating classifiers for each pair of training"
                                + " and testing set...");
                long wordGraphs = System.currentTimeMillis();
                classifiers.createClassifierForWordGraphs();
                long nGramGraphs = System.currentTimeMillis();
                classifiers.createClassifierForNGramGraphs();
                long bagOfWords = System.currentTimeMillis();
                classifiers.createClassifierForBagOfWords();

                System.out.println("Evaluating each classifier...");
                long wordGraphs2 = System.currentTimeMillis();
                classifiers.evaluateClassifierForWordGraphs();
                long nGramGraphs2 = System.currentTimeMillis();
                classifiers.evaluateClassifierForNGramGraphs();
                long bagOfWords2 = System.currentTimeMillis();
                classifiers.evaluateClassifierForBagOfWords();
                long end = System.currentTimeMillis();
                NumberFormat formatter = new DecimalFormat("#0.00000");

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

                output.write("Time for classifier of word graphs: ");
                output.write(formatter.format((nGramGraphs - wordGraphs) / 1000d)
                                + " seconds\n");
                output.write("Time for classifier of n-gram graphs: ");
                output.write(formatter.format((bagOfWords - nGramGraphs) / 1000d)
                                + " seconds\n");
                output.write("Time for classifier of bagOfWords: ");
                output.write(formatter.format((wordGraphs2 - bagOfWords) / 1000d)
                                + " seconds\n");

                output.write("Time for evaluating classifier of word graphs: ");
                output.write(formatter.format((nGramGraphs2 - wordGraphs2) / 1000d)
                                + " seconds\n");
                output.write("Time for evaluating classifier of n-gram graphs: ");
                output.write(formatter.format((bagOfWords2 - nGramGraphs2) / 1000d)
                                + " seconds\n");
                output.write("Time for evaluating classifier of bagOfWords: ");
                output.write(formatter.format((end - bagOfWords2) / 1000d)
                                + " seconds\n");
                output.close();
        }

        public File getResultsDirectory() {
                return resultsDirectory;
        }

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

```
package sentimentanalysis;
import java.io.File;
import java.util.ArrayList;
import java.util.Random;


public class FilenamePattern {

        private String patternForMatching;

        public FilenamePattern(int minRating, int maxRating) {

                patternForMatching = "(^\\d{1,5}[_](";

                for (int rating = minRating; rating < maxRating + 1; rating++)
                        patternForMatching += rating +"|";

                patternForMatching = patternForMatching.substring(0,
                                patternForMatching.length()-1) + ").txt$)";
        }

        public ArrayList<String> findFilenames(String reviewFilepath,
                        int noOfReviews, boolean shuffle) {

                File filepath = new File(reviewFilepath);
                String[] allFiles = filepath.list();

                if (shuffle)
                        allFiles = shuffle(allFiles);

                ArrayList<String> toFillArray = new ArrayList<String>();
                int size = 0;
                for (String s: allFiles)
                        if (match(s)) {
                                toFillArray.add(s);
                                if (++size >= noOfReviews)
                                        break;
                        }
                return toFillArray;
        }

        public ArrayList<String> findFilenames(String reviewFilepath,
                        int noOfReviews, boolean shuffle, long randomSeed) {

                if (randomSeed == -1)
                        return findFilenames(reviewFilepath, noOfReviews, shuffle);

                File filepath = new File(reviewFilepath);
                String[] allFiles = filepath.list();

                if (shuffle)
                        allFiles = shuffle(allFiles, randomSeed);

                ArrayList<String> toFillArray = new ArrayList<String>();
                int size = 0;
                for (String s: allFiles)
                        if (match(s)) {
                                toFillArray.add(s);
                                if (++size >= noOfReviews)
                                        break;
```

```
                }
            return toFillArray;
        }

        public ArrayList<String> findFilenames(String filepath, int noOfReviews,
                ArrayList<String> notMatchingArray, boolean shuffle) {

            File reviewFilepath = new File(filepath);
            String[] allFiles = reviewFilepath.list();
            ArrayList<String> toFillArray = new ArrayList<String>();
            int size = 0;

            if (shuffle)
                allFiles = shuffle(allFiles);

            for (String s: allFiles)
                if ((!notMatchingArray.contains(s)) && match(s)) {
                    toFillArray.add(s);
                    if (++size >= noOfReviews)
                        break;
                }
            return toFillArray;
        }

        public ArrayList<String> findFilenames(String filepath, int noOfReviews,
                ArrayList<String> notMatchingArray, boolean shuffle,
                long randomSeed) {

            if (randomSeed == -1)
                return findFilenames(filepath, noOfReviews,
                        notMatchingArray, shuffle);

            File reviewFilepath = new File(filepath);
            String[] allFiles = reviewFilepath.list();
            ArrayList<String> toFillArray = new ArrayList<String>();
            int size = 0;

            if (shuffle)
                allFiles = shuffle(allFiles, randomSeed);

            for (String s: allFiles)
                if ((!notMatchingArray.contains(s)) && match(s)) {
                    toFillArray.add(s);
                    if (++size >= noOfReviews)
                        break;
                }
            return toFillArray;
        }

        private String[] shuffle(String[] filenames) {
            Random randomIndex = new Random();

            for (int i = 0; i < filenames.length; i++) {
                int randomPosition = randomIndex.nextInt(filenames.length);
                String temp = filenames[i];
                filenames[i] = filenames[randomPosition];
                filenames[randomPosition] = temp;
            }
            return filenames;
        }
```

```
121
122         private String[] shuffle(String[] filenames, long randomSeed) {
123                 Random randomIndex = new Random();
124                 randomIndex.setSeed(randomSeed);
125
126                 for (int i = 0; i < filenames.length; i++) {
127                     int randomPosition = randomIndex.nextInt(filenames.length);
128                     String temp = filenames[i];
129                     filenames[i] = filenames[randomPosition];
130                     filenames[randomPosition] = temp;
131                 }
132                 return filenames;
133         }
134
135         private boolean match(String reviewFilename) {
136                 if (reviewFilename.matches(patternForMatching))
137                         return true;
138                 else
139                         return false;
140         }
141
142         public String getPatternForMatching() {
143                 return patternForMatching;
144         }
145 }
```