ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΤΟΜΕΑΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ Η/Υ, ΠΛΗΡΟΦΟΡΙΚΗΣ & ΔΙΚΤΥΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σπουδάστρια:

Μπάρε Αντονέλα

Θέμα:

"Μελέτη και ανάπτυξη εφαρμογής αξιολόγισης κατηγοριοποιητών συναισθηματικής ανάλυσης (sentiment analysis classifiers) με βάση κειμενική πληροφορία"

Εισηγητής:

Γεώργιος Ν. Πρεζεράκος

Πειραιας | Μάιος 2014

Περίληψη

Το θέμα της παρούσας πτυχιακής εργασίας είναι η εξαγωγή του συναισθήματος απο κείμενα χρησιμοποιώντας τεχνικές μηχανικής μάθησης και κατηγοριοποίησης κειμενικής πληροφορίας καθώς και άλλων κειμένων για την εκπαίδευση αυτών. Στη συνέχεια, παρουσιάζεται η διαδικτυακή εφαρμογή που υλοποιήθηκε, η οποία επιτρέπει σε ανθρώπους να κανουν χειροκίνητη αξιολόγιση κειμένων ,ούτως ώστε να κριθούν τα πειράματα για την αξιοπιστία των αποτελεσματων που εξήγαγαν.

Abstract

The subject of this thesis is the extraction of sentiment in texts using machine learning techniques, classification of textual information and other texts for training them. Afterwards, a web application that utilizes these models is presented. This application was developed to allow people to do manual text evaluation in order to classify the experiments on the reliability of the results they exported.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Κατάλογος εικόνων

Κατάλογος Πινάκων

1 ΚΕΦΑΛΑΙΟ: Εισαγωγή

1.1 Πρόλογος

Ο άνθρωπος απο την φύση του πάντα ενδιαφερόταν για την εύρεση και την παρακολούθηση της κοινής γνώμης. Εξαιτίας του Διαδικτύου επιταχύνεται η ήδη αλματώδης αύξηση εγγράφων κειμένου που δημοσιεύονται καθημερινά κυρίως απο τα μέσα κοινωνικής δικτύωσης(twitter,facebook) και αλλα websites.

Η εύρεση της επιθυμητής πληροφορίας γίνεται δύσκολη και η ανάλυση αυτής ακόμα δυσκολότερη λογο του όγκου των κειμενικών δεδομένων. Παρόλα αυτά, δημοσιευμένα κείμενα στο διαδίκτυο περιέχουν επαρκή δεδομένα για να ικανοποιήσουν αυτή την έμφυτη ανάγκη του ανθρώπου να γνωρίζει τη γνώμη άλλων.

Οι τεχνικές εξόρυξης γνώσης από δεδομένα (data mining), η εξόρυξη κειμένων (text mining) και η εφαρμογή τεχνικών μηχανικής μάθησης (machine learning), που αποτελεί κλάδο του ευρύτερου πεδίου της τεχνητής νουμοσύνης, μας επιτρέπουν να επιτύχουμε την κατανόηση κειμένων.

Το αντικείμενο της παρούσης εργασίας εξετάζει την ανάλυση κειμένων ως προς το συναίσθημα που αντικατοπτρίζεται απο το περιεχόμενο (sentiment analysis) και παρουσιάζει ένα εργαλείο αξιολόγισης sentiment detection classifiers που χρησιμοποιούνται για τέτοια συναισθηματική ανάλυση κειμένου.

Οι τεχικές συναισθηματικής ανάλυσης ανήκουν σε τρεις κυρίως κλάδους ,αυτόν της τεχνητής νοημοσύνης και μηχανικής εκμάθησης, της υπολογιστικής γλωσσολογίας (computational linguistics) και της εξόρυξης κειμένου (text mining). Στην παρούσα πτυχιακή εργασία ο συγγραφέας χρησιμοποίησε text mining και machine learning και ανέπτυξε λογισμικό στη γλώσσα Ruby και για την εφαρμογή αξιολόγισης χρησιμοποίησε το web development framework Ruby On Rails 4.0.0 .

1.2 *Δομή της πτυχιακής*

Η διπλωματική εργασία είναι διαρθρωμένη σε έξι κεφάλαια. Το παρόν κεφάλαιο είναι *το πρώτο* και σε αυτό γίνεται αναφορά στο αντικείμενο της πτυχιακής και στον τρόπο οργάνωσης του υπόλοιπου κειμένου.

Στο δεύτερο κεφάλαιο, με τίτλο «Εξόρυξη Γνώμης – Σημασιολογική Ανάλυση», παρατίθεται το θεωρητικό υπόβαθρο των τεχνικών μηχανικής μάθησης που εφαρμόζονται στην εξόρυξη γνώσης από κειμενική πληροφορία .

Στο *τρίτο κεφάλαιο* επιχειρείται η καταγραφή των πειραμάτων που διεξήχθησαν απο τις βιβλιοθήκες που χρησιμοποίησε ο συγγραφέας με σκοπό την σύγκριση των αποδόσεων για να καταλήξει στην επολογή της βιβλιοθήκης με τα πιο ικανοποιητικά αποτελέσματα. Στη συνέχεια παρουσιάζεται η συλλογή κειμένων που χρησιμοποίησε ο συγγραφεας, τα προβλήματα που αντιμετώπισε και πως προσέγγισε τις λύσεις των προβλημάτων αυτών.

Στο *τέταρτο κεφάλαιο* περιγράφονται τα μερη απο τα οποία απαρτίζεται η διαδικτυακή εφαρμογή μας, τις αρχιτεκτονικές και τα είδη των τεχνολογιών που χρησιμοποιήθηκαν κατα την κατασκευή αυτής.

Στο πέμπτο κεφάλαιο εξηγούνται οι λειτουργίες της και πως συνεργάζονται για να παρέχουν στον χρήστη το τελικό περιβάλλον καθώς επίσης τα σχεδιαστικά προβλήματα και διλήμματα που αντιμετωπίστηκαν και την τελική λύση αυτών.

2 ΚΕΦΑΛΑΙΟ: Εξόρυξη Γνώμης – Σημασιολογική Ανάλυση

Η σημασιολογική προσέγγιση της θεματολογίας σε επίπεδο εξαγωγής συναισθημάτων, τα οποία εκφράζονται από τους συγγραφείς των κειμένων, είναι ένα σημαντικό πεδίο έρευνας. Στη μελέτη της σημασιολογικής – συναισθηματικής ανάλυσης των αναρτήσεων στα κοινωνικά δίκτυα, στα ιστολόγια και στον παγκόσμιο ιστό γενικότερα, έχουν αποδοθεί διάφορες ονοματολογίες. Ένας όρος που αρχικά χρησιμοποιήθηκε και έγινε αποδεκτός στη συνέχεια είναι η Εξόρυξη Γνώμης (Opinion Mining), χωρίς όμως η ερμηνεία του να δίνει έμφαση στο συναισθηματικό κομμάτι αλλά κυρίως στην αναζήτηση και στην ανάκτηση πληροφορίας στο διαδίκτυο, αποτελώντας ουσιαστικά υποκατηγορία της Εξόρυξης Δεδομένων (Data Mining).

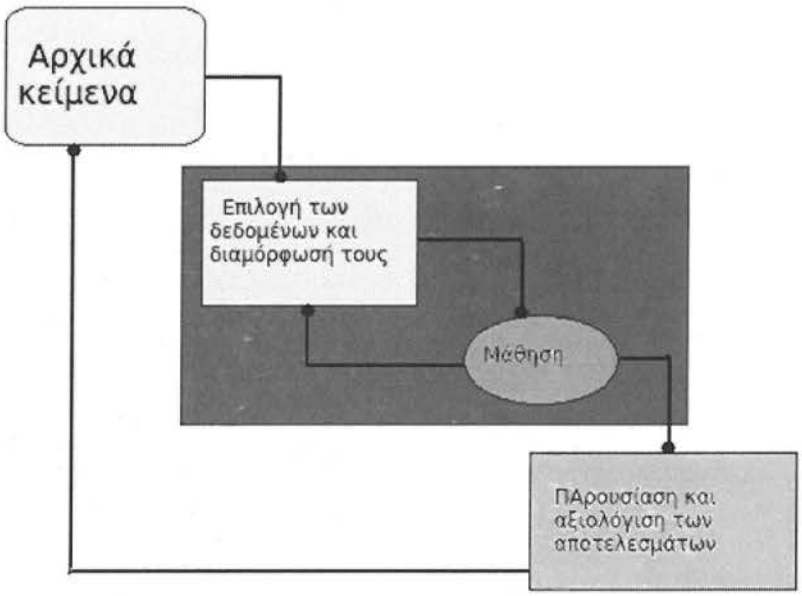
Η εξόρυξη γνώσης από κείμενα (Text Mining) αναφέρεται γενικά στη διαδικασία ανακάλυψης, με αυτοματοποιημένο τρόπο, πληροφοριών άγνωστων έως τότε από διαφορετικές γραπτές πηγές. Η διαδικασία της εξόρυξης περιλαμβάνει πάντα την υποδιαδικασία της προεπεξεργασίας του κειμένου (επιλογή και επεξεργασία γλωσσικών χαρακτηριστικών, εξάλειψη τυχών κειμενικών ιδιαιτεροτήτων και αποθήκευση των τελικών εξαχθέντων δεδομένων σε μια βάση δεδομένων), την επιλογή και εφαρμογή κάποιου μοντέλου επεξεργασίας (στατιστικού, μηχανικής μάθησης κ.τ.λ.) στα δομημένα πλέον στοιχεία του κειμένου και τελικά την αξιολόγηση και ερμηνεία των παραγόμενων αποτελεσμάτων.

Η εξόρυξη κειμένου στοχεύει στην εξαγωγή πληροφοριών από μεγάλο όγκο κειμένων οι οποίες μπορεί να φανούν χρήσιμες προς το χρήστη, μέσω της ανακάλυψης προτύπων απο τα μη δομημένα δεδομένα τους. Οι κυριότερες κατηγορίες των μεθόδων που χειρίζεται η εξόρυξη κειμένου είναι:

- Εξαγωγή χαρακτηριστικών γνωρισμάτων (Feature Extraction)
- Πλοήγηση βασισμένη στο κείμενο (Text Based Navigation)
- Περιληπτική Παρουσίαση της Πληροφορίας (Summarization)
- Κατηγοριοποίηση, κατάταξη με επίβλεψη (Categorization, Supervised Classification)
- Ομαδοποίηση, μη επιβλεπόμενη κατάταξη (Clustering, Unsupervised Classification)

- Γλωσσικός προσδιορισμός (Language Identification) και απόδοση κειμένου στο Συγγραφέα
- Συσχετίσεις (Associations)

Ο όρος text mining δεν πρέπει να συγχέεται με την αναζήτηση στον παγκόσμιο ιστό. Στην απλή αναζήτηση ο χρήστης αναζητά μια πληροφορία που είναι ήδη γνωστή και έχει γραφεί απο κάποιον αλλο. Το πρόβλημα που αντιμετωπίζει είναι να καθαρίσει ολη την πληροφορία που του εμφανίζεται από το υλικό που δεν είναι σχετικό με τις ανάγκες του. Στόχοι της εξόρυξης γνώσης είναι να ανακαλυφθεί άγνωστη μέχρι τότε πληροφορία, κάτι το οποίο κανείς δεν γνωρίζει και συνεπώς θα ήταν αδύνατο να έχει γραφεί. Η πληροφορία που θα εξαχθεί μπορεί να είναι η κατηγοριοποίηση κειμένων, η ομαδοποίηση κειμένων, η εξαγωγή απ' αυτά εννοιών η οντοτήτων, η ανάλυση συναισθήματος, η περιληπτική παρουσίαση της πληροφορίας του κειμένου, η ταξινόμιση των οντοτήτων του κειμένου και η εκμάθηση ενδιαφέροντων σχέσεων μεταξύ συγκεκριμένων οντοτήτων. Στην εικόνα που ακλουθεί παρουσιάζονται σχηματικά οι παραπάνω έννοιες:



Εικόνα 2.3.1-1: Ηδιαδικασία της εξόρυξης γνώσης απο κείμενα.

Η εξόρυξη πληροφορίας από κείμενα είναι ένας τομέας που αφορά την επιστήμη της ανάκτησης

πληροφοριών, της εξόρυξης δεδομένων, της μηχανικής μάθησης, τη στατιστική και την υπολογιστική γλωσσολογία. Δεδομένου ότι οι περισσότερες πληροφορίες (πάνω από 80%) αποθηκεύονται ως αδόμητο ή ημίδομημένο κείμενο, η εξόρυξη γνώσης από κείμενα φαίνεται να έχει υψηλή αξία, καθότι μπορεί να αποτελέσει χρήσιμο εργαλείο σε πολλές εμπορικές εφαρμογές. Πηγές κειμένων μπορούν να αποτελέσουν το διαδίκτυο, το ηλεκτρονικό ταχυδρομείο, οι ομάδες συζητήσεων (discussion groups και forums), οι ηλεκτρονικές βιβλιοθήκες, καθώς και αρχεία κειμένου.

Συμπληρωματικά αυτού του όρου, χρησιμοποιείται και ο όρος της Συναισθηματικής Ανάλυσης ή Εντοπισμού Συναισθήματος (Sentiment Analysis, Sentiment detection) προσδιορίζοντας καλύτερα την αξία της συλλογής και της ανάλυσης των συναισθηματικών μεταδεδομένων των αναρτήσεων. Στην εξόρυξη κειμένου είναι αρκετά συνήθης η προσπάθεια εύρεσης νοηματικής ομοιότητας των αρχείων κειμένου με κάποια θεματική περιοχή (πχ στη διάρκεια της κατηγοριοποίησης) είτε και των αρχείων μεταξύ τους (όπως στην ομαδοποίηση αρχείων κειμένου).

Έτσι λοιπόν, με το συνδυασμό των δύο μεθόδων (Εξόρυξη Γνώμης & Ανάλυση Συναισθήματος) μπορούν να προσδιοριστούν τόσο το συναίσθημα, όσο και οι αλληλεπιδράσεις, η υποκειμενικότητα, η αντικειμενικότητα αλλά και άλλες συναισθηματικές εκφάνσεις των χρηστών που δημιουργούν τις αναρτήσεις.

Ακολουθεί μία σειρά παραδειγματικών ερωτημάτων, στα οποία εξόρυξη γνώσης και η ανάλυση συναισθήματος καλούνται να προσφέρουν χρηστικές απαντήσεις:

- Ποιές είναι οι απόψεις των νέων ψηφοφόρων προς τους δημοκρατικούς και ρεπουμπλικάνους προεδρικούς υποψηφίους κατά τη διάρκεια της πιο πρόσφατης εκλογής;

- Ποιες είναι οι απόψεις και τα σχόλια των επενδυτών, των υπαλλήλων και των στελεχών για τις επιχειρηματικές πρακτικές των εταιριών τους;

- Ποιές είναι οι απόψεις του κοινού για διάσημους καλλιτέχνες που βρίσκονται στο προσκήνιο την παρούσα χρονική περίοδο;

Όπως προκύπτει από τη διαφορετικότητα των ερωτημάτων, η αξία της άντλησης σημασιολογικής και συναισθηματικής πληροφορίας από τις διαθέσιμες αναρτήσεις των κοινωνικών δικτύων βρίσκει,

όπως συζητήθηκε και στην εισαγωγή της παρούσας μελέτης, ευρύ πεδίο εφαρμογών. Η αποτύπωση της σημασιολογικής και συναισθηματικής πληροφορίας αποτελεί σημαντική πηγή άντλησης δεδομένων για τη χάραξη πολιτικών, κοινωνικών, οικονομικών, διοικητικών, καλλιτεχνικών και άλλων πολικών.

2.1 *Θεμελιώδεις έννοιες και Ορισμοί*

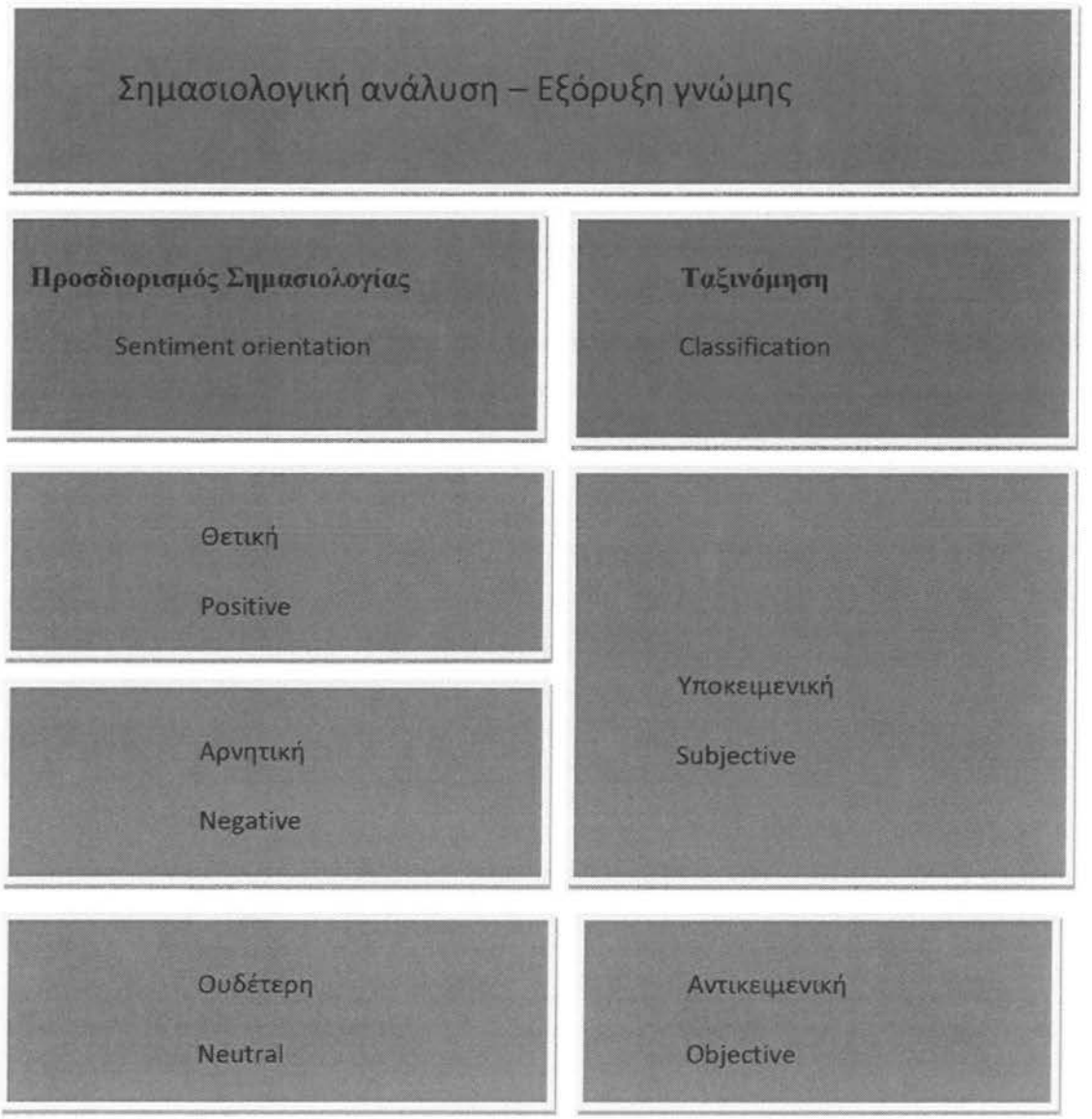
Λόγω της δυσκολίας κατανόησης των ερμηνιών και των ορισμών που προκύπτουν σε κάθε τομέα, αφιερώνουμε το παρών κεφάλαιο για να αναφέρουμε τους βασικότερους από αυτούς, που εξυπηρετούν στην κατανόηση των αρχών του πεδίου της σημασιολογίας κειμένων.

Η σημασιολογική ανάλυση (text mining) είναι ο συνδυασμός της δημιουργίας αυτοματοποιημένων συστημάτων, ικανών να αξιολογούν την ανθρώπινη γνώμη σ' ένα κείμενο που είναι γραμμένο σε φυσική γλώσσα και της μελέτης της άποψης, της σημασιολογίας και του συναισθήματος του κειμένου. Στοχεύει στο προσδιορισμό ή το προσανατολισμό της άποψης που εκφράζεται μέσα απο ένα κείμενο, μια διαδικασία που περικλείει ως επιμέρους στόχο και τη διάκριση μεταξύ υποκειμενικής και αντικειμενικής άποψης [30].

Η ταξινόμηση υποκειμενικότητας ταξινομεί τις προτάσεις σε δύο κλάσεις, υποκειμενικές και αντικειμενικές. Αντικειμενικές ονομάζονται οι προτάσεις οι οποίες μεταφέρουν πληροφορίες με τρόπο αμερόληπτο, ενώ υποκειμενικές αυτές οι οποίες εκφράζουν προσωπικές απόψεις [29]. Όσον αφορά στην κατηγοριοποίηση κάποιοι ερευνητές την κάνουν σ' επίπεδο εγγράφου (Document-level), ενώ κάποιοι άλλοι σ' επίπεδο πρότασης (Sentence level) κατηγοριοποιώντας κάθε πρόταση σαν μία υποκειμενική ή αντικειμενική πρόταση η οποία εκφράζει μία θετική ή αρνητική γνώμη. Επειδή η έκφραση αντικειμενικών γεγονότων μπορεί να αποπροσανατολίσει τον ταξινομητή, είναι σημαντικό να διακρίνουμε τις υποκειμενικές από τις αντικειμενικές προτάσεις.

Στην εργασία [11] αφού πραγματοποιείται κατηγοριοποίηση υποκειμενικότητας με 4 κλάσεις,

αποδεικύεται ότι μια υποκειμενική πρόταση μπορεί να μην είναι αξιολογήσιμη (απουσία θετικού ή αρνητικού συναίσθηματος) και ότι μια αντικειμενική πρόταση μπορεί να εκφράζει συναίσθημα. Αν μια πρόταση κατηγοριοποιηθεί ως υποκειμενική, ακολουθεί ο χαρακτηρισμός της ως θετική ή αρνητική. Για να συμβεί αυτό, εφαρμόζονται τεχνικές επιβλεπόμενης μάθησης όπως ακριβώς και στην ταξινόμηση συναισθήματος εγγράφων καθώς και τεχνικές οι οποίες βασίζονται σε λεξικά συναισθήματος.



Εικόνα 2.3.1-1: Κατηγορίες σημασιολογικού προσδιορισμού και ταξινόμησης

Στη βιβλιογραφία συναντούμε συχνά τους συνώνυμους όρους «Συναισθηματική Κατηγοριοποίηση» (Sentiment Classification) και «Εξαγωγή Γνώμης» (Opinion Extraction).

Ορισμός 1: *Κάτοχος άποψης* (opinion holder): Ο κάτοχος μιας άποψης είναι αυτός που την εκφράζει και μπορεί να είναι ένα υποκείμενο, ένας οργανισμός, ένα σύνολο κ.α.

Ορισμός 2: *Προσδιορισμός Άποψης* (Opinion orientation): Ο προσδιορισμός της άποψης, υποδεικνύει εάν η άποψη είναι θετική, αρνητική ή ουδέτερη.

Ορισμός 3: *Συναισθήματα* (emotions): Τα συναισθήματα είναι οι αισθήσεις και οι σκέψεις που εκφράζει ένα υποκειμένο.

Τα συναισθήματα είναι ένα ιδιαίτερο πεδίο έρευνας με το οποίο έχουν ασχοληθεί πολλοί επιστημονικοί τομείς, όπως η ψυχολογία, η φιλοσοφία, η κοινωνιολογία, η βιολογία κ.α. Παρόλα αυτά δεν υπάρχει ακόμα μια βάση αναφοράς των συναισθηματικών κατηγοριών η οποία να είναι κοινά αποδεκτή. Βασιζόμενοι στις εργασίες [28, 27, 26, 25] διακρίνουμε έξι κύριες κατηγορίες συναισθημάτων: αγάπη (love), χαρά (joy), έκπληξη (surprise), θυμό (anger), λύπη (fear) και φόβο (fear). Η καθεμία απο τις παραπάνω κατηγορίες μπορεί να εμπεριέχει και άλλες υποκατηγορίες κατηγοριοποιώντας εις περαιτέρω βάθος την ένταση των συναισθημάτων.

Ορισμός 4: *Αντικειμενική - Υποκειμενική πρόταση* (sentence objectivity – subjectivity): Μια αντικειμενική πρόταση εκφράζει κάποια πραγματική πληροφορία για μια θεματολογία, ενώ μια υποκειμενική εκφράζει απόψεις, πεποιθήσεις ακόμα και συναισθήματα.

Ορισμός 5: *Πρόταση με άποψη* (opinioned sentence) ονομάζεται μια πρόταση που εκφράζει άμεση ή έμμεση, θετική ή αρνητική άποψη. Η πρόταση αυτή μπορεί να είναι είτε υποκειμενική είτε αντικειμενική.

Οι υποκειμενικές προτάσεις έχουν διάφορες μορφές όπως για παράδειγμα ισχυρισμούς, επιθυμίες, υποθέσεις, υποψίες, όμως υπάρχει και η περίπτωση να μην περιέχουν καμία άποψη. Το ίδιο παρατηρείται και σε ορισμένες αντικειμενικές προτάσεις. Με τους ορισμούς 4 & 5 γίνεται σαφής ο διαχωρισμός μεταξύ μιας υποκειμενικής πρότασης και μιας πρότασης με άποψη. Οι προτάσεις με άποψη αποτελούν υποσύνολο των υποκειμενικών προτάσεων. Η τεχνική αναγνώρισης και ταξινόμησης των προτάσεων με άποψη ονομάζεται «Υποκειμενική Ταξινόμηση» (Subjectivity Classification) [30].

Η εργασία [20] αναφέρει οτι η υποκειμενική ταξινόμιση ενός κειμένου διαιρείται σε τρια πεδία που αλληλοεξαρτώνται.

- στον προσδιορισμό της υποκειμενικότητας (subjectivity) που ερευνά εάν σε ένα κείμενο, εκφράζεται ή όχι, θετική ή αρνητική άποψη σχετικά με ένα θέμα,

- στον προσδιορισμό του προσανατολισμού (orientation or polarity), που εξετάζει εάν σε ένα υποκειμενικό κείμενο εκφράζεται θετική ή αρνητική άποψη

- στον προσδιορισμό του σθένους του προσανατολισμού (strength of orientation), που εξετάζει αν η θετική ή αρνητική άποψη που εκφράζεται σε ένα κείμενο είναι κλιμακωτή (ασθενής, μερικώς ή έντονη).

2.2 *Προηγούμενες εργασίες*

Οι ερευνητικές προσεγγίσεις που έχουν γίνει στο πρόβλημα της ταξινόμησης συναισθήματος κατηγοριοποιούνται ανάλογα με το επίπεδο εφαρμογής τους. Μια από αυτές είναι η ταξινόμηση σε επίπεδο εγγράφου. Αυτή η προσέγγιση θεωρεί ότι κάθε έγγραφο περιέχει τις απόψεις ενός μόνο ατόμου γύρω από ένα συγκεκριμένο θέμα και έχει ως στόχο να χαρακτηρίσει το συναίσθημα που

εκφράζεται μέσα από το κείμενο ως θετικό, αρνητικό η ουδέτερο.

Οι περισσότερες εργασίες μελετούν και προτείνουν μεθόδους για τον προσδιορισμό του προσανατολισμού ενός κειμένου δηλαδή την εύρεση των προτάσεων που περιέχουν άποψη για ένα θέμα είτε θετική είτε αρνητική [24, 23, 22, 21, 19, 18, 16, 14, 13, 15, 17]. Πέρα από τις προκαθορισμένες μεθόδους μηχανικής μάθησης, οι ερευνητές έχουν προτείνει αρκετές τεχνικές ειδικά προσαρμοσμένες στην επίλυση του προβλήματος της κατηγοριοποίησης συναισθήματος. Η κατηγοριοποίηση συναισθήματος είναι ουσιαστικά ένα πρόβλημα ταξινόμησης κειμένου. Για την επίλυση του προβλήματος οποιαδήποτε υπάρχουσα μέθοδος επιβλεπόμενης μάθησης (Ταξινόμηση naïve Bayes, Μηχανισμοι Διανύσματος υποστήριξης (Support Vector Machines – SVM)) μπορεί να εφαρμοστεί [12, 14,29]. Οι B.Pang, L.Lee, και S.Vaithyanathan [24] ήταν οι πρώτοι που βασίστηκαν στην παραπάνω προσέγγιση και διαχώρισαν κριτικές ταινιών σε θετικές και αρνητικές. Έδειξαν ότι η χρήση μονογραμμάτων (unigrams) ως γνωρισμάτων για την κατηγοριοποίηση έχει αρκετά καλά αποτελέσματα είτε με τον naïve Bayes είτε με τα SVM, ενώ παράλληλα δοκίμασαν και εναλλακτικές επιλογές χαρακτηριστικών. Στην εργασία [1] επιτυγχάνεται συναισθηματική ανάλυση σε δεδομένα από σχόλια πελατών εκπαιδεύοντας γραμμικά SVM με μεγάλα διανύσματα χαρακτηριστικών (large feature vectors). Στη συνέχεια οι Pang, B. και Lee, L. [2] δημιούργησαν ένα γράφο μεταξύ των προτάσεων ενός κειμένου και εφάρμοσαν τον αλγόριθμο ελαχίστων κοψιμάτων (minimum cut), αφαιρώντας τις ακμές με το μικρότερο υποκειμενικό φορτίο, ώστε να διευκολύνουν την διαδικασία της κατηγοριοποίησης, η οποία έγινε με συνήθεις τεχνικές μηχανικής μάθησης.

Οι S.Matsumoto, H.Takamura και M.Okumura [17] χρησιμοποίησαν τις συντακτικές σχέσεις που υπήρχαν μεταξύ των λέξεων ως χαρακτηριστικά του SVM ενώ οι Kennedy και Inkpen [18] εξέτασαν την επίδραση των λέξεων που μπορούν να αλλάξουν το συναισθηματικό φορτίο στην ταξινόμιση κριτικών από ταινίες. Στην εργασία [19] εξέταστηκε μια μέθοδος ημιεπιβλεπόμενης μάθησης, η οποία πρώτα εξήγαγε τις ευδιάκριτες κριτικές των πελατών με χρήση φασματικών τεχνικών και στη συνέχεια τις χρησιμοποιούσε για να κατηγοριοποιήσει τις ασαφείς κριτικές συνδυάζοντας μεθόδους μάθησης που είτε αξιοποιούν την ανάδραση του χρήστη (ενεργητικές - active), είτε αποφασίζουν για τα άγνωστα δείγματα χωρίς να δημιουργήσουν ενδιάμεσο μοντέλο (μεταβιβαστικές - transductive) είτε συνδυάζουν τα αποτελέσματα επιμέρους μεθόδων (συνδυαστικές - ensemble). Το μοντέλο που

προτάθηκε από την έρευνα [20] περιλαμβάνει αρχικά μια επαναληπτική διαδικασία κατηγοριοποίησης κριτικών σύμφωνα με ένα λεξικό συναισθημάτων και σε δεύτερη φάση την εκπαίδευση ενός ταξινομητή επιβλεπόμενης μάθησης με κάποια από τα δεδομένα της πρώτης φάσης. Στη έρευνα [22] χρησιμοποιήθηκαν πολυεπίπεδα δομημένα μοντέλα για την εξαγωγή των χρήσιμων (π.χ. υποκειμενικών) προτάσεων ενός κειμένου καθώς και την πρόβλεψη του συναισθήματος αυτού, βάσει των προτάσεων που έχουν εξαχθεί.

Αντίθετα, ο Turney [31] πρότεινε μια μέθοδο μη επιβλεπόμενης μάθησης η οποία βασίζεται σε σταθερά συντακτικά μοντέλα, τα οποία είναι πιθανό να χρησιμοποιηθούν για την έκφραση απόψεων. Η κατηγοριοποίηση συναισθήματος γίνεται σύμφωνα με τον μέσο σημασιολογικό προσανατολισμό των φράσεων που περιέχουν επίθετα ή επιρρήματα. Η μέθοδος [32] χρησιμοποιεί λεξικά με λέξεις και φράσεις στις οποίες έχει προσημειωθεί η σημασιολογική πολικότητα (semantic polarity) και ισχύς (strength) και υπολογίζει μια βαθμολογία συναισθήματος για κάθε έγγραφο.

Αξίζει να αναφερθεί και η ταξινόμηση σε επίπεδο πρότασης. Σε αυτή την προσέγγιση υπάρχει η παραδοχή ότι μόνο μια άποψη μπορεί να υπάρξει μέσα σε κάθε πρόταση. Αυτό το πρόβλημα κατηγοριοποίησης συναισθήματος, μπορεί να αντιμετωπιστεί είτε ως ένα πρόβλημα ταξινόμησης τριών κλάσεων είτε ως δύο ξεχωριστά προβλήματα ταξινόμησης. Στην πρώτη περίπτωση, οι προτάσεις ταξινομούνται ως θετικές, αρνητικές ή ουδέτερες. Στην δεύτερη οι προτάσεις διαχωρίζονται σύμφωνα με το αν εκφράζουν κάποια άποψη ή όχι (ταξινόμηση υποκειμενικότητας) και στη συνέχεια όσες περιέχουν κάποια στοιχεία υποκειμενικής πληροφορίας ταξινομούνται ως θετικές ή αρνητικές. Στην ταξινόμηση υποκειμενικότητας οι προτάσεις κατατάσσονται σε δύο κλάσεις, υποκειμενικές και αντικειμενικές. Αντικειμενικές ονομάζονται οι προτάσεις οι οποίες μεταφέρουν πληροφορίες με χωρίς τη συναισθηματική φόρτιση του συγγραφέα, ενώ υποκειμενικές αυτές οι οποίες διατυπώνουν προσωπικές απόψεις [29]. Αν μια πρόταση κατηγοριοποιηθεί ως υποκειμενική, ακολουθεί ο χαρακτηρισμός της ως θετική ή αρνητική. Για να συμβεί αυτό, εφαρμόζονται τεχνικές επιβλεπόμενης μάθησης όπως ακριβώς και στην ταξινόμηση συναισθήματος εγγράφων καθώς και τεχνικές οι οποίες βασίζονται σε λεξικά συναισθήματος.

Οι Yu και Hatzivassiloglou [33] πραγματοποίησαν ταξινομήσεις υποκειμενικότητας χρησιμοποιώντας ένα ταξινομητή naïve Bayes και την ομοιότητα των προτάσεων. Η ομοιότητα των προτάσεων υπολογίζονταν από το σύστημα SIMFINDER [34], βάσει των κοινών τους λέξεων και φράσεων καθώς και των συνωνύμων του Word-Net. Οι Raaijmakers και Kraaij [35] έδειξαν ότι τα ν-γράμματα χαρακτήρων (μέρη λέξεων) είχαν καλύτερη απόδοση στην κατηγοριοποίηση υποκειμενικότητας σε σχέση με τα ν-γράμματα λέξεων και φωνημάτων. Τέλος, στην μελέτη [11] αφού πραγματοποιήθηκε κατηγοριοποίηση υποκειμενικότητας με 4 κλάσεις, αποδείχτηκε ότι μια υποκειμενική πρόταση μπορεί να μην είναι αξιολογήσιμη (απουσία συναισθήματος) και ότι μια αντικειμενική πρόταση μπορεί να εκφράζει συναίσθημα.

Σε κάποιες εργασίες με χρήση στατιστικών μεθόδων, επεκτείνουν περαιτέρω την υποκειμενικότητα ενός κειμένου και ταξινομούν τις απόψεις με βάση κάποια συναισθήματα, όπως στην εργασία [27] η οποία αναγνωρίζει τα έξι συναισθήματα (anger, disgust, fear, joy, sadness, surprise) σε σύνολα δεδομένων που προέρχονται από τίτλους ειδήσεων. Αντίστοιχα, στην εργασία [32] προσδιορίζουν την υποκειμενικότητα συζητήσεων από blogs, με βάση 8 συναισθηματικούς άξονες (acceptance, fear, anger, joy, anticipation, sadness, disgust, surprise).

Η τελευταία προσέγγιση είναι η ταξινόμηση σε επίπεδο λέξης. Η προσέγγιση αυτή ουσιαστικά χρησιμοποιείται για ταξινόμηση επιπέδου πρότασης ή κειμένου και βασίζεται στην παραδοχή ότι οι πιο σημαντικοί δείκτες συναισθημάτων είναι οι λέξεις γνώμης. Μια λίστα από τέτοιες λέξεις ονομάζεται λεξικό συναισθημάτων [29].

Για την δημιουργία λεξικών συναισθημάτων χρησιμοποιούνται πληροφορίες που προκύπτουν από την επεξεργασία είτε μεγάλων σωμάτων κειμένου(text corpora), είτε γλωσσολογικών πόρων όπως θησαυροί και λεξικά, με σκοπό την επέκταση μιας αρχικής λίστας με λέξεις γνώμης (seed words).

Στα λεξικά που προέρχονται από σώματα κειμένου, η επέκταση της λίστας αυτής, μπορεί να γίνει με χρήση συντακτικών μοτίβων, τα οποία ικανοποιούνται μέσα σε αυτά τα κείμενα. Ένας άλλος τρόπος

είναι με τη χρήση πληροφοριών που προκύπτουν από τη συχνότητα διάφορων μοτίβων από λέξεις [31].

Αντίθετα, τα λεξικά που βασίζονται σε γλωσσολογικούς πόρους προσπαθούν να πραγματοποιήσουν αυτή την επέκταση χρησιμοποιώντας τα συνώνυμα, τα αντώνυμα και την ιεραρχία αυτών των λέξεων μέσα σε γλωσσολογικούς θησαυρούς όπως το WordNet. Έτσι οι Kim και Hovy [36] χρησιμοποίησαν τις σχέσεις συνωνύμων και αντωνύμων του WordNet προκειμένου να επεκτείνουν το αρχικό σύνολο υποκειμενικών λέξεων. Αντίστοιχα, οι Hu και Liu [37] χρησιμοποίησαν τις ίδιες σχέσεις και το θησαυρό WordNet για ένα αρχικό σύνολο 30 επιθέτων, ενώ αργότερα οι Esuli και Sebastiani [38] βρήκαν το συναισθηματικό φορτίο για κάθε διαφορετική έννοια μιας λέξης αξιοποιώντας την ερμηνεία (gloss) της λέξης όπως αυτή δίνεται από το WordNet και ένα αρχικό σύνολο από υποκειμενικές λέξεις.

Χαρακτηριστικά παραδείγματα λεξικών συναισθημάτων είναι το Harvard General Inquirer, το Linguistic Inquiry and Word Counts (LIWC), το Bing Liu's Opinion Lexicon, το MPQA Subjectivity Lexicon και το SentiWordNet. Το Harvard General Inquirer είναι ένα λεξικό που επισυνάπτει συντακτικές, σημασιολογικές και πραγματικές πληροφορίες σε λέξεις με επισημείωση σχετικά με το μέρος του λόγου στο οποίο ανήκουν (π.χ. ουσιαστικό, ρήμα κτλ.). Το LIWC είναι μια εμπορικά διαθέσιμη βάση δεδομένων που περιέχει ένα μεγάλο αριθμό κατηγοριοποιημένων κανονικών εκφράσεων. Το Bing Liu's Opinion Lexicon αποτελείται από 6789 λέξεις οι οποίες είναι χωρισμένες σε θετικές και αρνητικές και βασίζεται στην προσέγγιση των Hu και Liu [37] όπως αυτή περιγράφηκε στην προηγούμενη παράγραφο. Το MPQA Subjectivity Lexicon επεκτείνει μια λίστα από στοιχεία (λέξεις) υποκειμενικότητας (subjectivity clues) των Riloff και Wiebe [7] με χρήση γλωσσολογικών πόρων, οι οποίες αφού πρώτα ομαδοποιηθούν ανάλογα με την αξιοπιστία (reliability) τους, χαρακτηρίζονται ως θετικές, αρνητικές, ουδέτερες ή ως θετικές και αρνητικές ταυτό- χρονα. Τέλος, το SentiWordNet, το οποίο στηρίζεται όπως είναι προφανές στο WordNet, διαχωρίζει τη συναισθηματική πολικότητα κάθε έννοιας μιας λέξης σε 3 κατηγορίες: θετική, αρνητική και ουδέτερη, σύμφωνα με την προαναφερθείσα μέθοδο των Esuli και Sebastiani [38]

2.3 Μεθοδολογικές Προσεγγίσεις

Αναφορικά με την μελέτη της αποτύπωσης συναισθήματος θα χρησιμοποιήσουμε δύο κύριες πρακτικές:

1. Χρήση Λεξικών
2. Μηχανική Μάθηση

2.3.1 Χρήση Λεξικών

Η πρώτη έχει έως βάση την χρήση λεξικών (Lexicon – based) στα οποία υπάρχουν εισηγμένες λέξεις, διαχωρισμένες ανάλογα με το μέρος του λόγου στο οποίο ανήκουν και αποδίδονται σε αυτές βάρη ανάλογα με την συναισθηματική ένταση των λέξεων.

Με αυτόν τον τρόπο μπορεί να αποδοθεί η υποκειμενική στάση των χρηστών απέναντι στη θεματολογία που περιέχεται στο σύνολο των δεδομένων. Ο τρόπος χρήσης των λεξικών, τόσο αναφορικά με την κλίμακα απόδοσης βαρών και τον χειρισμό αυτών, όσο και με την ολιστική ή με βάση συγκεκριμένα κριτήρια χρήση τους, διαφέρει ανάμεσα σε διαφορετικές μελέτες.

Έτσι μπορούμε να κατηγοριοποιήσουμε τον τρόπο χρήσης λεξικών ανάλογα με το αν:

- Κάνουν χρήση των λεξικών ολιστικά, αποδίδουν τη σημασιολογία και τη συναισθηματική στάση των χρηστών, απέναντι δηλαδή σε ολο το σύνολο δεδομένων (document level analysis),

ή

- Κάνουν χρήση των λεξικών τμηματικά, σε συγκεκριμένα τμήματα του συνόλου δεδομένων τα οποία διαχωρίζονται κατά την ανάλυση τους, έτσι επιτυχγάνεται καλύτερη επικεντρώση σε συγκεκριμένες θεματολογίες (sentence level analysis).

Στην πρώτη κατηγορία για να εξαχθεί το συναίσθημα σε ολόκληρο το σύνολο δεδομένων λαμβάνεται υπόψη η εγγύτητα των επιθέτων που περιλαμβάνονται στο προς μελέτη κείμενο με τα βασικά ουσιαστικά που χαρακτηρίζουν τη θεματολογία, και υπολογίζεται με τη βοήθεια αλγορίθμου η θετική ή αρνητική στάση απέναντι στη θεματολογία.

Στην δεύτερη κατηγορία, όπως προαναφέρθηκε, εντάσσονται οι μελέτες που διαχωρίζουν το περιεχόμενο και μελετούν τμηματικά την αποτύπωση του συναισθήματος στο σύνολο των δεδομένων. Υιοθετώντας αυτήν τη λογική τα λεξικά χρησιμοποιούνται διαφορετικά προσθέτοντας ή αφαιρώντας μεταβλητές. Χαρακτηριστικό παράδειγμα αποτελεί η μελέτη των Xiaowen Ding, Bing Liu, Philip S. Yu [3] οι οποίοι προσθέτοντας το συμπλεκτικό σύνδεσμο «και» προσπαθούν να απομονώσουν τη σημασιολογία των αποδιδόμενων χαρακτηριστικών σε συγκεκριμένα θέματα, τα οποία έχουν επιλέξει, από το σύνολο των δεδομένων. Αξίζει να αναφερθεί και το γεγονός ότι εφαρμόζουν τη μεθοδολογία τους τόσο σε επίπεδο κειμένου όσο και σε επίπεδο πρότασης.

Οι διαφορές που παρουσιάζουν οι δύο μεθοδολογίες, που βασίζονται στη χρήση των λεξικών, δεν έχουν να κάνουν με την επιλογή της μεθόδου, αλλά με το τον τρόπο χειρισμού των συνόλων των δεδομένων (data set).

Μια αλλη κατηγορία στην οποία μπορούν να ενταχθεί η τεχνική χρήσης λεξικών είναι και ο χειρισμός των βαρών – τιμών που αποδίδονται στις λέξεις.
Διακρίνουμε τις εξείς δυο κατηγορίες:

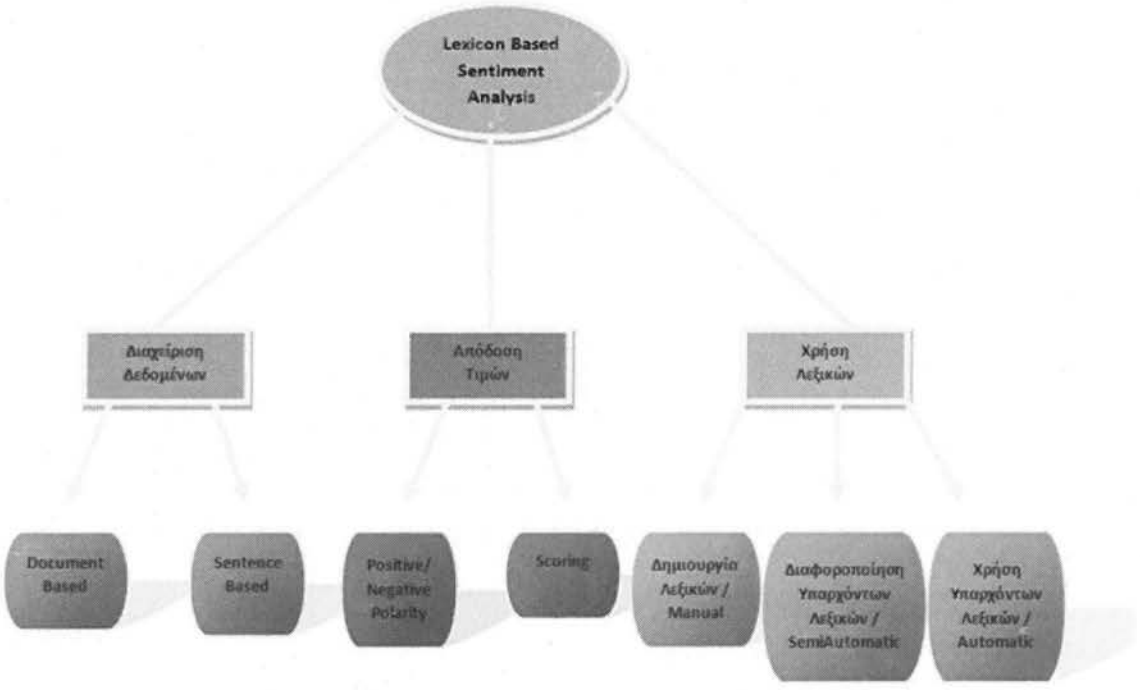
1. Η πρώτη θεωρεί τις λέξεις θετικές ή αρνητικές και τις κατηγοριοποιεί ανάλογα. Για παράδειγμα η λέξη "great" είναι μία θετική λέξη, ενώ η λέξη "ugly" είναι μία αρνητική λέξη. Έτσι αποδίδεται στις λέξεις πολικότητα (polarity), και εκφράζει την θετική ή αρνητική υποκειμενική άποψη , που αποτυπώνεται στο σύνολο δεδομένων.

2. Η δεύτερη αποδίδει τόσο θετικό όσο και αρνητικό βάρος στις λέξεις. Για παράδειγμα η λέξη "pretty" είναι 0,8 positive και 0,2 negative, ενώ η λέξη "bad" είναι 1,0 negative και 0 positive. Ο αναμενόμενος χειρισμός σε αυτήν την περίπτωση μπορεί να πραγματοποιηθεί με δύο τουλάχιστον τρόπους:

a. να οριοθετηθεί μία κλίμακα βαθμονόμησης σε κανονικοποιημένο διάστημα, για παράδειγμα [-1,1], όπου μπορούν να αποδοθούν χαρακτηρισμοί [-1, -0.5] strong negative, [- 0.49, 0] negative, [0.1, 0.5] positive και [0.51, 1] strong positive.

b. να υπολογιστεί η θετική ή αρνητική πολικότητα, τόσο σε document όσο και σε sentence level, προσθέτοντας ή αφαιρώντας το maximum score της κάθε λέξης.

Σημαντική παράμετρος που αφορά αυτές τις δύο κατηγοριοποιήσεις είναι ο τρόπος χρήσης των λεξικών που μπορεί να ποικίλει από την «χειροκίνητη» (manual) χρήση των λεξικών, τις ημιαυτόματες προσεγγίσεις και τη σχεδόν αυτοματοποιημένη διαχείριση των υπαρχόντων λεξικών . Συνοπτικά, στο παρακάτω σχήμα παρουσιάζονται οι κατηγοριοποιήσεις που αφορούν στη χρήση των λεξικών για τη σημασιολογική – συναισθηματική προσέγγιση, όπως αναλύθηκαν παραπάνω.



Εικόνα 2.3.1-1: Οι κατηγοριοποιήσεις που αφορούν στη χρήση των λεξικών για τη σημασιολογική – συναισθηματική προσέγγιση

2.3.2 Πλεονεκτήματα - Μειονεκτήματα

Ένα σημαντικό πλεονέκτημα των τεχνικών που χρησιμοποιούν λεξικά απόψεων είναι ότι δεν χρειάζονται σύνολα εκπαίδευσης για να κάνουν προβλέψεις, αφού στηρίζονται εξ' ολοκλήρου σε λεξικά που περιέχουν ένα προκαθορισμένο σύνολο λέξεων με άποψη. Τέτοιες τεχνικές χαρακτηρίζονται ως *μη επιβλεπόμενες τεχνικές μάθησης* και συνήθως, αλλά όχι αποκλειστικά, χρησιμοποιούνται σε περιπτώσεις όπου δεν έχουν χτιστεί ακόμα σύνολα εκπαίδευσης [4].

Οι συλλογές (corpus) που χρησιμοποιούνται για το «χτίσιμο» ενός λεξικού είναι πολύ σημαντικές για την ακρίβεια των αποτελεσμάτων της μεθόδου. Παρατηρείται όμως το φαινόμενο, λεξικά που έχουν χρησιμοποιήσει βασικές λέξεις ενός πεδίου (domain specific) να μην έχουν εξίσου αξιόλογα αποτελέσματα όταν χρησιμοποιούνται σε διαφορετικό πεδίο εφαρμογής. Για να μπορέσει ένα λεξικό να είναι ολοκληρωμένο και να περιέχει την πλειονότητα των λέξεων, απαιτούνται πολύ μεγάλες συλλογές δεδομένων.

Συχνά οι ερευνητές υποστηρίζουν ότι ο σημασιολογικός προσδιορισμός που προέρχεται από κάποιο λεξικό εχει να κανει με πιθανοτητες (probabilistic)[30]. Η σημασιολογία μιας πρότασης δεν εξαρτάται μόνο από την κατηγοριοποίηση των λέξεων ή/και των συνωνύμων τους σε θετικές ή αρνητικές. Η πρόταση «I am pretty tired» δεν μπορεί να ταξινομηθεί ως θετική, μόνο και μόνο επειδή το επίθετο «pretty» έχει κατηγοριοποιηθεί στο λεξικό ως θετική λέξη, η σημασιολογία των λέξεων και των προτάσεων εξαρτάται και απο άλλους παράγοντες, όπως οι σύνδεσμοι και η διαφορετικότητα χρήσης της γλώσσας. Σε κάθε περίπτωση τα λεξικά απόψεων αποτελούν το θεμέλιο λίθο πολλών μελετών και σχετικών εργασιών. Οι τεχνικές και οι μέθοδοι ποικίλουν και ενδεχομένως, να πρέπει να εμπλουτιστούν ακόμα περισσότερο αλλά εν κατακλείδι είναι μια τεχνική που ενισχύει σημαντικά τις διαδικασίες της σημασιολογική ανάλυσης.

2.3.3 Μηχανική Μάθηση

Η χρήση των μεθόδων της μηχανικής μάθησης στην ανάλυση συναισθήματος και τη σημασιολογική ανάλυση κειμένων, όπως και σε άλλα επιστημονικά πεδία – τεχνητή νοημοσύνη, συστήματα διαχείρισης γνώσης, λήψη αποφάσεων, κ.α. – αποσκοπεί στον εντοπισμό και στην χρήση του πλέον κατάλληλου αλγόριθμου για την εξαγωγή αποτελεσμάτων.

Η εξαγωγή αποτελεσμάτων απαιτεί πολλους πειραματισμούς των ερευνητών με διαφορετικού τύπου αλγόριθμους, οι οποίοι εκπαιδεύονται σε πολλά και διαφορετικά σύνολα δεδομένων έτσι ώστε να αποδώσουν τελικά την κατηγοριοποίηση των αγνώστων περιπτώσεων [24, 23, 21, 5, 6, 16, 13, 15].

Στην εργασία [21] μελετάται η υποκειμενική ταξινόμηση σύντομων μηνυμάτων από το Twitter, χρησιμοποιούνται κατηγοριοποιητές Μ.Μ. όπως ο Naive Bayes, Maximun Entropy, Support Vector Machines ενώ η εξαγωγή των χαρακτηριστικών στηρίζεται σε unigrams, bigram και part-of-speech ετικέτες. Για την δημιουργία του συνόλου εκπαίδευσης (training set) χαρακτηρίζονται χειροκίνητα 177 αρνητικά και 184 θετικά μηνύματα. Ο αλγόριθμος που κατηγοριοποίησε με τα υψηλότερα ποσοστά ταξινόμησης είναι ο MaxEntropy με ποσοστά επιτυχίας 83%. Στην εργασία [13] εφαρμόζεται ίδια μέθος απλά διπλασιάζεται το σύνολο εκπαίδευσης, με ισάριθμα θετικά και αρνητικά μηνύματα. Το σύνολο εκπαίδευσης αυτή τη φορά αποτελείται από 370 θετικά και 370 αρνητικά μηνύματα που χαρακτηρίσθηκαν χειροκίνητα ενώ ο κατηγοριοποιητής που είχε την μεγαλύτερη ακρίβεια ήταν ο Naive Bayes με ποσοστά 64%. Εδά ίσως να προκύπτει το ζήτημα της υπερκάλυψης (overfitting) του αλγορίθμου.

2.3.4 Κατηγοριοποίηση κειμένων

Ένα απο τα προβλήματα της εξόρυξης κειμένων είναι η εκτίμιση ομοιότητας μεταξύ εγγράφων διαφορετικού περιεχομένου. Αυτό σημαίνει είτε διαχωρισμό των εγγράφων σε προκαθορισμένες κατηγορίες είτε ομαδοποίηση εγγράφων σε φυσικές ομάδες.

Η κατηγοριοποίηση κειμένων (text classification ή categorization) ή αλλιώς κατηγοριοποίηση εγγράφων (document classification ή categorization) είναι η διαδικασία ανάθεσης ηλεκτρονικών εγγράφων, που περιέχουν κείμενο σε φυσική γλώσσα, σε μια ή περισσότερες κατηγορίες, σύμφωνα με το περιεχομένό τους. Κατατάσουμε την κατηγοριοποίηση κειμένων στίς εξής τρεις κατηγορίες:

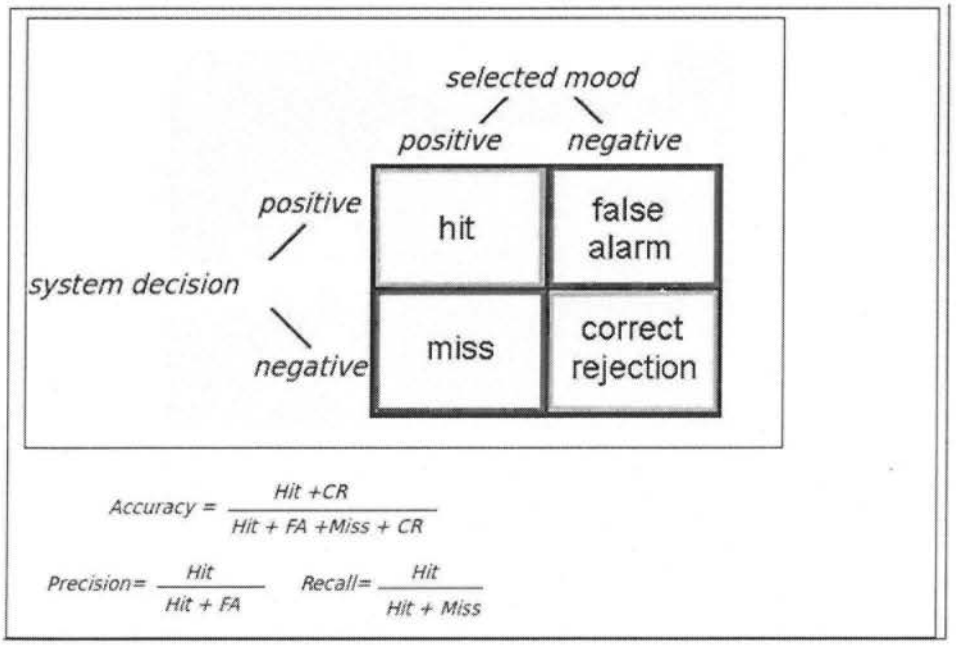
- Κατηγοριοποίηση με επίβλεψη (supervised classification), όπου κάποιος εξωτερικός μηχανισμός όπως η ανθρώπινη κρίση συμμετέχει στην σωστή κατηγοριοποίηση.
- Κατηγοριοποίηση με ημι-επίβλεψη (semi-supervised classification), όπου τμήματα των κειμένων έχουν ανατεθεί με ετικέτες από εξωτερικό μηχανισμό.
- Κατηγοριοποίηση χωρίς επίβλεψη (unsupervised classification), όπου η κατηγοριοποίηση γίνεται χωρίς καμία αναφορά σε εξωτερικό μηχανισμό.

Η μέθοδος που χρησιμοποιήσαμε στην παρούσα εργασία ανήκουν στο πρώτο είδος. Σε αυτή την περίπτωση έχουμε ένα σύνολο δεδομένων (data set), που αποτελείται από έγγραφα, που ονομάζεται και συλλογή εγγράφων (collection ή corpus). Η συλλογή αυτή χωρίζεται σε δύο μέρη: στο σύνολο εκπαιδευσης (training set) και το σύνολο δοκιμής (test set). Το πρώτο αποτελεί συνήθως το 80% περίπου ολόκληρης της συλλογής, ενώ το δεύτερο το υπόλοιπο 20%. Τα έγγραφα τα οποία προορίζονται για το σύνολο εκπαίδευσης είναι γνωστό σε ποια κατηγορία ανήκουν. Αυτά είναι τα δεδομένα με τα οποία λειτουργεί ο αλγόριθμος. Επομένως, γίνεται εύκολα αντιληπτό ότι όσο μεγαλύτερο είναι αυτό το σύνολο (περισσότερα έγγραφα), τόσο πιο ενημερωμένος είναι ο αλγόριθμος, αποδίδοντας καλύτερα. Από την άλλη, για τα έγγραφα που ανήκουν στο σύνολο δοκιμής δεν είναι γνωστή η κατηγορία στην οποία ανήκει το καθένα (η πληροφορία της κατηγορίας υπάρχει αλλά δεν παρέχεται στον αλγόριθμο). Έτσι, ο αλγόριθμος καλείται να τα κατατάξει στις σωστές κατηγορίες και στη συνέχεια τα αποτελέσματά του συγκρίνονται με τα πραγματικά

δεδομένα. Άρα, με το σύνολο εκπαίδευσης ο αλγόριθμος εκπαιδεύεται ενώ με το σύνολο δοκιμής δοκιμάζεται και αξιολογείται. Το σύνολο δοκιμής ονομάζεται και ground truth ή gold standard (test) διότι η πληροφορία που παρέχει αναφέρεται στήν ακρίβεια της κατηγοριοποίησης με επίβλεψη.

Γνωστές τεχνικές κατηγοριοποίησης κειμένων είναι: κατηγοριοποιητής naïve Bayes, tf-idf, Latent Semantic Indexing (LSI), Support Vector Machines (SVM), τεχνητά νευρωνικά δίκτυα, αλγόριθμος k-κοντινότερων γειτόνων (kNN), δέντρα αποφάσης (αλγόριθμοι ID3 και C4.5), concept mining και άλλα. Στην εργασία [7] προτείνεται μια τέτοια μέθοδος χρησιμοποιώντας κατηγοριοποιητές υψηλής ακρίβειας (HP-Subj, HP-Obj) οι οποίοι αυτόματα αναγνωρίζουν κάποιες υποκειμενικές και αντικειμενικές προτάσεις. Οι κατηγοριοποιητές χρησιμοποιούν μια λίστα από λεξικογραφικούς όρους οι οποίοι αποτελούν μια πρώτη ένδειξη. Μια πρόταση θα ταξινομηθεί ως υποκειμενική εάν περιέχει δύο ή περισσότερους ενδεικτικούς όρους της λίστας και ως αντικειμενική στην αντίθετη περίπτωση. Οι κατηγοριοποιητές συνήθως δίνουν υψηλά ποσοστά ακρίβειας (high precision) αλλά χαμηλά ποσοστά ολοκλήρωσης (low recall).

Οι όροι ακρίβεια (precision) και recall (ανάκληση) ορίζονται ως εξής:



Εικόνα 2.3.4-1: Ορισμός των όρων ακρίβεια (precision) και recall (ανάκληση)

Για παράδειγμα, στην περίπτωση ενός ταξινομήτη που λαμβάνει τη διαδική απόφαση για το αν ένα μήνυμα είναι θετικό ή αρνητικό ισχύουν τα εξής:

Hit: Το μήνυμα να ήταν θετικό και να προβλέφθηκε ως θετικό.
Miss: Το μήνυμα να ήταν θετικό και να μην προβλέφθηκε ως θετικό
False Alarm: Το μήνυμα να ήταν μην ήταν θετικό και να προβλέφθηκε ως θετικό
Correct Rejection: Το μήνυμα να ήταν μην ήταν θετικό και να προβλέφθηκε ως μη θετικό

Οι δύο όροι ανάκληση και ακρίβεια είναι αντιστρόφως ανάλογοι, οπότε συνήθως υπολογίζεται η ακρίβεια σε διάφορα επίπεδα ανάκλησης. Το μέτρο F είναι ενδεικτικό της ακρίβειας του συστήματος, ως συνάρτησης των recall και Precision λαμβάνοντας τιμές από 0 έως 1 και υπολογίζεται ως εξής:

$$F = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Σε κάθε έρευνα σημασιολογικού προσδιορισμού κειμένων, ανεξαρτήτου τεχνικής προσέγγισης, η ανάλυση εξαρτάται άμεσα από το πεδίο εφαρμογής των συνόλων [30]. Μια μέθοδος που έχει αξιόλογα αποτελέσματα σε ένα πεδίο μπορεί να μην είναι αξιόπιστη σε κάποιο άλλο. Το γεγονός αυτό οφείλεται στις ιδιαιτερότητες της φυσικής γλώσσας όπου η ερμηνεία πολλών λέξεων δεν είναι μονοσήμαντη. Όπως επισημαίνεται στην εργασία [24] το επίθετο «απρόβλεπτο» ερμηνεύεται αρνητικά όταν αφορά στη συμπεριφορά ενός αυτοκινήτου (απρόβλεπτη συμπεριφορά), ενώ θετικά όταν σχετίζεται με μια κινηματογραφική ταινία (απρόβλεπτη πλοκή).

3 *ΚΕΦΑΛΑΙΟ :* Συγκεντρωτική Παρουσίαση Αποτελεσμάτων της Πειραματικής Διαδικασίας

Οι lexicon-based τεχνικές που υλοποιήσαμε έδειξαν ότι η χρήση ενός σημασιολογικά βαθμολογημένου λεξικού απόψεων ενισχύει σημαντικά τις μεθόδους της υποκειμενικής ταξινόμησης. Αν και το λεξικό που χρησιμοποιήσαμε επιδέχεται βελτιώσεις, παρόλα αυτά συνέβαλε σημαντικά στο υψηλό ποσοστό των κειμένων που ταξινομήθηκαν και επιπλέον είναι ανεξάρτητο του πεδίου εφαρμογής γεγονός που ενισχύει την χρηστικότητά του.

Το σύνολο δοκιμής (test set) απο την συλλογή εγγράφων μας (corpus) για τις τεχνικές lexicon-based αποτελείται απο απο πέντε csv αρχεία όπου το καθένα έχει είκοσι κείμενα. Η συλλογή του συνόλου δοκιμής έγινε από τη μεγαλύτερη διαδικτυακή βάση δεδομένων για ταινίες, imdb (Internet Movie Database) και έχει ίδια θεματολογία – θεματικό προσανατολισμό.

- Negative.csv (περιέχει είκοσι αρχεία με αρνητικό συναίσθημα)
- Negtraining.csv (περιέχει είκοσι αρχεία με αρνητικό συναίσθημα)
- Positive.csv (περιέχει είκοσι αρχεία με θετικό συναίσθημα)
- Postraining.csv (περιέχει είκοσι αρχεία με θετικό συναίσθημα)
- Neutral.csv (περιέχει δεκαεννέα αρχεία χωρίς συναίσθημα)

Στη συνέχεια του κεφαλαίου εξετάζουμε επιμέρους τις τρείς βιβλιοθήκες που χρησιμοποίησαμε και σχολιάζουμε για κάθε μία από αυτές τις ιδιαιτερότητες που εντοπίσαμε από την πειραματική διαδικασία.

3.1 Βιβλιοθήκη "Sentimental"

Η Sentimental είναι μια απλή βιβλιοθήκη για συναισθηματική ανάλυση γραμμένη στη γλώσσα προγραμματισμού Ruby. Η βιβλιοθήκη εφαρμόζει τη μεθοδολογία της τόσο σε επίπεδο κειμένου όσο και σε επίπεδο πρότασης. Αποδίδεται στις λέξεις πολικότητα (polarity), και εκφράζει την υποκειμενική άποψη, θετική ή αρνητική, που αποτυπώνεται στο σύνολο δεδομένων. Οι προτάσεις διασπώνται σε μικρότερα τμήματα και υπολογίζει τη θετική ή την αρνητική πολικότητα, τόσο σε ολόκληρο το κείμενο όσο και σε επίπεδο πρότασης.

Για παράδειγμα η προεπιλεγμένη πολικότητα είναι το 0.0. Αν μια πρόταση έχει άθροισμα 0.0 τότε χαρακτηρίζεται ως ουδέτερη. Υψηλότερο άθροισμα συνεπάγεται σε θετικό αποτέλεσμα ενώ χαμηλότερο σε αρνητικό.
Εάν ορίσουμε την προεπιλεγμένη πολικότητα σε μη μηδενική τιμή, π.χ. 0.25

- Θετικά αθροίσματα είναι > 0.25
- Ουδέτερα αθροίσματα είναι -0.25 - 0.25
- Αρνητικά αθροίσματα είναι < -0.25

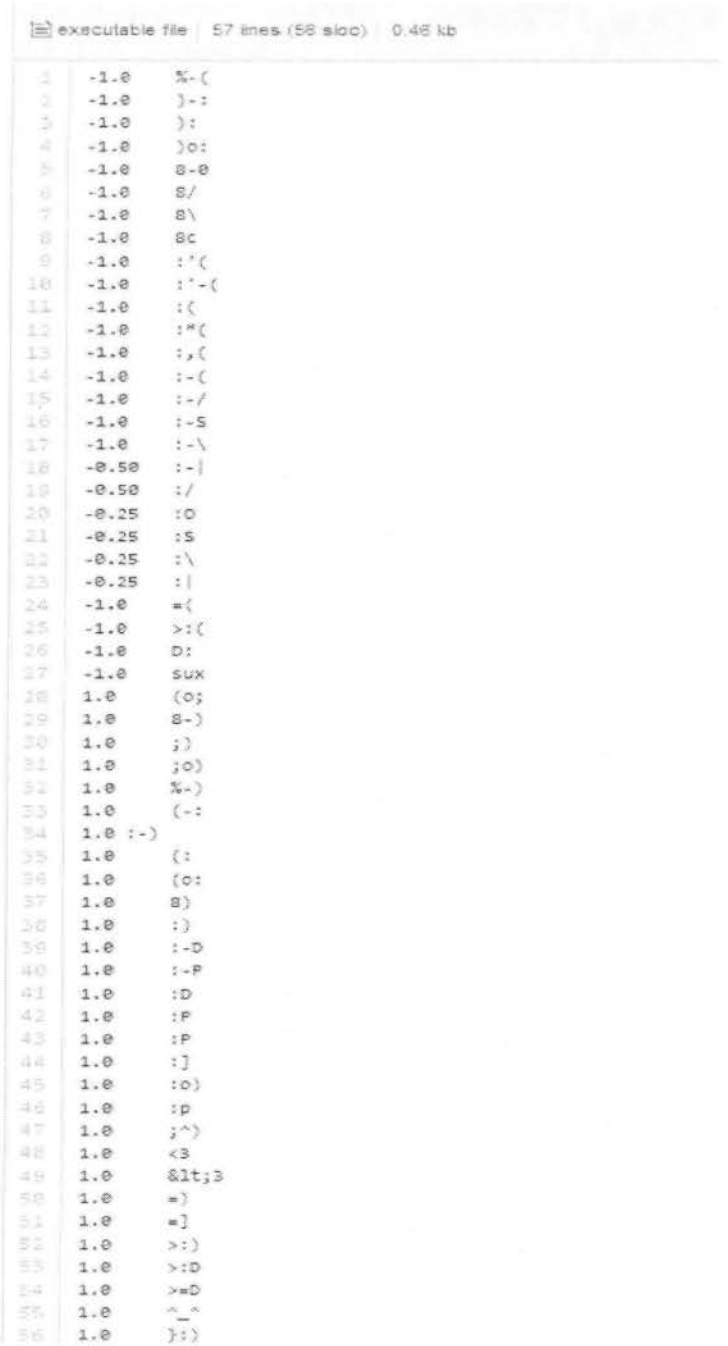
3.1.1 Συναισθηματικά λεξικά

Συνοπτικά, παρακάτω παρουσιάζεται η συναισθηματική προσέγγιση των λέξεων που χρησιμοποίηθηκαν στη βιβλιοθήκη Sentimental και η πολικότητα που τους δόθηκε.
Οπως βλέπουμε εχουν βαθμολογηθεί 18550 λέξεις με βαθμολογία απο [-1, 1]

executable file | 18551 lines (18550 sloc) | 303.569 kb

```
1.0     epic
1.0     good
1.0     upright
0.958333333333  fortunate
0.875   wonderfulness
0.875   wonderful
0.875   wide-eyed
0.875   wholesomeness
0.875   well-to-do
0.875   well-situated
0.875   well-heeled
0.875   well-fixed
0.875   vermifuge
0.875   utter
0.875   unpretentious
0.875   unmannerly
0.875   unmannered
0.875   understated
0.875   topping
0.875   top-hole
0.875   top-flight
0.875   therapeutic
0.875   sweetmeat
0.875   superb
0.875   successfulness
0.875   staring
0.875   spiffing
0.875   sodding
0.875   serendipity
0.875   self-respecting
0.875   self-respectful
0.875   self-respect
0.875   self-regard
0.875   seamy
0.875   sanative
0.875   salutary
0.875   rosy-cheeked
0.875   rose-cheeked
0.875   recuperative
0.875   radiate
0.875   radiance
0.875   prosperity
0.875   principled
0.875   picturesqueness
0.875   parasiticidal
0.875   overrefined
0.875   out-and-outer
```

Εικόνα 3.1.1-1: Το λεξικό της βιβλιοθήκης Sentimental



executable file | 57 lines (58 sloc) | 0.46 kb

```
-1.0    %-(
-1.0    )-:
-1.0    ):
-1.0    )o:
-1.0    8-0
-1.0    8/
-1.0    8\
-1.0    8c
-1.0    :'(
-1.0    :'-(
-1.0    :(
-1.0    :*(
-1.0    :,(
-1.0    :-(
-1.0    :-/
-1.0    :-S
-1.0    :-\
-0.50   :-|
-0.50   :/
-0.25   :O
-0.25   :S
-0.25   :\
-0.25   :|
-1.0    =(
-1.0    >:(
-1.0    D:
-1.0    sux
1.0     (o;
1.0     8-)
1.0     ;)
1.0     ;o)
1.0     %-)
1.0     (-:
1.0 :-)
1.0     (:
1.0     (o:
1.0     8)
1.0     :)
1.0     :-D
1.0     :-P
1.0     :D
1.0     :P
1.0     :P
1.0     :]
1.0     :o)
1.0     :p
1.0     ;^)
1.0     <3
1.0     &lt;3
1.0     =)
1.0     =]
1.0     >:)
1.0     >:D
1.0     >=D
1.0     ^_^
1.0     }:)
```

Εικόνα 3.1.1-2: Τα emoticons της βιβλιοθήκης Sentimental

Εκτός απο το λεξικό ήταν αναγκαία και η χρήση ειδικών χαρακτήρων που εκφράζουν συναίσθημα περιεκτικά, για αυτο έχουν χαρακτηριστεί επιπλέον και 56 emoticons στη βιβλιοθήκη .

Αυτή η ανάγκη προκύπτει από το γεγονός ότι στα κοινωνικά δίκτυα, που χρησιμοποιούν το Micro – Blogging, οι επιτρεπόμενοι χαρακτήρες ανάρτησης είναι περιορισμένοι. Έτσι λοιπόν, προκειμένου να γίνει πιο περιεκτική η επικοινωνία ανάμεσα στους χρήστες, αναπτύχθηκε και η χρήση των emoticons τα οποία μπορούν να εκφράζουν τα συναισθήματα των χρηστών με τη χρήση των ειδικών χαρακτήρων.

Πιο συγκριμένα, η άντληση χρήσιμης πληροφορίας και πιο συγκεκριμένα του συναισθήματος από την ανάλυση των ειδικών χαρακτήρων που εμφανίζονται στο Twitter, και ιδιαίτερα των emoticons, έχει γίνει αντικείμενο μελέτης από πολλούς ερευνητές [8, 9, 10] .

Κοινός παρονομαστής των μελετών τους είναι οι ειδικοί χαρακτήρες που χρησιμοποιούνται για τη δημιουργία των emoticons, με βάση την οποία προσδιορίζεται η συναισθηματική και σημασιολογική στάση των χρηστών απέναντι στη θεματολογία. Στην παρακάτω εικόνα παρουσιάζονται τα πιο συνηθισμένα emoticons και οι χαρακτήρες από τους οποίους αποτελούνται. Κρίνεται σκόπιμο να παρατεθούν διότι αφενός παρουσιάζουν μία συμπληρωματική προσέγγιση του σημασιολογικού και συναισθηματικού περιεχομένου του Παγκόσμιου Ιστού και αφετέρου τονίζουν την αξία της μελέτης και της εξαγωγής συμπερασμάτων από τη μελέτη του κοινωνικού περιεχομένου που διαμορφώνουν οι χρήστες.

Εικόνα 3.1.1-3: Emoticons και ειδικοί χαρακτήρες

3.1.2 Αποτελέσματα πειραματικής διαδικασίας

Ο σκοπός – στόχος της παρούσας μελέτης, εξετάζει την αποτύπωση του συναισθήματος των χρηστών, όπως αυτή προκύπτει από το περιεχόμενο των αναρτήσεων τους, στα προς μελέτη σύνολα δεδομένων.

Η αποτύπωση του συναισθήματος γίνεται μέσω της σύγκρισης των συνόλων δεδομένων με τη χρήση του λεξικού της Sentimental. Στο συγκεκριμένο λεξικό, αποδίδονται στην εκάστοτε λέξη δύο score, ένα «Θετικό» και ένα «Αρνητικό». Όπως είναι εμφανές και αναμενόμενο μπορεί να προκύψουν και να παρουσιαστούν και στα πειραματικά αποτελέσματα, οι περιπτώσεις όπου το score των tweet μηδενίζεται.

Με βάση λοιπόν την παραπάνω συλλογιστική, διαμορφώνονται τρεις βαθμίδες απόδοσης συναισθήματος που είναι οι παρακάτω:

- *Θετική Στάση – Positive*, αν το score του tweet είναι μεγαλύτερο από το 0.
- *Αρνητική Στάση – Negative*, αν το score του tweet είναι μικρότερο από το 0.
- *Ουδέτερη Στάση – Neutral*, αν το score του tweet είναι ίσο με το 0.

Παρακάτω παρουσιάζονται όλα τα πειράματα όπως για όλα τα σύνολα δεδομένων , με τις εντολές που τα τρέξαμε αλλά και τα αποτελέσματα τους όπως αποτυπώθηκαν στην οθόνη του υπολογιστή.

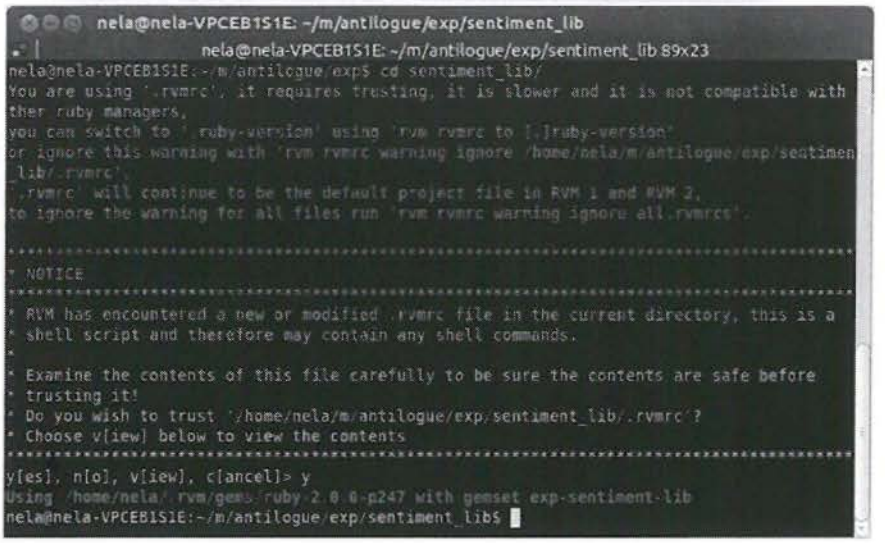
Το αρχείο Gemfile που έχει ολες τις βιβλιοθήκες που χρησιμοποιήσαμε για να στήσουμε το περιβάλλον του πρώτου set πειραμάτων με την χρήση της βιβλιοθήκης Sentimental.

```
1  source 'http://rubygems.org'
2  gem 'sentimental'
3  gem 'pry'
4  gem 'pry-doc'
5  gem 'pry-rescue'
6  gem 'pry-stack_explorer'
```

Το εκτελέσιμο πρόγραμμα που χρησιμοποιήσαμε για να τρέξουμε τα πειράματα:

```ruby
require 'csv'
require 'sentimental'
require 'open-uri'

csv_filename = ARGV[0]

def load_class_table(csv_filename)
csv_file = File.new(csv_filename, "r")
# file name , class
table = CSV.read(csv_file, headers: true, header_converters: :symbol ,
col_sep: " , ")
csv_file.close
return table
end
def sentiment_classify(table , classifier)
correct = 0
table.each do |tuple|
test_object = open(tuple[:filename]).read
test_class = classifier.get_sentiment(test_object)
real_class = tuple[:class].to_sym
if real_class == test_class
   a. correct += 1
end
end
accuracy = 0.0
accuracy = correct / Float(table.count)
return accuracy , correct
end


def setup_classifier(dataset_filename = nil)
Sentimental.load_defaults
Sentimental.load_senti_file(dataset_filename) if dataset_filename
classifier = Sentimental.new
return classifier
end

test_objects = load_class_table(csv_filename)

test_objects.each do |tuple|
puts "#{tuple[:filename]} #{tuple[:class]}"
end

classifier = setup_classifier()
acc, tp = sentiment_classify(test_objects,classifier)
puts "Overall accuracy: #{acc} and #{tp} objects were classified correctly"
```

Επιπλέον για κάθε βιβλιοθήκη χρησιμοποιήθηκε διαφορετικό σύνολο βιβλιοθηκών(gems) το οποιό λέγεται gemspec, για να μην παρουσιαστεί κάποιο σφάλμα.



Εικόνα 3.1.2-1:: Το gemspec της βιβλιοθήκης sentimental

Στην εικόνα 3.1.2-2 βλέπουμε το πρώτο πείραμα με το negative.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Παρατηρούμε οτι έντεκα από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 55%.

Εικόνα 3.1.2-2: Πείραμα βιβλιοθήκης Sentimental με το negative.csv αρχείο

Στην εικόνα 3.1.2-3 βλέπουμε το δεύτερο πείραμα με το negtraining.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Παρατηρούμε οτι δώδεκα από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 60%.

Εικόνα 3.1.2-3: Πείραμα βιβλιοθήκης Sentimental με το negtraining.csv

Στην εικόνα 3.1.2-4 βλέπουμε το τρίτο πείραμα με το positive.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Παρατηρούμε οτι δεκαέξι από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 80%.

Εικόνα 3.1.2-4: Πείραμα βιβλιοθήκης Sentimental με το positive.csv αρχείο

Στην εικόνα 3.1.2-5 βλέπουμε το τέταρτο πείραμα με το postraining.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Παρατηρούμε οτι δεκαπέντε από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 75%.

Εικόνα 3.1.2-5: Πείραμα βιβλιοθήκης Sentimental με το postraining.csv αρχείο

Στην εικόνα 3.1.2-6 βλέπουμε το πρώτο πείραμα με το negative.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Παρατηρούμε οτι δεν κατηγοριοποιήθηκε κανένα κείμενο στη σωστή του κλάση.

Εικόνα 3.1.2-6: Πείραμα βιβλιοθήκης Sentimental με το neutral.csv αρχείο

Συγκεντρωτικός Πίνακας Αποτελεσμάτων

Βιβλιοθήκη Sentimental		
Όνομα Αρχείου	Ποσοστό %	Σωστά ταξινομημένα αρχεία
Negative.csv	55 %	11/20
Negtrainig.csv	60 %	12/20
Positive.csv	80 %	16/20
Postraining.csv	75 %	15/20
Neutral.csv	0	0/19

Πίνακας 3.1.2-1: Συγκεντρωτικά αποτελέσματα της βιβλιοθήκης Sentimental

Η παρατήρηση που μπορούμε να κάνουμε πάνω στα αποτελέσματα είναι ότι η απόδοση στα αρνητικά είναι μέτρια ενώ στα θετικά είναι καλύτερη. Παρατηρούμε επίσης ότι η βιβλιοθήκη δεν κατηγοριοποιεί κανένα ουδέτερο κείμενο, το οποίο μπορεί να οφείλεται στη προεπιλεγμένη πολικότητα της βιβλιοθήκης.

3.2 *Βιβλιοθήκη " Sentiment Lib "*

Η sentiment lib είναι άλλη μια απλή βιβλιοθήκη για συναισθηματική ανάλυση γραμμένη στη γλώσσα προγραμματισμού Ruby. Η διαφορά της από την sentimental είναι οτι έχει δύο στρατηγικές ανάλυσης οι οποίες χρησιμοποιούνται ανάλογα με τον τομέα που θέλει να αναλύσει ο συγγραφέας.

Η πρώτη στρατηγική (BasicDictStrategy) χρησιμοποιεί ένα αρχέιο λεξικού (at lib/sentiment_lib/data/analysis/basic_dict/words.txt) το οποίο είναι ίδιο με το λεξικό που χρησιμοποιεί και η αρχική μας βιβλιοθήκη.

Η δεύτερη στρατηγική (FinancialDictStrategy) χρησιμοποιεί το οικονομικό συναισθηματικό λεξικό των the Loughran και McDonald όπου οι αρνητικές και θετικές λέξεις είναι σε διαφορετικές κατηγορίες αφού έχουν βαθμολογηθεί πρώτα με -1 και +1. Το αποτέλεσμα προκύπτει απο το άθροισμα των βαρών των λέξεων.

Το αρχείο Gemfile που έχει ολες τις βιβλιοθήκες που χρησιμοποιήσαμε για να στήσουμε το περιβάλλον του δεύτερου set πειραμάτων με την χρήση της βιβλιοθήκης Sentiment Lib.

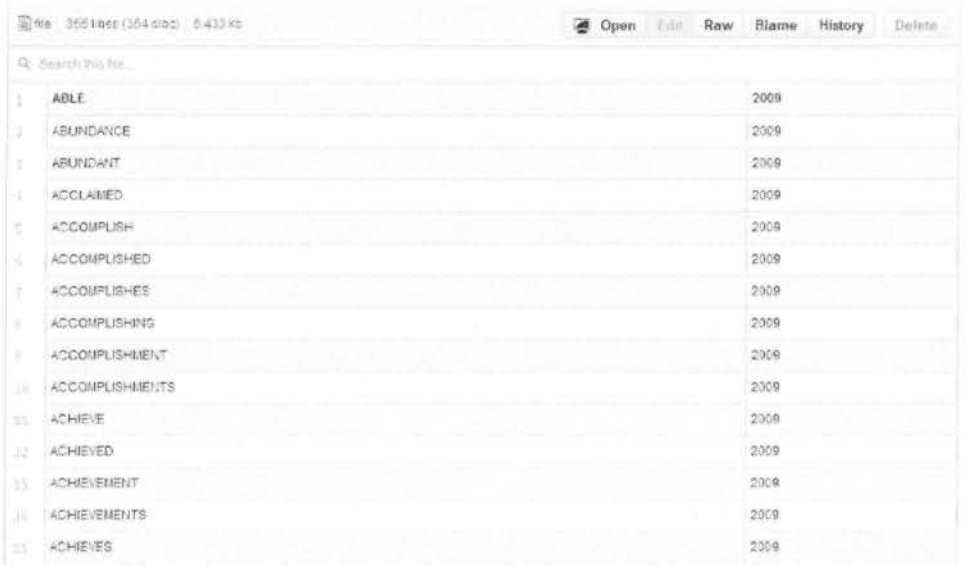
```
1  source 'http://rubygems.org'
2  gem 'classifier'
3  gem 'pry'
4  gem 'pry-doc'
5  gem 'pry-rescue'
6  gem 'pry-stack_explorer
```

Το εκτελέσιμο πρόγραμμα που χρησιμοποιήσαμε για να τρέξουμε τα πειράματα, κάθε φορά που χρησιμοποιείται η μια στρατηγική, η άλλη μπαίνει σε σχόλια.

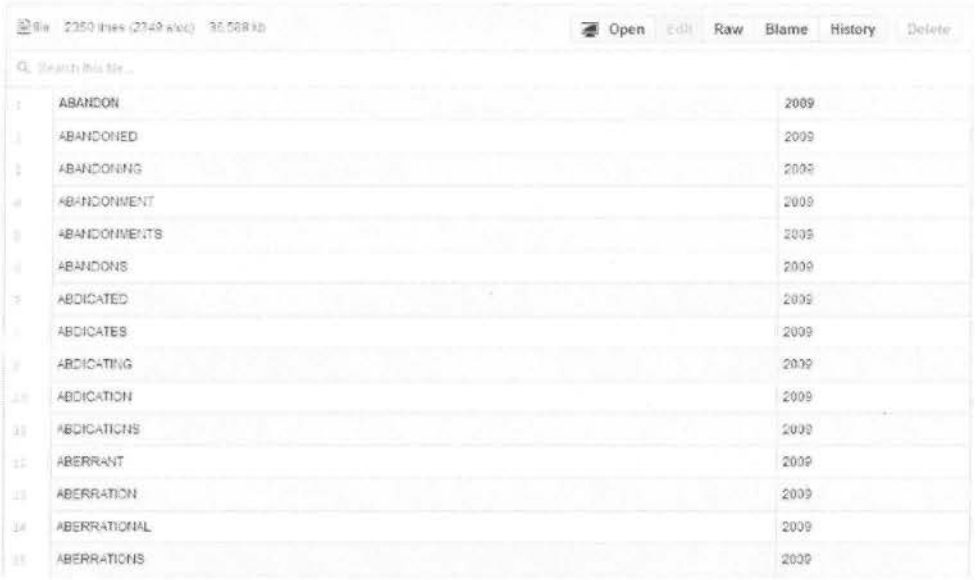
```
require 'csv'
require 'open-uri'
require 'sentiment_lib'
csv_filename = ARGV[0]

def load_class_table(csv_filename)
  csv_file = File.new(csv_filename, "r")
  # file name , class
  table = CSV.read(csv_file, headers: true, header_converters: :symbol , col_sep: "
, ")
  csv_file.close
 return table
end
def sentiment_classify(table , classifier)
  correct = 0
  table.each do |tuple|
    test_object = open(tuple[:filename]).read
    p test_object
    test_class = classifier.analyze(test_object)
    real_class = tuple[:class].to_sym
p real_class
        case
      when test_class < -0.0
            test_class = :negative
    when test_class > -0.0 && test_class < 0.0
            test_class = :neutral
        when test_class > 0.0
            test_class = :positive
        end
       if real_class == test_class
      correct += 1
    end
  end
  accuracy = 0.0
  accuracy = correct / Float(table.count)
  return accuracy , correct
end
def setup_classifier(dataset_filename = nil)
  #strategy = SentimentLib::Analysis::Strategies::BasicDictStrategy.new
    strategy = SentimentLib::Analysis::Strategies::FinancialDictStrategy.new
    classifier = SentimentLib::Analyzer.new({:strategy => strategy})
  return classifier
end
test_objects = load_class_table(csv_filename)
classifier = setup_classifier()
test_objects.each do |tuple|
  puts "#{tuple[:filename]} #{tuple[:class]}"
end

acc, tp = sentiment_classify(test_objects,classifier)
puts "Overall accuracy: #{acc} and #{tp} objects were classified correctly"
```

3.2.1 Συναισθηματικά λεξικά

Συνοπτικά, παρακάτω παρουσιάζεται η συναισθηματική προσέγγιση των λέξεων που χρησιμοποίηθηκαν στη βιβλιοθήκη Sentiment Lib και η πολικότητα που τους δόθηκε. Όπως αναφέραμε η Sentiment Lib χρησιμοποιεί δυο διαφορετικά λεξικά για τις δυο στρατηγηκες της, το BasicDictStrategy το οποίο είναι ίδιο με αυτό της Sentimental και αποτελείται απο 18550 βαθμολογημένες λέξεις με βαθμό απο [-1, 1] και το FinancialDictStrategy. Το FinancialDictStrategy έχει για σύνολα εκπαίδευσης δύο αρχεία, το positive.csv και το negative.csv όπου περιέχουν 354 και 2349 θετικές και αρνητικές λέξεις αντίστοιχα όπως βλέπουμε στις εικόνες 3.2.1-1 και 3.2.1-2. Αξίζει επίσης να αναφέρουμε ότι στην πρώτη στρατηγική χρησιμοποιούνται και ειδικοί χαρακτήρες (emoticons) ενω στην FinancialDictStrategy οχι.



Εικόνα 3.2.1-1: positive.csv αρχείο του FinancialDictStrategy



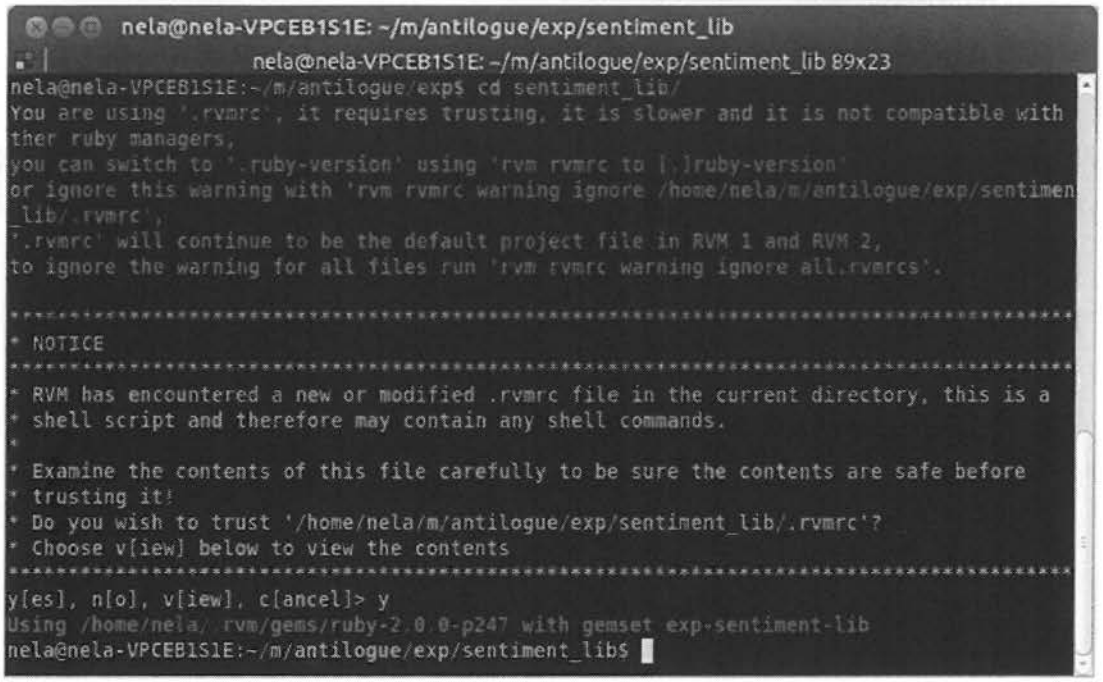
1	ABANDON	2009
2	ABANDONED	2009
3	ABANDONING	2009
4	ABANDONMENT	2009
5	ABANDONMENTS	2009
6	ABANDONS	2009
7	ABDICATED	2009
8	ABDICATES	2009
9	ABDICATING	2009
10	ABDICATION	2009
11	ABDICATIONS	2009
12	ABERRANT	2009
13	ABERRATION	2009
14	ABERRATIONAL	2009
15	ABERRATIONS	2009

Εικόνα 3.2.1-2: negative.csv αρχείο του FinancialDictStrategy

3.2.2 Αποτελέσματα πειραματικής διαδικασίας

Η αποτύπωση του συναισθήματος γίνεται ακριβώς με τον ίδιο τρόπο όπως και της βιβλιοθήκης Sentimental, η διαφορά που περιμένουμε να δούμε είναι μεταξύ της χρήσης των διαφορετικών στρατηγικών.

Παρακάτω παρουσιάζονται όλα τα πειράματα για όλα τα σύνολα δεδομένων , με τις εντολές που τα τρέξαμε αλλά και τα αποτελέσματα τους όπως αποτυπώθηκαν στην οθόνη του υπολογιστή, εκτελώντας πρώτα την BasicDictStrategy και στη συνέχεια την FinancialDictStrategy για καλύτερη σύγκριση αποτελεσμάτων.



Εικόνα 3.2.2-1: Το gemspec της βιβλιοθήκης Sentiment Lib

Τρέχουμε τα πειράματα για το κάθε αρχείο και με τις δύο βιβλιοθήκες για να συγκρινουμε την απόδοσή τους.

Στις εικόνες 3.2.2-1 και 3.2.2-2, βλέπουμε το πρώτο πείραμα με το negative.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Στο πρώτο όπου γίνεται χρήση του BasicDictStrategy παρατηρούμε οτι δεκαεπτά από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 85%. *Αντίστοιχα στο δεύτερο με χρήση του* FinancialDictStrategy παρατηρούμε οτι δεκαπέντε από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 75%.

Εικόνα 3.2.2-2: Πείραμα βιβλιοθήκης Sentiment Lib με το negative.csv αρχείο με το BasicDictStrategy λεξικό

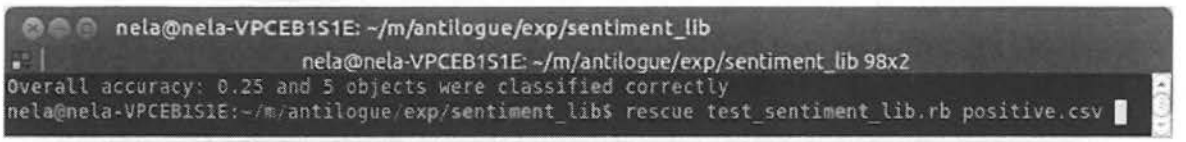
Εικόνα 3.2.2-3: Πείραμα βιβλιοθήκης Sentiment Lib με το negative.csv αρχείο με το FinancialDictStrategy λεξικό

Στις εικόνες 3.2.2-3 και 3.2.2-4, βλέπουμε το δεύτερο πείραμα με το negtraining.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Στο πρώτο όπου γίνεται χρήση του BasicDictStrategy παρατηρούμε οτι δεκαεννέα από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 95%. *Αντίστοιχα στο δεύτερο με χρήση του* FinancialDictStrategy παρατηρούμε οτι δεκαπέντε από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 75%.

Εικόνα 3.2.2-4: Πείραμα βιβλιοθήκης Sentiment Lib με το negtraining.csv αρχείο με το BasicDictStrategy λεξικό

Εικόνα 3.2.2-5: Πείραμα βιβλιοθήκης Sentiment Lib με το negtraining.csv αρχείο με το FinancialDictStrategy λεξικό

Στις εικόνες 3.2.2-5 και 3.2.2-6, βλέπουμε το τρίτο πείραμα με το positive.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Στο πρώτο όπου γίνεται χρήση του BasicDictStrategy παρατηρούμε οτι πέντε από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 25%. *Αντίστοιχα στο δεύτερο με χρήση του* FinancialDictStrategy παρατηρούμε οτι δώδεκα από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 60%.

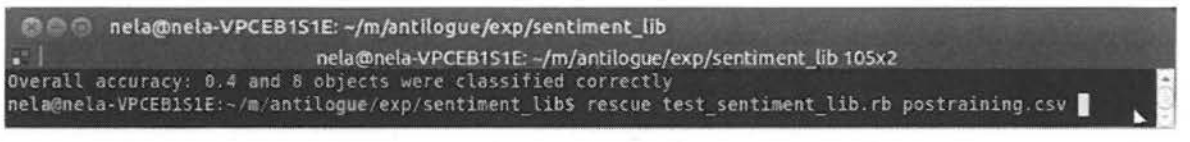


Εικόνα 3.2.2-6: Πείραμα βιβλιοθήκης Sentiment Lib με το positive.csv αρχείο με το BasicDictStrategy λεξικό

Εικόνα 3.2.2-7: Πείραμα βιβλιοθήκης Sentiment Lib με το positive.csv αρχείο με το FinancialDictStrategy λεξικό

Στις εικόνες 3.2.2-7 και 3.2.2-8, βλέπουμε το τέταρτο πείραμα με το postraining.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Στο πρώτο όπου γίνεται χρήση του BasicDictStrategy παρατηρούμε οτι έξι από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 30%. *Αντίστοιχα στο δεύτερο με χρήση του* FinancialDictStrategy παρατηρούμε οτι οχτώ από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 40%.

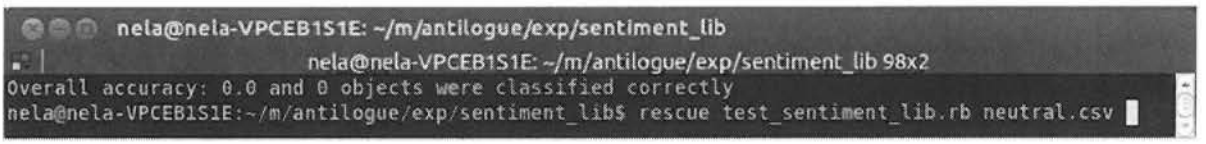
Εικόνα 3.2.2-8: Πείραμα βιβλιοθήκης Sentiment Lib με το postraining.csv αρχείο με το BasicDictStrategy λεξικό



Εικόνα 3.2.2-9: Πείραμα βιβλιοθήκης Sentiment Lib με το postraining.csv αρχείο με το FinancialDictStrategy λεξικό

Στις εικόνες 3.2.2-9 και 3.2.2-10, βλέπουμε το τελευταίο πείραμα με το neutral.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Παρατηρούμε οτι με τη χρήση και των δυο λεξικών δεν κατηγοριοποιείται σωστά κανένα κέιμενο

Εικόνα 3.2.2-10: Πείραμα βιβλιοθήκης Sentiment Lib με το neutral.csv αρχείο με το BasicDictStrategy λεξικό



Εικόνα 3.2.2-11: Πείραμα βιβλιοθήκης Sentiment Lib με το neutral.csv αρχείο με το FinancialDictStrategy λεξικό

Συγκεντρωτικός Πίνακας Αποτελεσμάτων

Βιβλιοθήκη Sentiment Lib		
Όνομα Αρχείου	Ποσοστό % (BasicDictStrategy)	Ποσοστό % (FinancialDictStrategy)
Negative.csv	85 %	75 %
Negtrainig.csv	95 %	75 %
Positive.csv	25 %	60 %
Postraining.csv	30 %	40 %
Neutral.csv	0	0

Πίνακας 3.2.2-1: Συγκεντρωτικά αποτελέσματα βιβλιοθήκης Sentiment Lib

Παρατηρούμε οτι η απόδοση του FinancialDictStrategy λεξικού έχει πιο ισορροπημένα αποτελέσματα από το βασικό λεξικό που όμως αποδίδει πολύ καλά στα θετικά, αλλά κάτω από 50% στα αρνητικά. Βλέπουμε όμως οτι και στις δύο στρατηγικές δεν κατηγοριοποιεί κανένα ουδέτερο κείμενο και αυτό οφείλεται όπως και στην παραπάνω βιβλιοθήκη στην προκαθορισμένη της πολικότητα.

3.3 Βιβλιοθήκη "Sad Panda"

Η Sad Panda είναι η τελευταία lexicon-based βιβλιοθήκη για συναισθηματική ανάλυση που χρησιμοποιήσαμε, γραμμένη και αυτή στη γλώσσα προγραμματισμού Ruby. Η βιβλιοθήκη πέρα από την θετική και αρνητική ταξινόμιση κειμένου κάνει και κατηγοριοποίηση συναισθήματος, αναγνωρίζοντας τα εξής συναισθήματα: "Οργή", "αποστροφή", "χαρά", "έκπληξη", "φόβο" και "θλίψη". Αποδίδεται στις λέξεις πολικότητα (polarity), με εύρος απο 1 εως 10 και εκφράζει την υποκειμενική άποψη, θετική ή αρνητική, που αποτυπώνεται στο σύνολο δεδομένων.

Το αρχείο Gemfile που έχει ολες τις βιβλιοθήκες που χρησιμοποιήσαμε για να στήσουμε το περιβάλλον του τρίτου set πειραμάτων με την χρήση της βιβλιοθήκης Sad Panda.

```
source 'http://rubygems.org'
gem 'sad_panda'
gem 'pry'
gem 'pry-doc'
gem 'pry-rescue'
gem 'pry-stack_explorer'
```

Το εκτελέσιμο πρόγραμμα που χρησιμοποιήσαμε για να τρέξουμε τα πειράματα:

```
require 'csv'
require 'open-uri'
require 'sad_panda'

csv_filename = ARGV[0]

def load_class_table(csv_filename)
csv_file = File.new(csv_filename, "r")
# file name , class
table = CSV.read(csv_file, headers: true, header_converters: :symbol ,
col_sep: " , ")
csv_file.close
return table
end


def sentiment_classify(table , classifier)
correct = 0
table.each do |tuple|
test_object = open(tuple[:filename]).read

#test_class = classifier.get_sentiment(test_object)
test_class = SadPanda.polarity(test_object)
real_class = tuple[:class].to_sym
    a. case
    b. when test_class < 5
        i. test_class = :negative
when test_class == 5
        i. test_class = :neutral
    b. when test_class > 5
        i. test_class = :positive
    c. end

if real_class == test_class
    a. correct += 1
end
end
accuracy = 0.0
accuracy = correct / Float(table.count)
```

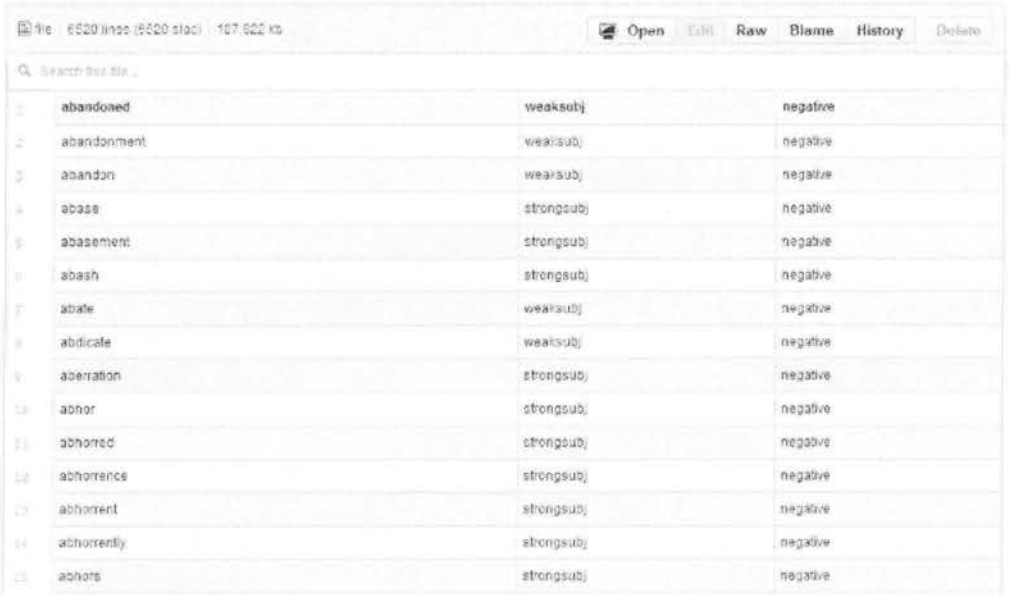
```
25 return accuracy , correct
26 end
27 def setup_sentimental_classifier(dataset_filename = nil)
28 Sentimental.load_defaults
29 Sentimental.load_senti_file(dataset_filename) if dataset_filename
30 classifier = Sentimental.new
31 return classifier
32 end


33 test_objects = load_class_table(csv_filename)

34 test_objects.each do |tuple|
35 puts "#{tuple[:filename]} #{tuple[:class]}"
36 end

37 #classifier = setup_classifier()
38 #acc, tp = sentiment_classify(test_objects,classifier)
39 acc, tp = sentiment_classify(test_objects,nil)

40 puts "Overall accuracy: #{acc} and #{tp} objects were classified correctly"
```

3.3.1 Συναισθηματικά λεξικά

Συνοπτικά, παρακάτω παρουσιάζεται η συναισθηματική προσέγγιση των λέξεων που χρησιμοποίηθηκαν στη βιβλιοθήκη Sad Panda και η πολικότητα που τους δόθηκε εικόνα 3.3.1-1 . Η βιβλιοθήκη χρησιμοποιεί και μια μέθοδο για να βρεί τον βαθμό υποκειμενικότητας ενός κειμένου δίνοντας βαρύτητα στις λέξεις (weaksubjectivity και strongsubjectivity), ανάλογα αν η λέξη είναι πολύ η λίγο υποκειμενική εικόνα 3.3.1-2. Τέλος στην εικόνα 3.3.1-3 φαίνεται η τεχνική κατηγοριοποίησης συναισθήματος αναγνωρίζοντας τα εξής συναισθήματα: "Οργή", "αποστροφή", "χαρά", "έκπληξη", "φόβο" και "θλίψη". Κάθε λέξη που χρησιμοποιείται κατατάσεται σε ένα από τα παραπάνω συναισθήματα.

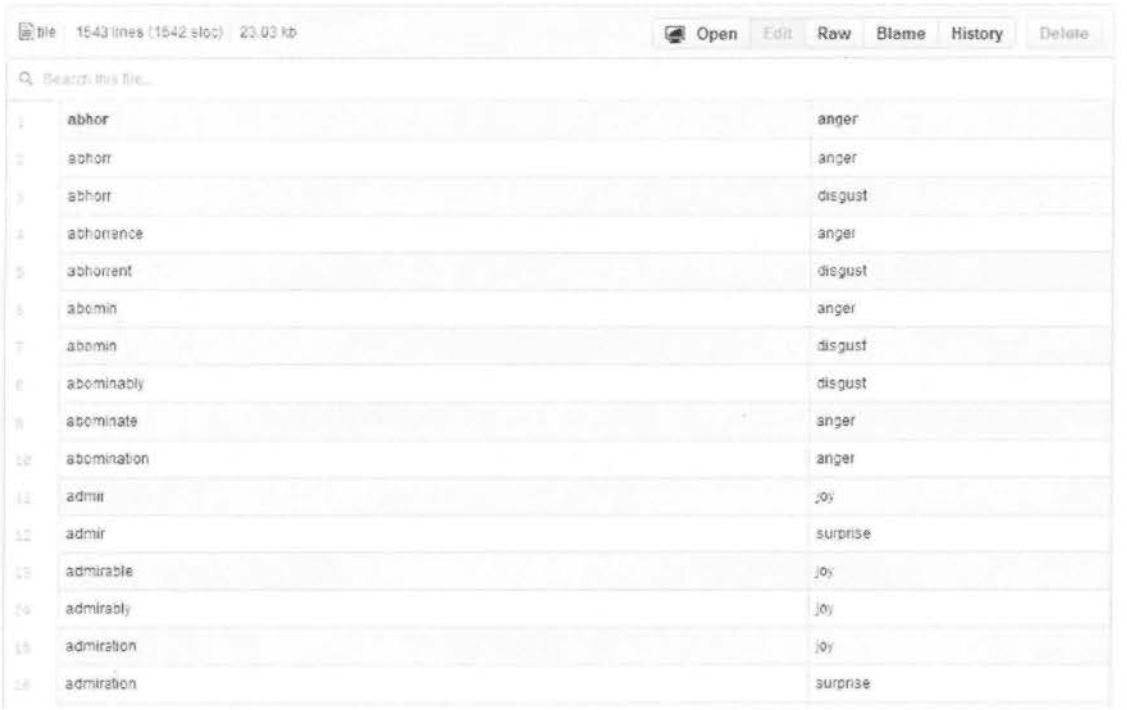
```
module TermPolarities
    # this method reads a csv file containing 'word,severity,positive/negative'
    # triplets, and returns a giant hash where the keys are individual words
        # and the values range between 0 and 2 (0 being more negative, 2 being most positive)
  def self.get_term_polarities
              @polarities = {"abandoned"=>2.5, "abandonment"=>2.5, "abandon"=>2.5, "abase"=>0,
                      "abasement"=>0, "abash"=>0, "abate"=>2.5, "abdicate"=>2.5, "aberration"=>0,
                      "abhor"=>0, "abhorred"=>0, "abhorrence"=>0, "abhorrent"=>0,
                      "abhorrently"=>0, "abhors"=>0, "abidance"=>10, "abide"=>10, "abject"=>0,
                      "abjectly"=>0, "abjure"=>2.5, "abilities"=>7.5, "ability"=>7.5, "able"=>7.5,
                      "abnormal"=>2.5, "abolish"=>2.5, "abominable"=>0, "abominably"=>0,
                      "abominate"=>0, "abomination"=>0, "above"=>7.5, "aboveaverage"=>7.5,
                      "abound"=>7.5, "abrade"=>2.5, "Abrasive"=>0, "abrupt"=>2.5, "abscond"=>0,
                      "absence"=>2.5, "absentee"=>2.5, "absentminded"=>0, "absolve"=>10, "absurd"=>0,
                      "absurdity"=>0, "absurdly"=>0, "absurdness"=>0, "abundant"=>7.5,
                      "abundance"=>7.5, "abuse"=>0, "abuses"=>2.5, "abusive"=>0, "abysmal"=>0,
                      "abysmally"=>0, "abyss"=>0, "accede"=>10, "accept"=>7.5, "acceptance"=>7.5,
                      "acceptable"=>7.5, "accessible"=>7.5, "accidental"=>2.5, "acclaim"=>10,
                      "acclaimed"=>10, "acclamation"=>10, "accolade"=>10, "accolades"=>10,
                      "accommodative"=>7.5, "accomplish"=>7.5, "accomplishment"=>7.5,
```

Εικόνα 3.3.1-1: Απόδοση πολικότητας στις λέξεις με εύρος 1-10 στη βιβλιοθήκη Sad Panda



	abandoned	weaksubj	negative
2	abandonment	weaksubj	negative
3	abandon	weaksubj	negative
4	abase	strongsubj	negative
5	abasement	strongsubj	negative
6	abash	strongsubj	negative
7	abate	weaksubj	negative
8	abdicate	weaksubj	negative
9	aberration	strongsubj	negative
10	abhor	strongsubj	negative
11	abhorred	strongsubj	negative
12	abhorrence	strongsubj	negative
13	abhorrent	strongsubj	negative
14	abhorrently	strongsubj	negative
15	abhors	strongsubj	negative

Εικόνα 3.3.1-2: Προσέγγιση λεξικού υποκειμενικότητας της βιβλιοθήκης Sad Panda



file 1543 lines (1542 sloc) 23.03 kb | Open | Edit | Raw | Blame | History | Delete

Search this file...

1	abhor	anger
2	abhorr	anger
3	abhorr	disgust
4	abhorrence	anger
5	abhorrent	disgust
6	abomin	anger
7	abomin	disgust
8	abominably	disgust
9	abominate	anger
10	abomination	anger
11	admir	joy
12	admir	surprise
13	admirable	joy
14	admirably	joy
15	admiration	joy
16	admiration	surprise

Εικόνα 3.3.1-3: Λεξικό απόδοσης συναισθημάτων της βιβλιοθήκης Sad Panda

3.3.2 Αποτελέσματα πειραματικής διαδικασίας

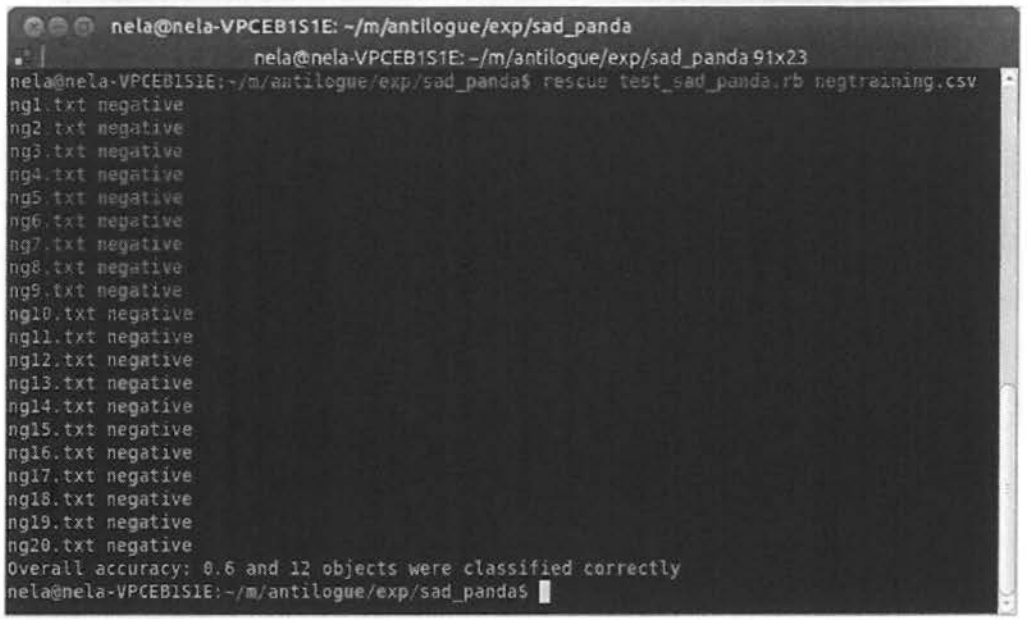
Παρακάτω παρουσιάζονται όλα τα πειράματα για όλα τα σύνολα δεδομένων , με τις εντολές που τα τρέξαμε αλλά και τα αποτελέσματα τους όπως αποτυπώθηκαν στην οθόνη του υπολογιστή.

Εικόνα 3.3.2-1: Το gemspec της βιβλιοθήκης sentiment lib

Στην εικόνα 3.3.2-2 Βλέπουμε το πρώτο πείραμα με το negative.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Παρατηρούμε οτι δεκαπέντε από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 75%.

Εικόνα 3.3.2-2: Πείραμα βιβλιοθήκης Sad Panda με το negative.csv αρχείο

Στην εικόνα 3.3.2-3 Βλέπουμε το δεύτερο πείραμα με το negtraining.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Παρατηρούμε οτι δώδεκα από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 60%.



Εικόνα 3.3.2-3: Πείραμα βιβλιοθήκης Sad Panda με το negtraining.csv αρχείο

Στην εικόνα 3.3.2-4 Βλέπουμε το τρίτο πείραμα με το positive.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Παρατηρούμε οτι δεκαοχτώ από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 90%.

Εικόνα 3.3.2-4: Πείραμα βιβλιοθήκης Sad Panda με το positive.csv αρχείο

Στην εικόνα 3.3.2-5 βλέπουμε το τέταρτο πείραμα με το postraining.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Παρατηρούμε οτι δεκατέσσερα από τα είκοσι κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 70%.

Εικόνα 3.3.2-5: Πείραμα βιβλιοθήκης Sad Panda με το postraining.csv αρχείο

Στην εικόνα 3.3.2-5 Βλέπουμε το πρώτο πείραμα με το neutral.csv το οποίο αποτελείται απο είκοσι αρνητικά κείμενα. Παρατηρούμε οτι τρία από τα δεκαεννέα κατηγοριοποιήθηκαν σωστά με ποσοστό επιτυχίας 15.8%

Εικόνα 3.3.2-6: Πείραμα βιβλιοθήκης Sad Panda με το neutral.csv αρχείο

Συγκεντρωτικός Πίνακας Αποτελεσμάτων

Βιβλιοθήκη Sad Panda		
Όνομα Αρχείου	Ποσοστό %	Σωστά ταξινομημένα αρχεία
Negative.csv	75 %	15/20
Negtrainig.csv	60 %	12/20
Positive.csv	90 %	18/20
Postraining.csv	70 %	14/20
Neutral.csv	15.8%	3/19

Πίνακας 3.3.2-1: Συγκεντρωτικά αποτελέσματα βιβλιοθήκης Sentiment Lib

Η τρίτη και τελευταία βιβλιοθήκη βασισμένη στα λεξικά έχει τα καλύτερα αποτελέσματα και στην ταξινόμιση των θετικών και των αρνητικών και είναι η μόνη που κατηγοριοποίησε ουδέτερα κείμενα αν και με πολύ χαμηλό ποσοστό, μολις 15.5%. Ο μόνος τρόπος για να έχουμε καλύτερη απόδοση στις παραπάνω βιβλιοθήκες είναι να αλλάξουμε τα λεξικά τους προσθέτοντας άλλες πιο βαρυσήμαντες λέξεις που πιθανόν να έχουν περισσοτερο συναισθηματικό φορτίο. Επειδή όμως αυτός ο τρόπος είναι πολύ χρονοβόρος και δεν μας εγγυάται καλύτερα αποτελέσματα καταφεύγουμε στην επόμενη λύση που είναι ο κατηγοριοποιητής Naïve Bayes.

3.4 Βιβλιοθήκη "Classifier"

Ο Classifier είναι ένα γενικό module που επιτρέπει Bayesian και άλλα είδη ταξινομήσεων. Στην τεκμηρίωση της βιβλιοθήκης αναφέρονται ο Naïve Bayes κατηγοριοποιητής και η συνάρτηση Λανθάνουσα Σημασιολογική Ευρετηριοποίηση (Latent Semantic Indexing ή LSI).

Ο αλγόριθμος αφελής Bayes (Naïve Bayes) είναι ένας κατηγοριοποιητής βασισμένος στις πιθανότητες. Η κατασκευή του εξαρτάται από ένα σύνολο εκπαίδευσης για να εκτιμήσει τις

παραμέτρους μιας κατανομής πιθανότητας, δεδομένων των τιμών των χαρακτηριστικών ενός νέου εγγράφου. Αυτές οι πιθανότητες εκτιμώνται με τη βοήθεια του θεωρήματος Bayes. Πρόσφατη μελέτη της Bayesian κατηγοριοποίησης έδειξε ότι υπάρχουν κάποιοι θεωρητικοί λόγοι για την καλή απόδοση του naïve τροπου[39]

Υποθέτουμε ότι έχουμε ένα σύνολο δεδομένων S και έστω ότι κάθε δείγμα δεδομένων X=(x1,x2,...,xn) με m κατηγορίες C1,C2,..,Cm. Δεδομένου ενός αγνώστου δείγματος δεδομένων X, ο κατηγοριοποιητής θα προβλέψει ότι το X ανήκει στην κατηγορία C που έχει την μέγιστη εκ των υστέρων(posterior) πιθανότητα με βάση το X. Αυτό σημαίνει ότι το X κατηγοριοποιείται στην Ci αν και μόνο αν:

$$p(Ci|X)>p(Cj|X) \text{ για κάθε } 1\leq j\leq m \text{ και } j\neq i$$

Ο στόχος, λοιπόν, είναι να βρούμε την μέγιστη posterior πιθανότητα, δηλαδή το μέγιστο p(Ci|X) για κάθε κλάση, με αποτέλεσμα ο Naïve Bayesian κατηγοριοποιητής να έχει υψηλή απόδοση. Η απόδοση του συγκρίνεται με αυτή των δέντρων απόφασης και κάποιους κατηγοριοποιητές που στηρίζονται σε νευρωνικά δίκτυα σε ορισμένες εφαρμογές.

Η λανθάνουσα σημασιολογική ευρετηριοποίηση (latent semantic indexing – LSI) [40] προσπαθεί να συλλάβει την υποκείμενη σημασιολογική δομή των δεδομένων. Συνίσταται στην εφαρμογή μιας τεχνικής της γραμμικής άλγεβρας, της αποσύνθεσης ιδιαζουσών τιμών (singular value decomposition), πάνω στον πίνακα εμφανίσεων των όρων ανά έγγραφο. Έτσι προκύπτουν νέες διαστάσεις, μαζί με τη βαρύτητα της καθεμίας, και τόσο οι αρχικοί όροι, όσο και τα έγγραφα αναπαριστώνται ενιαία ως ορθοκανονικά διανύσματα στον νέο διανυσματικό χώρο. Αντίθετα με την επιλογή και την ομαδοποίηση όρων, οι νέες διαστάσεις δεν είναι ερμηνεύσιμες διαισθητικά. Από αυτές επιλέγονται τελικά οι k σημαντικότερες για την αναπαράσταση, με το k να κυμαίνεται τυπικά μεταξύ 50 και 150. Η LSI πλεονεκτεί έναντι άλλων προσεγγίσεων σε περιπτώσεις που ένα πλήθος από όρους συνεισφέρει από λίγο ο καθένας στη διαμόρφωση σημαντικής πληροφορίας συνολικά. [41].

Το αρχείο Gemfile που έχει ολες τις βιβλιοθήκες που χρησιμοποιήσαμε για να στήσουμε το περιβάλλον του τέταρτου set πειραμάτων με την χρήση του Classifier.

```
source 'http://rubygems.org'
gem 'classifier'
gem 'pry'
gem 'pry-doc'
gem 'pry-rescue'
gem 'pry-stack_explorer'
```

Το εκτελέσιμο πρόγραμμα που χρησιμοποιήσαμε για να τρέξουμε τα πειράματα.

```
require 'csv'
require 'open-uri'
require 'fast_stemmer'
require 'classifier'
csv_filename = ARGV[0]

def load_class_table(csv_filename)
csv_file = File.new(csv_filename, "r")
# file name , class
table = CSV.read(csv_file, headers: true, header_converters: :symbol ,
col_sep: " , ")
csv_file.close
return table
end

def sentiment_classify(table , classifier)
correct_count = 0
table.each do |tuple , assessor = classifier, count = correct_count|

test_object = open(tuple[:filename]).read

    a. correct_count = yield(assessor, test_object , tuple, count)

end
accuracy = 0.0
accuracy = correct_count / Float(table.count)
return accuracy , correct_count
end

def setup_classifier(dataset = {})
pos_files = load_class_table(dataset["positive"])
neg_files = load_class_table(dataset["negative"])

classifier = Classifier::Bayes.new('negative','positive')

pos_files.each do |tuple|
training_object = open(tuple[:filename]).read
    a. classifier.train_positive(training_object)
end
neg_files.each do |tuple|
training_object = open(tuple[:filename]).read
```

```
      a. classifier.train_negative(training_object)
31 end

32 return classifier
33 end

34 test_objects = load_class_table(csv_filename)

35 test_objects.each do |tuple|
36 puts "#{tuple[:filename]} #{tuple[:class]}"
37 end

38 classifier = setup_classifier({"positive" => "positive.csv", "negative" =>
   "negative.csv"})
39 acc, tp = sentiment_classify(test_objects,classifier) do |klassifier,
   test_object , tuple, correct_count|
40 puts klassifier
41 puts correct_count
42 test_class = klassifier.classify(test_object)
43 real_class = tuple[:class]
44 puts "#{real_class} #{test_class}"
45 if real_class == test_class.downcase
      a. correct_count += 1
      b. puts correct_count
46 end
47 correct_count
48 end


49 puts "Overall accuracy: #{acc} and #{tp} objects were classified correctly"
```

3.4.1 Συλλογές δεδομένων

Όπως ήδη έχουμε αναφέρει οι Naïve Bayes κατηγοριοποιητές απαιτούν σύνολα εκπαίδευσης που κάνουν προβλέψεις για να εξάγουν αποτελέσματα. Οι συλλογές (corpus) που χρησιμοποιούνται παίζουν πολύ σημαντικό ρόλο στην ακρίβεια των αποτελεσμάτων της μεθόδου. Όπως αναφέραμε στο κεφάλαιο 2.3 για τον Naïve Bayes χρησιμοποιήσαμε κατηγοριοποίηση με επίβλεψη επειδή είναι απαραίτητη η ανθρώπινη κρίση για τον εντοπισμό συναισθημάτων σε ένα κείμενο λόγω των αμφίσημων της φυσικής γλώσσας. Η συλλογή εγράφων που θα χρησιμοποιήσουμε θα χωριστεί σε δύο κατηγορίες, το σύνολο δοκιμής που συνήθως αποτελεί το μικρότερο ποσοστό και το σύνολο εκπαίδευσης. Το σύνολο εκπαίδευσης πρέπει να αποτελείται απο πολλές κατηγορίες κειμένων για

να κατηγοριοποιεί σωστά όχι μόνο σε μια κατηγορία, όπως κριτικές ταινιών. Εκτός απο τα πέντε σύνολα δεδομένων που χρησιμοποιήσαμε για τις lexicon-based βιβλιοθήκες συλλέξαμε και άλλα δεδομένα :

Απο το Movie Review Data [13] χρησιμοποιήσαμε το polarity dataset v2.0 που αποτελείται απο 1000 θετικές και 1000 αρνητικές κριτικές ταινιών.

- cornell_mov_pos.csv (1000 θετικές κριτικές)
- cornell_mov_neg.csv (1000 αρνητικές κριτικές)
- pos_web.csv (20 κείμενα τεχνολογικού περιεχομένου με θετικό σημασιολογικό συναίσθημα)
- neg_web.csv (19 κείμενα τεχνολογικού περιεχομένου με αρνητικό σημασιολογικό συναίσθημα)
- pos_sentiment.csv (55 κείμενα τεχνολογικού περιεχομένου με θετικό σημασιολογικό συναίσθημα)
- neg_sentiment.csv (14 κείμενα τεχνολογικού περιεχομένου με αρνητικό σημασιολογικό συναίσθημα)
- pos_test.csv (22 κείμενα τεχνολογικού περιεχομένου με θετικό σημασιολογικό συναίσθημα για σύνολο δοκιμής)
- phealth.csv (30 κείμενα ιατρικού περιεχομένου με θετικό σημασιολογικό συναίσθημα)
- nhealth.csv (30 κείμενα ιατρικού περιεχομένου με αρνητικό σημασιολογικό συναίσθημα)

3.4.2 Αποτελέσματα πειραματικής διαδικασίας

Για να καταλήξουμε στο καλύτερο σε απόδοση σύνολο εκπαίδευσης για τον κατηγοριοποιητή μας, κάνουμε όλους τους πιθανούς συνδυασμούς με όλα τα δεδομένα μας.

Η εντολή που τρέχουμε έχει την εξής μορφή :

rescue classifier.rb TestSet.csv Positive_TrainingSet.csv Negative_TrainingSet.csv

Σύνολο Δοκιμής (Testset)	Σύνολο Εκπαίδευσης (Training Set)	
	Positive	**Negative**
pos_test.csv	pos_sentiment.csv	neg_sentiment.csv
18 / 21		
negtraining.csv	pos_sentiment.csv	neg_sentiment.csv
19 / 20		
negtraining.csv	positive.csv	neg_sentiment.csv
0 / 20		
negtraining.csv	postraining.csv	neg_sentiment.csv
3 / 20		
negtraining.csv	pos_sentiment.csv	negative.csv
20 / 20		
postraining.csv	positive.csv	neg_sentiment.csv
19 / 20		
postraining.csv	pos_sentiment.csv	neg_sentiment.csv
3 / 20		

Πίνακας 3.4.2-1Πρώτο πείραμα με όλους τους πιθανούς συνδυασμούς

Σύνολο Δοκιμής (Testset)	Σύνολο Εκπαίδευσης (Training Set)	
	Positive	**Negative**
Negative.csv	pos_sentiment.csv	neg_sentiment.csv
18 / 20		
positive.csv	pos_sentiment.csv	neg_sentiment.csv
2 / 20		

Πίνακας 3.4.2-2: Δεύτερο πείραμα με ίδιο σύνολο εκπαίδευσης

Σύνολο Δοκιμής (Testset)	Σύνολο Εκπαίδευσης (Training Set)	
	Positive	**Negative**
neg_sentiment.csv	pos_sentiment.csv	negative.csv
	1 / 14	
pos_sentiment.csv	positive.csv	neg_sentiment.csv
	1 / 82	
negative.csv	positive.csv	neg_sentiment.csv
	0 / 20	
positive.csv	pos_sentiment.csv	negative.csv
	2 / 20	

Πίνακας 3.4.2-3: Τρίτο πείραμα με όλους τους πιθανούς συνδυασμούς

Παρατηρούμε ότι τα αποτελέσματα κατηγοριοποίησης δεν είναι ικανοποιητικά μέ τα σύνολα εκπαίδευσης που ετοιμάσαμε. Ετοιμάζουμε νέο σετ πειραμάτων συνδυάζοντας κάποια από τα σύνολα δεδομένων που έχουμε.

- comb_postraining.csv (postrainig.csv+pos_sentiment.csv)
- comb_negtraining.csv (negtrainig.csv+neg_sentiment.csv)
- comb_positive.csv (positive.csv+pos_sentiment.csv)
- comb_negative.csv (negative.csv+neg_sentiment.csv)

Σύνολο Δοκιμής (Testset)	Σύνολο Εκπαίδευσης (Training Set)	
	Positive	**Negative**
positive	comb_postraining	comb_negraining
	1 / 20	
positive	comb_postraining	neg_sentiment
	10 / 20	
positive	comb_postraining	negative
	18 / 20	

Πίνακας 3.4.2-4: Δοκιμή απόδοσης θετικού συνόλου με το συνδυαστικό σύνολο εκπαίδευσης

Σύνολο Δοκιμής (Testset)	Σύνολο Εκπαίδευσης (Training Set)	
	Positive	**Negative**
postraining	comb_positive	comb_negative
0 / 20		
postraining	comb_positive	neg_sentiment
5 / 20		
postraining	positive	comb_negative
19 / 20		

Πίνακας 3.4.2-5: Δοκιμή απόδοσης θετικού συνόλου με το συνδυαστικό σύνολο εκπαίδευσης

Σύνολο Δοκιμής (Testset)	Σύνολο Εκπαίδευσης (Training Set)	
	Positive	**Negative**
negtraining	comb_positive	comb_negative
20 / 20		
negtraining	comb_positive	neg_sentiment
17 / 20		
negtraining	positive	comb_negative
1 / 20		

Πίνακας 3.4.2-6: Δοκιμή απόδοσης αρνητικού συνόλου με το συνδυαστικό σύνολο εκπαίδευσης

Στα παραπάνω πειράματα παρατηρούμε τις εξής ιδιαιτερότητες :

- Τα θετικά τείνουν να είναι domain specific, δηλαδή έχουμε καλύτερα αποτελέσματα όταν το σύνολο εκπάιδευσης ανήκει στην ίδια κατηγορία με το σύνολο δοκιμής. Η απόδοση μειώνεται όταν η κατηγορία του συνόλου εκπαίδευσης είναι διαφορετική απο αυτή του συνόλου δοκιμής.
- Τα αρνητικά δέν έχουν το ίδιο πρόβλημα, ο αλγόριθμος έχει καλή απόδοση ακόμη και όταν το σύνολο δοκιμής και το σύνολο εκπαίδευσης ανήκουν σε διαφορετικές κατηγορίες. Παρατηρούμε ότι αν η κατηγορία μεταξύ του συνόλου εκπαίδευσης είναι διαφορετική η απόδοση μειώνεται.

Συνδυάζουμε όλα τα δεδομένα που συγκεντρώσαμε γιά να τα δοκιμάσουμε με το έτοιμο corpus απο την πηγή [13]. Ετσι δημιουργήθηκαν τα :

- golpos.csv (περιέχει 112 αρχεία από ολες τις κατηγορίες)
- golneg.csv (περιέχει 73 αρχεία από ολες τις κατηγορίες)

Testset (Σύνολο Δοκιμής)	Trainingset (Σύνολο Εκπαίδευσης)		Ποσοστό επιτυχίας %	Σωστά κατηγ. απο το σύνολο
	Positive	**Negative**		
golneg.csv	**cornell_mov_pos.csv**	**cornell_mov_neg.csv**	**63%**	**46/73**
golpos.csv	**cornell_mov_pos.csv**	**cornell_mov_neg.csv**	**87,5%**	**97/112**
cornell_mov_neg.csv	**golpos.csv**	**golneg.csv**	**98.1%**	**981/1000**
cornell_mov_pos.csv	**golpos.csv**	**golneg.csv**	**18.9%**	**189/1000**

Πίνακας 3.4.2-7: Δοκιμή της συλλογής κειμένων του Cornell εναντι στη δική μας

Όπως βλέπουμε στον πίνακα 3.4.2-6 το σύνολο δοκιμής του Cornell [13] αποδίδει καλά στο δετικό σύνολο δοκιμής μας και λίγο χειρότερα στο αρνητικό. Στο επόμενο σετ πειραμάτων θα εξετάσουμε ξεχωριστά τα σύνολα δοκιμών με σύνολο εκπαίδευσης το Cornell, για να δούμε σε ποιό σύνολο αποτυγχάνει.

Testset (Σύνολο Δοκιμής)	Trainingset (Σύνολο Εκπαίδευσης)		Ποσοστό επιτυχίας %	Σωστά κατηγ. απο το σύνολο
	Positive	Negative		
pos_web.csv	cornell_mov_pos.csv	cornell_mov_neg.csv	85 %	17/20
positive.csv	cornell_mov_pos.csv	cornell_mov_neg.csv	90 %	18/20
postraining.csv	cornell_mov_pos.csv	cornell_mov_neg.csv	95 %	19/20
phealth.csv	cornell_mov_pos.csv	cornell_mov_neg.csv	63 %	19/30
pos_sentiment.csv	cornell_mov_pos.csv	cornell_mov_neg.csv	83%	46/55

Πίνακας 3.4.2-8: Ανάλυση θετικών συνόλων δοκιμών

Παρατηρούμε ότι με την χρήση του έτοιμου συνόλου δοκιμών όλα τα θετικά σύνολά μας αποδίδουν καλά. Στη συνέχεια εξετάζουμε και τα αρνητικά με τον ίδιο τρόπο και παρατηρούμε ότι 3 από τα συνολά μας έχουν πολύ χαμηλή απόδοση. Αυτό μπορεί να οφείλεται είτε στην ανομοιότητα μεταξύ του συνόλου δοκιμής και του συνόλου εκπαίδευσης, στον μή επαρκή αριθμό του αρνητικού συνόλου εκπαίδευης ή στον δύσκολο λόγο που μπορεί να έχουν τα σύνολα δοκιμης μας.

Testset (Σύνολο Δοκιμής)	Trainingset (Σύνολο Εκπαίδευσης)		Ποσοστό επιτυχίας %	Σωστά κατηγ. απο το σύνολο
	Positive	Negative		
neg_web.csv	cornell_mov_pos.csv	cornell_mov_neg.csv	25 %	5/19
negative.csv	cornell_mov_pos.csv	cornell_mov_neg.csv	85 %	17/20
negtraining.csv	cornell_mov_pos.csv	cornell_mov_neg.csv	100 %	20/20

nhealth.csv	cornell_mov_pos.csv	cornell_mov_neg.csv	26 %	8 / 30
neg_sentiment.csv	cornell_mov_pos.csv	cornell_mov_neg.csv	28%	4 /14

Πίνακας 3.4.2-9: Ανάλυση αρνητικών συνόλων δοκιμών

Δοκιμάζουμε να αλλάξουμε το αρνητικό σύνολο εκπαίδευσης γιά να υποστηρίξουμε ή να απορρίψουμε τους παραπάνω ισχυρισμούς για την ανομοιογένεια και την μη σωστή αναλογία τους. Εφόσον τό πρόβλημα μας ήταν το αρνητικό σύνολο εκπαίδευσης το αλλάζουμε με το δικό μας και κρατάμε το θετικό σύνολο εκπαίδευσης του cornell το οποίο περιέχει και κριτικές ταινιών τότε παρατηρούμε οτι όταν δοκιμάζουμε θετικά δουλευει καλά σε κατηγορία η οποία είναι ίδια με του cornell, δηλαδη κρητικές και καθόλου σε αλλες κατηγορίες.

Δηλαδή συμπεραίνουμε οτι το μοντέλο (cornell_pos gol_neg δικο μας) δουλεύει καλά σε σύνολα δοκιμών τα οποία ειναι συναφή με τη κλάση που κανει τη συνεισφορά.

Παρατηρούμε ότι ενώ τα σύνολα που πρίν δεν απέδωσαν καλα, τωρα έχουν πολύ καλυτερα ποσοστά. Αντίθετα τα θετικά σύνολα χειροτέρεψαν με αυτά τα σύνολα εκπαίδευσης. Άρα δεν μπορούμε να τα κρατήσουμε ως τελικό σύνολο εκπαίδευσης γιατί χρειαζόμαστε ένα σύστημα που να έχει πιο ισορροπημένα αποτελέσματα και να αποδίδει εξίσου καλά και στα θετικά και στα αρνητικά σύνολα δοκιμής.

Testset (Σύνολο Δοκιμής)	Trainingset (Σύνολο Εκπαίδευσης)	
	cornell_mov_pos	golneg_test
neg_web.csv	19/19	
negative.csv	1/20	
negtraining.csv	7/20	
neg_sentiment.csv	13/14	

Πίνακας 3.4.2-10: Δοκιμή θετικού συνόλου με νέο σύνολο εκπαίδευσης

Testset (Σύνολο Δοκιμής)	Trainingset (Σύνολο Εκπαίδευσης)	
	cornell_mov_pos	golneg_test
pos_web.csv	0/20	
positive.csv	16/20	
postraining.csv	19/20	
pos_sentiment.csv	0/55	

Πίνακας 3.4.2-11: Δοκιμή αρνητικού συνόλου με νέο σύνολο εκπαίδευσης

Στα παραπάνω πειράματα τα αποτελέσματα του ταξινομιτή μας δίνονται με ποσοστό ακρίβειας. Όπως αναφέραμε στο προηγούμενο κεφάλαιο ο καλύτερος τρόπος για να μετράμε την απόδοση ενός κατηγοριοποιητή είναι ως προς την ακρίβεια (recall) και την ανάκληση (precision) για να έχουμε καλύτερη εικόνα της πειραματικής διαδικασίας.

Τροποποιούμε το πρόγραμμα για να παίρνουμε τις μετρήσεις ως προς την ακρίβεια και την ανάκληση:

```
require 'csv'
require 'open-uri'
require 'fast_stemmer'
require 'classifier'
require 'sad_panda'

require_relative 'classifier_setup'

csv_filename = ARGV[0]
pos_csv = ARGV[1]
neg_csv = ARGV[2]
threshold = ARGV[3].to_f



def score_nb_class(neutral_threshold = 0.005, nb_scores_hash)
  pos_score = nb_scores_hash["Positive"].to_i.abs
  neg_score = nb_scores_hash["Negative"].to_i.abs

  test_class = "negative"
  score_avg = (neg_score + pos_score) / 2.0

  if pos_score < neg_score
    proximity = (neg_score - pos_score)
    puts "proximity #{proximity}" if $DEBUG
    if proximity <= neutral_threshold * pos_score
      test_class = "neutral"
    else
      test_class = "positive"
    end
  elsif pos_score > neg_score
    proximity = (pos_score - neg_score)
    puts "proximity #{proximity}" if $DEBUG
    if proximity <= neutral_threshold * neg_score
      test_class = "neutral"
    else
      test_class = "negative"
    end
  else
    test_class ="neutral"
  end

  return test_class , pos_score , neg_score
end


test_objects = load_class_table(csv_filename)
test_results = {}

nb_classifier = setup_classifier({"positive" => pos_csv, "negative" => neg_csv})

score1 = Proc.new {|t, t_obj, classifier|
score_nb_class(t,classifier.classifications(t_obj))}
score2 = Proc.new {|t, t_obj, classifier| classifier.classify(t_obj)}
score3 = Proc.new {|t, t_obj, classifier| score_sad_panda(t_obj)}

[score1,score2, score3].each do |scoring_method|
```

```
58      class_sum = Hash.new(0)
59      acc, tp = sentiment_classify(test_objects,nb_classifier) do |klassifier,
60    test_object , tuple, correct_count|
61
62          test_class , pos_score, neg_score = scoring_method.call(threshold,
63    test_object, klassifier)
64
65          real_class = tuple[:class]
66
67          if real_class.chomp == test_class.downcase
68            correct_count += 1
69          end
70
71          test_results[tuple[:filename]] = {}
72          test_results[tuple[:filename]][:real_class] = real_class.chomp
73          class_sum[real_class.chomp] += 1
74          test_results[tuple[:filename]][:test_class] = test_class.downcase
75          correct_count
76        end
77
78        res = Hash.new(0.0)
79        test_results.each_pair do |fname, test_result|
80          res = assess_results(res, test_result[:real_class], test_result[:test_class])
81        end
82
83        report_results(acc,res)
84    end # [score1,score2].each
```

Παρατηρούμε ότι τα αποτελέσματα κατηγοριοποίησης δεν είναι ικανοποιητικά με τα σύνολα εκπαίδευσης που δοκιμάσαμε. Ετοιμάζουμε το τελευταίο σετ πειραμάτων με τον τελικό συνδυασμό συνόλου κειμένων. Προσθέσαμε και ουδέτερα κείμενα για να δούμε αν συνεισφέρουν στην καλύτερη απόδοση του κατηγοριοποιητή μας. Ύστερα από πολλές δοκιμές καταλήξαμε ότι η αποδοτικότερη πολικότητα (threshold) είναι το 0.03.

Ο τελικός συνδυασμός του συνόλου κειμένων μας είναι:

- combined_testset.csv (αποτελείται από 32 ουδέτερα , 105 θετικά και 68 αρνητικά σύνολα κειμένων)
- combined_postraining.csv (αποτελείται από 137 θετικά σύνολα κειμένων)
- combined_negtraining.csv (αποτελείται από 99 αρνητικά σύνολα κειμένων)
- combined_negtraining_neutral.csv (αποτελείται από 99 αρνητικά και 22 ουδέτερα σύνολα κειμένων)
- combined_postraining_neutral.csv (αποτελείται από 137 θετικά και 22 ουδέτερα σύνολα κειμένων)

Ακολουθεί το τελευταίο σέτ πειραμάτων με τις εντολές που χρησιμοποιήθηκαν καθώς και τα αποτελέσματα που εξήγαγε ο αλγόριθμος Naïve Bayes, τα οποία δεν δίνοται μόνο ως προς την ποσοστιαία ολική ακρίβεια αλλά και ως προς την ακρίβεια (recall) και την ανάκληση (precision), καθώς επίσης μας δίνονται και τα λάθος θετικά (false positives) και λάθος αρνητικά (false negatives) κατηγοριοποιημένα κείμενα.

Εντολή: rescue test_neutrals.rb combined_testset.csv combined_postraining.csv combined_negtraining.csv 0.03	
Overall classifier accuracy: 0.6341463414634146 - 130/205	
POSITIVE	**NEGATIVE**
Precision: 0.7236842105263158	Precision: 0.625
Recall: 0.7051282051282052	Recall: 0.9183673469387755
false positives positive 21.0	false positives negative 27.0
false negatives positive 23.0	false negatives negative 4.0

Πίνακας 3.4.2-12: Πρώτο πείραμα με αναλυτικά αποτελέσματα ως προς την ακρίβεια και την ανάκληση

Εντολή: rescue test_neutrals.rb combined_testset.csv combined_postraining_neutral.csv combined_negtraining.csv 0.03	
Overall classifier accuracy: 0.624390243902439 - 128/205	
POSITIVE	**NEGATIVE**
Precision: 0.6746987951807228	Precision: 0.6461538461538462
Recall: 0.717948717948718	Recall: 0.8571428571428571
false positives positive 27.0	false positives negative 23.0
false negatives positive 22.0	false negatives negative 7.0

Πίνακας 3.4.2-13: Δεύτερο πείραμα με αναλυτικά αποτελέσματα ως προς την ακρίβεια και την ανάκληση

Εντολή: rescue test_neutrals.rb combined_testset.csv combined_postraining.csv combined_negtraining_neutral.csv 0.03	
Overall classifier accuracy: 0.5853658536585366 - 120/205	
POSITIVE	**NEGATIVE**
Precision: 0.7454545454545455	Precision: 0.5053763440860215

Recall: 0.5256410256410257	Recall: 0.9591836734693877
false positives positive 14.0	false positives negative 46.0
false negatives positive 37.0	false negatives negative 2.0

Πίνακας 3.4.2-14: Τρίτο πείραμα με αναλυτικά αποτελέσματα ως προς την ακρίβεια και την ανάκληση

Εντολή: rescue test_neutrals.rb combined_testset.csv combined_postraining_neutral.csv combined_negtraining_neutral.csv 0.03	
Overall classifier accuracy: 0.6048780487804878 - 124/205	
POSITIVE	**NEGATIVE**
Precision: 0.7454545454545455	Precision: 0.5053763440860215
Recall: 0.5256410256410257	Recall: 0.9591836734693877
false positives positive 14.0	false positives negative 46.0
false negatives positive 37.0	false negatives negative 2.0

Πίνακας 3.4.2-15: Τέταρτο πείραμα με αναλυτικά αποτελέσματα ως προς την ακρίβεια και την ανάκληση

Εντολή: rescue test_neutrals.rb combined_testset.csv combined_postraining_neutral.csv combined_negtraining_neutral.csv 0.03	
Overall classifier accuracy: 0.6048780487804878 - 124/205	
POSITIVE	**NEGATIVE**
Precision: 0.7101449275362319	Precision: 0.569620253164557
Recall: 0.6282051282051282	Recall: 0.9183673469387755
false positives positive 20.0	false positives negative 34.0
false negatives positive 29.0	false negatives negative 4.0

Πίνακας 3.4.2-16: Πέμπτο πείραμα με αναλυτικά αποτελέσματα ως προς την ακρίβεια και την ανάκληση

3.5 Προβλήματα που αντιμετωπίστηκαν - Συμπεράσματα

Μετά το πέρας της πειραματικής διαδικασίας στις βιβλιοθήκες οι οποίες βασίζονται στα λεξικά παρατηρήσαμε οτι για να μπορέσει ένα λεξικό να είναι ολοκληρωμένο και να περιέχει την πλειονότητα των λέξεων, απαιτούνται πολύ μεγάλες συλλογές δεδομένων. Για να μπορέσουμε να επιτύχουμε καλύτερα αποτελέσματα, πρέπει να αλλάξουμε την προεπιλεγμένη πολικότητα η οποία απλά μπορεί να βοηθήσει στην κατηγοριοποίηση και των μη συναισθηματικά φορτισμένων κειμένων (ουδέτερων) αλλά και να εμπλουτίσουμε τα λεξικά με μεγαλύτερο όγκο δεδομένων. Παρατηρείται όμως το φαινόμενο, λεξικά που έχουν στηριχθεί σε λέξεις ενός πεδίου να μην έχουν αξιόλογα αποτελέσματα όταν χρησιμοποιούνται σε διαφορετικό πεδίο εφαρμογής. Αυτό φαίνεται και στην δεύτερη βιβλιοθήκη που χρησιμοποιήσαμε (Sentiment Lib) όπου έχει δύο λεξικά τα οποία απαρτίζονται απο διαφορετικές κατηγορίες λέξεων. Επειδή υποστηρίζεται ότι ο σημασιολογικός προσδιορισμός που προέρχεται από κάποιο λεξικό δεν μπορεί να είναι παρά μόνο μια ένδειξη προτιμήσαμε να καταφύγουμε στην χρήση ενός Naïve Bayes κατηγοριοποιητή όπου τα αποτελέσματα που θα εξάγει εξαρτώνται από τα σύνολα δεδομένων που εμείς θα εισάγουμε.

Ένα από τα βασικότερα προβλήματα που είχαμε να αντιμετωπίσουμε ήταν η μεγάλη συλλογή δεδομένων αφού αυτή η τεχνική βασίζεται στις μη επιβλεπόμενες τεχνικές μάθησης. Τα δεδομένα που συλλέξαμε προορίζονται το μεγαλύτερο μέρος τους για εκπαίδευση και το υπόλοιπο για δοκιμή. Το σύνολο δοκιμής που αποτελεί το 20-30% του συνόλου δεδομένων δεν αποτελεί πρόβλημα διότι ο αλγόριθμος πρέπει να εκπαιδευτεί με τον καταλληλότερο τρόπο για να επιτυγχάνει την καλύτερη κατηγοριοποίηση. Κατά την εκπαίδευση του αλγορίθμου παρατηρήσαμε ότι το αρνητικό σύνολο εκπαίδευσης πρέπει να έιναι μικρότερο απο αυτό του θετικού συνόλου για την επίτευξη του καλύτερου δυνατού αποτελέσματος. Μια αλλη σημαντική παρατήρηση είναι ότι στα σύνολα εκπαίδευσης πρέπει να περιέχονται κείμενα από διάφορους τομείς για να μην καταλήξουμε να έχουμε καλά αποτελέσματα μόνο σε έναν συγκεκριμένο τομέα (domain specific). Σε γενικές γραμμές η κατηγοριοποίηση με επίβλεψη των κειμένων πρέπει να γίνεται προσεκτικά για να μην οδηγήσουμε τον αλγόριθμο σε εσφαλμένα αποτελέσματα.

Κατά την πειραματική διαδικασία είδαμε ότι η ακρίβεια δεν ήταν αρκετό μέτρο σύγκρισης, για αυτό τροποποιήσαμε τον κώδικα εκτέλεσης του αλγορίθμου έτσι ώστε να μας εξάγει τα αποτελέσματα σε ακρίβεια και ανάκλήση καθώς και την εμφάνιση των λάθος θετικών και λάθος αρνητικών κατηγοριοποιημένων κειμένων, για να έχουμε καλύτερη γενική εικόνα των αποτελεσμάτων.

Για τόν λόγο οτι μπορούμε να επέμβουμε στην πειραματική διαδικασία και τα αποτελέσματα της είναι πιο αξιόπιστα, ο αλγόριθμος ταξινόμησης που επιλέχθηκε για το χτίσιμο της εφαρμογής ήταν ο Naïve Bayes με σκοπό να υπολογιστεί η υπο συνθήκη πιθανότητα ένα μήνυμα d να είναι μέλος μίας κλάσης c, όταν οι πιθανές κλάσεις ήταν οι «αρνητικό» και «θετικό».

4 *ΚΕΦΑΛΑΙΟ :* Γενική Αρχιτεκτονική

4.1 Έρευνα Τεχνολογιών

4.1.1 Ruby [1]

Είναι γλώσσα ανοιχτού κώδικα και σαν χαρακτηριστικό της έχει την απλότητα στην κατανόηση και στη συγγραφή κώδικα. Ο δημιουργός της, Yukihiro Matsumoto, χρησιμοποίησε κομμάτια από τις αγαπημένες του γλώσσες (Perl, Smalltalk, Eiffel, Ada και Lisp) και δημοσίευσε την πρώτη έκδοση της Ruby το 1995. Το 1999 δημιουργήθηκε τεκμηρίωση (documentation) στα αγγλικά, κάτι που βοήθησε στην εξάπλωση της. Ο βασικός λόγος της σημερινής της

Εικόνα 4.1.1-1: Λογότυπο της Ruby

επιτυχίας στο διαδίκτυο, είναι η Ruby on Rails. Αυτή τη στιγμή βρίσκεται στην έκδοση 2.1.2 και αναπτύσσεται με γρήγορους ρυθμούς.

Η Ruby είναι μια γλώσσα επηρεασμένη από την Lisp και την Smalltalk, η οποία σύμφωνα με τον δημιουργό της, φτιάχτηκε για να «κάνει τους προγραμματιστές χαρούμενους». Αυτό φαίνεται από τους λίγους περιορισμούς που επιβάλλει στον τρόπο γραφής της και το απλό συντακτικό της με το οποίο όμως υπάρχει η δυνατότητα να δημιουργηθούν πολύ μεγάλα και περίπλοκα προγράμματα.

Τα πάντα στην Ruby θεωρούνται αντικείμενα, κάτι που την καθιστά από τις πιο αντικειμενοστραφείς γλώσσες που υπάρχουν. Ακόμα και οι αριθμοί ή οι ίδιες οι κλάσεις, αποτελούν αντικείμενα. Επίσης λόγω της επιρροής από την Lisp, έχει πολλά στοιχεία αναδρομικών (functional) γλωσσών. Αυτό, εξυπηρετεί στο καθαρό της συντακτικό και στην εύκολη συντήρηση του κώδικα, καθώς δίνεται η δυνατότητα να γράφονται συμπαγή και κατανοητά προγράμματα.

Είναι μια διερμηνευτική (interpreted) γλώσσα, γεγονός που την καθιστά ιδιαίτερα ευέλικτη. Ένα παράδειγμα της ευελιξίας της είναι οι ανοιχτές κλάσεις, δηλαδή η προσθήκη μεθόδων σε υπάρχουσες κλάσεις την ώρα εκτέλεσης ενός προγράμματος. Αυτή η δυνατότητα, θεωρείται από κάποιους αδυναμία γιατί ενδέχεται να γίνει μη ηθελημένη αντικατάσταση κάποιας μεθόδου.

Επίσης, ένα ακόμα από τα πιο γνωστά της στοιχεία, είναι η δυνατότητα «μετα-προγραμματισμού» ή αλλιώς να γράφεται κώδικας που γράφει κώδικα.

4.1.2 Ruby On Rails [2]

Η Ruby on Rails είναι ένα από τα πρώτα frameworks ιστού τα οποία ακολούθησαν την αρχιτεκτονική Μοντέλο-Προβολή-Ελεγκτής (MVC - Model-View-Controller). Δημιουργήθηκε το 2004, από την 37signals και τον David Heinemeier Hanson για την ανάπτυξη των εφαρμογών ιστού τους, οι οποίες πλέον χρησιμοποιούνται από εκατομμύρια χρήστες και σχεδόν από όλες τις μεγάλες εταιρείες (TOP 500) που αναφέρονται σε αναγνωρισμένα περιοδικά όπως το Forbes.

Εικόνα 4.1.2-1: Λογότυπο της Ruby On Rails

Η Ruby on Rails βασίστηκε εξαρχής πάνω σε ισχυρές αρχές και θεμέλια τα οποία ακολουθούνται πιστά από όσους βοηθάνε στην ανάπτυξη της. Μερικές αρχές είναι:

- Σύμβαση αντί για παραμετροποίηση (convention over configuration)
- Ευέλικτες τεχνικές υλοποίησης (agile)
- Ανάπτυξη εφαρμογής οδηγούμενη από δοκιμές (tests-driven development)

4.1.2.1 Σύμβαση αντί για παραμετροποίηση (Convention over Configuration)

Η Rails ακολουθώντας τις βέλτιστες πρακτικές και κατευθυντήριες αρχές, "υποθέτει" το τί προτίθεται να κάνει ο προγραμματιστής και τον τρόπο πραγματοποίησής του. Οι προεπιλογές αυτές εξοικονομούν αρκετό χρόνο και δεν περιορίζουν τον προγραμματιστή, αφού μπορούν να παρακαμφθούν.

4.1.2.2 "Μην επαναλαμβάνεσαι" (DRY - Don't repeat yourself)

Στόχος είναι να μένει ο κώδικας σύντομος και εύκολα αναγνώσιμο. Η αρχή "μην επαναλαμβάνεσαι" υπονοεί ότι κάθε κομμάτι κώδικα πρέπει να εκφράζεται σε μόνο ένα σημείο της εφαρμογής.

4.1.2.3 Αντιπροσωπευτική Μεταβίβαση Καταστάσεων (REST – Representational State Transfer)

Είναι μια αρχιτεκτονική λογισμικού για κατανεμημένα συστήματα όπως το Διαδίκτυο. Η αρχιτεκτονική αντιπροσωπευτικής μεταβίβασης καταστάσεων καθορίζει πώς πρέπει να χρησιμοποιούνται τα πρότυπα του παγκόσμιου ιστού, όπως το HTTP και τα URIs.

Οι βασικές αρχές είναι:

- **Όλοι οι πόροι να έχουν ταυτότητα**: αντίστοιχα με την αντικειμενοστραφή λογική, η λογική προσανατολισμένη στους πόρους (resource-oriented) προϋποθέτει ότι κάθε πόρος χρειάζεται μια ταυτότητα για να κληθεί. Πόροι εκτός από τις ανεξάρτητες οντότητες μπορεί να είναι και

ένα σύνολο αντικειμένων ή αποτελέσματα υπολογισμών.

- Οι εφαρμογές που ακολουθούν την REST αρχιτεκτονική δεν αποθηκεύουν παραμέτρους, είναι δηλαδή **stateless**. Είτε κάποιος πελάτης (client) έχει στείλει στοιχεία σε προηγούμενη αίτηση είτε στέλνει πρώτη φορά αίτηση, θα έχει την ίδια μεταχείριση από τον εξυπηρετητή. Ο εξυπηρετητής απαντάει σε αιτήσεις κι όχι σε πελάτες. Κάθε αίτηση προς τον εξυπηρετητή αντιμετωπίζεται ανεξάρτητα και μεμονωμένα. Γι αυτό το λόγο οι απαραίτητες παράμετροι θα πρέπει να στέλνονται σε κάθε HTTP αίτηση.
- **Υπερμέσα ως κινητήρια δύναμη της κατάστασης της εφαρμογής** (HATEOAS - hypermedia as the engine of application state): Χρήση συνδέσμων και δυνατότητα σύνδεσης πόρων που δεν ανήκουν απαραίτητα στην ίδια εφαρμογή.
- **Πόροι με πολλαπλές αναπαραστάσεις**: Είναι επιθυμητό για μια εφαρμογή, να έχει τη δυνατότητα απάντησης στις αιτήσεις με πολλαπλές αναπαραστάσεις. Αν παρέχεται HTML και XML αναπαράσταση ενός πόρου, είναι δυνατό να χρησιμοποιηθεί από φυλλομετρητές ιστού όπως επισής και από άλλες εφαρμογές. Για παράδειγμα, μια είδηση με δυνατότητα αναπαράστασης σε HTML και XML (HTML έχει σκοπό την εμφάνιση δεδομένων – XML έχει σκοπό τη μεταφορά δεδομένων), θα εμφανιστεί σε ένα φυλλομετρητή ιστού με HTML αναπαράσταση και σε ένα πρόγραμμα συγχρονισμού και ανάγνωσης ειδήσεων (RSS feeds reader) με XML αναπαράσταση. **Error! Reference source not found.**

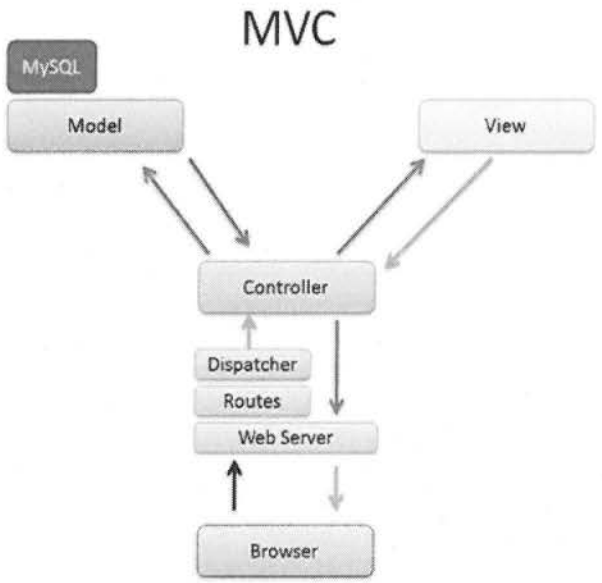
Στην Rails, τα ρήματα του HTTP πρωτοκόλλου (GET, POST κλπ) σχετίζονται άμεσα με τα URLs και τις

HTTP μέθοδοι	Διαδρομή	Λειτουργία βάσης	Τί κάνει
GET	/products	Read	Εμφανίζει μια λίστας προϊόντων
GET	/products/new	-	Επιστρέφει μια HTML φόρμα για τη δημιουργία νέου προϊόντος
POST	/products	Create	Δημιουργεί νέο προϊόν
GET	/products/:id	Read	Εμφανίζει συγκεκριμμένο προϊόν
GET	/products/:id/edit	-	Επιστρέφει μια HTML φόρμα για τη επεξεργασία συγκεκριμμένου προϊόντος
PUT	/products/:id	Update	Ενημερώνει τη βάση για τις αλλαγές στα δεδομένα συγκεκριμμένου προϊόντος
DELETE	/products/:id	Delete	Διαγράφει συγκεκριμμένο προϊόν

Πίνακας 4.1.2-1: Αντιστοίχιση HTTP μεθόδων, Διαδρομών (paths), Λειτουργιών βάσης

4.1.2.4 Μοντέλο – Προβολή – Ελεγκτής

Η Rails έχει φτιαχτεί γύρω από το μοντέλο Μοντέλο – Προβολή – Ελεγκτής (MVC – Model – View – Controller).



Εικόνα 4.1.2-2: Αναπαράσταση MVC αρχιτεκτονικής

Είναι μια απλή αρχή σύμφωνα με την οποία χωρίζονται τα δεδομένα από τη λογική και την προβολή του προγράμματος με αποτέλεσμα να διατηρείται η εφαρμογή καλά οργανωμένη.

Ο κώδικας που ελέγχει και χειρίζεται τα δεδομένα ή ορίζει τη λογική βρίσκεται στα μοντέλα. Τα μοντέλα είναι κλάσεις που επικοινωνούν με την βάση δεδομένων. Γράφοντας κώδικα Ruby και πολύ σπάνια SQL δίνεται η δυνατότητα για αναζήτηση, τροποποίηση, δημιουργία ή διαγραφή εγγραφών από τη βάση δεδομένων.

Οι ελεγκτές απαντούν στις αιτήσεις των χρηστών. Παίρνουν τα δεδομένα εισόδου, "αποφασίζουν" πώς θα τα διαχειριστούν, καλούν μεθόδους, αλληλεπιδρούν με τα μοντέλα και προωθούν τα δεδομένα εξόδου στις προβολές.

Οι προβολές εμφανίζουν τα δεδομένα εξόδου συνήθως σε HTML, ενώ η Rails κάνει επίσης απλή την δημιουργία προβολών σε κώδικα XML ή JSON.

4.1.2.5 Δομή φακέλων κάθε εφαρμογής Ruby on Rails

Κάθε εφαρμογή Ruby on Rails από τη δημιουργία της είναι οργανωμένη σε φακέλους. Οι εικόνες, το σχήμα της βάσης δεδομένων, ο κώδικας HTML, ο κώδικας CSS και κάθε άλλο κομμάτι της εφαρμογής, έχουν τη δική τους θέση. Το πλεονέκτημα αυτό συμβάλει στην εύκολη συντήρηση και επέκταση του κώδικα. Στην παρακάτω εικόνα φαίνεται η δομή των φακέλων και ακολουθεί αναλυτικότερη παρουσίαση για κάθε φάκελο.

`app/` Αυτός είναι ο φάκελος που χρησιμοποιείται περισσότερο από τον προγραμματιστή. Περιέχει τον κυρίως κώδικα της εφαρμογής (μοντέλο-προβολή-ελεγκτής) και θα αναλυθεί εκτενέστερα στην επόμενη παράγραφό με τίτλο "Μοντέλο-Προβολή-Ελεγκτής".

`config/` Η Ruby on Rails, όπως αναφέρθηκε προηγούμενα, χρησιμοποιεί την αρχή της σύμβασης αντί παραμετροποίησης. Οι επιπλέον ρυθμίσεις που απαιτούνται για την κάλυψη των αναγκών της εφαρμογής ή οι ρυθμίσεις που παρακάμπτουν τις συμβάσεις αποθηκεύονται στον φάκελο config.

config.ru Αρχείο για την παραμετροποίηση της διεπαφής με τον Rack εξυπηρετητή.

db/ Σχήμα βάσης δεδομένων και πληροφορίες για τα migrations.

doc/ Σ' αυτό το φάκελο βρίσκεται η τεκμηρίωση που δημιουργείται αυτόματα με την εντολή doc.

Gemfile Εδώ καθορίζονται τα gems από τα οποία εξαρτάται η εφαρμογή. Τα gems είναι πακέτα που περιέχουν προγράμματα ruby και βιβλιοθήκες.

lib/ Ο φάκελος αυτός φιλοξενεί κώδικα που είτε δεν ανήκει σε κανένα ή χρησιμοποιείται από περισσότερα από ένα εκ των μοντέλο, προβολή, ελεγκτής (Model-View-Controller).

log/ Όσο "τρέχει" μια Rails εφαρμογή δημιουργούνται καταγραφές (logs). Υπάρχουν τρεις φάκελοι για την ανάπτυξη, τις δοκιμές και την κατάσταση που η εφαρμογή έχει δημοσιευτεί. Καταγράφονται στοιχεία που αφορούν ερωτήματα προς τη βάση δεδομένων, δεδομένα της κρυφής μνήμης (cache) κ.α.

public/ Ο φάκελος αυτός είναι προσπελάσιμος από το διαδίκτυο και θεωρείται από τον εξυπηρετητή ο βασικός φάκελος της εφαρμογής. Εδώ, βρίσκονται στατικές σελίδες.

Rakefile Εδώ ορίζονται εργασίες που μπορεί να εκτελούν δοκιμές (tests), να δημιουργούν τεκμηρίωση, να εκτυπώσουν το σχήμα της βάσης κ.α.

README Οδηγίες εγκατάστασης και χρήσης.

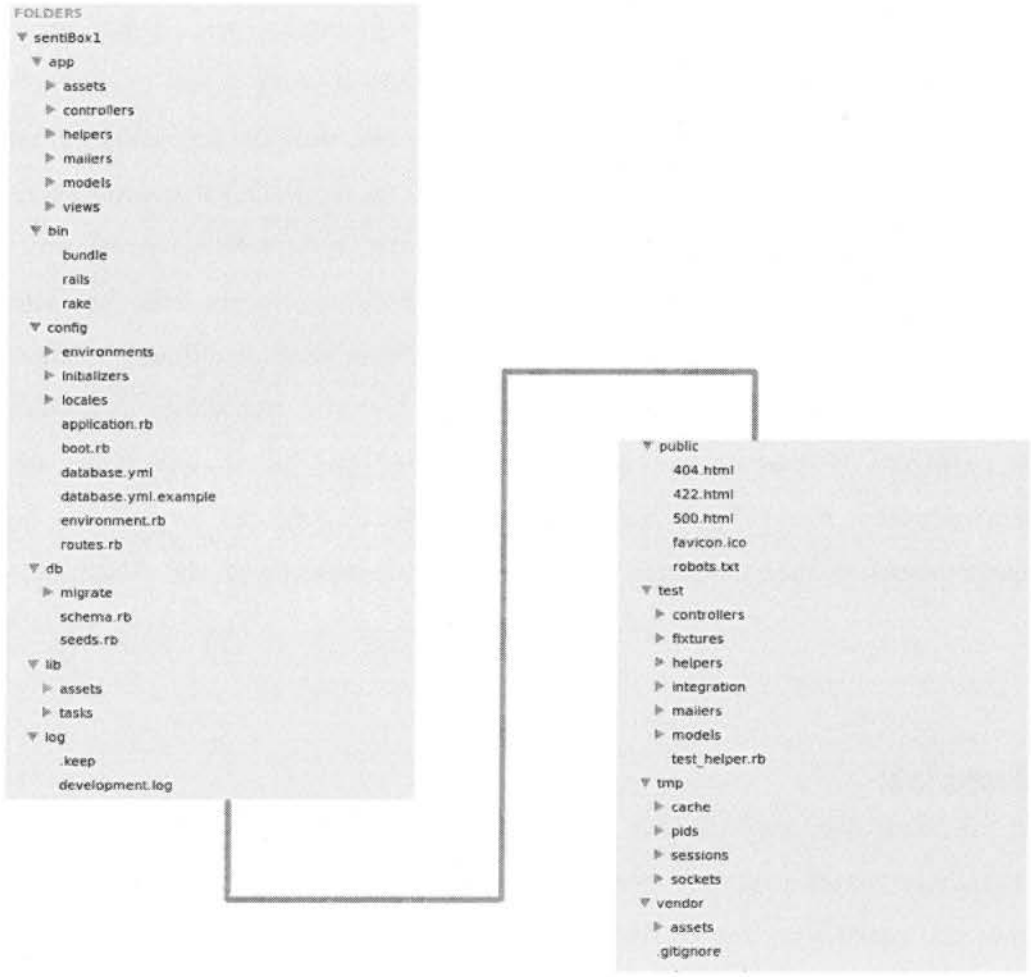
script/ Εδώ βρίσκονται τα σενάρια (scripts) της Rails, τα οποία "τρέχουν" αν στη γραμμή εντολών γράψουμε την εντολή rails και το όνομα της συνάρτηση που καλούμε, δηλαδή *rails console*, *rails generate*, *rails new* κ.λ.π. Επίσης, σ' αυτό το φάκελο αποθηκεύονται και τα σενάρια που γράφει ο προγραμματιστής.

test/ Η Rails προσφέρει ευρύ φάσμα εργαλείων για την δημιουργία και την εκτέλεση δοκιμών (tests). Σ' αυτό το φάκελο βρίσκεται ό,τι σχετίζεται με δοκιμές, όπως δοκιμές ενσωμάτωσης (integration tests), δοκιμές λειτουργικότητας (functional tests), δοκιμές επανάληψης (iteration tests).

tmp/ Ένας φάκελος για τα προσωρινά αρχεία, όπως τα περιεχόμενα της κρυφής μνήμης.

vendor/ Όλες σχεδόν οι εφαρμογές χρησιμοποιούν έτοιμο κώδικα ο οποίος προσθέτει

λειτουργικότητα. Ο πιο διαδεδομένος τρόπος στην Rails είναι τα gems. Παρόλ' αυτά αν κρίνεται σκόπιμο ο κώδικας αυτός μπορεί να αποθηκευτεί στο φάκελο vendor/plugin.



Εικόνα 4.1.2-3: Δομή φακέλων των εφαρμογών Ruby on Rails

4.1.2.6 Rack Error! Reference source not found.

Το μεσισμικό rack είναι ένας τρόπος ώστε να φιλτράρονται οι HTTP αιτήσεις κι απαντήσεις που

λαμβάνει μία εφαρμογή φτιαγμένη σε Ruby ή σε κάποιο πλαίσιο της Ruby όπως είναι η Ruby on Rails. Σκοπός είναι να προσφέρει μία πολύ απλή διεπαφή για την σύνδεση εξυπηρετητών ιστού και πλαισίων ανάπτυξης εφαρμογών ιστού (web frameworks).

4.1.2.7 Υποστήριξη διαφορετικών περιβαλλόντων

Συχνά οι προγραμματιστές συναντούν προβλήματα στο στάδιο δημοσίευσης της εφαρμογής τους λόγω των παραμετροποιήσεων που απαιτούνται για την μετάβαση αυτή. Κατά την ανάπτυξη συνήθως η εφαρμογή χρησιμοποιεί μία τοπική βάση δεδομένων ενώ κατά την δημοσίευση πολλές φορές η βάση φιλοξενείται σε άλλο εξυπηρετητή. Επίσης, κατά τη διάρκεια της ανάπτυξης είναι επιθυμητό να βλέπουμε τα σφάλματα που προκύπτουν, ενώ κατά τη δημοσίευση προτιμάται να κρύβονται από τους χρήστες όσο το δυνατό περισσότερα προβλήματα. Ακόμα, όταν διεξάγονται έλεγχοι γράφονται, τροποποιούνται ή διαγράφονται δεδομένα από τη βάση με πιθανούς κινδύνους απώλειας ή αλλαγής εγγραφών. Η παραμετροποίηση για όλες τις παραπάνω περιπτώσεις δυσχεραίνει την μετάβαση. Η Ruby on Rails λύνει αυτό το πρόβλημα με την υποστήριξη τριών ανεξάρτητων περιβαλλόντων: ανάπτυξης, ελέγχων και παραγωγής. Οι ρυθμίσεις γίνονται μια φορά στην αρχική παραμετροποίηση και η μετάβαση από στάδιο σε στάδιο γίνεται με την αλλαγή μεταξύ περιβαλλόντων.

4.1.2.8 Migrations [12]

Η αρχιτεκτονική της βάσης δεδομένων αλλάζει πολλές φορές κατά τη διάρκεια της ανάπτυξης. Για παράδειγμα, μπορεί να προστεθεί κάποιος πίνακας ή να διαγραφεί μια στήλη. Ιδιαίτερα οι ομάδες προγραμματιστών που εργάζονται ταυτόχρονα για κάποια εφαρμογή έρχονται αντιμέτωποι με αλλαγές στη βάση που έχουν γίνει από τους ίδιους ή από συναδέλφους. Προκύπτει λοιπόν η ανάγκη παρακολούθησης των αλλαγών αυτών. Η Rails δίνει λύση με τα migrations τα οποία περιγράφουν, με τη μορφή κώδικα, τις αλλαγές που γίνονται στη βάση δεδομένων και τις εκτελεί "τρέχοντας" μόνο μία γραμμή κώδικα.

Τα migrations είναι αρχεία κώδικα στο φάκελο db/migrate. Κάθε migration έχει τυπικά στο όνομά του δεκατέσσερα ψηφία που αφορούν την ημερομηνία και ώρα δημιουργίας του. Τα ψηφία αυτά μπορεί

να ακολουθούνται από "_" και κάποια ακολουθία χαρακτήρων όπως "add_order_id_to_line_item". Αποτελούν τη ταυτότητα του migration στην οποία αναφερόμαστε όταν το εκτελούμε από τη γραμμή εντολών: `rake db:migrate VERSION=20120529160455`. Ο κώδικας των migration συντηρεί στη βάση κάθε Rails εφαρμογής τον πίνακα schema_migration. Ο πίνακας αυτό διαθέτει μια μοναδική στήλη με όνομα version. Κάθε γραμμή της στήλης version περιέχει το όνομα ενός επιτυχώς εκτελεσμένου migration. Με την εντολή `rake db:migrate` αν δεν υπάρχει ο πίνακας schema_migration η Rails τον δημιουργεί και εκτελεί όλα τα διαθέσιμα migrations. Αν ο πίνακας υπάρχει εξετάζεται ποιά migrations δεν υπάρχουν και εκτελούνται. Τέλος, αν χρειάζεται να εκτελεστεί ξανά κάποιο migration δεν αρκεί η εντολή `rake db:migrate`, η οποία σε αυτή την περίπτωση δεν θα αλλάξει τίποτα, κι έτσι πρέπει να δηλώσουμε την έκδοση, `rake db:migrate VERSION=20120529160455`.

4.1.2.9 Scaffold [5]: [6]:

Έστω η περίπτωση δημιουργίας μιας εφαρμογής με προϊόντα. Θα χρειαστεί ένας πίνακας "προϊόντα" στη βάση δεδομένων, ένα μοντέλο "προϊόν", ένα σύνολο από προβολές για την καταχώρηση, ανάγνωση, τροποποίηση, διαγραφή προϊόντων και τέλος ένας ελεγκτής που θα συντονίζει τα προηγούμενα. Όλα αυτά μπορούν να γίνουν με μια εντολή γνωστή ως scaffold: `rails scaffold Product title:string description:text price:decimal`. Η εντολή scaffold συνεπώς, προσθέτει λειτουργικότητα στην εφαρμογή με απλό και γρήγορο τρόπο.

4.1.2.10 Active Record

Το Active Record είναι μια βιβλιοθήκη ORM της Ruby που επιτρέπει τη μεταφορά δεδομένων και συνεργάζεται με την υπόλοιπη εφαρμογή εξυπηρετώντας την καταχώρηση δεδομένων στη βάση, η οποία συνήθως είναι σχεσιακή. Το Active Record περιλαμβάνεται στη Ruby on Rails αλλά επίσης είναι διαθέσιμο και σαν Ruby Gem. Ακολουθώντας την αρχή των συμβάσεων αντί πραμετροποίησης ελαχιστοποιεί τις ρυθμίσεις που χρειάζεται να γίνουν, κάτι που δεν συναντάται συχνά στις υπόλοιπες ORM βιβλιοθήκες, όπως η Hibernate της Java.

Μερικές από τις λειτουργίες του είναι:

- Ελέγχει ποια migrations έχουν γίνει και εκτελεί τα κατάλληλα όταν εισάγουμε την εντολή `rake db:migrate`. Ενώ είναι αυτό που προσφέρει τη δυνατότητα αναίρεσης ή αλλαγής κάποιου migration (`rake db:rollback`, `rake db:migrate:up VERSION=20120522140000`)
- Παρέχει μεθόδους για εργασίες που επαναλαμβάνονται συχνά και αφορούν αλλαγές στη μορφή της βάσης (`add_column`, `add_index`, `change_column`, `change_table`, `create_table`, `drop_table`, `remove_column`, `remove_index`, `rename_column`)
- Παρέχει αρκετές μεθόδους αναζήτησης χώρις τη συγγραφή SQL ερωτημάτων. Μερικές μέθοδοι είναι οι `where`, `select`, `group`, `order`, `reorder`, `reverse_order`, `limit`, `offset`, `joins`, `includes`, `having` κλπ.
- Εκτελεί ερωτήματα προς τη βάση και είναι συμβατό με τις περισσότερες βάσεις δεδομένων (MySQL, PostgreSQL, SQLite κλπ).
- Ελέγχει τις επικυρώσεις (validations) και συμβάλλει στην εμφάνιση των σφαλμάτων στην κατάλληλη προβολή. Αν υπάρχουν σφάλματα δεν καταχωρεί ή ενημερώνει την εγγραφή στη βάση.
- Δύο μοντέλα μπορούν να συσχετίζονται με διάφορους τρόπους. Η Rails υποστηρίζει έξι: `belongs_to`, `has_one`, `has_many`, `has_many :through`, `has_one :through`, `has_and_belong_to_many`. Σύμφωνα με τους συσχετισμούς (associations) το Active Record διαμορφώνει ανάλογα το σχήμα της βάσης.
- Διαθέτει βοηθούς (helpers) οι οποίοι παρέχουν κάποιες ακόμα λειτουργίες όπως είναι η προσθήκη του timestamp, δηλαδή της χρονικής στιγμής (ημερομηνία και ώρα με ακρίβεια δευτερολέπτου), σε κάθε νέα καταχώρηση εγγραφής και σε κάθε τελευταία τροποποίηση. Προσθέτουν αυτόματα το πρωτεύον κλειδί (id) .

Το Active Record υποστηρίζει τους εξής τύπους δεδομένων για τη βάση: `binary`, `boolean`, `date`, `datetime`, `decimal`, `float`, `integer`, `primary_key`, `string`, `text`, `time`, `timestamp`.

Οι παραπάνω τύποι αντιστοιχίζονται από το Active Record με τύπους της βάσης δεδομένων. Για

παράδειγμα, ο τύπος `:string` αντιστοιχίζεται στον τύπο `VARCHAR(255)` της MySQL.

4.1.3 MySQL [3]

Η βάση δεδομένων MySQL είναι μια απ' τις πιο δημοφιλής βάσεις δεδομένων. Είναι ανοιχτού κώδικα, σχεσιακή, αξιόπιστη, εύκολη στη χρήση, υψηλών επιδόσεων συμβατή με τις περισσότερες πλατφόρμες όπως τα Linux, Max OS, Windows κ.α.

4.1.3.1 Σχεσιακή βάση δεδομένων (Relational database)

Οι σχεσιακές βάσεις δεδομένων αποθηκεύουν τα δεδομένα σε συσχετισμένους πίνακες όπου κάθε γραμμή ενός πίνακα αναπαριστά ένα αντικείμενο και κάθε στήλη αναπαριστά μία ιδιότητα του αντικειμένου. Κάθε αντικείμενο που αναπαρίσταται σε μια σχεσιακή βάση δεδομένων είναι στιγμιότυπο κάποιου μοντέλου. Επίσης, σε κάθε πίνακα υπάρχει μια στήλη το περιεχόμενο της οποίας είναι μοναδικό για κάθε γραμμή και ονομάζεται πρωτεύον κλειδί (primary key). Σύμφωνα με τις συμβάσεις της Rails, το πρωτεύων κλειδί είναι ακέραιος αριθμός και ονομάζεται id. Το σύνολο των πινάκων και η δομή τους αποθηκεύονται στο σχήμα της βάσης (database schema).

4.1.3.2 Αντικείμενο-σχεσιακή χαρτογράφηση (ORM – Object Relational Mapping)

Οι βιβλιοθήκες αντικειμενοσχεσιακής χαρτογράφησης είναι εργαλεία που επιτρέπουν την εύκολη και αυτοματοποιημένη αποθήκευση εγγραφών σε μια σχεσιακή βάση δεδομένων. Η λειτουργία τους είναι να κάνουν την αντιστοίχιση πινάκων με κλάσεις, γραμμών με αντικείμενα και στηλών με ορίσματα. Οι μέθοδοι των κλάσεων (class methods) επιδρούν σε πίνακες ενώ οι μέθοδοι των στιγμιοτύπων (instance methods) επιδρούν σε κάποια γραμμή του πίνακα.

Αποτέλεσμα της χρήσης ORM σε μια εφαρμογή είναι η μείωση του χρόνου ανάπτυξης του λογισμικού, του κόστους ανάπτυξης και συντήρησης, η σύνταξη απλούστερου και λιγότερου κώδικα αντί πολύπλοκων SQL ερωτημάτων. Στον αντίποδα, τα ORM εργαλεία δεν προτιμώνται στην ανάπτυξη πολύ μικρών εφαρμογών, οι οποίες μπορούν γρηγορότερα να υλοποιηθούν με απλά SQL ερωτήματα.

4.1.4 Git[7]

Μια εφαρμογή κατά την ανάπτυξή της χωρίζεται από τον προγραμματιστή ή την ομάδα ανάπτυξης της σε στάδια – κατηγορίες.

Εικόνα 4.1.3-1: Λογότυπο Git

Το git είναι ένα σύστημα ελέγχου και διαχείρισης εκδόσεων λογισμικού και δεδομένων που καταγράφει τις αλλαγές σε ένα αρχείο ή σύνολο αρχείων στην πάροδο του χρόνου, έτσι ώστε οι προγραμματιστές να μπορούν να ανακαλέσει τις συγκεκριμένες εκδόσεις αργότερα. Από τη γέννησή του το 2005, το Git έχει εξελιχθεί, για να είναι εύκολο στη χρήση αλλά και να διατηρεί τις αρχικές του ιδιότητες. Είναι απίστευτα γρήγορο, πολύ αποτελεσματικό με τα μεγάλα έργα, και έχει ένα σύστημα διακλάδωσης για μη-γραμμική ανάπτυξη και χρησιμοποιείται από μεγάλες εταιρείες και μεγάλα έργα όπως οι Google, Microsoft, gnome, linux, twitter, facebook, LinkedIn, perl, PostegreSQL, android, eclipse, ruby on rails.

Η εγκατάσταση είναι εύκολη σε κάθε σύστημα εκτελώντας:

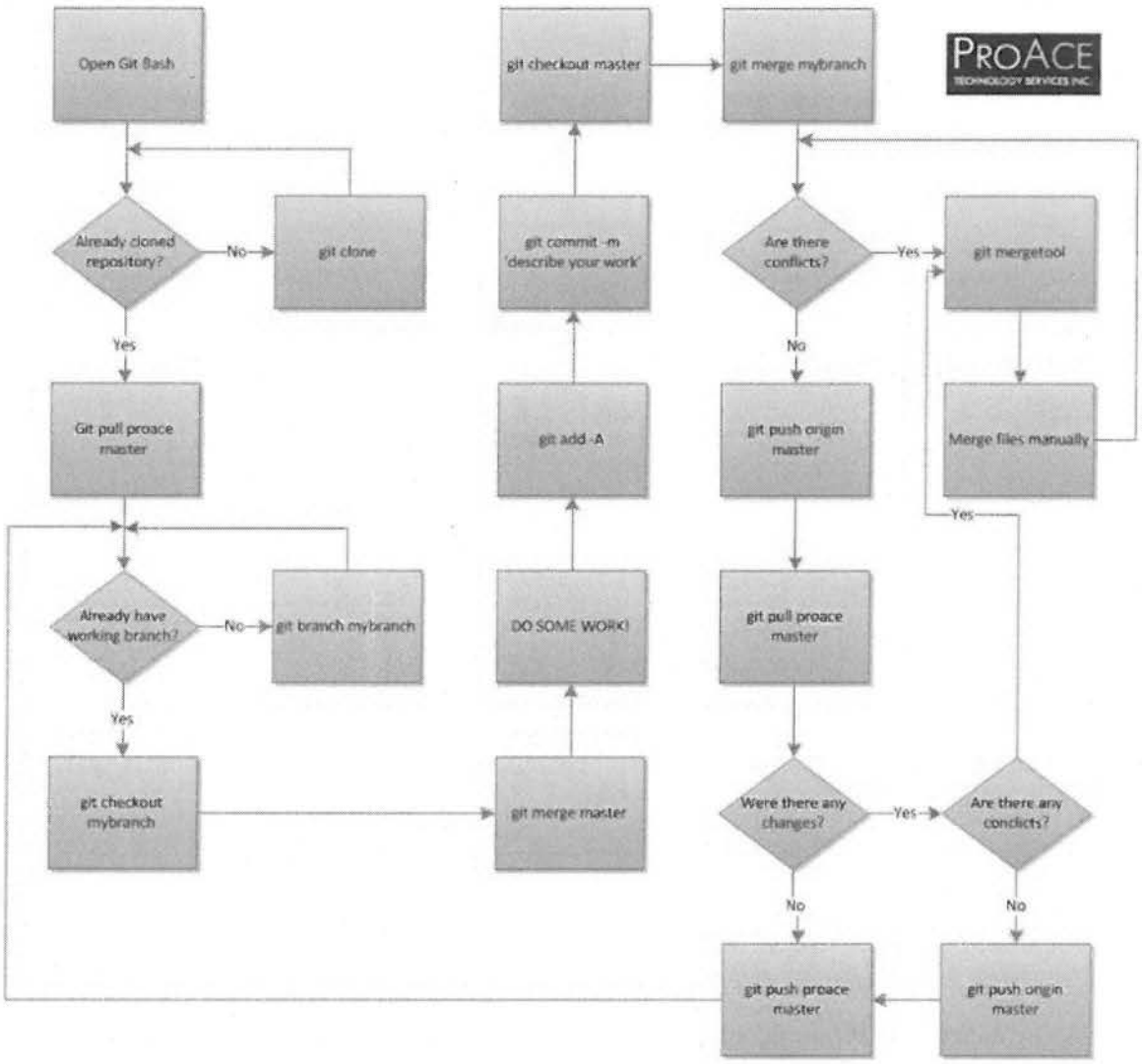
- `yum install git-core` (CentOS, fedora)
- `apt-get install git-core` (debian, ubuntu)
- `brew install git` (OS X)

Στα Windows, όπως ένα κοινό πρόγραμμα, μεταφορτώνεται και εγκαθίσταται το msysGit package.

Μετά την εγκατάσταση, αρχικοποιείται ένας κρυφός (hidden) φάκελος, με όνομα .git, που κρατάει στοιχεία, ρυθμίσεις και το ιστορικό των αλλαγών. Η εντολή για την αρχικοποίηση είναι `git init`.

Όταν γίνονται προσθήκες, αλλαγές ή ολοκληρώνονται κομμάτια της εφαρμογής καταγράφεται η τρέχουσα κατάσταση (stage files). Η εντολή που χρησιμοποιείται είναι `git add`. Τώρα, οι αλλαγές βρίσκονται σε κατάσταση αναμονής και δεν έχουν καταγραφεί ακόμα σε κάποιο αποθετήριο (repository). Αποθετήριο είναι το φυσικό σημείο που βρίσκονται αποθηκευμένα αρχεία και ρυθμίσεις

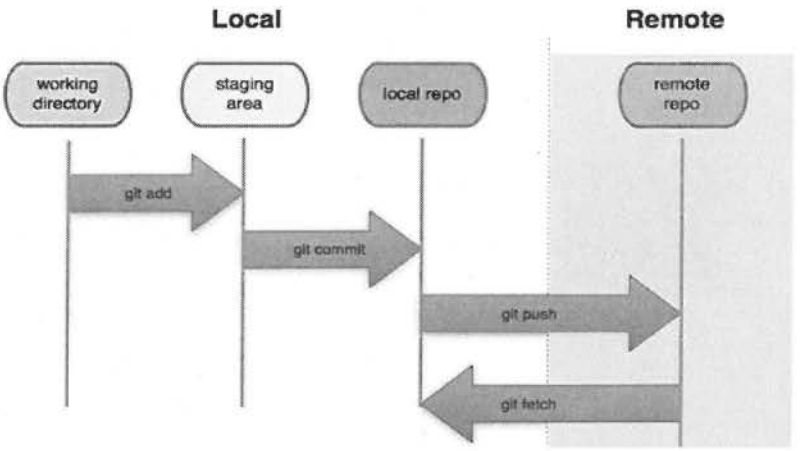
που αφορούν το έργο και τις εκδόσεις του. Για να αποθηκευτούν οι αλλαγές στο τοπικό αποθετήριο εκτελείται η εντολή `git commit`.



Εικόνα 4.1.4-2: Εντολές και διάγραμμα ροής στο Git

Τις περισσότερες φορές υπάρχουν ένα ή περισσότερα απομακρυσμένα αποθετήρια (remote repositories), εκτός από το τοπικό (local repository). Για να ενημερωθεί ένα απομακρυσμένο αποθετήριο για τις τελευταίες εκδόσεις εκτελείται η εντολή `git push`. Και αντίστροφα για την ενημέρωση του τοπικού αποθετηρίου από το απομακρυσμένο, εκτελείται η εντολή `git fetch`. Όμως με την εντολή `git fetch` ενημερώνεται το τοπικό αποθετήριο χωρίς να ενσωματώνει τις

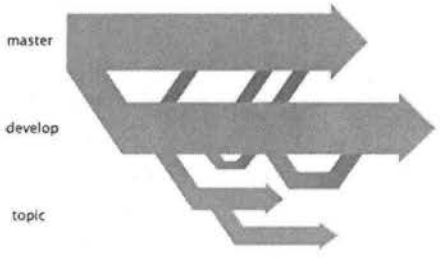
αλλαγές. Για την ενσωμάτωση η κατάλληλη εντολή είναι `git merge`. Ενώ, η εντολή `git pull` έχει την ίδια λειτουργικότητα με την διαδοχική εκτέλεση των εντολών `git fetch` και `git merge`.



Εικόνα 4.1.4-3: Ενημέρωση τοπικού και απομακρυσμένου αποθετηρίου

Όταν θέλουμε να πάρουμε ένα αντίγραφο τότε μπορούμε να κλονοποιήσουμε ένα απομακρυσμένο αποθετήριο μέ την εξής εντολή: `git clone https://github.com/webspirit/car-pooling-diploma-thesis.git`

Το git, επίσης υποστηρίζει διακλαδώσεις (branches). Συνηθίζεται να υπάρχει μια κύρια διακλάδωση (master) η οποία περιλαμβάνει μόνο το κώδικα που προορίζεται για την τελική έκδοση της εφαρμογής κι άλλες διακλαδώσεις στις οποίες μπορεί να δοκιμάζεται κώδικας κι αν πληρεί τις προϋποθέσεις στη να προστεθεί στη κύρια διακλάδωση.



Εικόνα 4.1.4-4: Διακλαδώσεις στο Git

Η δημιουργία, τροποποίηση και ενσωμάτωση διακλαδώσεων γίνεται με τις παρακάτω εντολές:

`git branch branch-name`, δημιουργία νέας διακλάδωσης με όνομα "branch-name"

`git branch -D branch-name`, διαγραφή της διακλάδωσης "branch-name"

`git branch`, προβολή λίστας με όλες τις διακλαδώσεις

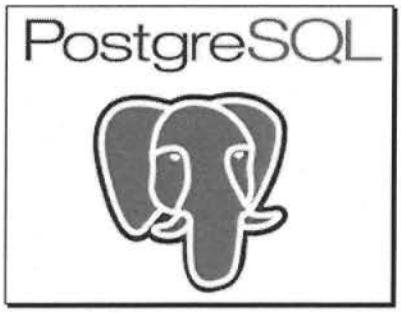
`git checkout branch-name`, μετάβαση στη διακλάδωση "branch-name"

`git merge`, ενσωμάτωση μιας διακλάδωσης στο κύριο κορμό της εφαρμογής

4.2 Επιλεγμένες Τεχνολογίες

4.2.1 Βάση δεδομένων - PostgreSQL

Η PostgreSQL αποτελεί μια ανοιχτού κώδικα σχεσιακή βάση δεδομένων.Η ανάπτυξη της ήδη διαρκεί πάνω από 20 χρόνια και βασίζεται σε μια αποδεδειγμένα καλή αρχιτεκτονική η οποία έχει δημιουργήσει μια ισχυρή αντίληψη των χρηστών της γύρω από την αξιοπιστία,την ακεραιότητα δεδομένων και την ορθή λειτουργία.



Εικόνα 4.2.1-1: Λογότυπο της PostgreSQL

Η PostgreSQL τρέχει σε όλα τα βασικά λειτουργικά συστήματα, περιλαμβάνοντας **Linux**, **UNIX** (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), και **Windows**.

Η PostgreSQL είναι τύπος βάσης δεδομένων που χρησιμοποιείτε από sites με ειδικές εφαρμογές. Τέτοιες βάσεις δεδομένων χρησιμοποιούνται για εφαρμογές όπως Forum,Blogs, Ηλεκτρονικά καταστήματα κ.α. Γενικότερα χρησιμοποιούνται για sites που έχουν ανάγκη για online διαχείριση μεγάλων όγκων δεδομένων ή κατασκευή ειδικών εφαρμογών. Είναι ACID συμβατή (ACID compliant),

έχει ολοκληρωμένη υποστήριξη για foreign keys, joins, views, triggers, και stored procedures (σε διάφορες γλώσσες προγραμματισμού). Συμπεριλαμβάνει τα περισσότερα SQL92 και SQL99 data types, συμπεριλαμβανομένων INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, και TIMESTAMP. επίσης υποστηρίζει αποθήκευση binary large objects, όπως εικόνες, ήχοι ή video. Διαθέτει native programming interfaces για **C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, κ.α.**

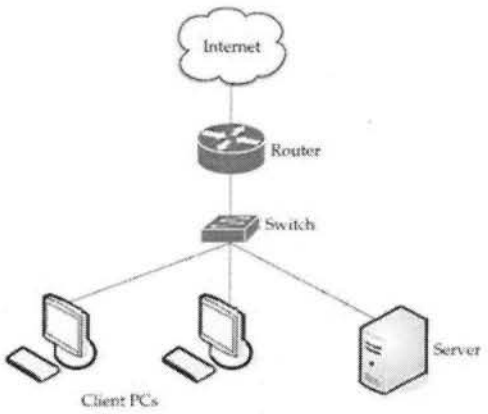
Η PostgreSQL υλοποιεί εξεζητημένα χαρακτηριστικά όπως Multi-Version Concurrency Control (MVCC), point in time recovery, tablespaces, asynchronous replication, nested transactions (savepoints), online/hot backups, a sophisticated query planner/optimizer, write ahead logging for fault tolerance. Υποστηρίζει διεθνή σετ χαρακτήρων, κωδικοποίηση χαρακτήρων σε πολλά byte, Unicode καθώς και δυνατότητα ταξινόμισης δεδομένων ανεξάρτητα απο το locale. Η PostgreSQL μπορεί να διαχειριστεί εύκολα μεγάλους αριθμούς ταυτόχρονων χρηστών καθώς και μεγάλο όγκο δεδομένων. Υπάρχουν ενεργές εγκαταστάσεις σε περιβάλλοντα παραγωγής που διαχειρίζονται πάνω από 4 terabytes δεδομένων. Μερικές Γενικές οριακές Τιμές συμπεριλαμβάνονται στον παρακάτω πίνακα:

Limit	Value
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Unlimited
Maximum Columns per Table	250 - 1600 depending on column types
Maximum Indexes per Table	Unlimited

Πίνακας 4.2.1-1: Οριακές τιμές ενεργών εγκαταστάσεων PostgreSQL

4.2.2 Cloud Computing – Heroku [8] [10]

Το Cloud computing ή αλλιώς Υπολογιστικό νέφος ή σύννεφο πήρε το όνομα του ως μεταφορά για το διαδίκτυο. Συνήθως το διαδίκτυο παρουσιάζεται στα διαγράμματα δικτύου σαν σύννεφο, όπως στην εικόνα 4.2.2-1. Με το σκίτσο του σύννεφου συνήθως προσπαθούμε να περιγράψουμε ένα απομακρυσμένο σύνολο αξιόπιστο. Όπως ακριβώς συμβαίνει και με το ηλεκτρικό ρεύμα όπου ο καταναλωτής ασχολείται μόνο με το που βρίσκεται μια πρίζα και όχι με το πώς παράγεται ή μεταφέρεται η ηλεκτρική ενέργεια (http://blogs.msdn.com/b/gkanel/archive/2010/10/29/cloud-computing.aspx).



Εικόνα 4.2.2-1: Προέλευση ονόματος Cloud Computing

Το σύννεφο χρησιμοποιείται στα διαγράμματα δικτύου για να απεικονίσουν το internet (Antony T.Velte, Toby J.Velte, Robert Elsenpeter ,2010, Cloud computing: a practical approach, The McGraw-Hill Companies).

Το υπολογιστικό νέφος βοηθάει στη μείωση των λειτουργικών εξόδων και των κεφαλαιουχικών εξόδων και κυρίως εξοικονομεί χρόνο στα τμήματα IT για να μπορούν να επικεντρώνονται σε στρατηγικά σχέδια αντί να αναλώνονται στην συντήρηση του κέντρου δεδομένων.

Στην ουσία το υπολογιστικό νέφος είναι μια κατασκευή που επιτρέπει να έχουν οι χρήστες πρόσβαση σε εφαρμογές που στη πραγματικότητα είναι εγκατεστημένες σε διαφορετική τοποθεσία από τους υπολογιστές τους ή οποιαδήποτε άλλη συσκευή σύνδεσης στο διαδίκτυο, η οποία είναι συνήθως κάποιο μακρινό κέντρο δεδομένων. Υπάρχουν πολλά οφέλη από αυτό. Για παράδειγμα μπορούμε να σκεφτούμε την εγκατάσταση του Microsoft Excel σε έναν οργανισμό, η οποία έγινε είτε τρέχοντας ένα CD ή DVD-ROM σε κάθε υπολογιστή ξεχωριστά, είτε έχοντας ρυθμίσει να γίνει αυτόματα η εγκατάσταση σε όλους από τον διακομιστή με κάποιο λογισμικό. Κάθε φορά που η Microsoft εκδίδει νέο service pack, θα πρέπει να γίνει η ενημέρωση σε όλους τους υπολογιστές. Όλοι οι υπάλληλοι της εταιρίας όμως δεν χρησιμοποιούν το Excel καθημερινά και ακόμα και αν δεν το χρησιμοποιούν η άδεια για το Excel πληρώνεται. Το πλεονέκτημα του υπολογιστικού νέφους είναι ότι μία άλλη εταιρία φιλοξενεί την εφαρμογή που σημαίνει ότι αυτοί χειρίζονται τα κόστη των servers (εξοπλισμό του server, ενέργεια που καταναλώνεται για την λειτουργία του αλλά και την ψύξη του), διαχειρίζονται τις αναβαθμίσεις του λογισμικού και ορισμένες φορές αμείβονται λιγότερο για οποιοδήποτε service. Τέλος πολύ σημαντικό πλεονέκτημα είναι η δυνατότητα χρησιμοποίησης των εφαρμογών από οποιοδήποτε σημείο θέλουμε σε περίπτωση που ταξιδεύουμε.

Υπάρχουν όμως και μειονεκτήματα στο cloud computing. Στην περίπτωση που υπάρχει διακοπή λειτουργίας του internet ή ο φορέας παροχής υπηρεσιών internet έχει πρόβλημα, τότε η εταιρία που χρησιμοποιεί cloud computing δεν θα είναι σε θέση να χρησιμοποιήσει τις εφαρμογές της και συνεπώς θα καθυστερεί τις εργασίες της. Αυτό το πρόβλημα δεν θα υπήρχε αν η εταιρία είχε εγκατεστημένες όλες τις εφαρμογές της. Από την άλλη μπορεί το site το οποίο επισκέπτεται η εταιρία για να χρησιμοποιήσει τις εφαρμογές της να έχει πρόβλημα. Αυτό συνέβη τον Ιούλιο του 2008 όταν το Amazon S3 έπεσε για δεύτερη φορά το ίδιο έτος. Πολλές από τις εφαρμογές που φιλοξενούσε και όλες οι υπηρεσίες του δεν είχαν πρόσβαση μέχρι να αποκατασταθεί το πρόβλημα. Πολλές από τις εφαρμογές ήταν εκτός λειτουργίας για 8 ώρες (Antony T.Velte, Toby J.Velte, Robert Elsenpeter ,2010).

4.2.2.1 Είδη υπηρεσιών Cloud Computing

Υπάρχουν τρία είδη υπηρεσιών cloud computing, το Software-as-as-a-Service, το Platform-as-a-Service και το Infrastructure-as-a-Service. Το κάθε ένα από αυτά, εξυπηρετεί διαφοραS εφαρμογή η προσαρμογή των προγραμμάτων είναι ευκολότερη και γίνεται ανάλογα με τις ανάγκες της εκάστοτε εταιρίας.

Υπάρχουν όμως και μειονεκτήματα στο μοντέλο SaaS. Το πρώτο μειονέκτημα είναι ότι μία εταιρία με πολύ συγκεκριμένες ανάγκες σε υπολογιστικά προγράμματα πιθανόν να μην μπορέσει να βρει την εφαρμογή που χρειάζεται μέσω του SaaS. Επιπλέον το μοντέλο SaaS αντιμετωπίζει προβλήματα με τις εφαρμογές ανοιχτού κώδικα και το φθηνότερο hardware. Οι εταιρίες που εκδίδουν τα προγράμματα μπροστά σε αυτή την απειλή μπορούν να ενσωματώσουν τις εφαρμογές ανοιχτού κώδικα σε hardware που έχει καλύτερη απόδοση και κοστίζει λιγότερο από ότι στο παρελθόν. Η Microsoft παρέχει τις εξής SaaS υπηρεσίες: Exchange Online (ηλεκτρονικό ταχυδρομείο), SharePoint Online (Σύστημα διαχείρισης κειμένων και περιεχομένου) CRM Online, Office Live Meeting (ηλεκτρονικός χώρος συναντήσεων), Office Communications Online (Instant Messaging), Hotmail, Live Messenger, LiveID. ετικές ανάγκες και προσφέρει διαφορετικές υπηρεσίες.

4.2.2.2 Software as a Service

Το μοντέλο Software-as-a-Service βασίζεται στη λογική της υπενοικίασης λογισμικού από έναν πάροχο υπηρεσιών, αντί της αγοράς της άδειας χρήσης. Software as a Service (SaaS) είναι το μοντέλο στο οποίο μια εφαρμογή φιλοξενείται ως υπηρεσία στον πελάτη μέσω του Internet. Το λογισμικό λειτουργεί σε ένα κεντρικοποιημένο δίκτυο servers προκειμένου να διατίθεται ως υπηρεσία από το web ή το διαδίκτυο. Το SaaS μοντέλο είναι πολύ αποτελεσματικό στη μείωση του κόστους αφού παρέχεται στην επιχείρηση ως μηνιαίο λειτουργικό κόστος το οποίο συνήθως είναι κατά πολύ οικονομικότερο από την αγορά των αντίστοιχων αδειών χρήσης και υποδομής. Όταν το μοντέλο φιλοξενείτε εκτός του χώρου της εταιρίας, ο πελάτης δεν χρειάζεται να συντηρεί ή να υποστηρίζει την εφαρμογή. Ο πάροχος αναλαμβάνει όλες τις αναβαθμίσεις καθώς και την συντήρηση της εφαρμογής.

Ένα από τα μεγαλύτερα οφέλη του μοντέλου SaaS είναι ότι κοστίζει λιγότερο από ότι η αγορά των εφαρμογών. Ο πάροχος της υπηρεσίας μπορεί να προσφέρει φθηνότερα πιο αξιόπιστες εφαρμογές. Επίσης όλοι οι υπάλληλοι πλέον έχουν πρόσβαση και είναι εξοικειωμένοι με τον παγκόσμιο ιστό (www) συνεπώς η εκμάθηση της χρήσης των εξωτερικών εφαρμογών είναι εύκολη. Οι εταιρίες δεν χρειάζονται το ίδιο εργατικό δυναμικό στα IT τμήματά τους αρά μειώνονται τα κόστη μισθοδοσίας και ασφάλισης των εργαζομένων αλλά και ο χώρος στέγασης των γραφείων τους στην εταιρία. Με την SaaS εφαρμογή η προσαρμογή των προγραμμάτων είναι ευκολότερη και γίνεται ανάλογα με τις ανάγκες της εκάστοτε

εταιρίας.

Υπάρχουν όμως και μειονεκτήματα στο μοντέλο SaaS. Το πρώτο μειονέκτημα είναι ότι μία εταιρία με πολύ συγκεκριμένες ανάγκες σε υπολογιστικά προγράμματα πιθανόν να μην μπορέσει να βρει την εφαρμογή που χρειάζεται μέσω του SaaS. Επιπλέον το μοντέλο SaaS αντιμετωπίζει προβλήματα με τις εφαρμογές ανοιχτού κώδικα και το φθηνότερο hardware. Οι εταιρίες που εκδίδουν τα προγράμματα μπροστά σε αυτή την απειλή μπορούν να ενσωματώσουν τις εφαρμογές ανοιχτού κώδικα σε hardware που έχει καλύτερη απόδοση και κοστίζει λιγότερο από ότι στο παρελθόν. Η Microsoft παρέχει τις εξής SaaS υπηρεσίες: Exchange Online (ηλεκτρονικό ταχυδρομείο), SharePoint Online (Σύστημα διαχείρισης κειμένων και περιεχομένου) CRM Online, Office Live Meeting (ηλεκτρονικός χώρος συναντήσεων), Office Communications Online (Instant Messaging), Hotmail, Live Messenger, LiveID.

4.2.2.3 Platform as a Service

Platform-as-a-Service παρέχει μια cloud πλατφόρμα εφαρμογών για εταιρείες ή ιδιώτες που κατασκευάζουν λογισμικό είτε για ιδία χρήση είτε για τρίτους. Το PaaS προμηθεύει όλους τους πόρους που χρειάζονται για τη δημιουργία εφαρμογών και υπηρεσιών πλήρως από το Internet, χωρίς να χρειάζεται να κατεβάσει και να εγκαταστήσει ο χρήστης το λογισμικό.

Οι υπηρεσίες του PaaS περιλαμβάνουν σχεδιασμό εφαρμογής, ανάπτυξη, δοκιμή, εγκατάσταση και φιλοξενία. Άλλες υπηρεσίες περιλαμβάνουν συνεργασία της ομάδας, web service integration, database integration, ασφάλεια, επεκτασιμότητα, αποθήκευση και ενημέρωση εκδόσεων.

Το αρνητικό του PaaS είναι η έλλειψη διαλειτουργικότητας και φορητότητας μεταξύ των παρόχων. Δηλαδή αν δημιουργήσουμε μια εφαρμογή με έναν πάροχο και αποφασίσουμε να αλλάξουμε πάροχο μπορεί να μην είναι δυνατό ή θα πρέπει να πληρώσουμε υψηλότερη τιμή. Επίσης αν ο πάροχος σταματήσει την λειτουργία του τότε τα δεδομένα θα χαθούν.

Το PaaS βασίζεται στο μοντέλο «Pay-per-use» με τέτοιο τρόπο έτσι ώστε να επιτυγχάνεται η πλήρης αξιοποίηση των υπολογιστικών πόρων που χρησιμοποιούνται σε σχέση με το κόστος χρήσης. Αν συνδυαστεί με το χαρακτηριστικό της αυτόκλιμάκωσης μπορούμε να πετύχουμε τη διάθεση υπηρεσιών

που να μπορούν να ανταποκρίνονται σε οποιαδήποτε ραγδαία ή αναμενόμενη μεταβολή χωρητικότητας (ισχύς, μνήμη, αποθηκευτικό χώρο, δίκτυο) που θα απαιτηθεί ανά πάσα χρονική στιγμή χωρίς να έχω δεσμευτεί εκ των προτέρων είτε με αγορά υποδομής, λογισμικού πλατφόρμας, δικτυακή γραμμή υψηλής χωρητικότητας κλπ. είτε με ένα συμβόλαιο παροχής υπηρεσιών φιλοξενίας υποδομής και πλατφόρμας συγκεκριμένης χωρητικότητας και χρονικής διάρκειας. Η Microsoft παρέχει τις παρακάτω PaaS υπηρεσίες: Windows Azure, SQL Azure, Windows Azure AppFabric.

Infrastructure-as-a-Service Το Infrastructure-as-a-Service (IaaS) είναι η επόμενη μορφή υπηρεσίας του cloud computing. Το Infrastructure-as-a-Service είναι η παροχή υπολογιστικών και δικτυακών υποδομών ως μια πλήρως outsourced υπηρεσία. Η εταιρεία ή ο ιδιώτης μπορεί να υπενοικιάσει υποδομή (όχι όμως και πλατφόρμα όπως στο PaaS) ανάλογα με τις απαιτήσεις εκείνης της χρονικής στιγμής με λογική, όπως και στο PaaS, «Pay as you go» αντί να προβεί στην αγορά εξοπλισμού (υπολογιστικού, δικτυακού, κλπ) ή στη σύναψη συμβολαίου παροχής υπηρεσιών φιλοξενίας υποδομής για συγκεκριμένο χρονικό διάστημα.

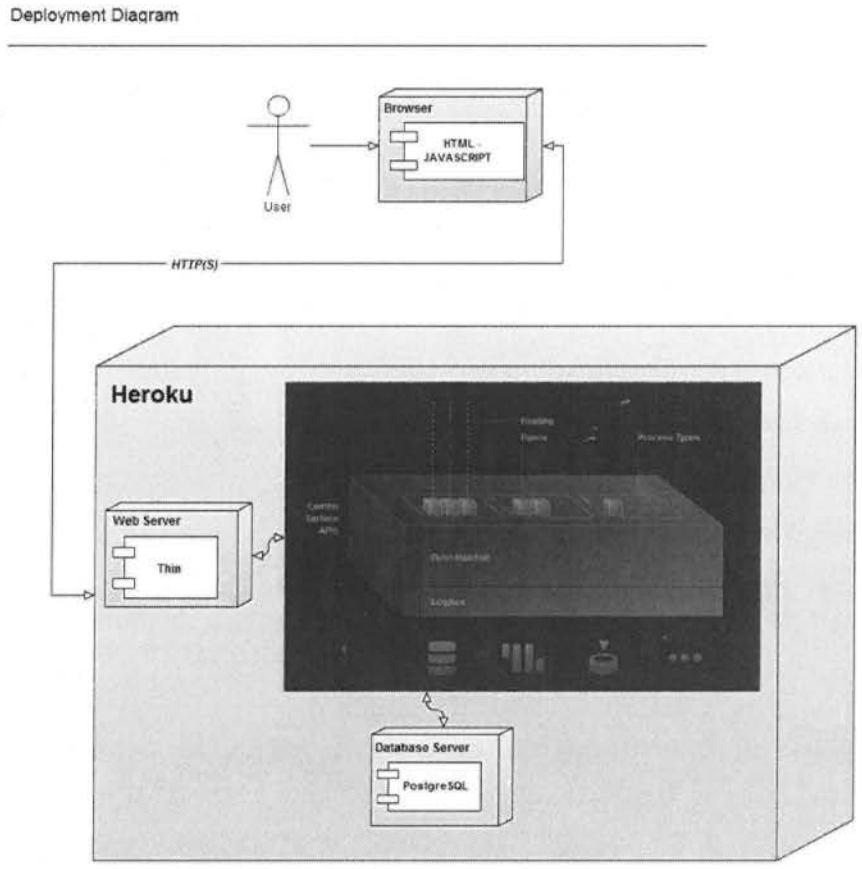
4.2.2.4 Heroku (PaaS) [9]

Εικόνα 4.2.2-2: Λογότυπο Heroku

Το Heroku ήταν μια από τις πρώτες πλατφόρμες στο υπολογιστικό σύννεφο,υποστηρίζει πολλές γλώσσες προγραμματισμού και συνεχίζει να εξελίσεται από το2007. Όταν έκανε την εμφάνισή της υποστήριζε μόνο την γλώσσα προγραμματισμού Ruby αλλά στη συνέχεια προστέθηκε υποστήριξη για τις Java, Node.js, Scala, Clojure, Python, Perl και για μη τεκμηριωμένη PHP. Το βασικό λειτουργικό της σύστημα είναι Debian βασισμένα σε Debian.

Η πλατφόρμα Heroku βοηθάει και διευκολύνει τους χρήστες να χτίζουν και να ανεβάζουν διαδικτυακές εφαρμογές στο σύννεφο. Βασικός στόχος είναι οι προγραμματιστές να μήν σπαταλούν χρόνο και πόρους στην εγκατάσταση της εφαρμογής (διαχείριση servers, επεκτασιμότητα, deployment κ.α.) αλλά να δίνουν περισσότερη έμφαση στην ανάπτυξη της εφαρμογής.

Παραθέτεται η σχέση που υπάρχει ανάμεσα στο χρήστη και την εφαρμογή. Εδώ φαίνεται πως ο χρήστης, μέσω του φυλλομετρητή και της αρχιτεκτονικής που αυτός χρησιμοποιεί (Html και JavaScript), συνεργάζεται με τον εξυπηρετητή και τη βάση δεδομένων (στην προκειμένη περίπτωση αυτά εδρεύουν στο υπολογιστικό νέφος και όχι σε κάποιο συγκεκριμένο φυσικό μηχάνημα).



Εικόνα 4.2.2-3: Αρχιτεκτονικό μοντέλο λογισμικού

4.2.3 HTML – HTML5 [14]

Η HTML5 είναι μια υπό ανάπτυξη γλώσσα σήμανσης για τον Παγκόσμιο Ιστό και είναι η επόμενη μεγάλη έκδοση της HTML (Γλώσσα Υπερκειμένου, HyperText Markup Language).

Η ομάδα Web Hypertext Application Technology Working Group (WHATWG) άρχισε δουλειά σε αυτή την έκδοσητον Ιούνιο του 2004 με το όνομα Web Applications 1.0.

Εικόνα 4.2.3-1: Λογότυπο HTML5

Η HTML5 προορίζεται για αντικατάσταση της HTML 4.01, της XHTML 1.0, και της DOM Level 2 HTML. Ο σκοπός είναι η μείωση της ανάγκης για ιδιόκτητα plug-in και πλούσιες διαδικτυακές εφαρμογές (RIA) όπως το Adobe Flash, το Microsoft Silverlight, το Apache Pivot, και η Sun JavaFX.

Οι ιδέες πίσω από την HTML5 εμφανίστηκαν αρχικά το 2004 από την ομάδα WHATWG. Η HTML5 εμπεριέχει το πρότυπο *Web Forms 2.0* που είναι επίσης της WHATWG.

Το πρότυπο HTML5 υιοθετήθηκε ως αρχικό βήμα για τις εργασίες της νέας ομάδας εργασίας HTML του W3C το 2007. Αυτή η ομάδα εργασίας δημοσίευσε το Πρώτο Δημόσιο Working Draft του προτύπου στις 22 Ιανουαρίου 2008. Το πρότυπο είναι ακόμη υπό ανάπτυξη, και αναμένεται να παραμείνει έτσι για πολλά χρόνια, παρόλο που μέρη της HTML5 θα τελειώσουν και θα υποστηριχτούν από περιηγητές πριν το όλο πρότυπο φτάσει στη τελική κατάσταση Recommendation. Οι συντάκτες της HTML5 είναι ο Ίαν Χίκσον της εταιρίας Google και ο Ντέιβ Χίατ της εταιρίας Apple.

4.2.4 Twitter Bootstrap [15]

Το Bootstrap είναι μια συλλογή εργαλείων ανοιχτού κώδικα (Ελεύθερο λογισμικό) για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει HTML και CSS για τις μορφές τυπογραφίας, κουμπιά πλοήγησης και άλλων στοιχείων του περιβάλλοντος, καθώς και προαιρετικές επεκτάσεις JavaScript. Έχει το πιο δημοφιλές πρόγραμμα στο GitHub και έχει χρησιμοποιηθεί από τη NASA και το MSNBC, μεταξύ άλλων.

Εικόνα 4.2.4-1: Λογότυπο Bootstrap

Το Bootstrap έχει σχετικά ελλιπή υποστήριξη για HTML5 και CSS, αλλά είναι συμβατό με όλους τους φυλλομετρητές (browsers). Βασικές πληροφορίες συμβατότητας των ιστοσελίδων ή εφαρμογές είναι διαθέσιμες για όλες τις συσκευές και τα προγράμματα περιήγησης.

Το Bootstrap είναι σπονδυλωτό και αποτελείται ουσιαστικά από μια σειρά στυλ(stylsheets) που εφαρμόζουν τα διάφορα συστατικά του πακέτου εργαλείων. Ένα στυλ που ονομάζεται bootstrap.less περιλαμβάνει τα συστατικά stylesheets. Οι προγραμματιστές μπορούν να προσαρμόσουν το αρχείο Bootstrap, επιλέγοντας τα στοιχεία που θέλουν να χρησιμοποιήσουν στο έργο τους.

Προσαρμογές είναι δυνατές σε περιορισμένη έκταση μέσω ενός κεντρικού στυλ διαμόρφωσης. Η χρήση γλώσσας στυλ επιτρέπει τη χρήση για μεταβλητές, λειτουργίες και φορείς (operators), ένθετους επιλογείς, γνωστά και ως μείγματα mixin.

Από την έκδοση 2.0, η διαμόρφωση του Bootstrap έχει επίσης μία ειδική επιλογή "Προσαρμογή " στην τεκμηρίωση (documentation). Επιπλέον, ο σχεδιαστής του έργου επιλέγει σε μια φόρμα τα επιθυμητά συστατικά και τα προσαρμόζει, εάν είναι αναγκαίο, σε τιμές διαφόρων εναλλακτικών λύσεων για τις ανάγκες του. Στη συνέχεια δημιουργείται ένα πακέτο που περιλαμβάνει ήδη το προ-χτισμένο CSS στυλ.

Εκτός από τα βασικά HTML στοιχεία, το Bootstrap περιέχει και άλλα στοιχεία περιβάλλοντος που χρησιμοποιούνται συχνά. Αυτά περιλαμβάνουν κουμπιά με προηγμένα χαρακτηριστικά (π.χ. ομαδοποίηση κουμπιών ή drop-down επιλογή, οριζόντιες και κάθετες καρτέλες, πλοήγηση, σελιδοποίηση, κ.λπ.), ετικέτες, προηγμένες τυπογραφικές δυνατότητες, εικονίδια, προειδοποιητικά μηνύματα και μια γραμμή προόδου.

4.2.5 JavaScript [16]

Εικόνα 4.2.5-1: Λογότυπο JavaScript

Η JavaScript είναι γλώσσα προγραμματισμού, η οποία έχει σαν σκοπό την παραγωγή δυναμικού περιεχομένου και την εκτέλεση κώδικα στην πλευρά του πελάτη (client-side) σε ιστοσελίδες.

Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).

Το πρότυπο της γλώσσας κατά τον οργανισμό τυποποίησης ECMA ονομάζεται ECMAscript.

Η γλώσσα προγραμματισμού JavaScript δημιουργήθηκε αρχικά από τον Brendan Eich της εταιρείας Netscape με την επωνυμία Mocha. Αργότερα, Mocha μετονομάστηκε σε LiveScript, και τελικά σε JavaScript, κυρίως επειδή η ανάπτυξή της επηρεάστηκε περισσότερο από τη γλώσσα προγραμματισμού Java. LiveScript ήταν το επίσημο όνομα της γλώσσας όταν για πρώτη φορά κυκλοφόρησε στην αγορά σε βήτα (beta) εκδόσεις με το πρόγραμμα περιήγησης στο Web, Netscape Navigator εκδοχή 2.0 τον Σεπτέμβριο του 1995. LiveScript μετονομάστηκε σε JavaScript σε μια κοινή ανακοίνωση με την εταιρεία Sun Microsystems στις 4 Δεκεμβρίου, 1995 , όταν επεκτάθηκε στην έκδοση του προγράμματος περιήγησης στο Web, Netscape εκδοχή 2.0B3

4.2.6 Highcharts [17]

Εικόνα 4.2.6-1: Λογότυπο Highcharts

Το Highcharts είναι μια βιβλιοθήκη διαγραμμάτων γραμμένη σε καθαρή JavaScript, προσφέροντας στον προγραμματιστή έναν εύκολο τρόπο να προσθέσει διδραστικά διαγράμματα στην ιστοσελίδα ή στην διαδικτυακή εγαρμογή του. Το Highcharts υποστηρίζει τους εξής τύπους διαγραμμάτων : line, spline, area, areaspline, column, bar, pie,

scatter, angular gauges polar chart types .

Οι χρήστες έχουν την δυνατότητα να εξάγουν το γράφημα απευθείας από την ιστοσελίδα. Είναι πλήρως συμβατή με όλους τους περιηγητές συμπεριλαμβάνοντας και τα iPhone/iPad και τον Internet Explorer από την έκδοση 6.

4.2.7 RVM [18]

Εικόνα 4.2.7-1: Λογότυπο του RVM

Το RVM (Ruby Version Manager) δημιουργήθηκε από τον Seguin Waynee με σκοπό την ταυτόχρονη εγκατάσταση πολλών εκδόσεων της γλώσσας Ruby. Επίσης υποστηρίζει την καλύτερη οργάνωση των βιβλιοθηκών σε gemsets.Κάθε gemset αποτελείται από ένα σύνολο βιβλιοθηκών (gems) της Ruby. Ανάλογα με την εφαρμογή χρησιμοποιείται η κατάλληλη έκδοση βιβλιοθηκών και οργανώνονται καλύτερα οι αλληλεξαρτήσεις των βιβλιοθηκών.

4.2.8 Bundler

Ο Bundler αναπτύχθηκε από έναν από τους πρωτοπόρους της βασικής ομάδας της Rails και της jQuery, τον Yehuda Katz. Ελέγχει τις αλληλεξαρτήσεις μεταξύ των βιβλιοθηκών της Ruby και φροντίζει ώστε να γίνεται με αυτοματοποιημένο τρόπο η μεταφόρτωση όπως και η εγκατάσταση όλων των βιβλιοθηκών μιας εφαρμογής σε διαφορετικά συστήματα.

4.2.9 Λόγοι χρήσης επιμέρους τεχνολογιών

Σε αυτό το σημείο παρουσιάζονται οι λόγοι για τους οποίους χρησιμοποιήκαν οι παραπάνω τεχνολογίες. Σε μια εποχή μεγάλων ανακατατάξεων στισ γλώσσες προγραμματισμού για τη

σχεδίαση διαδικτυακών εφαρμογών, αποφασίστηκε να ακολουθήσουμε το ρεύμα τής εποχής και να χρησιμοποιηθεί ένα ευέλικτο, απλό και ισχυρό web framework γραμμένο σε Ruby, το Ruby on Rails.

Το Ruby on Rails, με την τεράστια κοινότητα που διαθέτει αποδείχτηκε εξαίρετη επιλογή καθώς έδωσε τη δυνατότητα στους προγραμματιστές να μάθουν τα βασικά χαρακτιριστικά σε πολύ μικρό χρονικό διάστημα και να μπορέσουν να δημιουργίσουν τα βασικά στοιχεία της εφαρμογής.

Ένα επίσης πλεονέκτημα που παρατηρήθηκε κατά τη διάρκεια δημιουργίας της εφαρμογής είναι η τεράστια ποικιλία σε βιβλιοθήκες για το Ruby on Rails αλλά και τη Ruby που βοηθούν τον προγραμματιστή στην ανάπτυξη σημαντικών χαρακτθριστικών του συστήματος (user authentication, full text search, user interface).

Όσον αφορά τη βάση δεδομένων στα αρχικά στάδια, χρησιμοποιήθηκε η πιο διαδεδομένη σχεσιακή βάση δεδομένων, η MySQL. Όμως όταν έφτασε η στιγμή η εφαρμογή να περάσει στο στάδιο παραγωγής, η επιλογή της χρήσης του "υπολογιστικού νέφους" και κατ' επέκταση της πλατφόρμας ως εφαρμογή (platform as a service) Heroku προέκυψε η ανάγκη για τη μετάβαση στη βάση δεδομένων PostgreSQL. Το Heroku δίνει τη δυνατότητα στους χρήστες του να αναπτύξουν εφαρμογές επιλέγοντας μόνο ανάμεσα στην PostgreSQL και τη MongoDB.
Αυτή η αναγκαστική αλλαγή αποδίκτηκε η καλύτερη επιλογή όσον αφορά το κομμάτι της αποθήκευσης των δεδομένων. Η εύκολη και ταχύτατη διαχείριση από τους προγραμματιστές, η σταθερότητα και αξιοπιστία μέσω του Heroku αλλά και ηδυνατότητα επεκτασιμότητας είναι μερικοί από τους λόγους που κάνουν την PostgreSQL άριστη επιλογή πέρα από αναγκαία.

Το Heroku, το οποίο είναι μια πλατφόρμα ως εφαρμογή, μείωσε στο ελάχιστο δυνατό το κόστος και τον κόπο ανέγερσης της εφαρμογής στο διαδίκτυο. Το μόνο που χρειάστηκε ήταν μια πλήρως λειτουργική εφαρμογή σε Ruby on Rails, η αλλαγή μερικών αρχείων ρυθμίσεων (configuration files) και ένας λογαριασμός στην πλατφόρμα Heroku. Τα υπόλοιπα τα αναλαμβάνει το σύστημα Heroku με ικανοποιητικά αποτελέσματα.

Για τη δημιουργία ενός εύχρηστου και όμορφου περιβάλλοντος απεικόνισης , μετά από αρκετή μελέτη βρέθηκε η βιβλιοθήκη Twitter Bootstrap, η οποία μπορεί να βοηθήσει τον αρχάριο σχεδιστή εφαρμογών διαδικτύου να αναπτύξει με λίγη προσπάθεια ένα καλαίσθητο, επαγγελματικού επιπέδου αποτέλεσμα. Το Twitter Bootstrap προσφέρει μια συλλογή από εργαλεία που χρησιμοποιούν τεχνολογίες όπως HTML, Less CSS και JavaScript. Συνεπώς οι τεχνολογίες αυτές βοηθούν στην εύκολη παραμετροποίηση κάποιον έτοιμων χαρακτηριστικών που προσφέρει το Twitter Bootstrap και ήταν ο κύριος λόγος για την επιλογή του.

Ενα ακόμα χαρακτηριστικό της εφαρμογής είναι τα διαγράμματα (Charts). Μέσα από ένα διάγραμμα μπορεί ο χρήστης να καταλάβει την απόδοση του αλγορίθμου σε σχέση με τη δική του βαθμολογία. Για την απεικόνιση των διαγραμμάτων χρησιμοποιήθηκε η βιβλιοθήκη Highcharts η οποία είναι γραμμένη σε καθαρή JavaScript προσφέροντας γρήγορη απόδοση και όμορφο αποτέλεσμα. Η βιβλιοθήκη Highcharts δέχεται τις πληροφορίες που χρειάζεται από τη βάση δεδομένων.

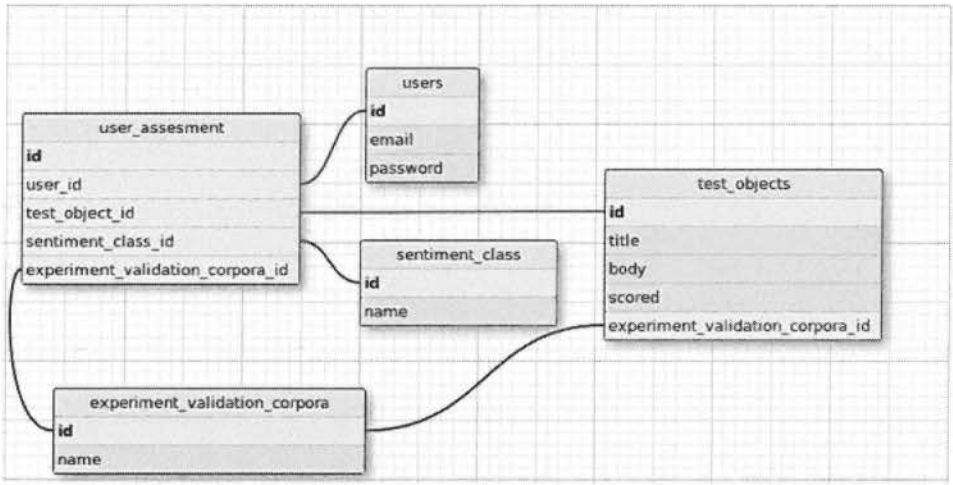
5 ΚΕΦΑΛΑΙΟ : Η Εφαρμογή αξιολόγισης sentiment analysis classifiers με βάση κειμενική πληροφορία SentiBox

5.1 Η δημιουργία του κορμού της εφαρμογής

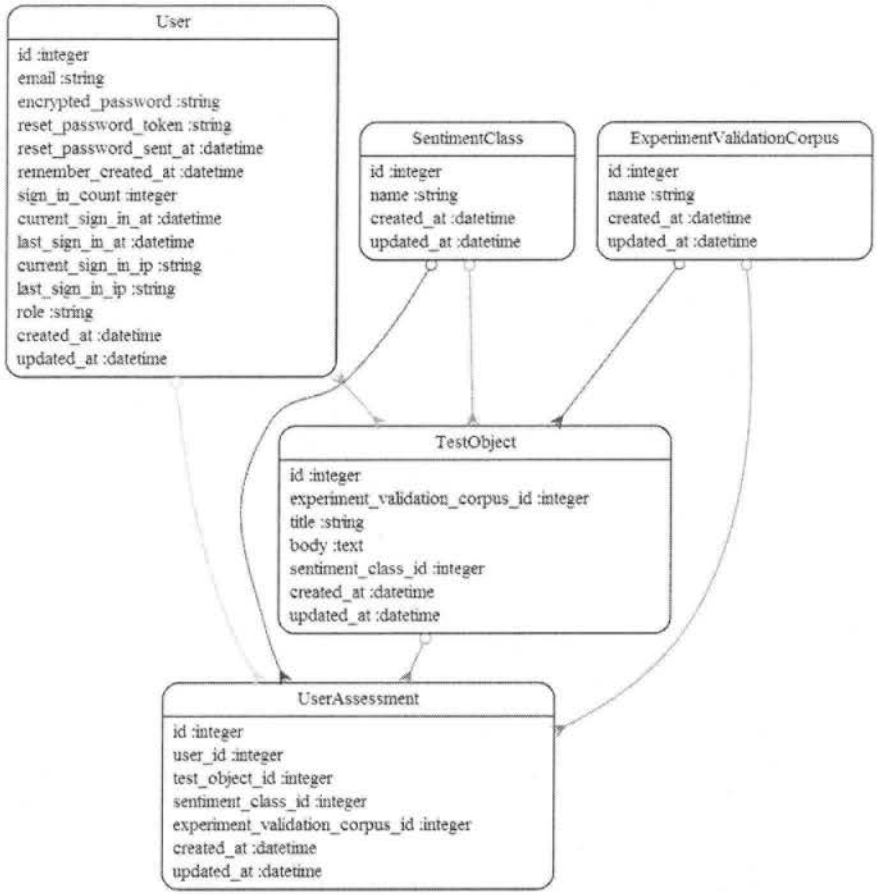
Ενώ οι αλγόριθμοι και τα διάφορα λογισμικά προγράμματα ξεπερνούν τον ανθρώπινο νου σε ταχύτητα και αποτέλεσμα όταν πρόκειται για ποσοτικές μεταβλητές και υπολογιστικές διαδικασίες, στην περίπτωση του opinion mining, που στηρίζεται κυρίως σε ποιοτικές μεταβλητές, η ακρίβεια της διαδικασίας κυμαίνεται αναμφίβολα σε χαμηλότερα επίπεδα. Με βάση τα παραπάνω προήλθε η ιδέα για τη δημιουργία μίας εφαρμογής η οποία εχει ως βασικό στόχο την αξιολόγηση των

αποτελεσμάτων που εξήξαγε ο αλγοριθμος βάση της ανθρώπινης νοϋμοσύνης μέσα απο ενα εύχρηστο και φιλικό περιβάλλον.

Ο κορμός της εφαρμογής είναι η βάση δεδομένων και κατ' επέκταση οι πίνακες που περιέχει. Στο πρώτο στάδιο δημιουργήθηκαν οι εξής πίνακες: Πίνακας του συλλογής επικύρωσης πειράματος (experiment validation corpora), συναισθηματικής κλάσης (sentiment class), αντικειμένου δοκιμής (test object), και ανάθεσης χρήστη (user assessment). Ο χρήστης θα έχει τη δυνατότητα να επιλέγει την κατηγορία του πειράματος που θέλει να αξιολογήσει και στη συνέχεια θα του εμφανίζονται τα κείμενα της επιλεγμένης κατηγορίας.



Εικόνα 4.2.9-1: Schema βάσης δεδομένων



Εικόνα 4.2.9-3: Schema βάσης δεδομένων όπως σχεδιάστηκε προγραμματιστικά

Στο δεύτερο στάδιο δημιουργήθηκαν οι αλγόριθμοι για τον υπολογισμό βασικών πληροφοριών ανάλογα με τα δεδομένα που εισήγαγε ο χρήστης. Οι πληροφορίες αυτές είναι οι εξής:

- Το σύνολο των αντικειμένων που έχει αξιολογήσει ο χρήστης ανά κατηγορία
- Οι πληροφορίες για την αξιολόγιση του χρήστη σε σύγκριση με τα αποτελέσματα που εξήγαγε ο αλγόριθμος
- Ο υπολογισμός των τελικών αποτελεσμάτων πάνω στα διαγράμματα

Στο τρίτο στάδιο δημιουργίθηκε το περιβάλλον διεπαφής του χρήστη με το σύστημα. Αρχικά σχηματίστηκαν οι φόρμες για την εισαγωγή των δεδομένων στους πίνακες. Έπειτα μορφοποιήθηκαν τα δεδομένα και παρουσιάστηκαν στο χρήστη μέσω αναλυτικών πινάκων και διαγραμμάτων.

Όλα τα παραπάνω προεκτάθηκαν για πολλούς χρήστες μέσω του πίνακα χρηστών (Users) και της διαδικασίας διαπίστευσης χρηστών (authentication). Επίσης αποκλείστηκαν οι σελίδες διεπαφής με τη βάση στους μη συνδεδεμένους χρήστες.

5.2 *Αρχεία εφαρμογής και εργαλεία ανάπτυξης*

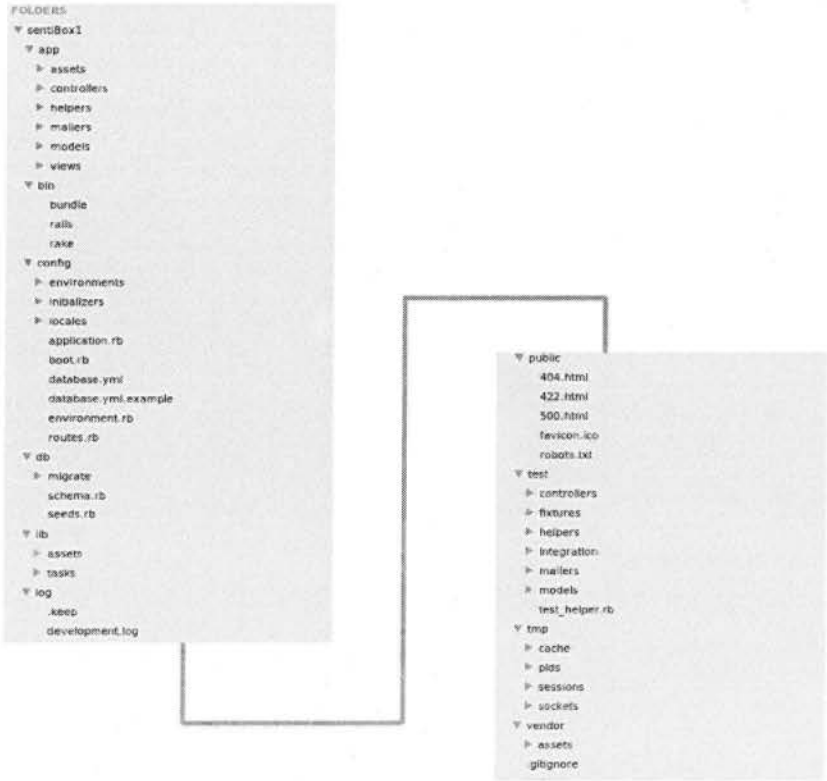
Τα gems που χρησιμοποιήθηκαν είναι τα παρακάτω:

```
source 'https://rubygems.org'

# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
gem 'rails', '4.0.2'

# Use postgreSQL as the database
gem 'pg'

# Application server
gem 'puma'

# Use SCSS for stylesheets
gem 'sass-rails', '~> 4.0.0'

# Use Uglifier as compressor for JavaScript assets
gem 'uglifier', '>= 1.3.0'

gem "therubyracer"
gem "less-rails" #Sprockets (what Rails 3.1 uses for its asset pipeline) supports LESS
gem "twitter-bootstrap-rails"

# Imports csv files
gem 'roo'

# Use CoffeeScript for .js.coffee assets and views
```

```
gem 'coffee-rails', '~> 4.0.0'

# See https://github.com/sstephenson/execjs#readme for more supported runtimes
# gem 'therubyracer', platforms: :ruby

# Use jquery as the JavaScript library
gem 'jquery-rails'

# Turbolinks makes following links in your web application faster. Read
more: https://github.com/rails/turbolinks
gem 'turbolinks'

# Build JSON APIs with ease. Read more: https://github.com/rails/jbuilder
gem 'jbuilder', '~> 1.2'

# User authentication
gem 'devise'
gem 'bcrypt-ruby', '~> 3.1.2'

#User authorization
gem 'cancan'

# Generates data
gem 'faker'


# Pagination
gem 'kaminari'

group :doc do
 # bundle exec rake doc:rails generates the API under doc/api.
 gem 'sdoc', require: false
end

group :production do

  gem 'rails_12factor'
end

group :development do
  # Debugging tool
  gem 'pry'
end
# Use ActiveModel has_secure_password
```

```
# gem 'bcrypt-ruby', '~> 3.1.2'

# Use unicorn as the app server
# gem 'unicorn'

# Use Capistrano for deployment
# gem 'capistrano', group: :development

# Use debugger
# gem 'debugger', group: [:development, :test]
ruby "2.0.0"
```

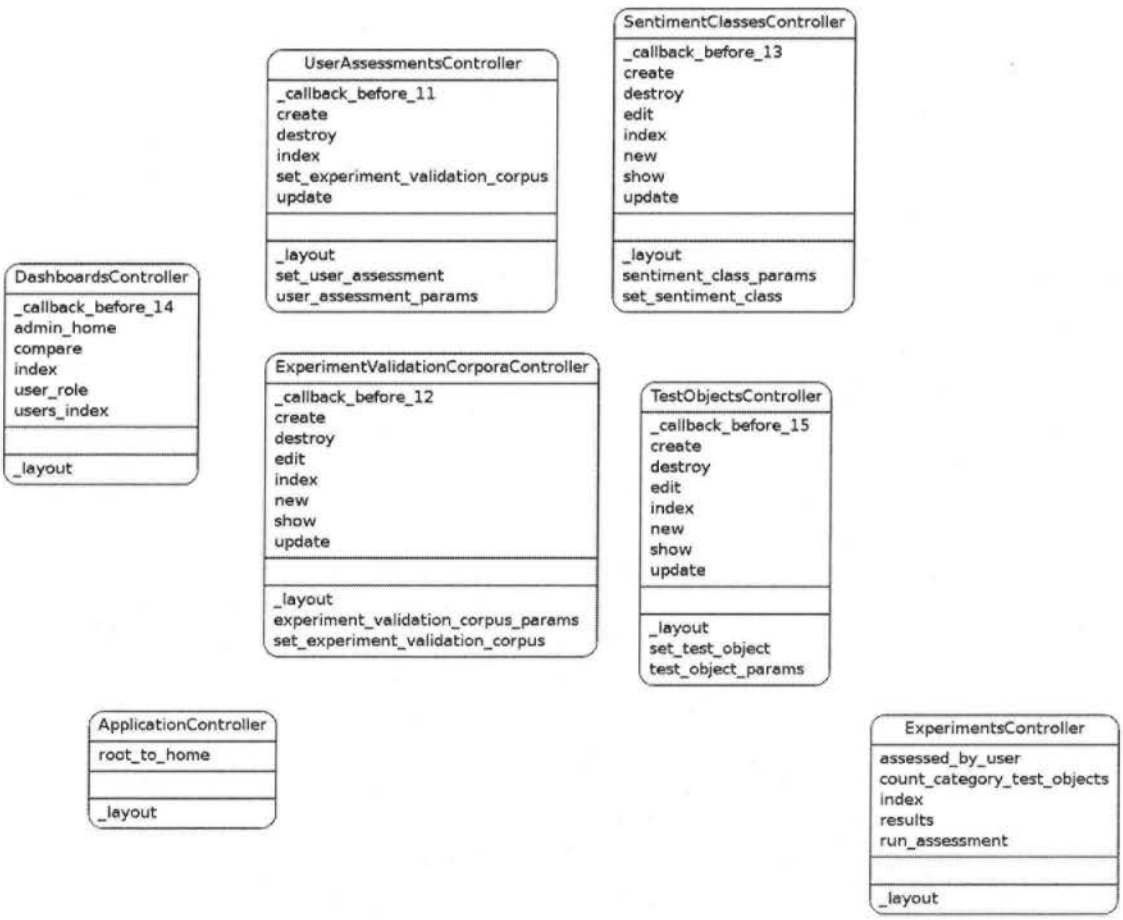
Τα αρχεία της εφαρμογής είναι τα παρακάτω:



Εικόνα 4.2.9-1: Δομή φακέλων της εφαρμογής Sentibox

5.3 Δομή της εφαρμογής

Παρατίθενται εικόνες που περιγράφουν τη δομή του κώδικα της εφαρμογής και πιο συγκεκριμένα την δομή και των τρόπο σύνδεσης των controllers που αποτελούν την καρδία του συστήματος αλλά και τις ενέργειες που ο χρήστης μπορεί να εκτελέσει στις επιμέρους σελίδες της εφαρμογής. Οι controllers οδηγούν τον χρήση μέσα απο μονοπάτια που ορίζουν στις ενέργειες που επιζητούν να κάνουν στην εφαρμογή. Η Ruby on Rails έχει μια βιβλιοθήκη η όποια καθιστά εύκολη την αναπαράσταση των controllers αλλά και την αποθήκευση στη μορφή εικόνας που επιθυμεί ο προγραμματιστής. Το διάγραμμα κλάσεων για τους controllers δημιουργήθηκε χρησιμοποιώντας το gem "railroady" με την εντολή railroady -C | neato -Tpng > controllers.png.



Εικόνα 4.2.9-1: Δομή και σύνδεση των controllers

5.4 Λειτουργίες χρηστών

Στην αρχική σελίδα οποιοσδήποτε χρήστης ενημερώνεται για τον σκοπό της εφαρμογής και του δίνεται η δυνατότητα συνδεθεί (sign in) στο σύστημα από την μπάρα που βρίσκεται στην κορυφή αν είναι ήδη εγγεγραμμένος ή να εγγραφεί (sign up).

Εικόνα 4.2.9-1: Αρχική σελίδα μη εγγεγραμμένου χρήστη

5.4.1 Δημιουργία χρήστη - σύνδεση χρήστη

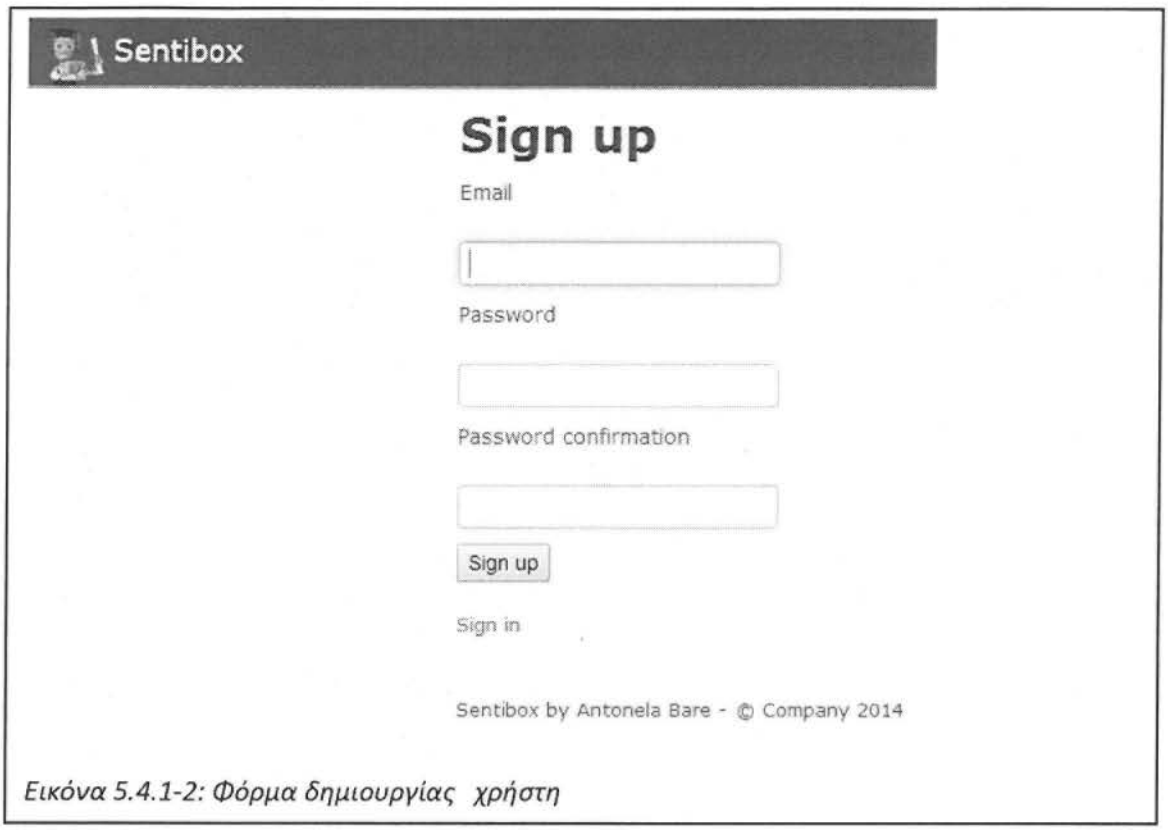
Η δημιουργία λογαριασμού στο σύστημα SentiBox είναι πολύ απλή. Στην αρχική σελίδα επιλέγοντας το σύνδεσμο sign up εμφανίζεται η φόρμα συμπλήρωσης με τρία υποχρεωτικά πεδία:

- E-mail. Υποχρεωτικά ενα έγκυρο e-mail το οποίο θα χρησιμοποιείται για τη σύνδεση του

χρήστη.

- Password (κωδικός). Ο μυστικός κωδικός που θα χρησιμοποιηθεί σε συνδυασμό με το e-mail για τη σύνδεση του χρήστη στο σύστημα. Ο κωδικός θα πρέπει να είναι τουλάχιστον 8 χαρακτήρες.
- Password Confirmation (επιβεβαίωση κωδικού). Στο πεδίο αυτό ο χρήστης επαναλαμβάνει το κωδικό του για να αποφευχθούν τυχόν λάθη.

Εικόνα 5.4.1-1: Φόρμα σύνδεσης χρήστη



Εικόνα 5.4.1-2: Φόρμα δημιουργίας χρήστη

Από τη στιγμή που ο χρήστης δημιουργήσει λογαριασμό μπορεί να συνδεθεί στο Sentibox εισάγοντας μόνο το e-mail και τον κωδικό που εισήγαγε κατά την εγγραφή του.

Επιλέγοντας το κουτί Remember me δε χρειάζεται να εισάγει τα στοιχεία σύνδεσης κάθε φορά που εισέρχεται στην εφαρμογή. Τέλος υπάρχει και η επιλογή Forgot your password όπου ο χρήστης έχει τη δυνατότητα να ανακτήσει τον κώδικό του εισάγοντας μόνο το email του.

Εικόνα 5.4.1-3: Λειτουργία ανάκτησης κωδικού

5.4.2 Λειτουργίες διαχειριστών - απλών χρηστών

Κάθε εφαρμογή έχει τους απλούς χρήστες που είναι περιορισμένες οι ενέργειες τους στη εφαρμογή από την κατασκευή της, έχει και τον διαχειριστή πού έχει πλήρη δικαιώματα στην εφαρμογή. Με τον ίδιο ακριβώς τρόπο κατασκευάστηκε και το Sentibox, παρέχοντας μόνο μια λειτουργία στούς απλούς χρήστες, αυτήν της αξιολόγησης κειμένων και τις υπολοίπες τις ανέθεσε στον διαχειριστή.

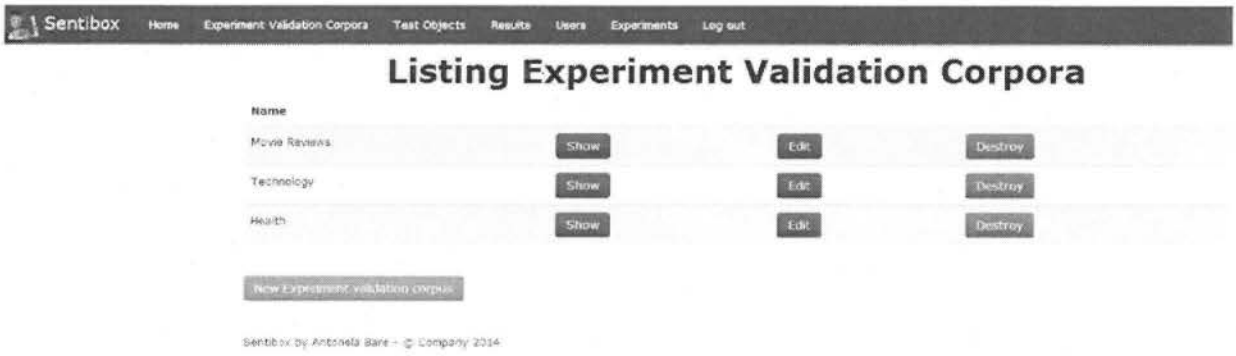
Αρχική σελίδα

Η αρχική σελίδα είναι αρκετά απλή και αποτελείται απο ένα γρήγορο μενού με όλες τις λειτουργίες όπου ο διαχειριστής μπορεί να βλέπει από όποια σελίδα και αν βρίσκεται.

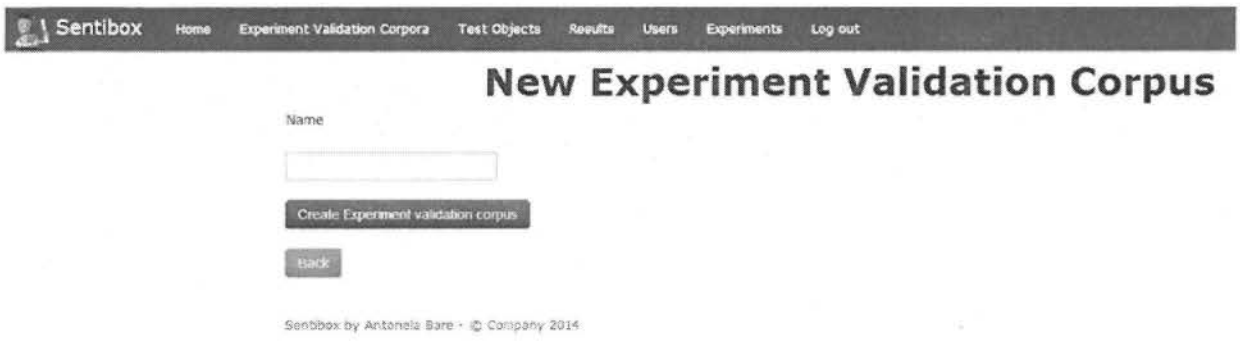
Εικόνα 5.4.2-1: Αρχική σελίδα διαχειριστή

Experiment Validation Corpora

Από αυτό το σημείο άν επιλέξει το κουμπί Experiment Validation Corpora μπορεί να δεί όλες τις κατηγορίες που υπάρχουν για κατηγοριοποίηση. Σε αυτό το σημείο έχει τη δυνατότητα να τις επεξεργαστεί να δημιουργήσει νέα κατηγορία αλλά και να τις διαγράψει μαζί με όλα τα κείμενα που περιέχουν.



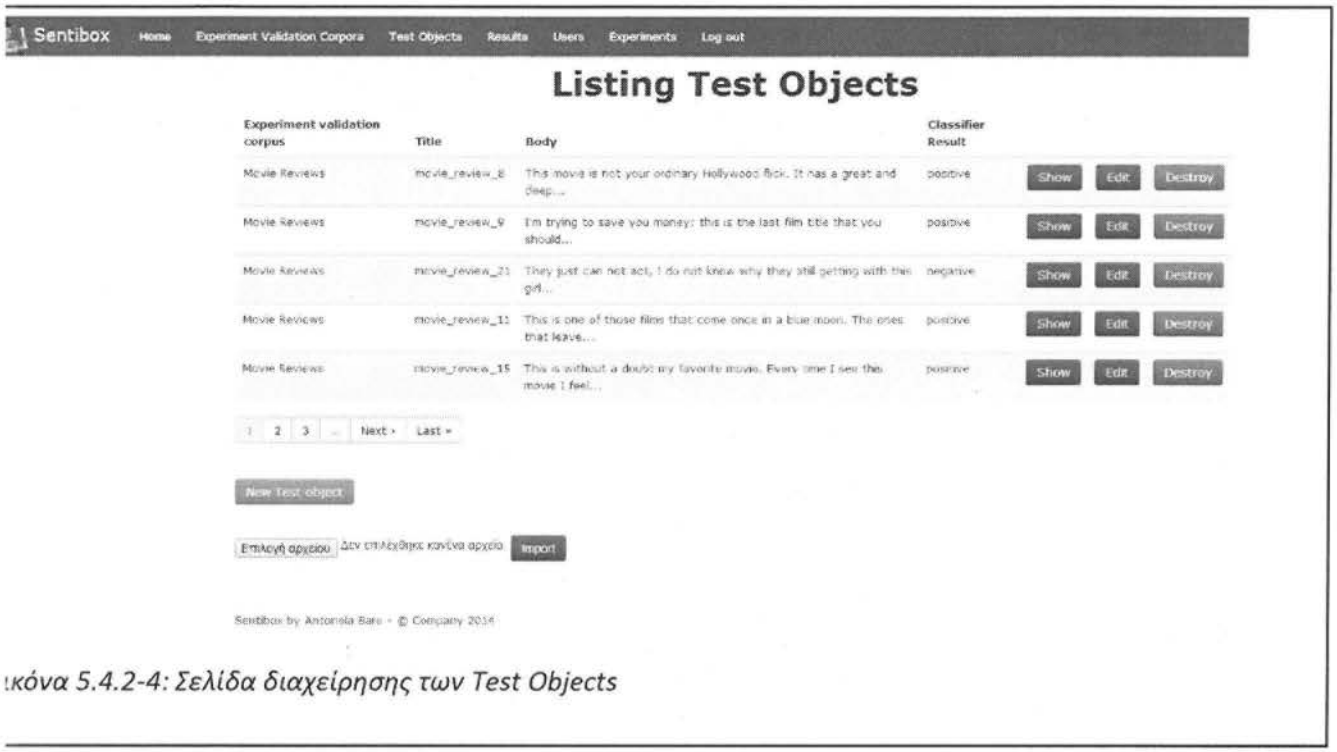
Εικόνα 5.4.2-2: Λίστα με τις κατηγορίες που διαχειρίζεται ο διαχειριστής



Εικόνα 5.4.2-3: Δημιουργία νέας κατηγορίας από το διαχειριστή

Test Objects

Αν επιλέξει από το μενού το κουμπί Test Objects έχει τη δυνατότητα να δεί όλα τα κείμενα που υπάρχουν προς κατηγοριοποίηση καθώς και σε ποία κατηγορία ανήκουν και πώς τα έχει βαθμολογήσει ο κατηγοριοποιητής μας. Όπως μπορούμε να δούμε μπορεί να δημιουργήσει ένα νέο Test Object πατώντας το πράσινο κουμπί κάτω αριστερά New Test Object. Δεξία από το κάθε Test Object έχει τα κουμπιά Show Edit και Destroy όπου ο διαχειριστής μπορεί να δεί, να επεξεργαστεί και να διαγράψει τα αντικείμενα. Επειδή όπως είναι φυσικό ο αριθμός των Test Objects μπορεί να είναι μεγάλος και αυτό να καθιστά τη πλοήγηση δύσκολη στον χρήστη προστέθηκε μία λειτουργία πλοήγησης. Τέλος γιά να μήν είναι αναγκασμένος ο χρήστης να ανεβάζει το κάθε κείμενο ξεχωριστά του δίνεται η δυνατότητα να τα ανεβάζει αυτοματοποιημένα επιλέγοντας ένα csv αρχείο με όλα τα δεδομένα.



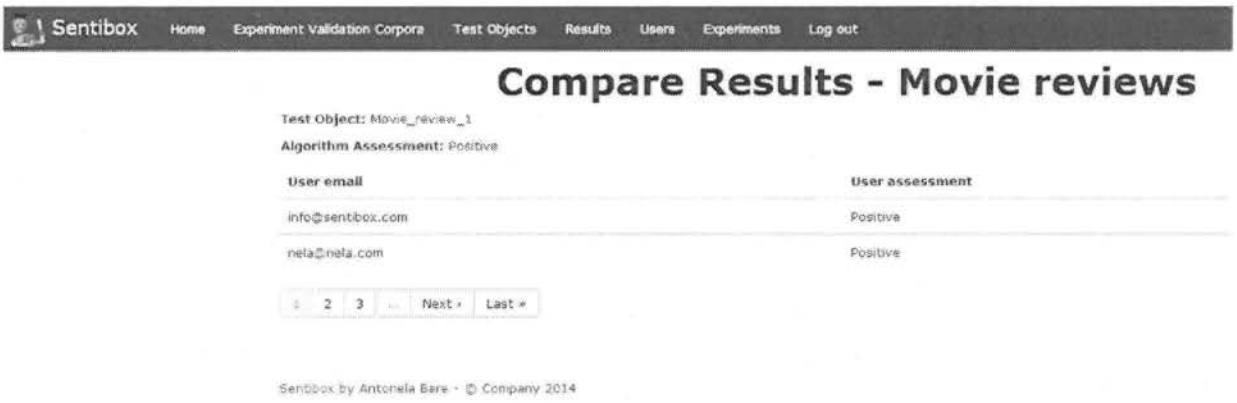
Εικόνα 5.4.2-4: Σελίδα διαχείρησης των Test Objects

Compare Results

Σε αυτό το σημείο ο διαχειριστής μπορεί να επιλέξει μία κατηγορία για να συγκρίνει τα αποτελέσματα όλων των χρηστών.

Εικόνα 5.4.2-5: Σελίδα σύγκρισης αποτελεσμάτων ανά κατηγορία

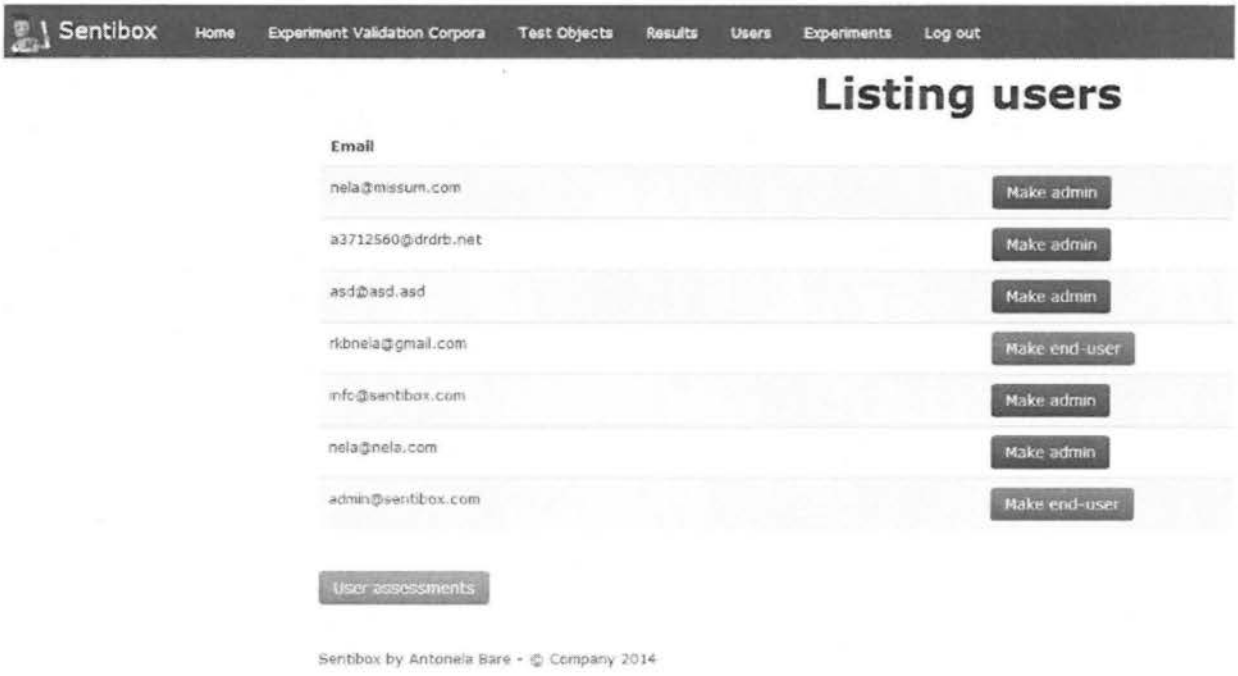
Αν επιλέξει μία κατηγορία του εμφανίζονται σε κάθε σελίδα τα κείμενά της καθώς επίσης την βαθμολογία του αλγορίθμου, τα email των χρηστών και πώς το βαθμολόγησε ο κάθε χρήστης το συγκεκριμένο κείμενο.



Εικόνα 5.4.2-6: Σελίδα σύγκρισης αποτελεσμάτων ανά Test Object

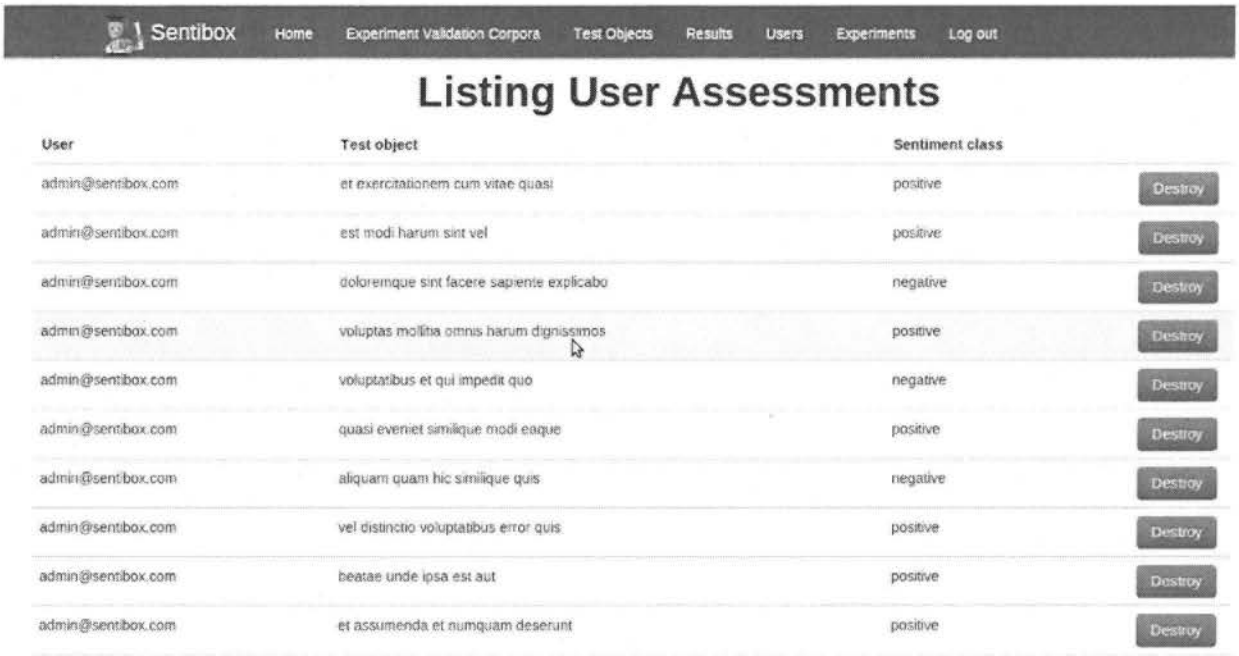
Users

Σε αυτό το σημείο εμφανίζονται όλοι οι εγγεγραμμένοι χρήστες είτε είναι απλοί είτε έχουν δικαιώματα διαχειριστή. Ο διαχειριστής έχει την δυνατότητα να μετατρέψει έναν απλό χρήστη σε διαχειριστή παρέχοντάς του όλα τα δικαιώματα αλλά και το αντίστροφο, δηλαδή να αφαιρέσει όλα τα δικαιώματα από έναν διαχειριστή και να τον κάνει απλό χρήστη.



Εικόνα 5.4.2-7: Λίστα εμφάνισης και επεξεργασίας χρηστών

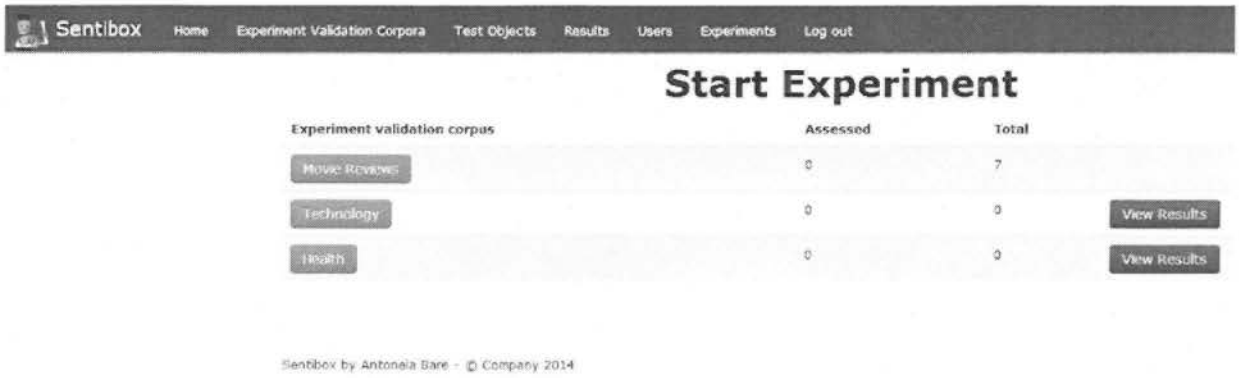
Πατώντας το πράσινο κουμπί ο διαχειριστής έχει τη δυνατότητα να δεί συγκεντρωτικά τα αποτελέσματα και να τα διαγραψει αν το θεωρήσει σκόπιμο.



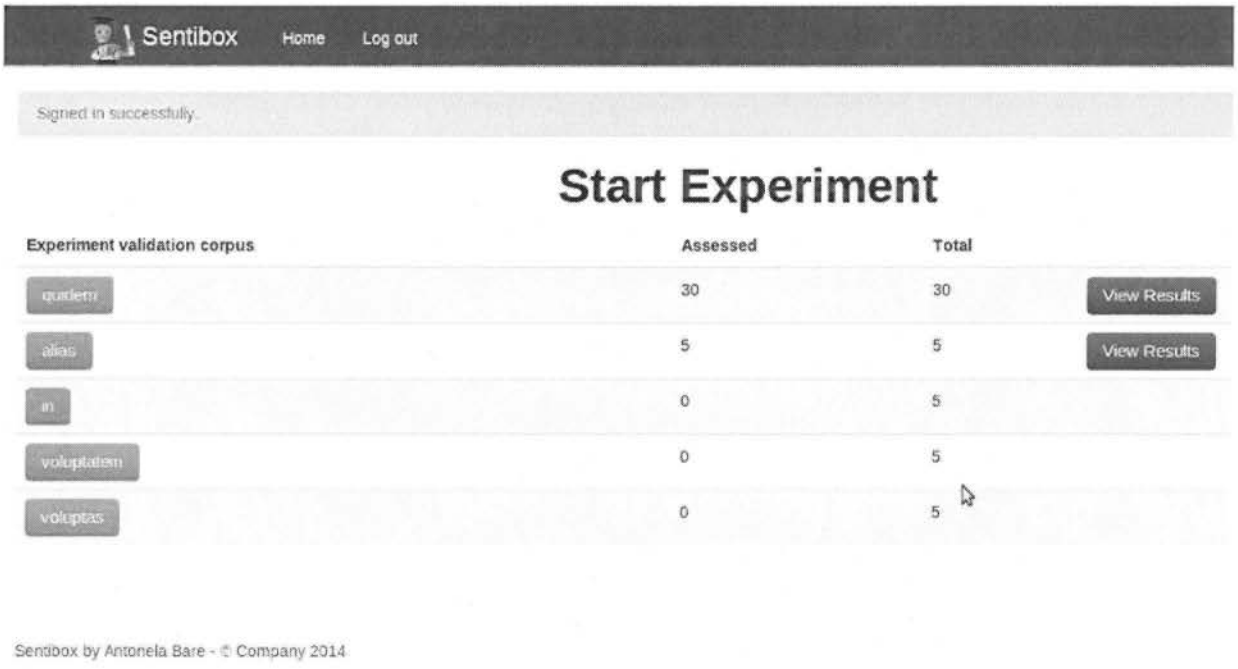
Εικόνα 5.4.2-8: Λίστα εμφάνισης και διαγραφής των Test Object

Experiments

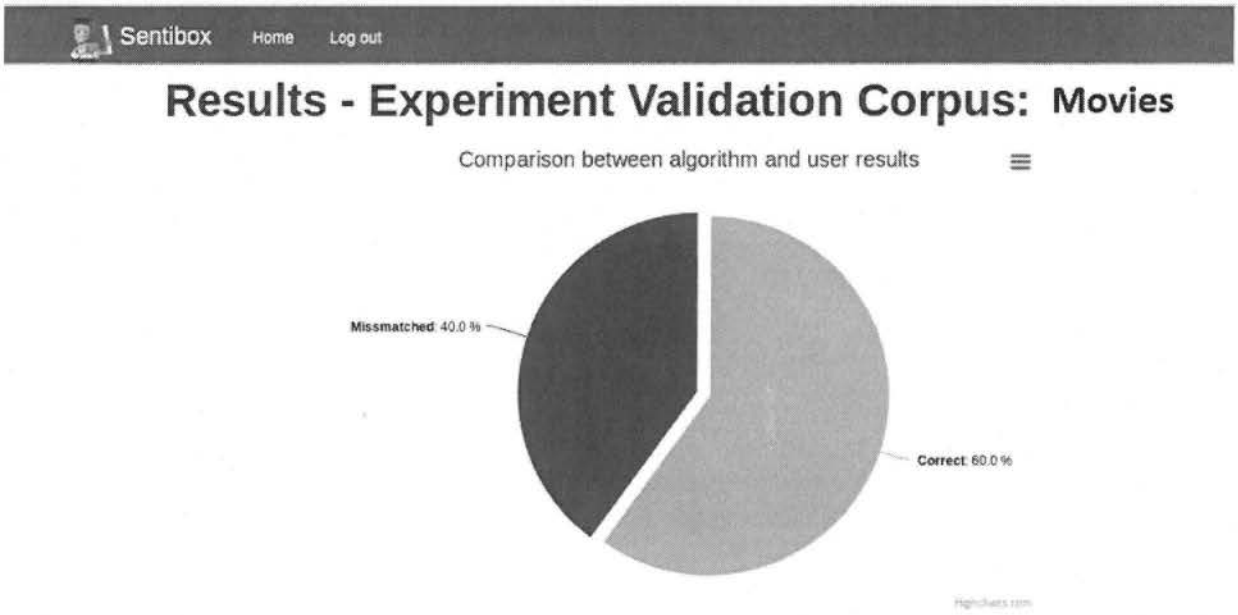
Τελευταία ουσιαστική λειτουργία του χρήστη πρίν απο την αποσύνδεσή του (log out), είναι το Experiments. Αυτή είναι και η μοναδική κοινή λειτουργία που έχουν οι διαχειριστές με τους απλούς χρήστες. Σε αυτή τη σελίδα παρουσιάζονται όλα τα πειράματα καθώς και ο συνολικός αριθμός των κειμένων που περιέχουν και πόσα απο αυτά ο χρήστης έχει βαθμολογήσει. Ο χρήστης έχει τη δυνατότητα να δεί τα αποτελέσματα μίας κατηγορίας μόνο εφόσον έχει βαθμολογήσει όλα της τα κείμενα. Εφόσον έχει ολοκληρώσει την βαθμολόγηση όλων των κειμένων μιας κατηγορίας του εμφανίζεται το μπλέ κουμί δεξιά View Results, αν το πατήσει μπορεί να δεί και να συγκρίνει τη βαθμολογία του με αυτή του κατηγοριοποιητή. Παράλληλα του εμφανίζεται και ένα γράφημα όπου απεικονίζει το ποσοστό επιτυχίας και αποτυχίας του συνόλου.



Εικόνα 5.4.2-9: Σελίδα διαχειριστή για εκκίνηση δικών του πειραμάτων



Εικόνα 5.4.2-10: Σελίδα εμφάνισης κατηγοριών στον απλό χρήστη



Εικόνα 5.4.2-11: Εμφάνιση τελικού γραφηματος αφου έχει ολοκληρωθεί ένα πείραμα

Εικόνα 5.4.2-12: Τρόπος ανάθεσης βαθμολογίας στα κείμενα απο τους χρήστες

5.5 *Προβλήματα που αντιμετωπίστηκαν - Συμπεράσματα*

Σε όλα τα επιχειρήματα του ανθρώπου μικρά η μεγάλα παρουσιάζονται προβλήματα και προβληματισμοί σε όλα τα στάδια υλοποιήσης τους. Έτσι και κατά την δημιουργία και σχεδίαση της εφαρμογής SentiBox εμφανίστηκαν προβλήματα τα οποία, έπειτα απο εμπεριστατωμένη έρευνα και αναζήτηση στο διαδίκτυο, επιλύθηκαν ωστε να καταλήξουμε στο επιθυμητό αποτέλεσμα.

Κατά τα πρώτα στάδια της εεγκατάστασης και εκμάθησης τησν γλώσσας προγραμματισμού Ruby on Rails παρουσιαστηκε το πρόβλημα της ύπαρξης διαφορετικών εκδόσεων λειτουργικού συστήματος. Οι εκδόσεις που χρησιμοποιήσαμε ήταν Ubuntu 12.04, Ubuntu 13.10 και Mac OSX 10.6. Ο λόγος ύπαρξής τους ήταν γιατί ήταν αδύνατη η εργασία μόνο από ένα σύστημα αλλά και για να παρατηρηθεί ο τρόπος λειτουργίας της εφαρμογής από διαφορετικά λειτουργικά συστήματα. Μετά από αναζήτηση στο διαδίκτυο βρέθηκε τρόπος της εύκολης εγκατάστασης της πλατφόρμας σε διαφορετικά λειτουργικά συστήματα.

Ένα άλλο χαρακτηριστικό της εφαρμογής είναι η εφαρμογή της βιβλιοθήκης Twitter Bootstrap. Κατά την εγκατάστασή της παρουσιάστηκαν αρκετά προβλήματα και εφαρμόστηκαν δίαφοροι τρόποι και διαφορετικές βιβλιοθήκες μέχρι να καταλήξουμε στην πιο κατάλληλη. Η λύση βρέθηκε έπειτα απο αναζήτηση στα εγχειρίδια της βιβλιοθήκης και της κοινότητας της Ruby on Rails.

Αξίζει να αναφερθεί και ο λόγος επιλογής των ονομάτων πού απαρτίζουν την εφαρμογή. Κατά τον σχεδιασμό του σχήματος της βάσης επιλέχθησαν απλά ονόματα όπως Categories, Scores, Texts, User και User Scores. Επειδή όμως η εφαρμογή μας διαχειρίζεται κυρίως επιστημονικούς όρους έπρεπε να επιλέξουμε πιο προσεκτικά την ονοματολογία και καταλήξαμε στην επιλογή των εξής όρων: Experiment Validation Corpus, User Assesment, Test Objects και User.

Λόγο της ύπαρξης μεγάλου αριθμό κειμένων ανα κατηγορία ο χρήστης έρχεται αντιμέτωπος να ανατρέξει σε μεγάλο αριθμό γραμμών για να βαθμολογήσει όλα τα κείμενα. Τη λύση στο πρόβλημα αυτό ήρθε να λύσει η βιβλιοθήκη Kaminari που είναι για σελιδοποίηση. Η βιβλιοθήκη αυτή ελαχιστοποίησε τον αριθμό των γραμμών προσφέροντας μία εύκολη πλοήγηση.

Ένα ακόμη σοβαρό πρόβλημα που αντιμετωπίστηκε είναι ο διαχωρισμός των λειτουργιών του απλού χρήστη και του διαχειριστή. Όπως αναφέραμε πιο πάνω η εφαρμογή μας βασίζεται περισσότερο στις λειτουργίες του διαχειριστή για να μπορέσει ο απλός χρήστης να κάνει την βασική του λειτουργία η οποία είναι η κατηγοριοποίηση κειμένων. Για αυτό χρησιμοποιήσαμε την βιβλιοθήκη device η οποία με πολύ ευκολο τρόπο παραχωρεί δικαιώματα στον χρήστη που θέλουμε και αφαιρεί άλλα από τον απλό χρήστη.

Τέλος, η εφαρμογή για να μας δώσει τα αποτελέσματα που θέλουμε, εκτός από αρχεία κώδικα Ruby περιέχει και αρχεία με javascript, CSS και εικόνες που λέγονται assets. Τα αρχεία αυτά πρέπει να περάσουν από σύνταξη (compilation) για να τρέξουν παράλληλα με τον κύριο κορμό της εφαρμογής. Η λειτουργία αυτή στο υπολογιστικό νέφος ήταν εσφαλμένη με αποτέλεσμα τη δυσλειτουργία της εφαρμογής. Η λύση ήρθε μέσω της αναζήτησης σε παρόμοια προβλήματα προγραμματιστών και συγκεκρημένα από τη σελίδα stackoverflow.com.

Μετά το πέρας του σχεδιασμού και της υλοποίησης της εφαρμογής καταλήξαμε στο επιθυμητό αποτέλεσμα το οποίο είναι να βοηθηθούν οι ερευνητές που δουλεύουν πάνω στο κομμάτι της συναισθηματικής ανάλυσης κειμένων. Οι χρήστες μέσω ενός απλού και εύχρηστου εργαλείου μπορούν να κάνουν χειροκίνητη αξιολόγιση του συναισθήματος των κειμένων βοηθώντας την τελική εξαγωγή συμπεράσματων για την καλή λειτουργία των classifiers.

Παράρτημα

Κατάλογος κώδικα – Κώδικας

File – H:\SentiBox\production.rb

```
#Εδώ βρίσκονται οι ρυθμίσεις για την εφαρμογή στο στάδιο της #παραγωγής.
#Η μόνη αλλαγή που έγινε είναι η προσθήκη του κώδικα στο τέλος για #την αποστολή
#και λήψη μηνυμάτων ηλεκτρονικού ταχυδρομείου μέσω του λογαριασμού #του
#συστήματος στο Add-on SendGrid.

Sentibox::Application.configure do
  # Settings specified here will take precedence over those in con-
fig/application.rb.

  # Code is not reloaded between requests.
  config.cache_classes = true

  # Eager load code on boot. This eager loads most of Rails and
  # your application in memory, allowing both thread web servers
  # and those relying on copy on write to perform better.
  # Rake tasks automatically ignore this option for performance.
  config.eager_load = true

  # Full error reports are disabled and caching is turned on.
  config.consider_all_requests_local       = false
  config.action_controller.perform_caching = true

  # Enable Rack::Cache to put a simple HTTP cache in front of your application
  # Add `rack-cache' to your Gemfile before enabling this.
  # For large-scale production use, consider using a caching reverse proxy like
nginx, varnish or squid.
  # config.action_dispatch.rack_cache = true

  # Disable Rails's static asset server (Apache or nginx will already do this).
  config.serve_static_assets = false

  # Compress JavaScripts and CSS.
  config.assets.js_compressor = :uglifier
  # config.assets.css_compressor = :sass

  # Do not fallback to assets pipeline if a precompiled asset is missed.
  config.assets.compile = false

  # Generate digests for assets URLs.
  config.assets.digest = true

  # Version of your assets, change this if you want to expire all your assets.
  config.assets.version = '1.0'

  # Specifies the header that your server uses for sending files.
  # config.action_dispatch.x_sendfile_header = "X-Sendfile" # for apache
  # config.action_dispatch.x_sendfile_header = 'X-Accel-Redirect' # for nginx
                          # Force all access to the app over SSL, use Strict-
Transport-Security, and use secure cookies.

  # config.force_ssl = true

  # Set to :debug to see everything in the log.
  config.log_level = :info
```

```
    # Prepend all log lines with the following tags.
    # config.log_tags = [ :subdomain, :uuid ]

    # Use a different logger for distributed setups.
    # config.logger = ActiveSupport::TaggedLogging.new(SyslogLogger.new)

    # Use a different cache store in production.
    # config.cache_store = :mem_cache_store

    # Enable serving of images, stylesheets, and JavaScripts from an asset server.
    # config.action_controller.asset_host = "http://assets.example.com"

    # Precompile additional assets.
    # application.js, application.css, and all non-JS/CSS in app/assets folder are
already added.
    # config.assets.precompile += %w( search.js )
    # Ignore bad email addresses and do not raise email delivery errors.
    # Set this to true and configure the email server for immediate delivery to raise
delivery errors.
    # config.action_mailer.raise_delivery_errors = false

    # the I18n.default_locale when a translation can not be found).
    config.i18n.fallbacks = true

    # Send deprecation notices to registered listeners.
    config.active_support.deprecation = :notify

    # Disable automatic flushing of the log to improve performance.
    # config.autoflush_log = false

    # Use default logging formatter so that PID and timestamp are not suppressed.
    config.action_mailer.default_url_options = { :host => 'www.sentibox.eu'}
    config.action_mailer.delivery_method = :smtp
    config.action_mailer.raise_delivery_errors = false
  config.action_mailer.smtp_settings = {
    address: "smtp.gmail.com",
    port: 587,
    domain: "www.sentibox.eu",
    authentication: "plain",
    enable_starttls_auto: true
    user_name: ENV['GMAIL_USERNAME'],
    password: ENV['GMAIL_PASSWORD'],

  }

    End
```

File – H:\SentiBox\application.rb

```
# Στο αρχείο αυτό βρίσκονται οι ρυθμίσεις για τον τύπο της βάσης #δεδομένων,
# τα regional settings, δεδομένα που θα φιλτράρονται στο log file
# αλλά και ρυθμίσεις που χρειάζονται για το υπολογιστικό νέφος Heroku

require File.expand_path('../boot', __FILE__)

require 'rails/all'
require 'CSV'

# Require the gems listed in Gemfile, including any gems
# you've limited to :test, :development, or :production.
Bundler.require(:default, Rails.env)

module Sentibox
  class Application < Rails::Application
    # Settings in config/environments/* take precedence over those specified here.
    # Application configuration should go into files in config/initializers
    # -- all .rb files in that directory are automatically loaded.

    # Set Time.zone default to the specified zone and make Active Record auto-
convert to this zone.
    # Run "rake -D time" for a list of tasks for finding time zone names. Default
is UTC.
     config.time_zone = 'Athens'

    # The default locale is :en and all translations from config/locales/*.rb,yml
are auto loaded.
    # config.i18n.load_path += Dir[Rails.root.join('my', 'locales',
'*.{rb,yml}').to_s]
    # config.i18n.default_locale = :de
  end
end
```

File – H:\SentiBox\Results.js.coffee

```
//Αρχείο που περιλαμβάνει τη συνάρτηση javascript για την εμφάνιση του διαγράμματος
// στην σελίδα της σύγκρισης αποτελεσμάτων
$ ->
  $("#comparison_chart").highcharts
    chart:
      plotBackgroundColor: null
      plotBorderWidth: null
      plotShadow: false

    title:
      text: "Comparison between algorithm and user results"

    tooltip:
      pointFormat: "{series.name}: <b>{point.percentage:.1f}%</b>"

    plotOptions:
      pie:
        allowPointSelect: true
        cursor: "pointer"
        dataLabels:
          enabled: true
          format: "<b>{point.name}</b>: {point.percentage:.1f} %"
          style:
            color: (Highcharts.theme and Highcharts.theme.contrastTextColor) or
"black"

    series: [
      type: "pie"
      name: "User assessments"
      data: [
        {
          name: "Correct"
          y: parseInt($('#user_correct').text())
          sliced: true
          selected: true
        }
        [
          "Missmatched"
          100 - parseInt($('#user_correct').text())
        ]

      ]
    ]

  return
```

1 **File – H:\SentiBox\custom.css.scss**

```
//Συγκεντρωτικοί κανόνες για την καλύτερη εφαρμογή της βιβλιοθήκης Twitter Bootstrap
body {
  background-image: white;
}

h1 {
  text-align: center
}

table {
    margin: 0 auto;
}

footer{
    margin-top: 40px;
}
.container-fluid {
background-color: #3E4449;
}
.navbar {
overflow: hidden;
margin-bottom: 0px;
}
.main-wrapper {
  width: 1170px;
  margin: 0 auto;
  background-color: white;
}
.navbar .nav {
  margin-top:5px;
}
.navbar .brand:hover {
  background-color: #3E4449;
  text-decoration: none;
  color: #777777;
  text-shadow: 0 1px 0 #777777;
}
.navbar .brand {
float: left;
display: block;
padding: 8px 20px 4px;
margin-left: 50px;
font-size: 20px;
font-weight: 200;
color: #ecf0f1;
text-shadow: 0 1px 0 #ecf0f1;
}
.navbar-inner {

  padding-left: 0px;
  padding-right: 0px;
  background-color: #3E4449;
  background-image: -webkit-linear-gradient(top, #333333, #222222);
```

```
  border-bottom: 2px solid #e67d21;
}
Custom.css.scss
.navbar .nav>li>a {
float: none;
padding: 10px 15px 10px;
color: #ecf0f1;
text-decoration: none;
text-shadow: 0 1px 0 #ecf0f1;
display: block;
}
.navbar .nav>li>a:hover {
  color: #777777;
  text-shadow: 0 1px 0 #777777;

}
.alert { margin-top: 20px;}
.dashboard a {
  margin: 30px 10px;
  padding: 10px;
}
a.btn-primary:visited, a.btn-success:visited, a.btn-danger:visited {color:#fff}

.btn-default {
    color: #523352;
}

h4.result_categories {
  float: left;
}
.result_categories li {
  list-style: none;
  float: left;
  margin: 5px 10px;
}
img.style_image{
  margin: 20px auto;
  display: block;
  width: 40%;
}

hr {
margin: 15px 0;
border: 0;
border-top: 6px solid #EBEBEB;
border-bottom: 17px solid #ffffff;
}

.text-red {color: rgb(228, 0, 43);}
p {
font-family: verdana, arial, helvetica, sans-serif;
font-size: 16px;
line-height: 24px;
color: #3b455f;
}
```

File – H:\SentiBox\ user_assessments_controller.rb

```
//Ο Controller αυτός είναι υπεύθυνος για την διαχείριση των βαθμολογιών των
//χρηστών.


class UserAssessmentsController < ApplicationController
  authorize_resource class: false
  before_action :set_user_assessment, only: [:show, :edit, :update, :destroy]

  # GET /user_assessments
  # GET /user_assessments.json
  def index
    @user_assessments = UserAssessment.all.page(params[:page]).per(20)
  end

  # POST /user_assessments
  # POST /user_assessments.json
  def create
    @user_assessment = UserAssessment.new(user_assessment_params)
    @user_assessment.user_id = current_user.id
    set_experiment_validation_corpus
    respond_to do |format|
      if @user_assessment.save
        format.html { redirect_to :back, notice: 'User assessment was successfully
created.' }
        format.json { render action: 'show', status: :created, location:
@user_assessment }
      else
        if @user_assessment.errors[:user_id] == ["has already been taken"]
          update and return
        end
        format.html { render action: 'new' }
        format.json { render json: @user_assessment.errors, status:
:unprocessable_entity }
      end
    end
  end

  def update
    @user_assessment = UserAssessment.where(user_id: current_user.id,
test_object_id: user_assessment_params[:test_object_id] ).first
    respond_to do |format|
      if @user_assessment.update_attributes(sentiment_class_id: us-
er_assessment_params[:sentiment_class_id])
        format.html { redirect_to :back, notice: 'User assessment was successfully
updated.' }
        format.json { head :no_content }
      else
        format.html { render action: 'edit' }
        format.json { render json: @user_assessment.errors, status:
:unprocessable_entity }
      end
    end
  end

```

```
56    # DELETE /user_assessments/1
57    # DELETE /user_assessments/1.json
58    def destroy
59      @user_assessment.destroy
60      respond_to do |format|
61        format.html { redirect_to user_assessments_url }
62        format.json { head :no_content }
63      end
64    end
65
66    def set_experiment_validation_corpus
67      @user_assessment.experiment_validation_corpus_id = Tes-
68  tObject.find(user_assessment_params[:test_object_id]).experiment_validation_corpus_
69  id
70    end
71
72    private
73      # Use callbacks to share common setup or constraints between actions.
74      def set_user_assessment
75        @user_assessment = UserAssessment.find(params[:id])
76      end
77
78      # Never trust parameters from the scary internet, only allow the white list
79  through.
80      def user_assessment_params
81        params.require(:user_assessment).permit(:user_id, :test_object_id,
82  :sentiment_class_id)
83      end
84  end
```

File – H:\SentiBox\ Test_objects_ controller.rb

```
//O Controller αυτός είναι υπεύθυνος για την διαχείριση του συνόλου κειμένων των
//κατηγοριων που υπάρχουν
class TestObjectsController < ApplicationController
  authorize_resource class: false
  before_action :set_test_object, only: [:show, :edit, :update, :destroy]

  # GET /test_objects
  # GET /test_objects.json
  def index
    @test_objects = TestObject.all.page(params[:page]).per(5)
  end

  # GET /test_objects/1
  # GET /test_objects/1.json
  def show
  end

  # GET /test_objects/new
  def new
    @test_object = TestObject.new
  end

  # GET /test_objects/1/edit
  def edit
  end

  # POST /test_objects
  # POST /test_objects.json
  def create
    @test_object = TestObject.new(test_object_params)

    respond_to do |format|
      if @test_object.save
        format.html { redirect_to @test_object, notice: 'Test object was success-
fully created.' }
        format.json { render action: 'show', status: :created, location:
@test_object }
      else
        format.html { render action: 'new' }
        format.json { render json: @test_object.errors, status:
:unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /test_objects/1
  # PATCH/PUT /test_objects/1.json
  def update
    respond_to do |format|
      if @test_object.update(test_object_params)
        format.html { redirect_to @test_object, notice: 'Test object was success-
fully updated.' }
        format.json { head :no_content }
      else
```

```
        format.html { render action: 'edit' }
        format.json { render json: @test_object.errors, status:
:unprocessable_entity }
      end
    end
  end

  # DELETE /test_objects/1
  # DELETE /test_objects/1.json
  def destroy
    @test_object.destroy
    respond_to do |format|
      format.html { redirect_to test_objects_url }
      format.json { head :no_content }
    end
  end

  def import_csv
    TestObject.import(params[:file])
    redirect_to test_objects_path, notice: "Test objects imported."
  end

  def import_body
    test_object = TestObject.find(params[:test_object_id])
    file_name = test_object.body
    file = File.open("#{Rails.root}/tmp/#{file_name}")
    test_object.body = file.read()
    test_object.save
    redirect_to(:back)
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_test_object
      @test_object = TestObject.find(params[:id])
    end

    # Never trust parameters from the scary internet, only allow the white list
through.
    def test_object_params
      params.require(:test_object).permit(:experiment_validation_corpus_id, :title,
:body, :sentiment_class_id)
    end
end
```

File – H:\SentiBox\ Sentiment_classes_ controller.rb

```
//Ο Controller αυτός είναι υπεύθυνος για την διαχείρηση των κλάσεων του συνόλου
//κειμένων των κατηγοριών που υπάρχουν

class SentimentClassesController < ApplicationController
  authorize_resource class: false
  before_action :set_sentiment_class, only: [:show, :edit, :update, :destroy]

  # GET /sentiment_classes
  # GET /sentiment_classes.json
  def index
    @sentiment_classes = SentimentClass.all
  end

  # GET /sentiment_classes/1
  # GET /sentiment_classes/1.json
  def show
  end

  # GET /sentiment_classes/new
  def new
    @sentiment_class = SentimentClass.new
  end

  # GET /sentiment_classes/1/edit
  def edit
  end

  # POST /sentiment_classes
  # POST /sentiment_classes.json
  def create
    @sentiment_class = SentimentClass.new(sentiment_class_params)

    respond_to do |format|
      if @sentiment_class.save
        format.html { redirect_to @sentiment_class, notice: 'Sentiment class was
successfully created.' }
        format.json { render action: 'show', status: :created, location:
@sentiment_class }
      else
        format.html { render action: 'new' }
        format.json { render json: @sentiment_class.errors, status:
:unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /sentiment_classes/1
  # PATCH/PUT /sentiment_classes/1.json
  def update
    respond_to do |format|
      if @sentiment_class.update(sentiment_class_params)
        format.html { redirect_to @sentiment_class, notice: 'Sentiment class was
successfully updated.' }
        format.json { head :no_content }
      else
```

```
57          format.html { render action: 'edit' }
58          format.json { render json: @sentiment_class.errors, status:
59  :unprocessable_entity }
60        end
61      end
62    end
63
64    # DELETE /sentiment_classes/1
65    # DELETE /sentiment_classes/1.json
66    def destroy
67      @sentiment_class.destroy
68      respond_to do |format|
69        format.html { redirect_to sentiment_classes_url }
70        format.json { head :no_content }
71      end
72    end
73
74    private
75      # Use callbacks to share common setup or constraints between actions.
76      def set_sentiment_class
77        @sentiment_class = SentimentClass.find(params[:id])
78      end
79
80      # Never trust parameters from the scary internet, only allow the white list
81  through.
82      def sentiment_class_params
83        params.require(:sentiment_class).permit(:name)
84      end
85  end
```

File – H:\SentiBox\ Experiments_ controller.rb

```
//Ο Controller αυτός είναι υπεύθυνος για την διαχείριση του συνόλου των κατηγοριών
//που υπάρχουν

class ExperimentsController < ApplicationController
  def index
    @experiments = []
    ExperimentValidationCorpus.all.each do |experiment|
      exp = {}
      exp[:name] = experiment.name
      exp[:assessed] = assessed_by_user(experiment).length
      exp[:total] = count_category_test_objects(experiment)
      @experiments << exp
    end
  end

  def assessed_by_user experiment
    UserAssessment.where(user_id: current_user.id, experiment_validation_corpus_id:
experiment.id)
  end

  def count_category_test_objects experiment
    TestObject.where(experiment_validation_corpus_id: experiment.id).length
  end

  def results
    @experiment_validation_corpus = ExperimentValidationCor-
pus.find_by_name(params[:format])
    # in case someone tries to see results (if he knows the URL) without having as-
sessed all test objects
    if assessed_by_user(@experiment_validation_corpus).length ==
count_category_test_objects(@experiment_validation_corpus)
      @results = UserAssessment.where(experiment_validation_corpus_id:
@experiment_validation_corpus.id, user_id: current_user.id)

      @score = UserAssessment.algorithm_comparison(@results, current_user)
    end
  end

  def run_assessment
    @experiment_validation_corpus = ExperimentValidationCor-
pus.find_by_name(params[:format])
    @test_objects = TestObject.where(experiment_validation_corpus_id:
@experiment_validation_corpus.id).page(params[:page]).per(1)
    @user_assessment = UserAssessment.new
  end
end
```

File – H:\SentiBox\ Experiments_Validation_Corpora_controller.rb

```
//Ο Controller αυτός είναι υπεύθυνος για την διαχείριση του συνόλου των κατηγοριών
//που υπάρχουν

class ExperimentValidationCorporaController < ApplicationController
  authorize_resource class: false
  before_action :set_experiment_validation_corpus, only: [:show, :edit, :update,
:destroy]

  # GET /experiment_validation_corpora
  # GET /experiment_validation_corpora.json
  def index
    @experiment_validation_corpora = ExperimentValidationCorpus.all
  end

  # GET /experiment_validation_corpora/1
  # GET /experiment_validation_corpora/1.json
  def show
  end

  # GET /experiment_validation_corpora/new
  def new
    @experiment_validation_corpus = ExperimentValidationCorpus.new
  end

  # GET /experiment_validation_corpora/1/edit
  def edit
  end

  # POST /experiment_validation_corpora
  # POST /experiment_validation_corpora.json
  def create
    @experiment_validation_corpus = ExperimentValidationCor-
pus.new(experiment_validation_corpus_params)

    respond_to do |format|
      if @experiment_validation_corpus.save
        format.html { redirect_to @experiment_validation_corpus, notice: 'Experi-
ment validation corpus was successfully created.' }
        format.json { render action: 'show', status: :created, location:
@experiment_validation_corpus }
      else
        format.html { render action: 'new' }
        format.json { render json: @experiment_validation_corpus.errors, status:
:unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /experiment_validation_corpora/1
  # PATCH/PUT /experiment_validation_corpora/1.json
  def update
    respond_to do |format|
      if @experiment_validation_corpus.update(experiment_validation_corpus_params)
        format.html { redirect_to @experiment_validation_corpus, notice: 'Experi-
ment validation corpus was successfully updated.' }
```

```
57            format.json { head :no_content }
58          else
59            format.html { render action: 'edit' }
60            format.json { render json: @experiment_validation_corpus.errors, status:
61    :unprocessable_entity }
62          end
63        end
64      end
65
66      # DELETE /experiment_validation_corpora/1
67      # DELETE /experiment_validation_corpora/1.json
68      def destroy
69        @experiment_validation_corpus.destroy
70        respond_to do |format|
71          format.html { redirect_to experiment_validation_corpora_url }
72          format.json { head :no_content }
73        end
74      end
75
76      private
77        # Use callbacks to share common setup or constraints between actions.
78        def set_experiment_validation_corpus
79          @experiment_validation_corpus = ExperimentValidationCorpus.find(params[:id])
80        end
81
82        # Never trust parameters from the scary internet, only allow the white list
83    through.
84        def experiment_validation_corpus_params
85          params.require(:experiment_validation_corpus).permit(:name)
86        end
87    end
```

File – H:\SentiBox\ Dashboards_ controller.rb

```
//Ο Controller αυτός είναι υπεύθυνος για την διαχείρηση των λειτουργιων του
//διαχειριστη

class DashboardsController < ApplicationController
  authorize_resource class: false
  def admin_home
  end

  def index
    @experiment_validation_corpora = ExperimentValidationCorpus.all
  end

  def compare
    @experiment_validation_corpus = ExperimentValidationCor-
pus.find(params[:format])
    @test_objects_by_corpus =
@experiment_validation_corpus.test_objects.page(params[:page]).per(1)
  end

  def users_index
    @users = User.all
  end

  def user_role
    @user = User.find(params[:format])
    if @user.role.present?
      @user.update_attributes(role: "")
    else
      @user.update_attributes(role: "admin")
    end
    redirect_to :back, notice: 'User role was successfully updated.'
  end
end
```

File – H:\SentiBox\ application.html.erb

```
1  //Ο Κώδικας HTML και Ruby ο οποίος δημιουργεί την αρχική σελίδα.
2
3  <!DOCTYPE html>
4  <html lang="en">
5    <head>
6      <meta charset="utf-8">
7      <meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1">
8      <meta name="viewport" content="width=device-width, initial-scale=1.0">
9      <title><%= content_for?(:title) ? yield(:title) : "Sentibox" %></title>
10     <%= csrf_meta_tags %>
11
12     <!-- Le HTML5 shim, for IE6-8 support of HTML elements -->
13     <!--[if lt IE 9]>
14       <script src="//cdnjs.cloudflare.com/ajax/libs/html5shiv/3.6.1/html5shiv.js"
15 type="text/javascript"></script>
16     <![endif]-->
17
18     <%= stylesheet_link_tag "application", :media => "all" %>
19
20     <!-- For third-generation iPad with high-resolution Retina display: -->
21     <!-- Size should be 144 x 144 pixels -->
22     <%= favicon_link_tag 'apple-touch-icon-144x144-precomposed.png', :rel => 'ap-
23 ple-touch-icon-precomposed', :type => 'image/png', :sizes => '144x144' %>
24
25     <!-- For iPhone with high-resolution Retina display: -->
26     <!-- Size should be 114 x 114 pixels -->
27     <%= favicon_link_tag 'apple-touch-icon-114x114-precomposed.png', :rel => 'ap-
28 ple-touch-icon-precomposed', :type => 'image/png', :sizes => '114x114' %>
29
30     <!-- For first- and second-generation iPad: -->
31     <!-- Size should be 72 x 72 pixels -->
32     <%= favicon_link_tag 'apple-touch-icon-72x72-precomposed.png', :rel => 'apple-
33 touch-icon-precomposed', :type => 'image/png', :sizes => '72x72' %>
34
35     <!-- For non-Retina iPhone, iPod Touch, and Android 2.1+ devices: -->
36     <!-- Size should be 57 x 57 pixels -->
37     <%= favicon_link_tag 'apple-touch-icon-precomposed.png', :rel => 'apple-touch-
38 icon-precomposed', :type => 'image/png' %>
39
40     <!-- For all other devices -->
41     <!-- Size should be 32 x 32 pixels -->
42     <%= favicon_link_tag 'favicon.ico', :rel => 'shortcut icon' %>
43
44     <%= javascript_include_tag "application" %>
45     <script src="http://code.highcharts.com/highcharts.js"></script>
46     <script src="http://code.highcharts.com/modules/exporting.js"></script>
47   </head>
48   <body>
49
50     <div class="navbar navbar-fluid-top">
51       <div class="navbar-inner">
52         <div class="container-fluid">
53           <a class="btn btn-navbar" data-target=".nav-collapse" data-
54 toggle="collapse">
55
```

```
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </a>
          <a class="brand" href="/"><%= image_tag("logo1.png", size: "38") %>
Sentibox</a>
          <div class="container-fluid nav-collapse">
            <ul class="nav">
              <%= render '/layouts/menu' if current_user %>
            </ul>
          </div><!--/.nav-collapse -->
        </div>
      </div>
    </div>

    <div class="container-fluid main-wrapper">
      <div class="row-fluid">
        <div class="span12">
          <%= bootstrap_flash %>
          <%= yield %>
        </div>
      </div><!--/row-->

      <footer>
        <p>Sentibox by Antonela Bare - &copy; Company 2014</p>
      </footer>

    </div> <!-- /container -->

  </body>
</html>
```

File – H:\SentiBox\ schema.rb

```
//Το σχήμα των δεδομένων όπως δημιουργήθηκε αυτόματα κατά το migration. Όλοι οι οι
//πίνακες της βάσης έχουν οριστεί σε ξεχωριστά αρχεία στον υποκατάλογο της
//εφαρμογής db.

# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migra-
tions
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended that you check this file into your version control sys-
tem.

ActiveRecord::Schema.define(version: 20140111133824) do

  # These are extensions that must be enabled in order to support this database
  enable_extension "plpgsql"

  create_table "experiment_validation_corpora", force: true do |t|
    t.string   "name"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  create_table "sentiment_classes", force: true do |t|
    t.string   "name"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  create_table "test_objects", force: true do |t|
    t.integer  "experiment_validation_corpus_id"
    t.string   "title"
    t.text     "body"
    t.integer  "sentiment_class_id"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  add_index "test_objects", ["experiment_validation_corpus_id"], name: "in-
dex_test_objects_on_experiment_validation_corpus_id", using: :btree
  add_index "test_objects", ["sentiment_class_id"], name: "in-
dex_test_objects_on_sentiment_class_id", using: :btree

  create_table "user_assessments", force: true do |t|
    t.integer  "user_id"
    t.integer  "test_object_id"
    t.integer  "sentiment_class_id"
```

```
56      t.integer  "experiment_validation_corpus_id"
57      t.datetime "created_at"
58      t.datetime "updated_at"
59    end
60
61    add_index "user_assessments", ["experiment_validation_corpus_id"], name: "in-
62  dex_user_assessments_on_experiment_validation_corpus_id", using: :btree
63    add_index "user_assessments", ["sentiment_class_id"], name: "in-
64  dex_user_assessments_on_sentiment_class_id", using: :btree
65    add_index "user_assessments", ["test_object_id"], name: "in-
66  dex_user_assessments_on_test_object_id", using: :btree
67    add_index "user_assessments", ["user_id", "test_object_id"], name: "in-
68  dex_user_assessments_on_user_id_and_test_object_id", unique: true, using: :btree
69    add_index "user_assessments", ["user_id"], name: "in-
70  dex_user_assessments_on_user_id", using: :btree
71
72    create_table "users", force: true do |t|
73      t.string   "email",                  default: "", null: false
74      t.string   "encrypted_password",     default: "", null: false
75      t.string   "reset_password_token"
76      t.datetime "reset_password_sent_at"
77      t.datetime "remember_created_at"
78      t.integer  "sign_in_count",          default: 0,  null: false
79      t.datetime "current_sign_in_at"
80      t.datetime "last_sign_in_at"
81      t.string   "current_sign_in_ip"
82      t.string   "last_sign_in_ip"
83      t.string   "role"
84      t.datetime "created_at"
85      t.datetime "updated_at"
86    end
87
88    add_index "users", ["email"], name: "index_users_on_email", unique: true, using:
89  :btree
90    add_index "users", ["reset_password_token"], name: "in-
91  dex_users_on_reset_password_token", unique: true, using: :btree
92
93  end
```

Αναφορές

[1]: David Thomas, Andrew Hunt, (2001), *The Pragmatic Programmer's Guide*. Διαθέσιμο: www.ruby-doc.org/docs/ProgrammingRuby/html/index.html. Προσπελάστηκε: 20η Απρ. 2012

[2]: Ruby on Rails community, (2014), *Rails Guides*. Διαθέσιμο: guides.rubyonrails.org/generators.html. Προσπελάστηκε: 1η Απρ. 2014

[3]: Oracle Corporation, (2014), MySQL. Διαθέσιμο: www.mysql.com/why-mysql. Προσπελάστηκε: 26η Μαρτίου 2014

[4]: Christian Neukirchen, (2012), Rack: a Ruby Webserver Interface. Διαθέσιμο: rack.github.com. Προσπελάστηκε: 26 Απρ. 2012

[5]: Michael Hartl, (2012), *Ruby on Rails Tutorial - Learn Rails by Example*. Διαθέσιμο: ruby.railstutorial.org/chapters. Προσπελάστηκε: 21η Απρ. 2012

[6]: Ruby on Rails community, (2012), *Rails Guides*. Διαθέσιμο: guides.rubyonrails.org/generators.html. Προσπελάστηκε: 21η Απρ. 2012

[7]: git community, (2014), *git website*. Διαθέσιμο: git-scm.com. Προσπελάστηκε: 28η Μαΐου 2014

[8]: Antony T.Velte, Toby J.Velte, Robert Elsenpeter ,2010, Cloud computing: a practical approach, The McGraw-Hill Companies.

[9]: heroku, (2014), *heroku website*. Διαθέσιμο: www.heroku.com. Προσπελάστηκε: 25η Μαΐου 2014

[10]: C.Baun et al. (2011), Cloud Computing. 2nd ed. London: Springer. Κεφάλαιο 6 pp. 49-62

[11]: Roy T. Fielding, (2008), *REST APIs must be hypertext-driven*. Διαθέσιμο: roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven. Προσπελάστηκε: 21η Απρ. 2014

[12]: Ruby on Rails Community, (2014), *Active Record Migrations*. Διαθέσιμο: http://api.rubyonrails.org/classes/ActiveRecord/Migration.htmlΠροσπελάστηκε: 28η Μαΐου 2014

[13]: http://www.cs.cornell.edu/people/pabo/movie-review-data

[14]: What is HTML?, (2014), Διαθέσιμο: www.yourhtmlsource.com. Προσπελάστηκε: 25η Μαΐου 2014

[15]: Wikipedia, (2014), Διαθέσιμο: en.wikipedia.org/wiki/Bootstrap_(front-end_framework) . Προσπελάστηκε: 25η Μαΐου 2014

[16]: Wikipedia, (2014), Διαθέσιμο: en.wikipedia.org/wiki/JavaScript. Προσπελάστηκε: 25η Μαΐου 2014

[17]: Highcharts, (2014), Διαθέσιμο: www.highcharts.com. Προσπελάστηκε: 25η Μαΐου 2014

[18]: RVM, (2014), Διαθέσιμο: rvm.io. Προσπελάστηκε: 25η Μαΐου 2014

Βιβλιογραφία

[1] GAMON, M. 2004. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In Proceedings the 20th International Conference on Computational Linguistics, pp. 841–847.
[2] Pang, B., & Lee, L. (2004, July). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In Proceedings of the 42nd annual meeting on Association for Computational Linguistics (p. 271). Association for Computational Linguistics.
[3] Xiaowen Ding, Bing Liu and Philip S. Yu. "A Holistic Lexicon-Based Appraoch to Opinion Mining." Proceedings of First ACM International Conference on Web Search and Data Mining (WSDM-2008), Feb 11-12, 2008, Stanford University, Stanford, California, USA
[4] BO PANG AND LILLIAN LEE, OPINION MINING AND SENTIMENT ANALYSIS, FOUNDATIONS AND TRENDS IN INFORMATION RETRIEVAL, VOL 2, NOS 1-2(2008) 1-135
[5] MAITE TABOADA- JULIAN BROOKE- MANFRED STEDE, GENRE-BASED PARAGRAPH CLASSIFICATION FOR SENTIMENT ANALYSIS, SIGDIAL 2009
[6] S. DAS AND M. CHEN, EXTRACTING SENTIMENT MARKET FROM STOCK MESSAGE BOARDS, IN PROCEEDINGS OF THE ASIA PACIFIC FINANCE ASSOCIATION ANNUAL CONFERENCE (APFA), 2001.
[7] E. RILOFF AND J. WIEBE, LEARNING EXTRACTION PATTERNS FOR SUBJECTIVE EXPRESSIONS, EMNLP, 2003.
[8] SES: Sentiment Elicitation System for Social Media Data. Kunpeng Zhang, Yu Cheng, Yusheng Xie, Daniel Honbo, Ankit Agrawal, Diana Palsetia, Kathy Lee, Wei-keng Liao, Alok Choudhary. s.l. : 11th IEEE International Conference on Data Mining Workshops, 2011.
[9] Albert Bifet, Eibe Frank. Sentiment Knowledge Discovery in Twitter Streaming Data. Lecture Notes in Computer Science, Volume 6332, Discovery Science, Pages 1-15. 2010.
[10] Richard D. Waters, Jia Y. Jamal. Tweet, tweet, tweet: A content analysis of nonprofit organizations' Twitter updates. Public Relations Review. Elsevier Inc, September 2011, Τόμ. Volume 37, Issue 3, Pages 321-324.
[11] Farah Benamara, Baptiste Chardon, Yvette Yannick Mathieu, Vladimir Popescu: Towards Context-Based Subjectivity Analysis. IJCNLP 2011: 1180-1188
[12] Thorsten Joachims, Transductive Inference for Text Classification using Support Vector Machines. International Conference on Machine Learning (ICML), 1999.
[13] RAVI PARIKH - MATIN MOVASSATE, SENTIMENT ANALYSIS OF USERGENERATED TWITTER UPDATES USING VARIOUS CLASSICATION TECHNIQUES, JUNE 2009, STANDFORD UNIVERSITY
[14] An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Nello Cristianini and John Shawe-Taylor, Cambridge University Press, 2000
[15] V. PANDEY, K. IYER, SENTIMENT ANALYSIS OF MICROBLOGS, STANDFORD UNIVERSITY
[16] PREM MELVILLE - WOJCIECH GRYC - RICHARD D. LAWRENCE , SENTIMENT ANALYSIS OF BLOGS BY COMBINING LEXICAL KNOWLEDGE WITH TEXT CLASSIFICATION, KDD JUNE 2009
[17]Matsumoto, S., Takamura, H., & Okumura, M. (2005). Sentiment classification using word sub-sequences and dependency sub-trees. In Advances in Knowledge Discovery and Data Mining (pp.301-311). Springer Berlin Heidelberg.

[18]Kennedy, A., & Inkpen, D. (2006). Sentiment classification of movie reviews using contextual valence shifters. Computational Intelligence, 22(2), 110-125.
[19]Dasgupta, S., & Ng, V. (2009, August). Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2 (pp. 701-709). Association for Computational Linguistics.
[20] Qiu, L., Zhang, W., Hu, C., & Zhao, K. (2009, November). Selc: a self-supervised model for sentiment classification. In Proceedings of the 18th ACM conference on Information and knowledge management (pp. 929-936). ACM.
[21] ALEC GO, RICHA BHAYANI , LEI HUANG, TWITTER SENTIMENT CLASSIFICATION USING DISTANT SUPERVISION, STANFORD UNIVERSITY
[22]Yessenalina, A., Yue, Y., & Cardie, C. (2010, October). Multi-level structured models for document-level sentiment classification. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (pp. 1046-1056). Association for Computational Linguistics.
[23] ALEC GO, RICHA BHAYANI, EXPLOITING THE UNIQUE CHARACTERISTICS OF TWEETS FOR SENTIMENT ANALYSIS, STANDORD UNIVERSITY
[24] B. PANG, L. LEE, AND S. VAITHYANATHAN, THUMBS UP? SENTIMENT CLASSIFICATION USING MACHINE LEARNING TECHNIQUES, PROCEEDINGS OF THE CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP), 2002.
[25] A. GILL, R. FRENCH, D. GERGLE, J. OBERLANDER, IDENTIFYING EMOTIONAL CHARACTERISTICS FORM SHORT BLOG TEXTS
[26] KLAUS R. SCHERER, TRENDS AND DEVELOPMENTS: RESEARCH ON EMOTIONS, SOCIAL SCIENCE INFORMATION & 2005 SAGE PUBLICATIONS VOL 44(4), PP.695–729
[27] C.STRAPPARAVA - R.MIHALCEA, LEARNING TO IDENTIFY EMOTIONS IN TEXT, SAC'08
[28] W. PARROTT, EMOTIONS IN SOCIAL PSYCHOLOGY, PSYCHOLOGY PRESS, PHILADELPHIA, 2001.
[29] BING LIU, SENTIMENT ANALYSIS AND OPINION MINING, MORGAN & CLAYPOOL PUBLISHERS, MAY 2012.
[30] BING LIU, SENTIMENT ANALYSIS AND SUBJECTIVITY, HANDBOOK OF NATURAL LANGUAGE PROCESSING, SECOND EDITION, 2010
[31]Turney, P. D. (2002, July). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th annual meeting on association for computational linguistics (pp. 417-424). Association for Computational Linguistics.
[32]Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. Computational linguistics, 37(2), 267-307.
[33]Yu, H., & Hatzivassiloglou, V. (2003, July). Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In Proceedings of the 2003 confe rence on Empirical methods in natural language processing (pp. 129-136). Association for Computational Linguistics.
[34] Hatzivassiloglou, V., Klavans, J. L., Holcombe, M. L., Barzilay, R., Kan, M. Y., & McKeown, K. (2001). Simfinder: A flexible clustering tool for summarization. Proceedings of the NAACL Workshop on Automatic Summarizatio.
[35]Raaijmakers, S., & Kraaij, W. (2008). A Shallow Approach to Subjectivity Classification. In ICWSM.

[36] Kim, S. M., & Hovy, E. (2004, August). Determining the sentiment of opinions. In Proceedings of the 20th international conference on Computational Linguistics (p. 1367). Association for Computational Linguistics.
[37] Hu, M., & Liu, B. (2004, August). Mining and summarizing customer reviews. InProceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 168-177). ACM.
[38]Esuli, A., & Sebastiani, F. (2011). Enhancing opinion extraction by automatically annotated lexical resources. In Human Language Technology. Challenges for Computer Science and Linguistics (pp.500-511). Springer Berlin Heidelberg.
[39]Harry Zhang. Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference FLAIRS 2004, AAAI Press, (2004)
[40] Deerwester, S., Dumais S.T., Furnas G.W., Landauer T.K., and Harshman R. 1990. Indexing by Latent Semantic Indexing. Journal of the American Society for Information Science 41, 6, pp.391 –407.
[41] Sebastiani, F. 2001. Machine learning in automated text categorization. Revised Version of Technical Report IEI-B4-31-1999, Consiglio Nazionale delle Ricerche, Pisa, Italy, 1999.