ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEB INTELLIGENCE

Ανάπτυξη εργαλείων για αυτόματη εκτίμηση διάθεσης Ελληνικών κειμένων με χρήση Ημι-Επιβλεπόμενων Αναδρομικών Αυτοσυσχετιστών

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΟΤΡΟΤΣΙΟΥ ΚΩΝΣΤΑΝΤΙΝΟΥ

Επιβλέπων : Κωνσταντίνος Διαμαντάρας
Καθηγητής, ΑΤΕΙΘ

Θεσσαλονίκη, Μάιος 2015

ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEBINTELLIGENCE

Ανάπτυξη εργαλείων για αυτόματη εκτίμηση διάθεσης Ελληνικών κειμένων με χρήση Ημι-Επιβλεπόμενων Αναδρομικών Αυτοσυσχετιστών

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΟΤΡΟΤΣΙΟΥ ΚΩΝΣΤΑΝΤΙΝΟΥ

Επιβλέπων : Κωνσταντίνος Διαμαντάρας
Καθηγητής ΑΤΕΙΘ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 30 Ιουνίου 2015.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Όνομα Επώνυμο	Όνομα Επώνυμο	'Ονομα Επώνυμο
Choose an item.Α.Τ.Ε.Ι.Θ.	Choose an item.Α.Τ.Ε.Ι.Θ.	Choose an item.Α.Τ.Ε.Ι.Θ.

Θεσσαλονίκη, Choose a date

(Υπογραφή)

.....

Κοτρότσιος Κωνσταντίνος

Μηχανικός Πληροφορικής Τ.Ε.

Περίληψη

Στην παρούσα μεταπτυχιακή διπλωματική εργασία αναπτύσσονται εργαλεία με σκοπό να χρησιμοποιηθούν σε εφαρμογές αυτόματης εκτίμησης διάθεσης κειμένων στην Ελληνική γλώσσα. Κατασκευάζεται εργαλείο λεξικού της ελληνικής γλώσσας για βάσεις δεδομένων MySQL και Mongodb, το οποίο περιέχει τις λέξεις σε διαφορετικές πτώσεις, αριθμούς, γένη, χρόνους κτλ. Δημιουργείται συλλογή κειμένων από σχόλια χρηστών για προϊόντα τεχνολογίας χαρακτηρισμένα ως προς το συναίσθημα. Αναπτύσσονται εφαρμογές για αυτόματη διόρθωση ορθογραφικών λαθών και λημματοποίησης κειμένων. Γίνεται χρήση μοντέλου ημι-επιβλεπόμενων αναδρομικών αυτοσυσχετιστών, οι οποίοι εκπαιδεύονται να αναπαριστούν τις προτάσεις σε διανυσματικούς χώρους. Η συγκεκριμένη μέθοδος στην αυτόματη εκτίμηση διάθεσης κειμένων έχει υψηλότερες επιδόσεις από άλλες προσεγγίσεις σε ευρέως χρησιμοποιημένες συλλογές. Το μοντέλο εφαρμόζεται στην συλλογή σχολίων που αναπτύξαμε και εξετάζονται τα αποτελέσματα και το πως αυτά επηρεάστηκαν από την εφαρμογή των εργαλείων μας. Με την χρήση των εργαλείων για την προ επεξεργασία των σχολίων βελτιώσαμε τα αποτελέσματα μέχρι και 24% .

Λέξεις Κλειδιά: Αυτόματη εκτίμηση διάθεσης, Ημι-Επιβλεπόμενοι αναδρομικοι αυτοσυσχετιστές, Βαθιά μάθηση, Λημματοποίηση.

Abstract

In this master thesis our goal is to develop tools in order to be used in sentiment analysis applications for texts in Greek language. We have developed a Greek language dictionary using MySQL and MongoDB database systems, which contains words in different numbers, genders, tenses etc. A dataset, from users' opinions about technology products characterized according to sentiment, has been developed. Two applications, one for automatic spelling correction and another for lemmatization have also been developed. A framework of semi-supervised recursive autoencoders, which are trained in vector space representations for multi-word phrases, is used. These representations in automatic assessment of text provision outperform other state-of–the-art approaches on commonly used datasets. This model is applied in on our users' opinions dataset and the results were examined to see how they were affected by using our tools use, are tested. Using these tools for the pretreatment of the dataset, we have improved the results up to 24%.

Keywords: Sentiment Analysis, Semi-supervised Recursive Autoencoders, Deep Learning, Lemmatization

Ευχαριστίες

Θα ήθελα να ευχαριστήσω όλους εκείνους που συνέβαλαν, ο καθένας με τον τρόπο του για την συγγραφή – υλοποίηση αυτής της εργασίας.

Τον επιβλέποντα μου, καθηγητή Διαμαντάρα Κωνσταντίνο για την πολύτιμη βοήθεια του όλο αυτό το διάστημα και την άψογη συνεργασία που είχαμε. Τον Δρ Βοζαλή Εμμανουήλ, με τον οποίο συνεργαστήκαμε για την κατασκευή του λεξικού και της συλλογής σχολίων. Την Γιάτσογλου Μαρία για την βοήθεια της στο να μπορέσω να κατανοήσω την λειτουργία των RAE. Την συμφοιτήτρια μου Γρηγορίου Ελισάβετ για την βοήθεια της στην κατανόηση άρθρων. Τον φιλόλογο Αλεξίου Γιώργο για τις πολύτιμες διορθώσεις του στα κείμενα. Όλους τους φίλους μου που διάβασαν και χαρακτήρισαν τα σχόλια για την δημιουργία της συλλογής.

Πίνακας περιεχομένων

Πίνακας Εικόνων

Πίνακας Πινάκων

1

Εισαγωγή

1.1 Ανάγκη αυτόματης εκτίμησης της διάθεσης κειμένων

Σχεδόν μια δεκαπενταετία μετά την εισαγωγή του όρου Web 2.0 ή Κοινωνικός Ιστός (Social Web) για την περιγραφή των καινοτόμων τεχνολογιών και εργαλείων που προωθούν την επικοινωνία και τη συνεργασία στο διαδίκτυο. Οι τεχνολογίες αυτές έχουν σήμερα τον κυρίαρχο ρόλο στην καθημερινότητα του Διαδικτύου σε παγκόσμιο επίπεδο.

Οι τεχνολογίες και τα εργαλεία του Web 2.0 καλύπτουν σήμερα ένα ευρύτατο φάσμα εφαρμογών, από κοινωνικά δίκτυα (Facebook, Google+), εφαρμογές άμεσων μηνυμάτων (Skype, Viber, Yahoo! Messenger), ιστολόγια (Blogger, Wordpress), μικροϊστολόγια (Twitter), εφαρμογές διαμοιρασμού ψηφιακού περιεχομένου (Dropbox, Google drive), φωτογραφιών (Instagram), βίντεο (YouTube) και ημερολόγια (Google calendar), κ.λπ.

Η ανάπτυξη των τεχνολογιών και των υπηρεσιών του Web 2.0 υπήρξε καθοριστικός πολλαπλασιαστής της ισχύος και του μεγέθους του Διαδικτύου. Πολύ συνοπτικά αναφέρουμε κάποιους ενδεικτικούς αριθμούς:

Τα μεγαλύτερα και δημοφιλέστερα κοινωνικά δίκτυα και εφαρμογές άμεσων μηνυμάτων έχουν περισσότερους από 2 δισεκατομμύρια χρήστες. Περίπου το 1/4 του χρόνου που παραμένει κάποιος OnLine καταναλώνεται σε κοινωνικά δίκτυα. Το 95% των νέων στο ηλικιακό διάστημα 18-24 έχει χρησιμοποιήσει τουλάχιστον μια φορά ένα κοινωνικό δίκτυο.

Το περιεχόμενο (κείμενα, φωτογραφίες, αρχεία εικόνας και ήχου, κ.λπ.) που προστίθεται επί καθημερινής βάσης στον Παγκόσμιο Ιστό και ιδιαίτερα το περιεχόμενο που παράγεται από τους χρήστες (User Generated Content) γνωρίζει πραγματική έκρηξη μεγεθών, όπως φαίνεται και στην εικόνα 1.

Εικόνα 1 Περιεχόμενο που παράγεται από τους χρήστες στο internet κάθε λεπτό

Δεκάδες εκατομμύρια άνθρωποι σε καθημερινή βάση εκφράζουν σε παγκόσμιο επίπεδο την άποψη τους για προϊόντα, υπηρεσίες και προσφορές σε ιστολόγια, wikis και κοινωνικά δίκτυα, καθιστώντας κρίσιμη την όσο το δυνατόν με μεγαλύτερη ακρίβεια εξαγωγή με αυτοματοποιημένο τρόπο συναισθημάτων και απόψεων από τα αντίστοιχα κείμενα.

1.2 Αντικείμενο διπλωματικής

Η αυτόματη εκτίμηση της διάθεσης (sentiment analysis) κειμένων ή προφορικού λόγου είναι μια τεχνολογία, η οποία προβλέπεται να έχει μεγάλη ανάπτυξη τα επόμενα χρόνια λόγω του συνεχώς αυξανόμενου περιεχομένου που παράγεται στο διαδίκτυο από χρήστες.

Στόχος της εργασίας είναι να εφαρμόσει μοντέλα αυτόματης εκτίμησης της διάθεσης κειμένων στην ελληνική γλώσσα.

Πιο συγκεκριμένα, στόχος είναι να αναπτύξουμε εργαλεία με τα οποία θα γίνει αυτό εφικτό, να τα εφαρμόσουμε σε υπάρχοντα μοντέλα και να μετρήσουμε την αποτελεσματικότητά τους στην αυτόματη εκτίμηση της διάθεσης κείμενων στην Ελληνική γλώσσα.

Στα πλαίσια της παρούσας εργασίας αναπτύχθηκαν τα εξής:

- Εργαλείο λεξικού, το οποίο περιέχει λέξεις της Ελληνικής γλώσσας σε όλες τους τις πτώσεις, κλίσεις, χρόνους, κτλ. με επί πλέον πληροφορίες και χαρακτηριστικά για την κάθε λέξη.
- Συλλογή προτάσεων χαρακτηρισμένες ως προς το συναίσθημα που περιέχουν.
- Εφαρμογές για την αυτόματη διόρθωση ορθογραφικών λαθών σε κείμενα και την μετατροπή λέξεων στην αρχική τους μορφή (λημματοποίηση).

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

- Μελετήθηκαν συστήματα αυτόματης εκτίμησης της διάθεσης.
- Υλοποιήθηκαν εργαλεία για την εφαρμογή μοντέλων στην Ελληνική γλώσσα.
- Αναπτύχθηκε λεξικό της Ελληνικής γλώσσας, που πέρα από την χρήση του στην εργασία μας θα μπορούσε να χρησιμοποιηθεί και σε μελλοντικές εργασίες.
- Αναπτύχθηκε συλλογή προτάσεων χαρακτηρισμένων ως προς το συναίσθημα που περιέχουν.
- Αναπτύχθηκε εφαρμογή αυτόματης διόρθωσης κειμένων.
- Αναπτύχθηκε εφαρμογή λημματοποίησης.
- Εφαρμόστηκαν τα παραπάνω εργαλεία σε μοντέλα αυτόματης εκτίμησης διάθεσης.
- Αξιολογήθηκε η επίδοση των μοντέλων.

1.3 Οργάνωση κειμένου

Σχετικές εργασίες για την αυτόματη εκτίμηση της διάθεσης κειμένων παρουσιάζονται στο 2^{ο} Κεφάλαιο, ενώ στο 3^{ο} Κεφάλαιο γίνεται μια εισαγωγή στην βαθιά μάθηση (Deep Learning) πεδίο της μηχανικής μάθησης, στο οποίο ανήκει και η αρχιτεκτονική του μοντέλου που χρησιμοποιούμε, το οποίο παρουσιάζεται στο ίδιο κεφάλαιο. Το Κεφάλαιο 4 συζητά προβλήματα και ιδιαιτερότητες που έχει ένα σύστημα αυτόματης εκτίμησης της διάθεσης με ιδιαίτερη έμφαση στην ελληνική γλώσσα. Στο Κεφάλαιο 5 περιγράφουμε τον τρόπο ανάπτυξης των εργαλείων. Τα πειράματα και τα αποτελέσματα περιγράφονται στο Κεφάλαιο 6. Ενώ στο Κεφάλαιο 7 αναφερόμαστε στις τεχνικές λεπτομέρειες για την ανάπτυξη των εργαλείων.

2

Σχετικές εργασίες

Η αυτόματη εκτίμηση της διάθεσης (sentiment analysis) ή εξόρυξη απόψεων (opinion mining) αναφέρεται στην επεξεργασία της φυσικής γλώσσας ή την ανάλυση κειμένων με στόχο τον εντοπισμό και την εξαγωγή υποκειμενικής πληροφορίας για το κείμενο. Σε γενικές γραμμές η αυτόματη εκτίμηση της διάθεσης έχει ως στόχο να καθορίσει τη στάση που έχει ο ομιλητής ή ο συγγραφέας για κάποιο συγκεκριμένο θέμα ή και την συνολική άποψη που περιγράφεται σε ένα έγγραφο.

2.1 Αυτόματη εκτίμηση της διάθεσης κειμένων

Ο κύριος σκοπός της αυτόματης εκτίμησης της διάθεσης είναι να χαρακτηρίσει την πολικότητα του κειμένου μίας φράσης ή μιας πρότασης και να το χαρακτηρίσει ως θετικό, αρνητικό ή ουδέτερο. Επίσης, κάτι πιο προχωρημένο είναι, εκτός από την «πολικότητα» (θετικό, αρνητικό, ουδέτερο), το σύστημα να ταξινομεί το κείμενο με βάση την συναισθηματική κατάσταση όπως «θυμωμένος», «λυπημένος», «φοβισμένος», «χαρούμενος» κτλ.

Δύο πρωτοπόρες εργασίες στον τομέα είναι του Turney [1] και του Pang [2], στις οποίες εφαρμόζουν διαφορετικές μεθόδους για την ανίχνευση της πολικότητας σε κριτικές προϊόντων και ταινιών αντίστοιχα.

Ο Turney [1] παρουσιάζει έναν απλό αλγόριθμο, χωρίς επίβλεψη, που ταξινομεί τις κριτικές προϊόντων ως προτεινόμενες (thumps up) και μη (thumps down). Η ταξινόμηση της κριτικής

προβλέπεται από τον μέσο σημασιολογικό προσανατολισμό της φράσης, που προκύπτει από τα επίθετα και τα επιρρήματα που περιέχει. Μια κριτική ταξινομείται ως προτεινόμενη αν ο μέσος σημασιολογικός προσανατολισμός της είναι θετικός. Ο αλγόριθμος επιτυγχάνει μέση ακρίβεια 74%, όταν αξιολογήθηκαν 410 κριτικές, δείγματα από τέσσερις διαφορετικούς τομείς αξιολογήσεις (κριτικές αυτοκινήτων, τραπεζών, ταινιών, και ταξιδιωτικών προορισμών). Η ακρίβεια κυμαίνεται από 84% για τα σχόλια στα αυτοκίνητα έως 66% για τις κριτικές ταινιών.

Ο αλγόριθμος έχει τρία στάδια: 1^{ο} Εξαγωγή από την φράση των επίθετων και των επιρρημάτων. 2^{ο} Εκτίμηση του σημασιολογικού προσανατολισμού της κάθε φράσης, και 3^{ο} Κατάταξη του κειμένου με βάση το μέσο σημασιολογικό προσανατολισμό των προτάσεων. Ο πυρήνας του αλγόριθμου είναι το δεύτερο βήμα, το οποίο χρησιμοποιεί τον αλγόριθμο PMI-IR, για να υπολογίσει τον σημασιολογικό προσανατολισμό.

Ο αλγόριθμος PMI-IR (Pointwise Mutual Information and Information Retrieval) μετράει την ομοιότητα σε λέξεις ή φράσεις. Ο σημασιολογικός προσανατολισμός μιας δεδομένης φράσης υπολογίζεται συγκρίνοντας την ομοιότητα της φράσης σε σχέση με μια θετική αναφορά (πχ. «εξαιρετική») και με μία αρνητική αναφοράς (πχ. «κακή»).

Πιο συγκεκριμένα, σε μια φράση αποδίδεται μια αριθμητική βαθμολογία που υπολογίζεται από την ομοιότητα ανάμεσα στην φράση και την λέξη "εξαιρετική" και αφαιρώντας την βαθμολογία ομοιότητας μεταξύ της δεδομένης φράσης και της λέξης "κακή". Εκτός από τον καθορισμό του σημασιολογικού προσανατολισμού της φράσης (θετική ή αρνητική, με βάση το πρόσημο της αξιολόγησης), η αριθμητική αξιολόγηση φανερώνει το πόσο ισχυρός είναι ο σημασιολογικού προσανατολισμός (με βάση το μέγεθος του αριθμού).

Ο Pang [2] χρησιμοποίησε κριτικές ταινιών ως δεδομένα και αλγορίθμους μηχανικής μάθησης για την εργασία του. Ωστόσο, οι τρείς μέθοδοι μηχανικής μάθησης, που χρησιμοποίησε (Naive Bayes, Maximum Entropy Classification, και Support Vector Machines) δεν απέδωσαν εξίσου καλά στην ταξινόμηση συναισθημάτων, όπως σε παραδοσιακά προβλήματα ταξινόμησης.

Μία άλλη πιο προχωρημένη προσέγγιση στο πρόβλημα είναι η ταξινόμηση να μην γίνεται μόνο σε δύο κλάσεις (θετικό, αρνητικό), αλλά σε μία κλίμακα πολλαπλών κλάσεων, το οποίο επιχειρήθηκε από τον Pang [3] και τον Snyder [4].

Ο Pang [3] εξελίσσει την προηγούμενη εργασία του, όπου εκτιμούσε κριτικές προϊόντων σε προτεινόμενες και μη (thumbs up, thumbs down) και προσπαθεί να εκτιμήσει την αξιολόγηση ταινιών σε ένα σύστημα βαθμολόγησης. (παράδειγμα ένα σύστημα βαθμολόγησης από 1 μέχρι 5 αστέρια). Η συγκεκριμένη εργασία είναι μια ενδιαφέρουσα προσέγγιση για ένα σύστημα ταξινόμησης κειμένου σε πολλές κλάσεις, γιατί υπάρχει διαφορετικός βαθμός

ομοιότητας μεταξύ των κλάσεων. Παράδειγμα η κλάση «3 αστέρια» είναι πιο κοντά στην κλάση «4 αστέρια» απ' ότι στην κλάση «1 αστέρι».

Στην εργασία του Pang τα κείμενα βαθμολογήθηκαν αρχικά από ανθρώπους, με σκοπό να δείξουν ότι και οι βαθμολογίες των ανθρώπων διαφέρουν μεταξύ τους, αλλά είναι κοντά στις ίδιες κλάσεις (παράδειγμα μια ταινία που κάποιος την βαθμολογεί με 4 αστέρια είναι πιο πιθανό και οι υπόλοιποι βαθμολογητές να την βαθμολογήσουν με 4, 3 ή 5 αστέρια απ' ότι με 1). Στη συνέχεια, δοκιμάζονται τρείς διαφορετικοί αλγόριθμοι (ένας-εναντίον-όλων (one-vs-all), παλινδρόμηση (regression) και μετρική σήμανση (metric labeling)) και όλοι βασίζονται σε Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines SVM).

Ο Snyder [4] στην εργασία του αντιμετωπίζει την ανάλυση πολλαπλών απόψεων που σχετίζονται μεταξύ τους και υπάρχουν σε ένα κείμενο. Για παράδειγμα, η κριτική ενός εστιατορίου μπορεί να περιλαμβάνει κριτικές για το φαγητό, την ατμόσφαιρα και την εξυπηρέτηση. Αυτό το πρόβλημα αντιμετωπίστηκε ως πρόβλημα κατηγοριοποίησης πολλαπλών απόψεων, όπου στόχος είναι η παραγωγή μιας σειράς από αριθμητικά δεδομένα που το καθένα αντιστοιχεί σε μία άποψη. Ο αλγόριθμος που χρησιμοποιείται μαθαίνει να ταξινομεί τις ανεξάρτητες κριτικές. Ο αλγόριθμος προβλέπει τις βαθμολογίες των ανεξάρτητων κριτικών αναλύοντας τις σχέσεις ανάμεσα στις κριτικές για το αν συμφωνούν ή διαφωνούν και αποδεικνύει, ότι το μοντέλο για τις κριτικές που συμφωνούν μεταξύ τους είναι πιο ακριβές σε σχέση με τα ανεξάρτητα μοντέλα.

Παρόλο που στα περισσότερα μοντέλα στατιστικής ταξινόμησης η ουδέτερη κλάση αγνοείται με την παραδοχή, ότι τα ουδέτερα κείμενα βρίσκονται κοντά στο όριο του δυαδικού ταξινομητή, αρκετοί ερευνητές προτείνουν ότι, όπως και σε κάθε πρόβλημα πολικότητας, πρέπει να προσδιοριστούν τρεις κατηγορίες. Επί πλέων συγκεκριμένοι ταξινομητές όπως η μέγιστη εντροπία (Max Entropy) και οι Μηχανές Διανυσμάτων Υποστήριξης (SVMs) μπορούν να έχουν καλύτερα αποτελέσματα από την εισαγωγή της ουδέτερης τάξης και να βελτιωθεί η συνολική ακρίβεια τους [5].

Μια διαφορετική μέθοδος για τον προσδιορισμό του συναισθήματος είναι η χρήση ενός συστήματος διαβάθμισης, στο οποίο οι λέξεις ταξινομούνται σε κλάσεις συναισθήματος (θετικό, αρνητικό ή ουδέτερο) ή και σε μία κλίμακα, όπου βαθμολογούνται (παράδειγμα από το -10 έως το 10, από το πιο αρνητικό στο πιο θετικό). Όταν ένα κομμάτι κειμένου αναλύεται, χωρίζεται στις λέξεις που περιέχει και στο πως αυτές σχετίζονται με το περιεχόμενο. Στη συνέχεια σε κάθε λέξη – έννοια δίνεται μια βαθμολογία με βάση την κλίμακα, στην οποία ανήκει η κάθε λέξη. Αυτό επιτρέπει μια πιο εξελιγμένη κατανόηση του συναισθήματος που βασίζεται σε μια κλίμακα σημείων.

Στην εργασία τους ο Mike Thelwall [6] και η ομάδα του αναπτύξαν αλγόριθμο, ο οποίος βασίζεται σε ένα λεξικό συναισθηματικής δύναμης των λέξεων. Μία συλλογή από 298

θετικούς και 465 αρνητικούς όρους, ταξινομημένους με μία τιμή από το 2 έως το 5 για τους θετικούς και αρνητικούς όρους. Αρχικά, οι λέξεις ταξινομήθηκαν από ανθρώπους και στην συνέχεια τροποποιηθήκαν από τον αλγόριθμο εκπαίδευσης. Ο αλγόριθμος περιέχει, επίσης, μία λίστα με λέξεις «ώθησης», οι οποίες αυξάνουν ή μειώνουν την τιμή του συναισθήματος (πχ πολύ, ελάχιστα κτλ.) και μια λίστα με λέξεις αναστροφής που αλλάζουν την πολικότητα της λέξης. (Παράδειγμα αν η φράση πολύ χαρούμενος έχει θετική τιμή 4, τότε η φράση όχι πολύ χαρούμενος θα έχει αρνητική τιμή 4).

Μια άλλη ερευνητική προσέγγιση είναι η αναγνώριση της υποκειμενικότητας / αντικειμενικότητας. Σε αυτή την προσέγγιση, συνήθως, ταξινομείται κάποιο κείμενο (συνήθως μια πρόταση) σε μία από τις δύο κατηγορίες: αντικειμενική ή υποκειμενική. Αυτή η προσέγγιση μπορεί να είναι πιο δύσκολή από την πολικότητα. Η υποκειμενικότητα των λέξεων και των φράσεων μπορεί να εξαρτάται από το περιεχόμενό τους. Όπως αναφέρθηκε από τον Su [7], τα αποτελέσματα εξαρτώνται σε μεγάλο βαθμό από τον ορισμό της υποκειμενικότητας που χρησιμοποιείται στον σχολιασμό των κειμένων. Ωστόσο, Ο Pang [8] έδειξε, ότι η αφαίρεση των αντικειμενικών προτάσεων από ένα έγγραφο πριν από την ταξινόμηση του βοηθάει να βελτιωθούν οι επιδόσεις.

Ένα άλλο μοντέλο αναφέρεται στον καθορισμό των γνωμοδοτήσεων ή των συναισθημάτων που εκφράζονται για διαφορετικά χαρακτηριστικά σε διαφορετικά θέματα π.χ., ένα κινητό τηλέφωνο, μια ψηφιακή φωτογραφική μηχανή ή μια τράπεζα. Ένα χαρακτηριστικό ή πτυχή είναι ένα συστατικό μέρος μιας οντότητας, π.χ., η οθόνη ενός κινητού τηλεφώνου ή η ποιότητα της εικόνας μιας φωτογραφικής μηχανής. Το πρόβλημα αυτό περιλαμβάνει διάφορα επιμέρους προβλήματα, π.χ., ο εντοπισμός των σχετικών οντοτήτων, προσδιορισμός των απόψεων για κάθε χαρακτηριστικό είναι θετική, αρνητική ή ουδέτερη [9].

Δύο εργασίες, που έχουν ασχοληθεί με το θέμα της αυτόματης εκτίμησης της διάθεσης κειμένων στην ελληνική γλώσσα, είναι μιας ομάδας ερευνητών του Ινστιτούτου Τεχνολογιών Πληροφορικής και Επικοινωνιών σε συνεργασία με τον οργανισμό του διεθνούς φεστιβάλ κινηματογράφου της Θεσσαλονίκης [10] και του Αγαθαγγέλου και τις ομάδας του [11].

Στην εργασία του διεθνούς φεστιβάλ κινηματογράφου της Θεσσαλονίκης [10] παρουσιάζεται η εφαρμογή EventSense, που στόχο έχει την ανίχνευση των θεμάτων συζήτησης και το συναίσθημα σε μηνύματα χρηστών, που τα δημοσιεύουν σε κοινωνικά δίκτυα και σχετίζονται με κάποιο γεγονός-φεστιβάλ. Η εφαρμογή μπορεί να ανιχνεύσει το συναίσθημα (θετικό / αρνητικό / ουδέτερο) τόσο σε επίπεδο οντότητας (π.χ. μόνο για μια ταινία), αλλά και για όλο το γεγονός (φεστιβάλ κινηματογράφου). Η συγκεκριμένη εφαρμογή έχει εφαρμοστεί στο 53ο Διεθνές Φεστιβάλ Κινηματογράφου Θεσσαλονίκης σε μια συλλογή μηνυμάτων που συλλέχθηκαν.

Στην αρχή, γίνεται κάποια προ επεξεργασία των μηνυμάτων (καθαρισμός σημείων στίξης, φράσεις από social media) και χωρισμός σε λέξεις, πριν να προωθηθούν στα επόμενα στάδια. Επόμενο στάδιο είναι η ανίχνευση της οντότητας του μηνύματος, όπου για την περίπτωση του φεστιβάλ οντότητες είναι οι ταινίες που προβλήθηκαν. Έπειτα στο στάδιο ανίχνευσης θέματος χρησιμοποιούν την μέθοδο από την εργασία των Sasa Petrovic [12], με την οποία ταξινομούν τα μηνύματα σε κατηγορίες. Το τελικό στάδιο είναι η ανίχνευση του συναισθήματος, όπου αντιμετωπίζεται σαν ένα πρόβλημα μηχανικής μάθησης με επίβλεψη. Για την δημιουργία των δεδομένων εκπαίδευσης για τις αρνητικές και θετικές κατηγορίες κατασκευάστηκαν αυτόματα με βάση τα emoticons των μηνυμάτων (π.χ. μηνύματα που περιέχουν τα :), :-), :D χαρακτηρίζονται θετικά και μηνύματα με τα :(, :-(χαρακτηρίζονται αρνητικά). Ο αλγόριθμος μηχανικής μάθησης, που χρησιμοποιήσανε είναι ένας Naive Bayes ταξινομητής.

Ο Αγαθαγγέλου και η ομάδα του [11] παρουσιάζουν την εφαρμογή «NiosTo», που επιτρέπει την δημιουργία λεξικού και ανάλυση συναισθήματος σε ένα δεδομένο αντικείμενο. Η δικιά τους προσέγγιση αποτελείται από μια σειρά 6 βημάτων, όπου το κάθε βήμα τροφοδοτεί το επόμενο.

1. **Προ επεξεργασία:** Φιλτράρουν και καθαρίζουν τα δεδομένα. Στο συγκεκριμένο βήμα γίνεται και ο διαχωρισμός του κειμένου σε προτάσεις.
2. **Προπαρασκευή βοηθητικής λίστας λέξεων**: Εντοπίζονται οι σημαντικές λέξεις για τον αλγόριθμο στο κείμενο όπως, ρήματα, συγκριτικά στοιχεία, σύνδεσμοι, μειώσεις (π.χ. λιγότερο), πολλαπλασιαστές (π.χ. έξτρα), αρνήσεις (π.χ. όχι) και αντωνυμίες. Αυτά τα στοιχεία θα αποτελέσουν την κύρια τροφοδοσία του αλγορίθμου.
3. **Εξαγωγή πολικότητας από γνωστές λέξεις:** Αρχικά, εντοπίζονται οι λέξεις με γνωστή πολικότητα, (π.χ. κακό, άσχημο, υπέροχο) οι οποίες είναι κοινές ανεξάρτητα από το θέμα των κειμένων. Επίσης, λαμβάνεται υπόψη και η θέση της λέξης μέσα στο κείμενο από την οποία, μπορεί να αλλάξει και η πολικότητα της. Γενικά κάνουν χρήση κάποιων κανόνων στην γλώσσα, από τους οποίους μπορεί να αλλάξει η πολικότητα μιας λέξης.
4. **Εξαγωγή από συνδυασμό λέξεων:** Εξάγεται το συμπέρασμα από συνδυασμούς λέξεων ή σύνθετες λέξεις.
5. **Εξαγωγή νέων λέξεων:** Εξάγονται νέες λέξεις, οι οποίες περιέχουν κάποιο συναίσθημα σχετικό με το αντικείμενο του κειμένου. Η υπόθεση είναι πως κάθε λέξη, που περιέχει κάποιο συναίσθημα έχει ως στόχο κάποιο αντικείμενο, στο οποίο αναφέρεται (π.χ. ωραία οθόνη). Το «ωραία» περιέχει το συναίσθημα και «οθόνη» είναι το αντικείμενο. Βασιζόμενοι σε αυτή την υπόθεση και χρησιμοποιώντας το υπάρχον

λεξικό με λέξεις, που περιέχουν συναίσθημα προσδιορίζουν τους στόχους αυτών των λέξεων. Χρησιμοποιώντας το σύνολο στόχων εξάγουν νέες λέξεις συναισθήματος.

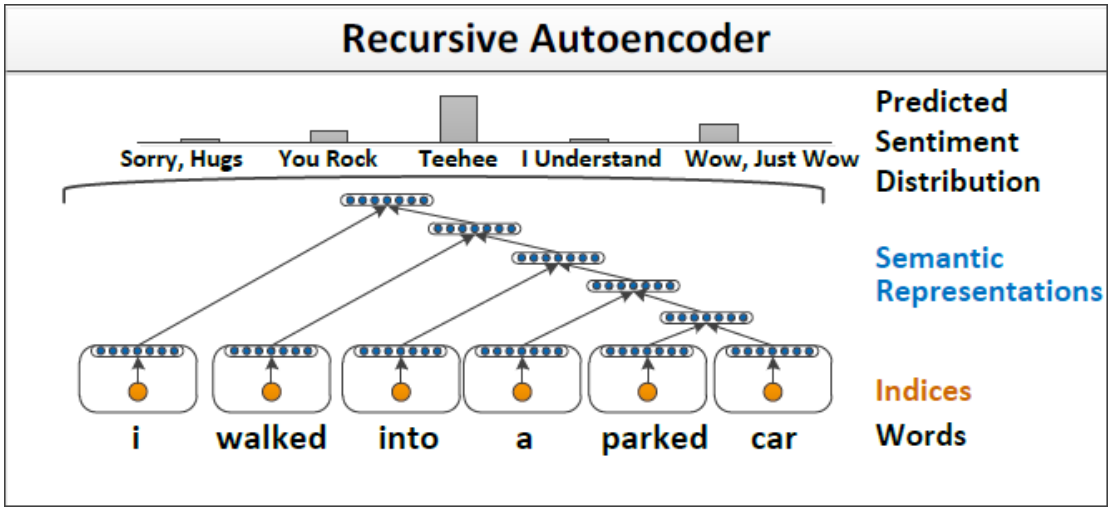
6. **Υπολογισμός συναισθήματος λέξεων:** Για τις λέξεις από το προηγούμενο βήμα αν ισχύουν κάποιοι κανόνες, τότε προστίθενται και αυτές στο λεξικό για τον υπολογισμό του συναισθήματος.

Η συλλογή, που χρησιμοποίησαν για τα πειράματα τους, είναι σχόλια χρηστών για προϊόντα από την ιστοσελίδα "Skroutz". Το dataset τους αναφέρεται σε 7 διαφορετικά προϊόντα (αντικείμενα). Η ακρίβεια (accuracy) πρόβλεψη του συναισθήματος κυμαίνεται από 59,95% μέχρι 83,09%.

Μία πολύ ενδιαφέρουσα προσέγγιση στο πρόβλημα της αυτόματη εκτίμηση της διάθεσης κειμένων γίνεται από τον Socher και την ομάδα του από το Stanford University [13]. Οι ερευνητές αυτοί εισάγουν ένα μοντέλο μηχανικής μάθησης, που βασίζεται σε αναδρομικούς αυτοσυσχετιστές (recursive autoencoders) για την πρόβλεψη του συναισθήματος με κατανομή σε επίπεδο προτάσεων. Η μέθοδός αυτή μαθαίνει να αναπαριστά προτάσεις ως διάνυσμα. Στην πρόβλεψη του συναισθήματος αυτή η προσέγγιση έχει καλύτερα αποτελέσματα από άλλες μεθόδους, που δεν κάνουν χρήση προκαθορισμένων λεξικών ή κανόνων αλλαγής πολικότητας στις ευρέως χρησιμοποιημένες συλλογές (datasets), όπως οι κριτικές ταινιών. Το συγκεκριμένο μοντέλο αξιολογήθηκε, επίσης, στην ικανότητα να προβλέψει κατανομές συναισθημάτων σε μία νέα συλλογή δεδομένων. Η νέα συλλογή δεδομένων αποτελείται από προσωπικές ιστορίες χρηστών σχολιασμένες με πολλαπλές ετικέτες συναισθημάτων, οι οποίες περιγράφουν διάφορες συναισθηματικές αντιδράσεις. Ο αλγόριθμος μπορεί να προβλέψει με μεγαλύτερη ακρίβεια την κατανομή αυτών των συναισθημάτων σε σχέση με άλλους αλγόριθμούς.

Η εργασία του Socher ασχολείται με τρία κύρια ζητήματα.

1. Δεν χρησιμοποιεί τις λέξεις μιας πρότασης αγνοώντας την θέση τους μέσα στην πρόταση (bag of words), αλλά εκμεταλλεύεται την ιεραρχική δομή και την χρησιμοποιεί για να κατανοήσει την συναισθηματική σημασιολογία.
2. Το σύστημά μπορεί να εκπαιδευτεί για οποιαδήποτε δεδομένα και δεν απαιτεί ειδικά συναίσθημα λεξικά, αναλυτές, κ.λπ. γ) αντί να περιοριστεί η πρόβλεψη για τα συναισθήματα σε θετικό / αρνητικό, ταξινομούνται σε μια πολυδιάστατη κατανομή από συναισθηματικές καταστάσεις.



Εικόνα 2 Αναπαράσταση των recursive auto encoders [13] *οι οποίοι μαθαίνουν σημασιολογικές αναπαραστάσεις φράσεων. Οι λέξεις αρχικά αναπαρίστανται σε ένα διάνυσμα και έπειτα συμπιέζονται σε διάνυσμα προκαθορισμένου μήκους με τους ίδιους auto encoders*

Ο Socher και οι συνεργάτες του εισάγουν μια προσέγγιση που βασίζεται σε ημι-επιβλεπόμενους, αναδρομικούς αυτοσυσχετιστές (semisupervised autoencoders RAE), οι οποίοι χρησιμοποιούν ως είσοδο συνεχή διανύσματα λέξεων. Στην εικόνα 2 απεικονίζεται το μοντέλο, που μαθαίνει να αναπαριστά διανύσματα προτάσεων, καθώς και την ιεραρχική δομή τους. Στην συνέχεια το μοντέλο μαθαίνει να κατανέμει σε μία σειρά συναισθημάτων για κάθε κόμβο.

Η δική μας εργασία στηρίζεται στην εργασία του Socher, όπου κάνουμε χρήση των ίδιων μοντέλων για την πρόβλεψη του συναισθήματος κειμένων στα ελληνικά.

3

Θεωρητικό υπόβαθρο

Οι επαναληπτικοί αυτοσυσχετιστές (Recursive Autoencoders – RAE) είναι μια αρχιτεκτονική βαθιάς μάθησης (Deep Learning), η οποία τα τελευταία χρόνια έχει κυριαρχήσει στο χώρο της μηχανικής μάθησης.

3.1 DeepLearning

Τα Deep Neural Networks αποδίδουν πολύ καλύτερα από άλλες τεχνικές μηχανικής μάθησης στους περισσότερους τομείς που έχουν εφαρμοστεί, όπως οπτική αναγνώριση, αναγνώριση φωνής, εύρεση προτύπων κτλ.

Το Deep Learning μιμείται τον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου, ο οποίος είναι ικανός να επεξεργαστεί περίπλοκα δεδομένα, να μάθει διαφορετικές γνώσεις γρήγορα, και να επιλύσει διαφορετικούς τύπους προβλημάτων και πολυπλοκότητας. Μεταφέροντας αυτά τα χαρακτηριστικά του ανθρώπινου εγκεφάλου σε μοντέλα μηχανικής μάθησης ευελπιστούμε πως τα μοντέλα αυτά θα μπορέσουν να υποστηρίζουν γρηγορότερους και πιο αποδοτικούς αλγορίθμους και να ανταποκριθούν με μεγαλύτερη ακρίβεια σε περίπλοκα προβλήματα της τεχνητής νοημοσύνης, όπως η μηχανική όραση και αναγνώριση φωνής. Το Deep Learning έχει τα χαρακτηριστικά ενός τέτοιου μοντέλου, με καλούς αλγόριθμους μάθησης και καλές επιδόσεις στην επίλυση προβλημάτων τεχνητής νοημοσύνης[14].

Σε αντίθεση με τις αρχιτεκτονικές των ρηχών δικτύων, που περιέχουν ένα μικρό αριθμό στρωμάτων, η αρχιτεκτονική του Deep Learning αναφέρεται σε ένα δίκτυο πολλών

στρωμάτων, όπου κάθε ζευγάρι γειτονικών στρωμάτων συνδέονται μεταξύ τους με κάποιον τρόπο. Όπως σημειώθηκε και από τους Bengio και Lecun [15] «Οι βαθιές αρχιτεκτονικές αποτελούνται από πολλά στρώματα μη γραμμικών συστατικών, με άλλα λόγια, προωθούν παραμετροποιημένα μη γραμμικά μοντέλα, που περιέχουν παραμέτρους εκπαίδευσης σε όλα τα επίπεδα».

Σύμφωνα με τον παραπάνω ορισμό θα βγάζαμε το συμπέρασμα ότι η μεγαλύτερη διαφορά ανάμεσα στις ρηχές και τις βαθιές αρχιτεκτονικές είναι ο αριθμός στρωμάτων.

Οι Bengio και Lecun [15] [16] πρότειναν την αξιολόγηση των δυνατοτήτων μιας αρχιτεκτονικής με βάση τα παρακάτω χαρακτηριστικά:

- Ευελιξία στο να καθορίζει την αρχική γνώση και έναν αλγόριθμο μάθησης, που να μπορεί να προσαρμόζεται σε διάφορες αρχιτεκτονικές.
- Αλγόριθμο εκπαίδευσης, που να μπορεί να προσαρμοστεί σε βαθιές αρχιτεκτονικές, στις οποίες μια απόφαση συνεπάγεται τη χειραγώγηση από πολλές ενδιάμεσες έννοιες και πολλαπλά επίπεδα μη γραμμικών βημάτων.
- Αλγόριθμο εκπαίδευσης, που να μπορεί να χειριστεί πολλές συναρτήσεις και να μπορεί να παραμετροποιείται με εκατομμύρια παραμέτρους.
- Αλγόριθμο εκπαίδευσης, που να μπορεί να εκπαιδευτεί ακόμη και όταν ο αριθμός των παραδειγμάτων εκπαίδευσης είναι πολύ μεγάλος. Αυτό αποκλείει αλγορίθμους εκπαίδευσης που θα αποθηκεύουν και να επαναλαμβάνουν πολλές φορές τα δεδομένα εκπαίδευσης.
- Αλγόριθμο εκπαίδευσης, που να μπορεί να ανακαλύψει τα χαρακτηριστικά που μπορούν να χρησιμοποιηθούν για πολλαπλά καθήκοντα (μάθηση πολλαπλών καθηκόντων (multi-task learning) και να μπορούν να εκμεταλλευτούν μεγάλες ποσότητες μη επισημασμένων δεδομένων (semi-supervised learning).

Από τα παραπάνω χαρακτηριστικά, που πρέπει να έχει μια καλή αρχιτεκτονική βγάζουμε το συμπέρασμα, ότι τα περισσότερα σχετίζονται με τον αλγόριθμο μάθησης. Συνοψίζοντας, στόχος μας είναι να σχεδιάσουμε μια αρχιτεκτονική η οποία:

- Να μπορεί να μάθει από τα δεδομένα εισόδου.
- Να έχει πολλαπλά στρώματα, όπου κάθε δύο γειτονικά στρώματα να συνδέονται μέσω μη-γραμμικών βημάτων.
- Να έχει διάφορες παραμέτρους εκπαίδευσης.
- Να έχει καλή κλιμάκωση.
- Να μπορεί να υποστηρίξει διαφορετικά παραδείγματα εκπαίδευσης, όπως πολλαπλών καθηκόντων και ημιεποπτευόμενη μάθηση.

Με αυτά τα κριτήρια, ο Bengio και ο Lecun κατέληξαν σε 2 συμπεράσματα:

- Οι ρηχές αρχιτεκτονικές μπορεί να είναι αναποτελεσματικές όσον αφορά τον απαιτούμενο αριθμό των νευρώνων και δεδομένων.
- Οι βαθιές αρχιτεκτονικές δεν περιορίζονται σε γκαουσιανές κατανομές, αλλά έχουν τη δυνατότητα να γενικεύουν με μη-τοπικό τρόπο.

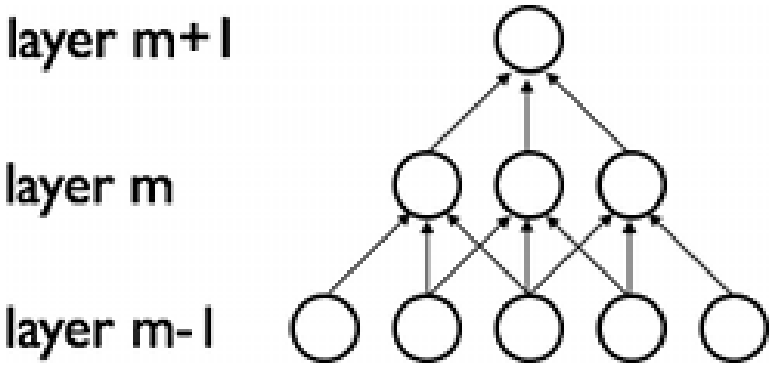
Υπάρχουν διάφορες αρχιτεκτονικές Deep Learning, αλλά τα Convolutional Neural Networks (CNNs) και τα Deep Belief Networks (DBNs) είναι ορόσημα στο πεδίο του Deep Learning. Πριν το 2006 δοκιμαστήκαν διάφορες μέθοδοι για την εκπαίδευση νευρωνικών δικτύων πολλών στρωμάτων, καμία όμως δεν είχε τα επιθυμητά αποτελέσματα με εξαίρεση τα CNNs. Το 2006, ο Hinton παρουσίασε μία νέα αρχιτεκτονική [17], που την ονόμασε Deep Belief Networks (DBNs), η οποία είναι ένα υβριδικό μοντέλο, που αποτελείται από ένα μη-κατευθυνόμενο και έναν κατευθυνόμενο γράφο (πολλαπλά στρώματα Restricted Boltzmann Machine).

3.1.1 Convolutional Neural Networks

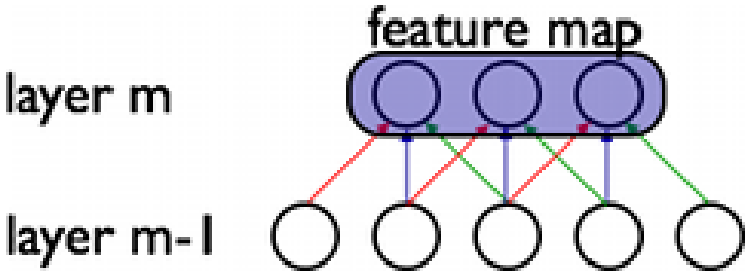
Τα Convolutional Neural Networks [18] ανήκουν στην οικογένεια νευρωνικών δικτυών πολλαπλών στρωμάτων, σχεδιασμένα για χρήση σε δεδομένα 2 διαστάσεων, όπως εικόνες και βίντεο. Τα CNNs είναι η πρώτη επιτυχημένη προσέγγιση Deep Learning, όπου πολλά στρώματα εκπαιδεύονται ιεραρχικά με επιτυχία.

Τα CNNs συνδυάζουν τρεις αρχιτεκτονικές ιδέες: Τα τοπικά πεδία λήψης (local receptive field), κοινόχρηστα βάρη (shared weights,) και χωρική ή χρονική υποδειγματοληψία (spatial or temporal subsampling). Τα τοπικά πεδία λήψης είναι η χρήση ενός μικρού τμήματος των δεδομένων. Τα κοινόχρηστα βάρη κάνουν τα χαρακτηριστικά στο ίδιο στρώμα να παραμετροποιούνται με τον ίδιο τρόπο. Η χωρική ή χρονική υποδειγματοληψία συρρικνώνει τα χαρακτηριστικά σχηματίζοντας ένα νέο σύνολο χαρακτηριστικών με μικρότερη διάσταση.

Στην εικόνα 3 φαίνεται η αραιή συνδεσιμότητα των CNNs. Στην εικόνα 4 βλέπουμε τα κοινόχρηστα βάρη μεταξύ δυο γειτονικών στρωμάτων. Αν τα κόκκινα, μωβ και πράσινα βέλη δείχνουν ένα σύνολο από φίλτρα, μπορούμε να πούμε, ότι το κάθε χαρακτηριστικό στο στρώμα m παραμετροποιείται από το ίδιο σύνολο φίλτρων.

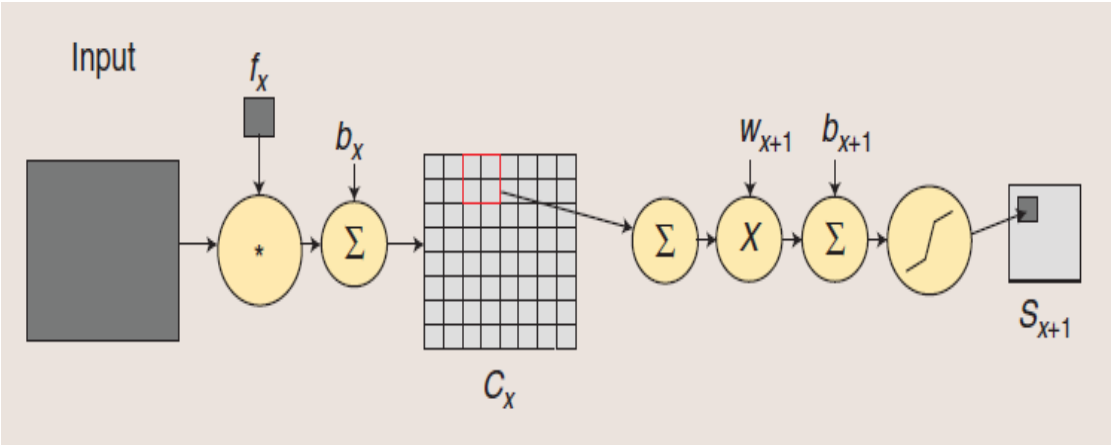


Εικόνα 3 Αραιή συνδεσιμότητα των CNNs) [19]



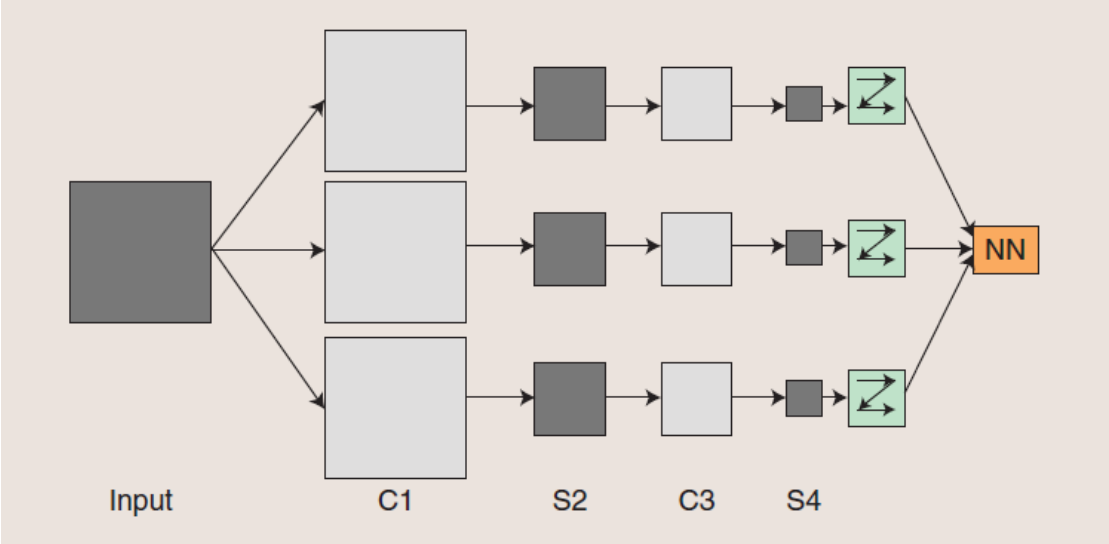
Εικόνα 4 Κοινόχρηστα βάρη μεταξύ δυο γειτονικών στρωμάτων στα CNNs [19]

Τα CNNs αποτελούνται από διαδοχικές διαδικασίες συνέλιξης και υποδειγματολειψίας. Στην εικόνα 5 παρουσιάζεται η διαδικασία της συνέλιξης, όπου αναλύονται τα δεδομένα μίας εισόδου με ένα εκπαιδευόμενο φίλτρο f_x, που αθροίζονται με μία πόλωση b_x για να παραχθεί το συνελεκτικό στρώμα C_x. Στη συνέχεια η διαδικασία της υποδειγματοληψίας αθροίσει τα γειτονικά στοιχεία (τέσσερα κόκκινα pixel) με τα βάρη W_{x+1} και την πόλωση b_{x+1}. Έπειτα μία σιγμοειδής συνάρτηση παράγει ένα μικρότερο πλήθος χαρακτηριστικών το S_{x+1}. Το C_x λαμβάνεται σαν η πληροφορία εισόδου, μετατρέποντας το στο S_{x+1} μικρότερων διαστάσεων χαρακτηριστικά μέσω της υποδειγματολειψίας.



Εικόνα 5 Οι διαδικασίες της συνέλιξης και της υποδιγματολιψίας[18]. Όπου τα δεδομένα εισόδου παράγουν το συνελεκτικό στρώμα C_x μέσω της διαδικασίας της συνέλιξης. Έπειτα το C_x αντιμετωπίζεται ως δεδομένα εισόδου, μετατρέποντας τα σε σετ μικρότερων διαστάσεων χαρακτηριστικών S_{x+1} μέσω της υποδειγματολειψίας.

Στην εικόνα 6 παρουσιάζεται ολόκληρη η διαδικασία, που υπολογίζει μια έξοδο από το CNNs. Βλέπουμε ότι μετά από κάθε στρώμα οι διαστάσεις των χαρακτηριστικών μειώνονται κι αυτό σημαίνει πως τα CNNs είναι δίκτυα, που μειώνουν τον αριθμό των παραμέτρων, με τους οποίους θα πρέπει να εκπαιδευτούν μέσω ενός αλγορίθμου Backpropagation ή μιας παραλλαγής του.



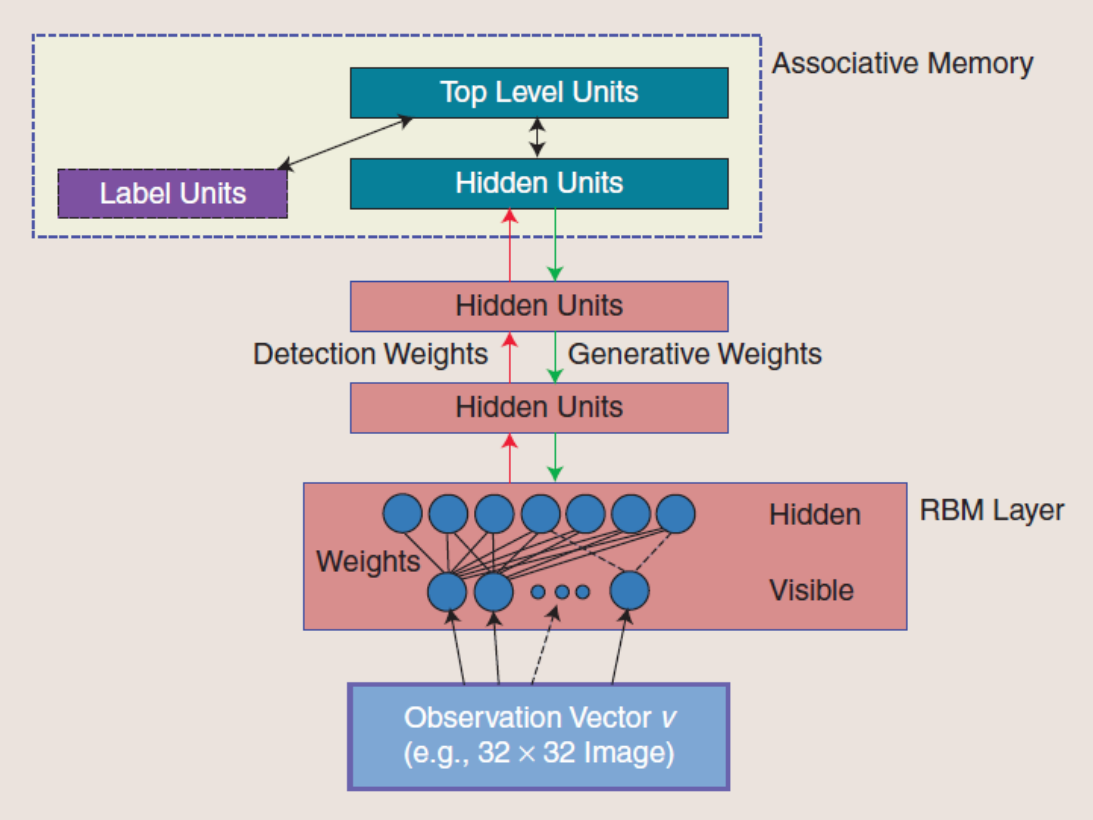
Εικόνα 6 Ολοκληρωμένο παράδειγμα CCN[18]. Τα δεδομένα εισόδου (μια εικόνα) περνάν από τρείς σειρές διαδικασιών συνέλιξης και της υποδειγματοληψίας όπως αυτό της εικόνας 5. Το «NN» είναι η τελική έξοδος από την διαδικασία της συνέλιξης.

Τα CCNs έχουν εφαρμοστεί για την επίλυση διαφόρων προβλημάτων (οπτική αναγνώριση, ανάλυση βίντεο, επεξεργασία φυσικής γλώσσας κτλ.). Στο τομέα της οπτικής αναγνώρισης έχουν επιτύχει ποσοστό σφάλματος 0,23% στην συλλογή δεδομένων MNIST[1], το οποίο είναι και το ελάχιστο που έχει επιτευχθεί [20]. Έχουν εφαρμοστεί και στην αναγνώριση προσώπων, όπου και έχουν μειώσει κατά πολύ το ποσοστό σφάλματος.

3.1.2 Deep Belief Networks(DBNs)

Τα Deep Belief Networks [14] είναι ένα υβριδικό μοντέλο, που αποτελείται από δύο μέρη. Στην εικόνα 7 βλέπουμε τα δύο πάνω στρώματα, τα οποία είναι μη-κατευθυνόμενοι γράφοι, που αποτελούν την συσχετιστική μνήμη. Τα υπόλοιπα στρώματα είναι κατευθυνόμενοι γράφοι, στην ουσία είναι Restricted Boltzmann Machines (RBM). Η διαφορά από τα CCNs είναι πως τα DBNs είναι στοχαστικές αρχιτεκτονικές που εξαρτώνται από το στόχο μάθησης. Γενικά τα DBNs μπορούν να εκπαιδευτούν για διάφορα προβλήματα (προβλήματα συμπερασμάτων, ταξινόμησης κτλ.) ή να δημιουργούν δεδομένα για εκπαίδευση.

[1] Η συλλογή χειρόγραφων ψηφίων που αποτελείται από 60000 παραδείγματα για εκπαίδευση και 10000 για έλεγχο. Χρησιμοποιείται ως αναφορά για σύγκριση μεταξύ αλγορίθμων αναγνώρισης προτύπων.



Εικόνα 7 Αναπαράσταση των Deep Belief Networks.

Βλέπουμε την διαδικασία από κάτω προς τα πάνω (κόκκινο βέλος), όπου μαθαίνει τα βάρη βελτιώνοντας την πιθανότητα $P(Label|Obversion)$. Το παραγόμενο μοντέλο είναι το από πάνω προς τα κάτω (πράσινο βέλος), το οποίο μαθαίνει τα παραγόμενα βάρη βελτιώνοντας την πιθανότητα $P(Label, Obversion)$.

Το μεγαλύτερο πλεονέκτημα των DBNs είναι η δυνατότητα τους να μαθαίνουν τα χαρακτηριστικά εκπαίδευσης από στρώμα σε στρώμα μέσω μεθόδων, όπου τα χαρακτηριστικά στα υψηλότερα στρώματα μαθαίνονται μέσω των προηγούμενων στρωμάτων. Τα χαρακτηριστικά στα υψηλότερα στρώματα είναι πιο περίπλοκα και περιγράφουν καλύτερα την πληροφορία εισόδου. Αποδεικνύεται, ότι προσθέτοντας ακόμη ένα στρώμα χαρακτηριστικών βελτιώνεται η εκτίμηση της κατανομής των δεδομένων εισόδου.

Οι εφαρμογές των DBNs κυμαίνονται από ταξινόμηση εικόνων και αναγνώριση ομιλίας μέχρι και ταξινόμηση ήχων. Μια ενδιαφέρουσα εφαρμογή των DBNs είναι του Geoffrey Hinton και της ομάδας του, οι οποίοι ανάπτυξαν εφαρμογή για την κατανόηση της φυσικής γλώσσας [21]. Στόχος της εφαρμογής (Spoken Language Understanding SLU) τους είναι η επικοινωνία μεταξύ ανθρώπου και μηχανής. Το σύστημα SLU εντοπίζει την πρόθεση του χρήστη από την φωνή του. Εξάγοντας πληροφορίες από τις λέξεις και εκτελώντας ερωτήματά σε μία βάση δεδομένων ικανοποιεί τα αιτήματα του χρήστη.

3.1.3 Αυτοσυσχετιστές

Οι αυτοσυσχετιστές (Autoencoders, AE) είναι τεχνητά νευρωνικά δίκτυα, όπου η είσοδος είναι ίση με το στόχο τους και εκπαιδεύονται, έτσι ώστε να μπορούν να ανακατασκευάσουν την είσοδο τους μέσω μιας ενδιάμεσης αναπαράστασης συχνότερα μικρότερης διάστασης από την είσοδο.

Ένα δίκτυο αυτοσυσχετιστών μπορεί να αποτελείται και από ένα μόλις κρυφό στρώμα και να εκπαιδευτεί με τη χρήση του αλγορίθμου Backpropagation ή κάποιας παραλλαγής του, έτσι ώστε η έξοδος του να ανακατασκευάζει με όσο το δυνατόν καλύτερη ακρίβεια την είσοδο του. Η δομή του δικτύου μοιάζει με τη δομή ενός Multilayer Perceptron ενός μόνο κρυφού στρώματος. Η διαφορές τους είναι, ότι στα δίκτυα αυτοσυσχετιστών συνήθως το κρυφό στρώμα είναι μικρότερης διάστασης από το στρώμα εισόδου και ενώ το MLP εκπαιδεύεται να αντιστοιχεί μία είσοδο x σε μία έξοδο y, ένα δίκτυο autoencoder εκπαιδεύεται να μπορέσει να ανακατασκευάσει την ίδια είσοδο, δηλαδή για μία είσοδο x να έχουμε μία έξοδος επίσης x.[22]

Ένας αυτοσυσχετιστής δέχεται μία είσοδο x και την κωδικοποιεί σε μία κρυφή αναπαράσταση y (κρυφό στρώμα) μέσω μιας αντιστοίχισης: $y = S(Wx + b)$, όπου S είναι κάποια συνάρτηση, όπως η σιγμοειδής ή υπερβολική εφαπτομένη ή άλλη. Στη συνέχεια, η κρυφή αναπαράσταση y αποκωδικοποιείται σε μια ανακατασκευή x_r ίδιου μεγέθους με την είσοδο, μέσω μιας παρόμοιας διαδικασίας: $x_r = S(Wy + b)$ x_r είναι η πρόβλεψη για την εισόδου x μέσο της αναπαράστασης y.

Στόχος είναι η εύρεση κατάλληλων παραμέτρων στο μοντέλο, έτσι ώστε το σφάλμα ανακατασκευής της εισόδου να είναι το ελάχιστο. Ένας τρόπος μέτρησης του σφάλματος είναι το μέσο τετραγωνικό σφάλμα, δηλαδή $l = E\{\|x - x_r\|^2\}$.

Αφού, ο αριθμός νευρώνων στο κρυφό στρώμα είναι μικρότερος από τους νευρώνες στα στρώματα εισόδου και εξόδου η αναπαράσταση της πληροφορίας στο κρυφό στρώμα είναι μικρότερη. Έτσι, οι ενεργοποιήσεις των νευρώνων στο κρυφό στρώμα μπορούν να θεωρηθούν ως μία συμπιεσμένη αναπαράσταση των αρχικών δεδομένων. [16]

3.1.4 Others

Εκτός από τα CNNs, τα DBNs και τους αυτοσυσχετιστές, υπάρχουν και πολλές άλλες παραλλαγές τους. Όπως το 3D CNNs [23], όπου η $3^{η}$ διάσταση είναι μόνιμα χαρακτηριστικά. Το CDBNs, που συνδυάζει τα CNNs με τα DBNs, όπου διαθέτει ένα επιπλέον στρώμα που ονομάζεται «στρώμα συγκέντρωσης» μεταξύ κάθε δύο παρακείμενων στρωμάτων RBM. Το CDBNs [24] προορίζεται για εικόνες με μεγάλες διαστάσεις, που αποδειχτικέ πώς αποδίδει καλύτερα από τα DBNs

3.2 Επαναληπτικοί Αυτοσυσχετιστές (Recursive Autoencoders –RAE)

Το μοντέλο των επαναληπτικών αυτοσυσχετιστών έχει ως στόχο να βρει το διάνυσμα αναπαράστασης για φράσεις μεταβλητού μεγέθους. Οι αναπαραστάσεις μπορεί να χρησιμοποιηθούν για μετέπειτα διεργασίες. Αρχικά, θα αναπαρασταθούν οι λέξεις στους νευρώνες εισόδου και στη συνέχεια θα γίνει επεξεργασία από το νευρωνικό δίκτυο, στο όποιο βασίζονται οι αυτοσυσχετιστές και μπορεί να τροποποιηθεί για να εκπαιδεύεται με φράσεις αναπαραστάσεων, φράσεις δομών και συναισθημάτων.

3.2.1 Αναπαράσταση λέξεων στους νευρώνες

Οι λέξεις αναπαρίστανται ως ένα συνεχές διάνυσμα παραμέτρων και μπορούν να αναπαρασταθούν με 2 τρόπους:

Στην πρώτη περίπτωση, η κάθε λέξη αρχικά αναπαριστάται ως $x \in R^n$ μέσω κανονικής γκαουσιανής (Gaussian) κατανομής και έπειτα τα διανύσματα αναπαράστασης τοποθετούνται σε έναν πίνακα λέξεων $L \in R^{n \times v}$ v είναι το μέγεθος του λεξικού. Αυτή η διαδικασία λειτουργεί καλά στην επιβλεπόμενη περίπτωση, όπου ένα δίκτυο μπορεί να τροποποιήσει τα διανύσματα για να λάβει ορισμένες κατανομές.

Στην δεύτερη περίπτωση, γίνεται προ-εκπαίδευση των διανυσμάτων αναπαράστασης των λέξεων με μη-επιβλεπόμενα μοντέλα. Αυτά τα μοντέλα μαθαίνουν από κοινού να ενσωματώνουν τις λέξεις σε ένα διανυσματικό χώρο και να χρησιμοποιούν αυτά τα διανύσματα για να προβλέψουμε πόσο πιθανόν είναι να ανήκει μια λέξη σε ένα δεδομένο πλαίσιο. Μετά την εκπαίδευση μέσω ανάβασης δυναμικού, τα διανύσματα λέξεων συμπεριλαμβάνουν συντακτική και σημασιολογική πληροφορία των λέξεων.

Και στις δύο περιπτώσεις μπορεί να χρησιμοποιηθεί ο πίνακας με τα διανύσματα των λέξεων L για τις επόμενες εργασίες ως εξής: Υποθέτουμε πως δίνουμε μία πρόταση ως ταξινομημένη λίστα m λέξεων. Η κάθε λέξη έχει έναν δείκτη στο λεξικό του πίνακα, που χρησιμοποιείται για να πάρουμε την διανυσματική αναπαράσταση. Αυτή η λειτουργία μαθηματικά μπορεί να θεωρηθεί ως ένα απλό δυαδικό διάνυσμα b που χρησιμοποιεί το 0 σε όλες τις θέσεις εκτός από την θέση του δείκτη k .

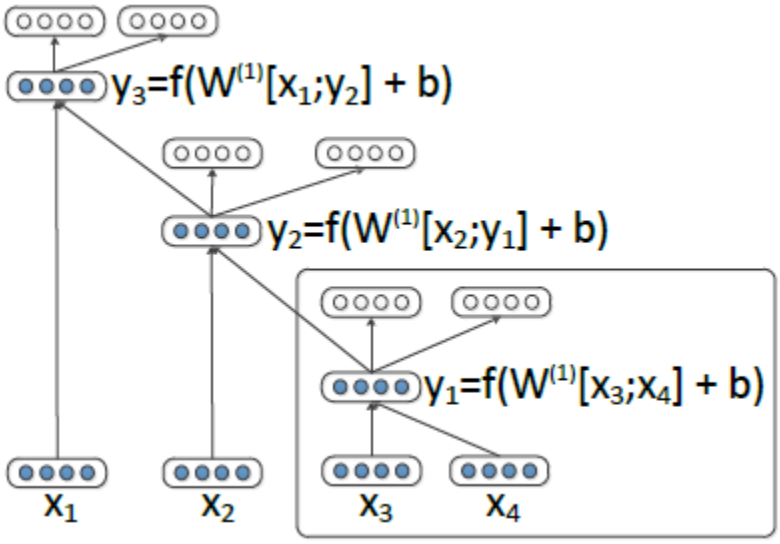
$$x_i = Lb_k \in R^n$$

Εξίσωση 1

Η αναπαράσταση μιας πρότασης ως διατεταγμένη λίστα από διανύσματα $(x_1, \dots, x_{m)}$. είναι πιο συμβατή με τους αυτοσυσχετιστές, σε σχέση με την αναπαράσταση δυαδικού αριθμού.

3.2.2 Παραδοσιακοί Επαναληπτικοί Αυτοσυσχετιστές

Στόχος των αυτοσυσχετιστών είναι να μάθουν να αναπαριστούν τις εισόδους τους. Στην εικόνα 8 παρουσιάζεται ένα παράδειγμα ενός επαναληπτικού αυτοσυσχετιστή που εφαρμόζεται σε ένα δέντρο.



Εικόνα 8 Αναπαράσταση εφαρμογής επαναληπτικών αυτοσυσχετιστών

Ας υποθέσουμε πως μας δίνεται μια λίστα με διανύσματα λέξεων $x = (x_1, \dots, x_m)$.

Ο κάθε κόμβος του δένδρου θα μπορεί να είναι είτε ένα διάνυσμα λέξης x_i ή ένας μη τερματικός κόμβος του δένδρου. Για το παράδειγμα της εικόνας 8, θα έχουμε τις εξής τριπλέτες: $((y_1 \to x_3 x_4), (y_2 \to x_2 y_1), (y_3 \to x_1 y_2))$. Για να μπορέσουμε να εφαρμόσουμε το ίδιο νευρωνικό δίκτυο σε κάθε ζευγάρι παιδιών, οι κρυφές αναπαραστάσεις y_i θα πρέπει να έχουν την ίδια διάσταση με το x_i.

Δεδομένης αυτής της δομής του δένδρου, είμαστε σε θέση να υπολογίσουμε τις αναπαραστάσεις των γονέων. Το πρώτο διάνυσμα γονέα y_1 υπολογίζεται από τα παιδιά $(c_1, c_2) = (x_3, x_4)$,

όπου πολλαπλασιάζοντας τον πίνακα παραμέτρων $W^{(1)} \in R^{n \times 2n}$ από την συνένωση των δυο παιδιών. Μετά την προσθήκη μιας πόλωσης εφαρμόζουμε ένα στοιχείο ενεργοποίησης μιας συνάρτησης tanh για να πάρουμε το διάνυσμα εξόδου.

$$p = f\left(W^{(1)}[c_1; c_2] + b^{(1)}\right)$$

Εξίσωση 2

Ένας τρόπος για να αξιολογηθεί το διάνυσμα που αντιπροσωπεύει τα παιδιά του είναι να ανακατασκευαστούν τα παιδιά σε ένα στρώμα ανακατασκευής:

$$[c_1'; c_2'] = W^{(2)}p + b^{(2)}$$

Εξίσωση 3

Κατά τη διάρκεια της εκπαίδευσης, στόχος είναι να ελαχιστοποιηθούν τα σφάλματα ανακατασκευής αυτών των ζευγών εισόδου. Για κάθε ζεύγος, μπορούμε να υπολογίσουμε την Ευκλείδια απόσταση μεταξύ της αρχικής εισόδου και της ανακατασκευασμένης:

$$E_{rec}([c_1; c_2]) = \frac{1}{2}\|[c_1; c_2] - [c_1'; c_2']\|^2$$

Εξίσωση 4

Αυτό είναι το πρότυπο μοντέλο αυτοσυσχετιστών. Εφόσον, έχει καθορίσει πως μπορεί να χρησιμοποιηθεί ένας αυτοσυσχετιστής για τον υπολογισμό ενός διανύσματος (p) αναπαράστασης δυο παιδιών c_1, c_2, μπορούμε να περιγράψουμε και πώς ένα τέτοιο δίκτυο μπορεί να χρησιμοποιηθεί και για το υπόλοιπο δένδρο.

Ουσιαστικά επαναλαμβάνονται τα ίδια βήματα. Τώρα, με δεδομένο το y_1 μπορούμε να χρησιμοποιήσουμε την εξίσωση 2 για να υπολογίσουμε το y_2 θέτοντας ως παιδιά τα $(c_1, c_2) = (x_2, y_1)$. Μετά τον υπολογισμό του διανύσματος y_2, μπορούμε να εκτιμήσουμε πόσο καλά αυτό το διάνυσμα λαμβάνει το περιεχόμενο των παιδιών του υπολογίζοντας το λάθος στην ανακατασκευή, όπως στην εξίσωση 4. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να κατασκευαστεί το πλήρες δένδρο και να έχουμε ένα λάθος ανακατασκευής σε κάθε μη τερματικό κόμβο. Αυτό το μοντέλο είναι παρόμοιο με το RAAM[25] μοντέλο, το οποίο επίσης απαιτεί μια σταθερή δεντρική δομή.

3.2.3 Αναδρομικοί Αυτοσυσχετιστές χωρίς επίβλεψη για την πρόβλεψη δομής

Υποθέτουμε ότι δεν δίνεται καμία δενδρική δομή για τα διανύσματα εισόδου x. Ο στόχος της πρόβλεψης δομής των RAE είναι να ελαχιστοποιήσουν το κατασκευαστικό σφάλμα όλων των ζευγαριών διανυσμάτων των παιδιών σε ένα δένδρο. Ορίζουμε την $A(x)$ ως το σύνολο των πιθανών δένδρων, που μπορούν να κατασκευαστούν από μια πρόταση εισόδου x και η $T(y)$ να είναι μια συνάρτηση, η οποία επιστρέφει όλες τις τριπλέτες των μη τερματικών κόμβων του δένδρου y. Χρησιμοποιώντας το σφάλμα ανακατασκευής της εξίσωσης 4, υπολογίζουμε:

$$RAE_\theta(x) = arg\min_{y \in A(x)} \sum_{s \in T(y)} E_{rec}([c_1; c_2]_s)$$

Εξίσωση 5

3.2.3.1 Άπληστη προσέγγιση κατασκευής δένδρου:

Για μια πρόταση από m λέξεις εφαρμόζουμε τους αυτοσυσχετιστές αναδρομικά. Παίρνοντας το πρώτο ζευγάρι γειτονικών διανυσμάτων, τα καθορίζουμε σαν πιθανά παιδιά μιας φράσης $(c_1, c_2) = (x_1, x_2)$ τα συνενώνουμε και τα δίνουμε ως είσοδο στους αυτοσυσχετιστές. Για κάθε ζευγάρι λέξεων αποθηκεύουμε τον πιθανό κόμβο γονέα p και το λάθος ανακατασκευής, που προκύπτει.

Μετά τον υπολογισμό του λάθους ανακατασκευής για το πρώτο ζεύγος, το δίκτυο μετατοπίζεται κατά μια θέση και λαμβάνει ως διάνυσμα εισόδου $(c_1, c_2) = (x_2, x_3)$ και ξανα-υπολογίζει τον πιθανό κόμβο γονέα και το λάθους ανακατασκευής. Αυτή η διαδικασία επαναλαμβάνεται μέχρι το τελευταίο ζευγάρι λέξεων μιας πρότασης $(c_1, c_2) = (x_{m-1}, x_m)$. Έπειτα, επιλέγεται το ζευγάρι, το οποίο έχει το μικρότερο σφάλμα ανακατασκευής E_{rec} και η αναπαράσταση των γονέων του p. Θα αναπαραστήσουν την φράση, που θα αντικαταστήσει και τα δυο παιδιά στην λίστα με τις λέξεις των προτάσεων. Για παράδειγμα, υποθέτουμε πως έχουμε την πρόταση (x_1, x_2, x_3, x_4) και το χαμηλότερο ανακατασκευαστικό σφάλμα το έχει το ζεύγος (x_3, x_4). Μετά την πρώτη διέλευση, η νέα αλληλουχία θα περιλαμβάνει $(x_1, x_2, p_{(3,4)})$. Αυτή η διαδικασία επαναλαμβάνεται και διαχειρίζεται το νέο διάνυσμα $p_{(3,4)}$, όπως οποιοδήποτε άλλο διάνυσμα εισόδου.

3.2.3.2 *Σταθμισμένη ανακατασκευή:*

Ένα πρόβλημα με την χρήση του λάθους ανακατασκευής και των δυο παιδιών, όπως περιγράφεται στην εξίσωση 4, είναι, ότι κάθε παιδί θα μπορούσε να αναπαριστά ένα διαφορετικό αριθμό από προηγούμενες λέξεις και έτσι θα είχε μεγαλύτερη σημασία η συνολική ανακατασκευή της πρότασης. Για παράδειγμα, σε μια περίπτωση, όπου $(x_1, p_{(2,(3,4))})$ θα μπορούσε να δοθεί περισσότερη σημασία στην ανακατασκευή του p, παρά του x_1. Έτσι μπορούμε να ρυθμίζουμε το σφάλμα ανακατασκευής επανακαθορίζοντας το λάθος ανακατασκευής ως:

$$E_{rec}([c_1 c_2]; \theta) = \frac{n_1}{n_1 + n_2} \left\| c_1 - c_1' \right\|^2 + \frac{n_2}{n_1 + n_2} \left\| c_2 - c_2' \right\|^2$$

Εξίσωση 6

Όπου n_1, n_2 είναι ο αριθμός των λέξεων ενός πιθανού παιδιού.

3.2.3.3 *Ημι-Επιβλεπόμενοι Αναδρομικοι Αυτοσυσχετιστές*

Μέχρι τώρα παρατηρήσαμε ότι, οι RAE δουλεύουν χωρίς επίβλεψη και παράγουν γενικές αναπαραστάσεις, που λαμβάνουν την σημασιολογία από φράσεις με πολλές λέξεις. Μπορούμε να προεκτείνουμε τους υπάρχοντες RAE σε ημι-επιβλεπόμενους, ώστε να προβλέπουν μια πρόταση ή φράση ενός κατανεμημένου στόχου t.

Ένα από τα κύρια πλεονεκτήματα των RAE είναι, ότι κάθε κόμβος του δένδρου που δημιουργείται από τους RAE συσχετίζεται με ένα κατανεμημένο διάνυσμα αναπαράστασης, το οποίο επίσης θα μπορούσε να χαρακτηρίσει αυτή τη φράση. Θα μπορούσαμε να το αξιοποιήσουμε με την προσθήκη στην κορυφή κάθε κόμβου γονέα, ένα επίπεδο softmax[26] για την πρόβλεψη των κατανεμημένων κλάσεων.

$$d(p; \theta) = softmax(W^{label} p)$$

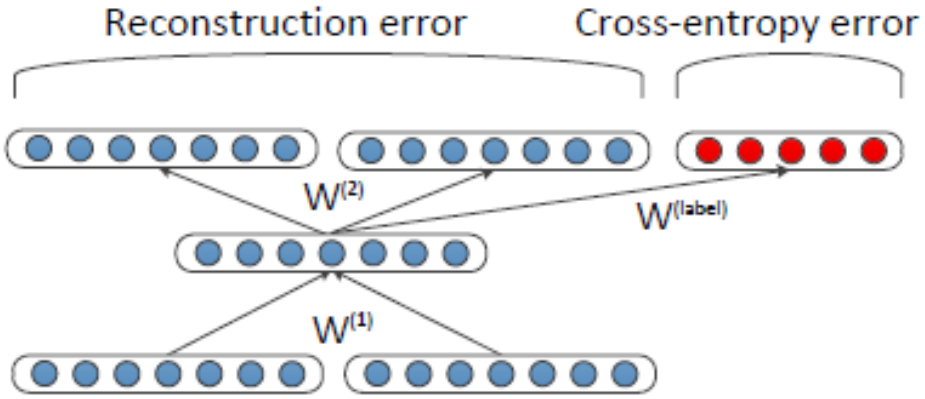
Εξίσωση 7

Υποθέτοντας, οτι υπάρχουν K κλάσεις, $d \in R^k$ είναι μια k-διάστατη πολυωνυμική κατανομή και $\sum_{k=1} d_k = 1$. Στην εικόνα 9 φαίνεται μια ημι-επιβλεπόμενη αναπαράσταση RAE.

Αν υποθέτουμε, οτι το t_k είναι το k στοιχείο της t διανομής της ετικέτας του στόχου για μία είσοδο. Οι έξοδοι του επιπέδου softmax μεταφράζονται ως συνθήκες πιθανοτήτων $d_k = p(k|[c_1; c_2])$, εφόσον το λάθος της εντροπίας είναι:

$$E_{cE}(p, t; \theta) = -\sum_{k=1}^{K} t_k \log d_k(p; \theta)$$

Εξίσωση 8



Εικόνα 9 Παρουσίαση μιας μονάδας RAE σε έναν μη τερματικό κόμβο του δένδρου. Οι κόκκινοι κόμβοι δείχνουν το επίπεδο softmax με επίβλεψη για την πρόληψη κατανεμημένης ετικέτας

Χρησιμοποιώντας το λάθος της εντροπίας για κάθε ετικέτα και το λάθος ανακατασκευής από την εξίσωση, το τελικό αντικείμενο των RAE με ημι-επίβλεψη σε ζεύγη (προτάσεις, ετικέτες) γίνεται ως μια συλλογή:

$$J = \frac{1}{N} \sum_{(x,t)} E(x, t; \theta) + \frac{\lambda}{2} \|\theta\|^2$$

Εξίσωση 9

όπου έχουμε ένα σφάλμα για κάθε είσοδο στο σύνολο εκπαίδευσης, όπου το άθροισμα πάνω από το σφάλμα στους κόμβους του δένδρου κατασκευάζεται από τους άπληστους RAE.

$$E(x, t; \theta) = \sum_{s \in T(RAE_\theta(x))} E([c_1; c_2]_s, P_s, t, \theta)$$

Το σφάλμα σε κάθε μη τερματικό κόμβο είναι το ισοζυγισμένο σύνολο του σφάλματος ανακατασκευής και διασταυρωμένης εντροπίας:

$$E([c_1; c_2]_s, P_s, t, \theta) = aE_{rec}([c_1; c_2]_s; \theta) + (1 - a)E_{cE}(p_s, t; \theta)$$

Η υπερπαράμετρος α λαμβάνει υπόψη το σφάλμα ανακατασκευής και εντροπίας. Όταν ελαχιστοποιείται το σφάλμα εντροπίας στο επίπεδο softmax, το λάθος θα μετακληθεί προς τα πίσω και θα επηρεάσει τις παραμέτρους των RAE και των αναπαραστάσεων των λέξεων.

Αρχικά, οι λέξεις όπως "καλός" και "κακός" έχουν αρκετά παρόμοια αναπαράσταση. Όταν γίνεται η εκπαίδευση με θετικό/αρνητικό συναίσθημα, η ενοποίηση των λέξεων τροποποιείται και καταγράφει λιγότερη συντακτική πληροφορία και περισσότερη συναισθηματική.

Προκειμένου να προληφθεί η κατανομή του συναισθήματος μιας πρότασης με αυτό το μοντέλο, χρησιμοποιούμε το διάνυσμα μάθησης αναπαράστασης του κορυφαίου κόμβου του δένδρου και εκπαιδεύουμε ένα απλό λογιστικό ταξινομητή (logistic regression classifier).

3.2.4 Εκπαίδευση

Υποθέτουμε, πως $\theta = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(1)}, W^{label}, L)$ είναι οι παράμετροι του μοντέλου, τότε η κλίση γίνεται

$$\frac{\partial J}{\partial \theta} = \frac{1}{N} \sum_{(x,t)} \frac{\partial E(x,t;\theta)}{\partial \theta} + \lambda\theta$$

Εξίσωση 10

Για τον υπολογισμό της κλίσης, πρέπει αρχικά να κατασκευαστούν τα δέντρα και στην συνέχεια να υπολογιστεί η παράγωγος για αυτά τα δέντρα μέσω ενός αλγορίθμου Backpropagation σε όλη την δομή των δέντρων. Επειδή, ο αλγόριθμος είναι «άπληστος» και η παράγωγος του σφάλματος της εντροπίας αλλάζει από τον πίνακα $W^{(1)}$ η συνάρτηση κόστους δεν είναι συνέχεις και μια μετακίνηση προς την κατεύθυνση της κλίσης δεν σημαίνει πως θα μειωθεί και το κόστος για να μας φέρει ποιο κοντά στο στόχο. Ωστόσο, τρέχοντας τον αλγόριθμο L-BFGS[27] στα δεδομένα εκπαίδευσης για την εκτίμηση των παραμέτρων δείχνει πως λειτουργεί καλά και ότι η σύγκλιση είναι ομαλή, με τον αλγόριθμο να βρίσκει μια καλή και γρήγορη λύση.

4

Ανάλυση προβλημάτων και ιδιαιτεροτήτων για την αυτόματη εκτίμηση της διάθεσης κειμένων στην Ελληνική γλώσσα

Από τα πρώτα προβλήματα, που γίνονται αντιληπτά στην ανάπτυξη ενός μοντέλου αυτόματης εκτίμησης διάθεσης, είναι η ανάγκη χρήσης ενός λεξικού της γλώσσας, στην οποία θα εφαρμοστεί το μοντέλο και μια συλλογή από προτάσεις κειμένου χαρακτηρισμένες - βαθμολογημένες ως προς το συναίσθημα που περιέχουν για να μπορέσει να δοκιμαστεί και να αξιολογηθεί το μοντέλο.

4.1 Προβλήματα – Αδυναμίες μοντέλων αυτόματης εκτίμησης της διάθεσης

Τα κυριότερα πρόβλημα που αντιμετωπίζουν τα καθιερωμένα μοντέλα ανάλυσης συναισθήματος είναι:

- Τα περισσότερα μοντέλα βασίζονται σε προκαθορισμένα λεξικά, στα οποία θα πρέπει να έχουν χαρακτηριστεί οι λέξεις από ανθρώπους ως προς το βαθμό συνθήματος, που φέρει η κάθε λέξη.
- Ένα μοντέλο αναπτύσσεται για να εφαρμοστεί σε μία συγκεκριμένη γλώσσα, χωρίς να έχει την δυνατότητα ή να χρειάζεται σοβαρές τροποποιήσεις, για να μπορέσει να εφαρμοστεί σε μία άλλη γλώσσα.
- Πολλά μοντέλα κάνουν χρήση των λέξεων μιας πρότασης, χωρίς να λαμβάνουν σημασία την θέση της λέξης μέσα στην πρότασης (bag-of-words), το οποίο περιορίζει τα μοντέλα να αντιληφθούν πιο σύνθετα γλωσσικά φαινόμενα. Για παράδειγμα, οι προτάσεις «τα λευκά αιμοσφαίρια καταστρέφουν μια μόλυνση» και «μια μόλυνση καταστρέφει τα λευκά αιμοσφαίρια» αποτελούνται από τις ίδιες λέξεις, αλλά προς το συναίσθημα που φέρουν είναι εντελώς αντίθετες.
- Κάποια μοντέλα, τα οποία μπορούν να αντιληφθούν πιο σύνθετα γλωσσικά φαινόμενα [28] αυξάνουν σημαντικά την πολυπλοκότητα του μηχανισμού ανάλυσης. Απαιτούν την έντονη ανθρώπινη παρέμβαση στην διαδικασία -κατασκευή με το χέρι εκτεταμένων πηγών (λεξικά συναισθημάτων, αναλυτών κειμένων, κανόνων μετασχηματισμού της πολικότητας των λέξεων). Περιορίζονται από τη θεματολογία των κειμένων, καθώς και από την γλώσσα (ισχύουν μόνο για τη γλώσσα για την οποία προετοιμάζονται).
- Στην μεγαλύτερη πλειοψηφία τους τα μοντέλα περιορίζονται σε ένα μοναδικό αποτέλεσμα (θετικό/αρνητικό ή σε μια βαθμολογία πχ. αστέρια). Ένα τέτοιο μονοδιάστατο αποτέλεσμα δεν μπορεί να αντικατοπτρίσει τα σύνθετα ανθρώπινα συναισθήματα.

4.2 Πλεονεκτήματα χρήσης μοντέλου Επαναληπτικών Αυτοσυσχετιστών (Recursive Autoencoders –RAE)

Στην εργασία μας κάνουμε χρήση του μοντέλου ημι-επιβλεπόμενων Επαναληπτικών αυτοσυσχετιστών, που αναπτύχθηκε από τους Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng και Christopher D. Manning [13].

Ο τρόπος λειτουργίας του μοντέλου των RAE είναι: πως χρησιμοποιούνε ως εισόδους διανύσματα, που αντιστοιχούν σε μεμονωμένες λέξεις. Στη συνέχεια, το μοντέλο μαθαίνει τη

διανυσματική αναπαράσταση των προτάσεων, καθώς και την ιεραρχική δομή του κειμένου χωρίς επίβλεψη. Επίσης, το μοντέλο έχει τη δυνατότητα να «μαθαίνει» μια πολυδιάστατη κατανομή πάνω στα διάφορα επίπεδα του συναισθήματος, που θεωρούμε, σε κάθε κόμβο της ιεραρχημένης δομής μιας φράσης.

Πιο συγκεκριμένα:

1. Οι λέξεις αναπαρίστανται ως συνεχή n-διάστατα διανύσματα $x \in \mathbb{R}^n$.
2. Κατασκευάζεται ένας πίνακας L, όπου $L \in \mathbb{R}^{n \times V}$και V είναι το μέγεθος του θεωρούμενου λεξικού. Ο πίνακας L περιέχει το διάνυσμα αναπαράστασης τις κάθε λέξης που έχει προστεθεί στο λεξικό.
3. Η αντιστοίχιση λέξεων-διανυσμάτων μπορεί να γίνει με δύο τρόπους: (α) με απλή δειγματοληψία από μια κανονική Γκαουσιανή κατανομή $x \in \mathcal{N}(0, \sigma^2)$ (β) με εκπαίδευση των διανυσμάτων των λέξεων με ένα μοντέλο νευρωνικών δικτύων χωρίς επίβλεψη (π.χ. Support Vector Machines). Σε αυτή την περίπτωση τα διανύσματα των λέξεων εμπλουτίζονται με συντακτική και σημασιολογική πληροφορία, που περιέχεται στη στατιστική κατανομή της εμφάνισης των λέξεων στις φράσεις του δείγματος, που χρησιμοποιείται για την εκπαίδευση του συστήματος.
4. Για μια τυχαία δοθείσα πρόταση, που αποτελείται από m λέξεις αντιστοιχίζουμε στην πρόταση μια λίστα διανυσμάτων $(x_1, x_2, \dots, x_m)$, μέσω μιας απλής αναζήτησης των αντίστοιχων λέξεων στον πίνακα L.

Τα τελικά αποτελέσματα της εφαρμογής ενός μηχανισμού με RAE σε μια πρόταση κειμένου είναι:

- Μια αναπαράσταση σταθερού μήκους για το σύνολο της πρότασης, στην οποία συγχωνεύεται η σημασιολογική αναπαράσταση των λέξεων της πρότασης.
- Υπολογισμός των ενδιάμεσων αναπαραστάσεων κατά τα διαδοχικά βήματα εφαρμογής του αλγορίθμου.
- Υπολογισμός της πολικότητας του θετικού/αρνητικού συναισθήματος ή μια κατανομή πάνω σε διάφορα πιο σύνθετα και συσχετισμένα μεταξύ τους συναισθήματα.

Επιλέξαμε το παραπάνω μοντέλο γιατί:

- Στην πρόβλεψη του συναισθήματος επιτυγχάνει υψηλότερες επιδόσεις από άλλα μοντέλα στις πιο κοινές χρησιμοποιημένες συλλογές [13].
- Δεν κάνει χρήση κάποιου προκαθορισμένου λεξικού ή κανόνων αλλαγής της πολικότητας των λέξεων.
- Οι αναπαραστάσεις, που χρησιμοποιεί για τις προτάσεις μπορούν να αντιληφθούν και πιο σύνθετα γλωσσικά φαινόμενα.
- Μπορεί να εφαρμοστεί για κείμενα ανεξάρτητου γλώσσας.

- Η χρήση του παραπάνω μοντέλου μπορεί να βρει πολλαπλές εφαρμογές και σε μελλοντικές εργασίες.

4.3 Ανάγκη δημιουργίας Ελληνικού λεξικού

Το λεξικό είναι ένα βασικό εργαλείο για την ανάπτυξη της συγκεκριμένης εργασίας, με χρησιμότητα, όμως, που σαφέστατα ξεφεύγει από τα όρια της και επεκτείνεται σε εργασίες – έργα, που εμπεριέχουν επεξεργασία κειμένου ή/και ομιλίας, και όχι μόνο.

Γίνεται σαφές, πως για την σωστή λειτουργίας της εφαρμογής θα πρέπει να γίνει χρήση κάποιου λεξικού, που θα πρέπει να περιέχει όσο το δυνατόν μεγαλύτερο πλήθος λέξεων της Ελληνικής γλώσσας σε όλες τους τις πιθανές πτώσεις, κλίσεις, χρόνους κτλ.

Έχοντας πραγματοποιήσει μια έρευνα για υπάρχοντα λεξικά της Ελληνικής γλώσσας, που θα μπορούσαν να καλύψουν τις ανάγκες της εργασίας μας, καταλήξαμε στο συμπέρασμα, ότι τα σχετικά λεξικά δεν διαθέτουν την επιθυμητή μορφή για τις ανάγκες της εργασίας μας.

Οι δύο κύριοι λόγοι, για τους οποίους απορρίψαμε την χρήση κάποιο υπάρχοντος λεξικού είναι:

1. Τα OnLine λεξικά στην πλειοψηφία τους περιέχουν τις λέξεις μόνο στην αρχική τους μορφή και προορίζονται για ερμηνευτική και γραμματική χρήση των λέξεων.
2. Η μορφή, στην οποία προσφέρονται δεν μπορεί να χρησιμοποιηθεί από αυτοματοποιημένα συστήματα.

Ένα λεξικό για να καλύψει αποτελεσματικά τις ανάγκες της εργασία μας θα πρέπει:

- Να περιέχει τις λέξεις σε όλες τους τις κλίσεις. Η ανάγκη αυτή προκύπτει από το ότι μια λέξη, που θα βρεθεί σε ένα κείμενο δεν θα είναι πάντα στην αρχική της μορφή, αλλά θα βρίσκεται σε κάποια κλίση, πτώση, χρόνο κτλ. Άρα, για να μπορεί να γίνει η αντιστοίχιση της λέξης του κειμένου με την λέξη στο λεξικού, θα πρέπει το λεξικό να περιέχει την λέξη.
- Το λεξικό για να καλύψει τις ανάγκες της εργασίας μας θα έπρεπε να έχει μια συγκεκριμένη μορφή. Όπου, θα μας παρέχει εύκολη και γρήγορη πρόσβαση στις λέξεις, στην αρχική μορφή των λέξεων και σε πληροφορίες, που αφορούν το είδος, την πτώση, τον χρόνο κτλ. Επίσης, θα πρέπει να παρέχει και πληροφορίες, οι οποίες σχετίζονται με την εφαρμογή μας., όπως διανυσματική αναπαράσταση των λέξεων, κάποιο είδος κωδικοποίηση των λέξεων κτλ.
- Θα πρέπει, επίσης, να είναι σε ηλεκτρονική μορφή, η οποία θα μας παρέχει όσο το δυνατόν πιο εύκολη και γρήγορη πρόσβαση στις πληροφορίες των λέξεων.
- Αν σκεφτούμε το μεγάλο πλήθος λέξεων, που υπάρχουν στα κείμενα και το μεγάλο πλήθος λέξεων στο λεξικό καταλαβαίνουμε, ότι ο χρόνος ανταπόκρισης του συστήματος είναι ένας άλλος, επίσης, πολύ σημαντικός παράγοντας.

Ως εκ τούτου, και λαμβάνοντας σοβαρά υπόψη τον καθοριστικό ρόλο του λεξικού στη σωστή λειτουργία του όλου συστήματός και τις δυνατότητες, που θα είχε να χρησιμοποιηθεί και σε άλλες μελλοντικές εργασίες - έργα, πήραμε την απόφαση για τη δημιουργία ενός δικού μας ελληνικού λεξικού.

4.4 Συλλογή ελληνικών κειμένων

Ένας, επίσης, σημαντικός παράγοντας για την ολοκλήρωση της εργασίας είναι η συλλογή ικανοποιητικού πλήθους κατάλληλων κειμένων για την δημιουργία του dataset, τα οποία και θα χρησιμοποιηθούν για την εκπαίδευση και τη διεξαγωγή των πειραμάτων.

Η αναφορά σε «κατάλληλα κείμενα» έχει να κάνει με την επιλογή κειμένων, που να καλύπτουν συγκεκριμένες ανάγκες της εργασίας μας,

Όπως:

- Τα κείμενα θα πρέπει να περιέχουν κάποιο συναίσθημα και να μην είναι ουδέτερα (όπως δημοσιογραφικά κείμενα, άρθρα wiki, λογοτεχνικά κείμενα κτλ.).
- Να είναι μικρές προτάσεις, οι οποίες να μπορούν να ταξινομηθούν και όχι κείμενα με πολλά και διαφορετικά συναισθήματα.
- Ένας άλλος παράγοντας είναι τα κείμενα να αναφέρονται σε κάποιο συγκεκριμένο τομέα, ώστε να χρησιμοποιούν όσο το δυνατόν πιο κοινό λεξιλόγιο.
- Να είναι όσο το δυνατόν σωστότερες γραμματικά και ορθογραφικά, με σωστή σύνταξη και με σωστή χρήση σημείων στίξης.

Με βάση τις παραπάνω ανάγκες για την δημιουργία της συλλογής κειμένων αποφασίσαμε να συλλέξουμε κείμενα από σχόλια χρηστών σε γνωστή ιστοσελίδα σύγκρισης προϊόντων τεχνολογίας γιατί:

- Στην μεγαλύτερη πλειοψηφία τους τα σχόλια περιέχουν κάποιο συναίσθημα (η γνώμη του χρήστη για το προϊόν.)
- Σε μεγάλο βαθμό τα σχόλια των χρηστών είναι μικρά σε έκταση και αναφέρονται σε ένα συγκεκριμένο θέμα.
- Κείμενα που αναφέρονται σε προϊόντα τεχνολογίας είναι μια συλλογή δεδομένων, που θα μπορούσε να βρει χρήση και σε μελλοντικές εργασίες.
- Τα κείμενα προέρχονται από καθημερινούς χρήστες, άρα είναι κοντά στο σκοπό ενός συστήματος ανάλυσης συναισθήματος, που θα εφαρμοζόταν σε κείμενα, που παράγονται από τους χρήστες του διαδικτύου.

Τα πρόβλημα, που εντοπίσαμε στην συλλογή που αναπτύξαμε είναι πως τα κείμενα, που προέρχονται από χρήστες, ο τρόπος γραφής τους εμφανίζει πολλές ιδιαιτερότητες (greeklish, ορθογραφικά λάθη, συντομεύσεις, αγγλικές λέξεις κτλ).

4.5 *Διόρθωση ορθογραφίας*

Το κυριότερο πρόβλημα που εντοπίσαμε κατά την ανάπτυξη της εργασίας είναι ότι για να εφαρμοστεί αποτελεσματικά μια μέθοδος ανάλυσης συναισθήματος σε κείμενα, που προέρχονται από σχόλια χρηστών θα πρέπει να ελεγχθεί η σωστή μορφή των κειμένων όπως η ορθογραφία τους.

Το πρόβλημα που προκύπτει είναι πως τα σχόλια δύο χρηστών που περιγράφουν με τον ίδιο τρόπο την άποψη τους για κάποιο προϊόν και χρησιμοποιούν ίδιες λέξεις, αν οι λέξεις του ενός χρήστη είναι γραμμένες διαφορετικά (περιέχουν ορθογραφικό λάθος, κάνουν χρήση ελληνοαγγλικών (greeklish), είναι με συντομογραφία κτλ.) το μοντέλο δεν θα μπορέσει να τις αναγνωρίζει ως ίδιες γιατί έχουν γραφτεί διαφορετικά με αποτέλεσμα η κωδικοποίηση των λέξεων για το μοντέλο των RAE να είναι διαφορετική.

Ο τρόπος με τον οποίον δουλεύει το μοντέλο των RAE που κάνουμε χρήση στην εργασία μας είναι πως αγνοεί τις λέξεις που εμφανίζονται στην συλλογή λιγότερες φορές από κάποιο όριο το οποίο μπορούμε να θέσουμε. Γίνεται αντιληπτό πως αν σε μία πρόταση υπάρχει λέξη γραμμένη λάθος δεν θα συμπεριληφθεί από τον αλγόριθμό για την εξαγωγή του συναισθήματος.

Από τα παραπάνω προκύπτει η ανάγκη ανάπτυξης εφαρμογής η οποία θα μπορεί αυτόματα να διορθώσει τα κείμενα πριν την είσοδο τους στο μοντέλο.

4.6 *Διαφορετικές μορφές της ίδιας λέξης*

Ένα άλλο πρόβλημα το οποίο έχει να κάνει με την εφαρμογή του μοντέλου στην Ελληνική γλώσσα είναι ότι οι λέξεις χρησιμοποιούνται με διαφορετικές μορφές μέσα στα κείμενα όπως «οδηγώ», «οδηγεί» και «οδήγηση» Επιπλέον, υπάρχουν οικογένειες σχετικών μεταξύ τους παράγωγων λέξεων, με παρόμοιες σημασίες, όπως «δημοκρατία», «δημοκρατικός», και «εκδημοκρατισμός».

Στην περίπτωση της εργασίας μας δύο προτάσεις που αναφέρονται σε δύο διαφορετικά προϊόντα μπορεί να τα χαρακτηρίζουν με το ίδιο επίθετο αλλά σε διαφορετική πτώση, αριθμό ή γένος. Για παράδειγμά οι προτάσεις «Καταπληκτικό αυτοκίνητο» και «καταπληκτική οθόνη».

Όπως περιγράψαμε και πιο πάνω το μοντέλο των RAE θα έβλεπε αυτές τις δύο προτάσεις να περιέχουν διαφορετικές λέξεις. Σαν αποτέλεσμα θα ήταν οι λέξεις να κωδικοποιούνται διαφορετικά από το μοντέλο των RAE, να κάνουμε χρήση μεγαλύτερου λεξικού για την εκπαίδευση του μοντέλου και πολλές λέξεις να αγνοούνταν λόγω του περιορισμού των εμφανίσεων τους στα κείμενα.

Η λημματοποίηση είναι συνήθως ο ορθότερος τρόπος χειρισμού του παραπάνω προβλήματος. Με την διαδικασία τις λημματοποίησης αυτό που κάνουμε είναι να αντικαθιστούμε τις λέξεις των κειμένων οι οποίες βρίσκονται σε κάποια κλήση – πτώση – αριθμό με την λέξη στην αρχική της μορφή. Για παράδειγμά δύο φράσεις που αναφέραμε και ποιο πανό «Καταπληκτικό αυτοκίνητο» και «καταπληκτική οθόνη» θα αλλάζανε σε «Καταπληκτικό αυτοκίνητο» και «καταπληκτικό οθόνη» και η λέξη Καταπληκτικό θα εμφανιζόταν και στις δύο προτάσεις ίδια.

5

Υλοποίηση εφαρμογής

5.1 Εφαρμογή μοντέλου Επαναληπτικών αυτοσυσχετιστών

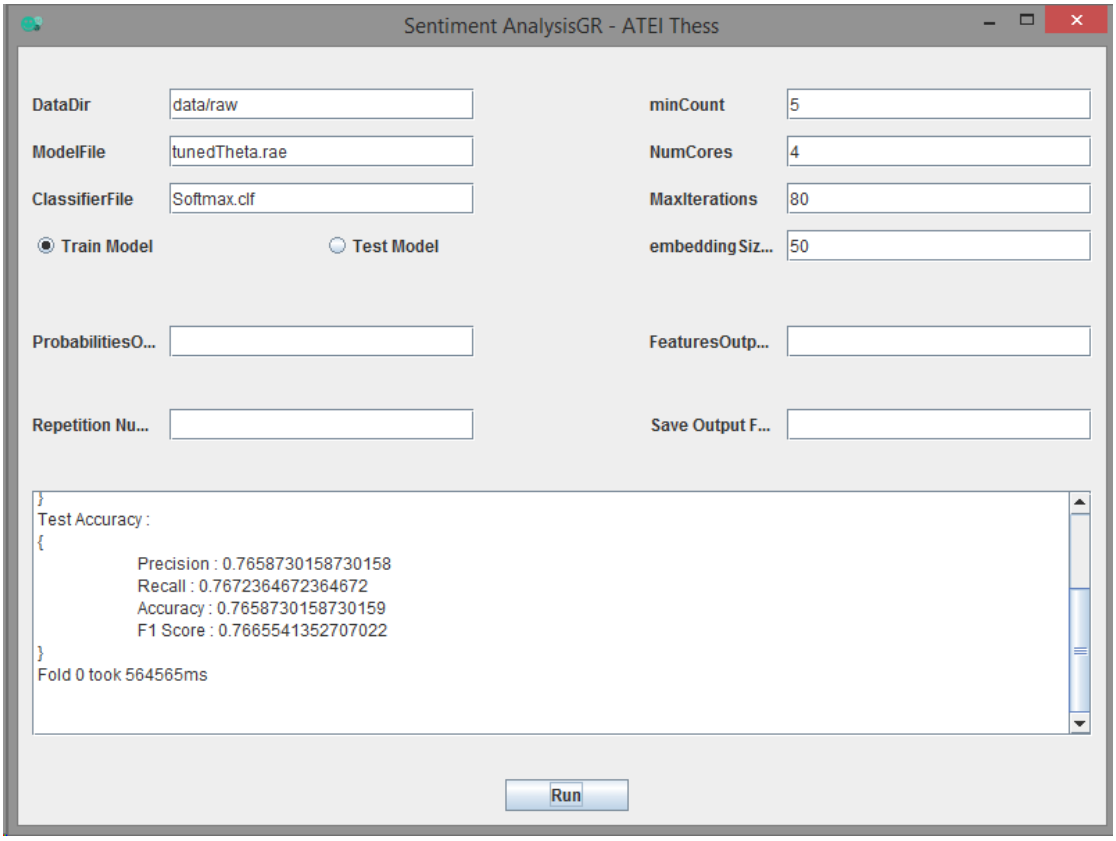
Για να εφαρμόσουμε το μοντέλο των RAE στην εργασία μας χρησιμοποιήσαμε κώδικα ο οποίος προέρχεται από τον σύνδεσμο https://github.com/sancha/jrae.

Η εφαρμογή που αναπτύξαμε «SentimentAnalysisGR» παρέχει γραφικό περιβάλλον μέσω του οποίου δίνεται η δυνατότητα χρήση των RAE.

Η εφαρμογή μέσο της φόρμα που παρουσιάζεται στην εικόνα 10 μας δίνει τις παρακάτω δυνατότητες:

- Να εκπαιδεύσουμε τους RAE και να αποθηκεύσουμε το μοντέλο εκπαίδευσης τους.
- Να κάνουμε έλεγχο σε προτάσεις κείμενων και οι RAE να τις χαρακτηρίσουν ως προς το συναίσθημα (θα πρέπει να υπάρχουν αποθηκευμένα μοντέλα εκπαίδευσης).
- Μπορούμε να διεξάγουμε πειράματα με τους RAE και να αποθηκεύσουμε τα αποτελέσματα.

Στα πεδία της φόρμα δίνουμε τις παραμέτρους που θα χρησιμοποιήσουν οι RAE κατά την εκτέλεση τους. Τα ονόματα που χρησιμοποιούμε για τα πεδία έχουν μείνει ίδια με τα ονόματα των παραμέτρων των RAE. Όταν φέρουμε τον δείκτη του ποντικιού πάνω στα πεδία εμφανίζονται πληροφορίες για την χρήση τους.



Εικόνα 10 Γραφικό περιβάλλον εφαρμογής Sentiment Analysis GR

Στο πεδίο **DataDir** δίνουμε την διαδρομή του φακέλου από την οποία θέλουμε οι RAE να διαβάσουν τα δεδομένα, είτε για εκπαίδευση είτε για τεστ. Τα αρχεία θα πρέπει να είναι οργανωμένα, έτσι ώστε όλες οι προτάσεις που ανήκουν σε μία κατηγορία να βρίσκονται στο ίδιο αρχείο, με όνομα αρχείου το όνομα της κατηγορίας. Κάθε γραμμή κειμένου πρέπει να είναι ξεχωριστή πρόταση. Τα δεδομένα για το test μπορούν να βρίσκονται και σε ένα αρχείο αν δεν είναι γνώστη η κατηγορία που ανήκουν. Αν γνωρίζουμε την κατηγορία τότε χρησιμοποιούμε το ίδιο όνομα με την κατηγορία με το πρόθεμα «test_». Για παράδειγμα οι προτάσεις εκπαίδευσης για τα θετικά σχόλια είναι στο αρχείο με όνομα pos.txt και οι προτάσεις για τον έλεγχο στο αρχείο με όνομα «test_pos.txt» Αν οι κατηγορίες ελέγχου είναι γνωστές τότε στο τέλος της εκτέλεσης θα εμφανιστούν οι παράμετροι αξιολόγησης.

- Το πεδίο **minCount** είναι το πλήθος που θα πρέπει να εμφανίζεται μια λέξη στην συλλογή για να προστεθεί στο λεξικό. Αν το αφήσουμε κενό θα πάρει την προεπιλεγμένη τιμή που είναι το 5.
- Το πεδίο **ModelFile** είναι προαιρετικό και δείχνει σε ποιο αρχείο θα αποθηκευτούν τα δεδομένα εκπαίδευσης αν εκτελούμε την εφαρμογή σε TrainModel. Αν είναι TestModel δείχνει το μοντέλο που θα χρησιμοποιήσει για να διεξάγει τα tests.

- Το πεδίο **NumCores** είναι ο αριθμός των παράλληλών νημάτων που θα χρησιμοποιηθούν για την επεξεργασία. Προεπιλεγμένη τιμή είναι το 4. Συνιστάται ο αριθμός των νημάτων να είναι ίσος με τον αριθμό πυρήνων του επεξεργαστή.
- Το πεδίο **ClassifierFile** είναι προαιρετικό και δείχνει σε ποιο αρχείο θα αποθηκευτεί ο ταξινομητής αν εκτελούμε TrainModel. Αν είμαστε σε TestModel δείχνει τον ταξινομητή που θα χρησιμοποιηθεί για να διεξάγει τα tests.
- Το πεδίο **MaxIterations** είναι ο αριθμός των επαναλήψεων για την εκπαίδευση των RAE. Προεπιλεγμένη τιμή είναι το 80.
- Με τα radiobuttons **TrainModel** και **TestModel** επιλέγουμε αν θα κάνουμε εκπαίδευση των RAE ή θα χρησιμοποιήσουμε υπάρχον ταξηνομητη για test.
- Το πεδίο **embeddingSize** είναι το μέγεθος του διανύσματος που χρησιμοποιείται για την αναπαράσταση των λέξεων στο λεξικό. Προεπιλεγμένη τιμή είναι το 50.
- Τα επόμενα δύο πεδία χρησιμοποιούνται μόνο στην περίπτωση που εκτελούμε τους RAE σε TestModel.
- Το πεδίο **ProbabilitiesOutputFile** είναι η παράμετρος για το αρχείο στο οποίο θα αποθηκευτούν οι ταξινομήσεις του μοντέλου. Χρησιμοποιείται μόνο σε TestModel.
- Το πεδίο **FeaturesOutputFile** είναι το αρχείο στο οποίο εξάγονται τα χαρακτηριστικά που χρησιμοποίησε το μοντέλο για το test. Το αρχείο δεν θα πρέπει να είναι .txt ή θα πρέπει να είναι σε άλλο φάκελο από τον –DataDir και αυτή η παράμετρος χρησιμοποιείται μόνο στο testing.
- Τα επόμενα δύο πεδία μπορούμε να τα χρησιμοποιήσουμε όταν εκτελούμε πειράματα με τους RAE και θέλουμε αποθηκεύσουμε τα αποτελέσματα των πειραμάτων σε αρχείο.
- Το πεδίο **Repetition Number** είναι ο αριθμό των επαναλήψεων που θέλουμε να εκτελεστεί το πείραμα με τις ίδιες παραμέτρους.
- Το πεδίο **Save Output File** είναι το αρχείο στο οποίο θέλουμε να αποθηκεύσουμε τα αποτελέσματα των πειραμάτων μας.
- Η περιοχή output (μεγάλη λευκή περιοχή) είναι για να εμφανίζονται τα αποτελέσματα κατά την εκτέλεση των RAE.

5.2 Υλοποίηση Λεξικού

Για την ανάπτυξη του λεξικού προχωρήσαμε σε συλλογή και καταγραφή όλων των ελληνικών λέξεων, στην αρχική τους μορφή, και στη συνέχεια επιχειρήσαμε να εξάγουμε τις επιπλέον μορφές τους (π.χ. πτώσεις, αριθμοί για ουσιαστικά, πρόσωπα, χρόνους για ρήματα, κοκ.), εφόσον υπήρχαν, με ημι-αυτόματο τρόπο. Για το σκοπό αυτό, στηριχτήκαμε στα

κλιτικά παραδείγματα που αντιστοιχούν σε κάθε βασικό μέρος του λόγου (ουσιαστικά, επίθετα, ρήματα, μετοχές)

Τα κλιτικά παραδείγματα ομαδοποιούν τις λέξεις με βάση τις διαφοροποιήσεις στην κλίση τους, οι οποίες καταγράφονται τόσο ως διαφορές στις καταλήξεις τους, όσο και ως μετατοπίσεις του τόνου εντός ή/και εκτός του θέματος της αντίστοιχης λέξης.

Το αποτέλεσμα ήταν η δημιουργία γεννητριών λέξεων, πλήθους ίσο με τα βασικά μέρη του λόγου. Κάθε γεννήτρια λέξεων υλοποιήθηκε μέσα από τη συγγραφή κώδικα στη γλώσσα προγραμματισμού Java.

Ο συγκεκριμένος κώδικας, λαμβάνοντας σαν είσοδο μία λέξη στην αρχική της μορφή μαζί με το γνωστό κλιτικό της παράδειγμα, πρώτα απομονώνει το θέμα από την κατάληξή της, και στη συνέχεια συνθέτει το σύνολο των εναπομεινασών μορφών της λέξης, συνενώνοντας το θέμα με τις υπόλοιπες καταλήξεις που αντιστοιχούν στο κλιτικό παράδειγμα.

Παράδειγμα

Λέξη: κανόνας

Θέμα λέξης: κανόν-

Κατάληξη λέξης: -ας

Κλιτικό παράδειγμα λέξης: Ο2

Ανάλυση κλιτικού παραδείγματος:

Ενικός				**Πληθυντικός**			
Ονομαστική	**Γενική**	**Αιτιατική**	**Κλητική**	**Ονομαστική**	**Γενική**	**Αιτιατική**	**Κλητική**
-ας	-α	-α	-α	-ες	-ων	-ες	-ες

Πλήρης ανάπτυξη της λέξης με βάση το κλιτικό παράδειγμα:

Ενικός				**Πληθυντικός**			
Ονομαστική	**Γενική**	**Αιτιατική**	**Κλητική**	**Ονομαστική**	**Γενική**	**Αιτιατική**	**Κλητική**
κανόν-ας	κανόν-α	κανόν-α	κανόν-α	κανόν-ες	κανόν-ων	κανόν-ες	κανόν-ες

Στη συνέχεια καταγράφονται αναλυτικά οι παράμετροι που λάβαμε υπόψη για την παραγωγή του συνόλου των μορφών των λέξεων καθώς και τα σχήματα που προέκυψαν για την αποτελεσματικότερη αποθήκευσή τους. Η καταγραφή γίνεται για κάθε βασικό μέρος του λόγου, ξεκινώντας από τα κλιτά, δηλαδή αυτά που κλίνονται (ουσιαστικά, επίθετα, μετοχές, ρήματα) και φτάνοντας στα άκλιτα (προθέσεις, επιρρήματα, σύνδεσμοι, επιφωνήματα).

5.2.1 Ουσιαστικά

Για την κλίση των ουσιαστικών έπρεπε να λάβουμε υπόψη μας τους δύο αριθμούς, αλλά και σύνολο των πτώσεων, δηλαδή των διαφορετικών μορφών που παίρνει ένα ουσιαστικό όταν

κλίνεται. Αναλυτικά, οι δύο τρόποι διάκρισης των ουσιαστικών μαζί με τις πιθανές τιμές τους, είναι οι ακόλουθοι:

Αριθμός: Ενικός/Πληθυντικός

Πτώσεις: Ονομαστική/Γενική/Αιτιατική/Κλητική

Οι διαφορετικές μορφές ουσιαστικών που προκύπτουν με βάση το συνδυασμό των παραπάνω χαρακτηριστικών, συγκεντρώνονται στον πίνακα 1

Ενικός				Πληθυντικός			
Ονομαστική	Γενική	Αιτιατική	Κλητική	Ονομαστική	Γενική	Αιτιατική	Κλητική

Πίνακας 1 Διαφορετικές μορφές ουσιαστικών

Το ίδιο ακριβώς σχήμα χρησιμοποιήθηκε για την καταγραφή των υποκοριστικών (π.χ. ομαδούλα) και μεγεθυντικών (π.χ. ομαδάρα) των ουσιαστικών, όπου αυτά υπήρχαν.

Το πλήθος των κλιτικών παραδειγμάτων που καταγράφηκαν αναλυτικά και χρησιμοποιήθηκαν σε συνδυασμό με το παραπάνω σχήμα, για τη συγγραφή της γεννήτριας λέξεων ουσιαστικών είναι 69.

5.2.2 Επίθετα

Για την κλίση των επιθέτων έπρεπε, καταρχήν, να λάβουμε υπόψη μας τους δύο αριθμούς και τις πτώσεις, όπως στα ουσιαστικά. Εκτός αυτών, τα επίθετα χαρακτηρίζονται από διαφορετικά γένη αλλά και βαθμούς, που έπρεπε επίσης να περιληφθούν. Να διευκρινίσουμε ότι με τον όρο «βαθμοί» ενός επιθέτου αναφερόμαστε στις διαφορετικές μορφές που λαμβάνει ένα επίθετο, προκειμένου να δηλωθούν διάφορα είδη και βαθμοί σύγκρισης.

Αναλυτικά οι τρόποι διάκρισης των επιθέτων, μαζί με τις επιμέρους τιμές τους, είναι οι ακόλουθοι:

Αριθμός: Ενικός/Πληθυντικός

Πτώσεις: Ονομαστική/Γενική/Αιτιατική/Κλητική

Γένη: Αρσενικό/Θηλυκό/Ουδέτερο

Βαθμός: Θετικός/Συγκριτικός/Υπερθετικός

Οι διαφορετικές μορφές επιθέτων, οι οποίες προκύπτουν από τον συνδυασμό των τιμών των παραπάνω χαρακτηριστικών, συγκεντρώνονται στους Πίνακες 2, 3, 4.

Α. Θετικός Βαθμός

Ενικός				Πληθυντικός			
Ονομαστική	Γενική	Αιτιατική	Κλητική	Ονομαστική	Γενική	Αιτιατική	Κλητική
Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό
Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό
Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο

Πίνακας 2 Επίθετα θετικός βαθμός

Β. Συγκριτικός Βαθμός

Ενικός				Πληθυντικός			
Ονομαστική	Γενική	Αιτιατική	Κλητική	Ονομαστική	Γενική	Αιτιατική	Κλητική
Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό
Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό
Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο

Πίνακας 3 Επίθετα συγκριτικός βαθμός

Γ. Υπερθετικός Βαθμός

Ενικός				Πληθυντικός			
Ονομαστική	Γενική	Αιτιατική	Κλητική	Ονομαστική	Γενική	Αιτιατική	Κλητική
Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό	Αρσενικό
Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό	Θηλυκό
Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο	Ουδέτερο

Πίνακας 4 Επίθετα υπερθετικός βαθμός

Να σημειωθεί ότι η πλειοψηφία των επιθέτων, και συγκεκριμένα αυτά που δηλώνουν ύλη, καταγωγή ή συγγένεια, τόπο, χρόνο, ή κατάσταση που δεν αλλάζει, δεν σχηματίζουν παραθετικά.

Το ίδιο ακριβώς σχήμα, χωρίς τα παραθετικά, χρησιμοποιήθηκε για την καταγραφή των υποκοριστικών των επιθέτων (π.χ. τρελούτσικος), όπου αυτά υπήρχαν.

Το πλήθος των κλιτικών παραδειγμάτων των επιθέτων που καταγράφηκαν αναλυτικά και χρησιμοποιήθηκαν, σε συνδυασμό με το παραπάνω σχήμα, για τη συγγραφή της αντίστοιχης γεννήτριας λέξεων είναι 27.

5.2.3 Ρήματα

Για την κλίση των ρημάτων έπρεπε καταρχήν να λάβουμε υπόψη μας τους δύο αριθμούς και τα πρόσωπα. Εκτός αυτών, τα ρήματα διαφοροποιούνται από τους χρόνους, την έγκλιση καθώς και τη φωνή τους.

Αναλυτικά, οι τρόποι διάκρισης των ρημάτων καθώς και οι τιμές που παίρνουν, είναι οι ακόλουθοι:

Αριθμός: Ενικός/Πληθυντικός

Πρόσωπα: Πρώτο/Δεύτερο/Τρίτο

Χρόνος: Ενεστώτας/Παρατατικός/Αόριστος/Στιγμιαίος (ή Συνοπτικός) Μέλλοντας

Έγκλιση: Οριστική/Προστακτική/Απαρέμφατο

Φωνή: Ενεργητική/Παθητική

Να σημειωθεί ότι από τους χρόνους που περιλάβαμε στον τελικό πίνακα, λείπουν ορισμένοι, οι οποίοι μπορούν εύκολα να παραχθούν με τη συνένωση υπαρχόντων χρόνων σε συνδυασμό με τους συνδέσμους «να» ή «θα», ή/και κλίσεις του ρήματος «έχω». Οι χρόνοι που λείπουν μαζί με τον τρόπο που μπορούν να συντεθούν, καθώς και ένα παράδειγμα για κάθε περίπτωση, έχουν συγκεντρωθεί στον Πίνακα 5.

Χρόνος	**Πώς δημιουργείται**	**Παράδειγμα**
Εξακολουθητικός Μέλλοντας	«θα»+ενεστώτας	θα λύνω
Παρακείμενος	«έχω»+απαρέμφατο	έχω λύσει
Υπερσυντέλικος	«είχα»+απαρέμφατο	είχα λύσει
Συντελεσμένος Μέλλοντας	«θα έχω»+απαρέμφατο	θα έχω λύσει
Υποτακτική	«να»+χρόνος/απαρέμφατο	να λύνω

Πίνακας 5 Χρόνοι που λείπουν από το λεξικό

Τελικά, όλοι οι συνδυασμοί των χαρακτηριστικών των ρημάτων που είναι αναγκαίοι για να παράξουν το σύνολο των διαφορετικών μορφών τους, συγκεντρώνονται στους Πίνακες 6, 7.

4Α) Ενεργητική και Παθητική > Οριστική

Ενεστώτας						Παρατατικός						Αόριστος						Στιγ. Μέλλοντας					
Ενικός			Πληθ			Ενικός			Πληθ			Ενικός			Πληθ			Ενικός			Πληθ		
Α	Β	Γ	Α	Β	Γ	Α	Β	Γ	Α	Β	Γ	Α	Β	Γ	Α	Β	Γ	Α	Β	Γ	Α	Β	Γ

Πίνακας 6 Συνδυασμοί των χαρακτηριστικών των ρημάτων που είναι αναγκαίοι για να παράξουν το σύνολο των διαφορετικών μορφών τους - Οριστική

4Β) Ενεργητική και Παθητική > Προστακτική & Απαρέμφατο

Προστακτική						Απαρέμφατο	
Ενεστώτας		Αόριστος		Αόριστος Β		Αόριστος	Αόριστος Β
Ενικός	Πληθ	Ενικός	Πληθ	Ενικός	Πληθ		
Β	Β	Β	Β	Β	Β		

Πίνακας 7 Συνδυασμοί των χαρακτηριστικών των ρημάτων που είναι αναγκαίοι για να παράξουν το σύνολο των διαφορετικών μορφών τους - Προστακτική & απαρέμφατο

Σημείωση: Στον παραπάνω πίνακα, τα γράμματα Α, Β και Γ, περιγράφουν το πρώτο, δεύτερο και τρίτο πρόσωπο ενός ρήματος, αντίστοιχα.

5.2.4 Μετοχές

Διακρίνουμε τις μετοχές σε άκλιτες και κλιτές.

Στην πρώτη κατηγορία εντάσσουμε τις μετοχές ενεστώτα, ενεργητικής φωνής, με κατάληξη –«-οντας» (όταν αυτή τονίζεται στην προπαραλήγουσα, π.χ. παίζοντας) και «–ώντας» (όταν αυτή τονίζεται στην παραλήγουσα, π.χ. τραγουδώντας). Αποτελούν τη συχνότερα συναντημένη μορφή μετοχών. Οι συγκεκριμένες μετοχές συγκεντρώθηκαν σε ξεχωριστό πίνακα.

Για την κλίση των μετοχών της δεύτερης κατηγορίας λάβαμε υπόψη μας τις πτώσεις, τους αριθμούς, τα γένη, τους χρόνους, καθώς και τη φωνή τους.

Αναλυτικά, οι τρόποι διάκρισης των κλιτών μετοχών καθώς και οι τιμές που παίρνουν, είναι οι ακόλουθοι:

Πτώσεις: Ονομαστική/Γενική/Αιτιατική/Κλητική

Αριθμός: Ενικός/Πληθυντικός

Γένη: Αρσενικό/Θηλυκό/Ουδέτερο

Χρόνος: Ενεστώτας/Αόριστος/Παρακείμενος

Φωνή: Ενεργητική/Παθητική

Να σημειωθεί ότι οι κλιτές μετοχές ενεργητικής φωνής δεν χρησιμοποιούνται ευρέως στη Νεοελληνική γλώσσα. Για λόγους πληρότητας έχουμε περιλάβει τις μορφές τους σε Ενεστώτα (π.χ. εκπροσωπών) και Αόριστο (π.χ. εκπροσωπήσας). Απεναντίας, οι κλιτές μετοχές παθητικής φωνής χρησιμοποιούνται συχνότερα, και συναντώνται κυρίως στους χρόνους του Ενεστώτα (π.χ. αναδημοσιευόμενος) και του Παρακειμένου (π.χ. αναδημοσιευμένος), αλλά, πιο σπάνια, και του Αορίστου (π.χ. αναδημοσιευθείς).

Οι διαφορετικές μορφές των κλιτών μετοχών, οι οποίες προκύπτουν από τον συνδυασμό των τιμών των παραπάνω χαρακτηριστικών, συγκεντρώνονται στους Πίνακες 8, 9.

Α) Ενεργητική και Παθητική Φωνή

Ενεστώτας								Αόριστος							
Ενικός				Πληθ				Ενικός				Πληθ			
Ον	Γεν	Αιτ	Κλ	Ον	Γεν	Αιτ	Κλ	Ον	Γεν	Αιτ	Κλ	Ον	Γεν	Αιτ	Κλ
Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ
Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ
Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ

Πίνακας 8 Διαφορετικές μορφές των κλιτών μετοχών

Β) Παρακείμενος Παθητική Φωνή

Παρακείμενος							
Ενικός				Πληθ			
Ον	Γεν	Αιτ	Κλ	Ον	Γεν	Αιτ	Κλ
Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ	Αρσ
Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ	Θηλ
Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ	Ουδ

Πίνακας 9 Διαφορετικές μορφές των κλιτών μετοχών

5.2.5 Άκλιτα μέρη του λόγου

Σε αυτή τη μεγάλη κατηγορία τοποθετούμε τις λέξεις, οι οποίες δεν κλίνονται, δηλαδή διατηρούν την ίδια μορφή, ανεξαρτήτως της χρήσης ή της θέσης τους. Συγκεκριμένα, στα άκλιτα μέρη του λόγου περιλαμβάνονται τα επιρρήματα, οι προθέσεις, οι σύνδεσμοι, τα μόρια και τα επιφωνήματα.

- Τα επιρρήματα χρησιμοποιούνται κυρίως για να προσδιορίσουν τα ρήματα και φανερώνουν τόπο (π.χ. εδώ), χρόνο (π.χ. αύριο), τρόπο (π.χ. αλλιώς), ποσό (π.χ. λίγο) κλπ.
- Οι προθέσεις μπαίνουν συνήθως μπροστά από κλιτές λέξεις και φανερώνουν σχέσεις, όπως και τα επιρρήματα (π.χ. πριν, δίχως, εξαιτίας).
- Οι σύνδεσμοι χρησιμοποιούνται για να συνδέσουν προτάσεις ή λέξεις που ανήκουν στην ίδια πρόταση, μεταξύ τους. Διακρίνονται σε πλήθος κατηγοριών, όπως συμπλεκτικοί (π.χ. και), διαχωριστικοί (π.χ. ή), αντιθετικοί (π.χ. όμως), συμπερασματικοί (π.χ. ώστε), κ.α.
- Τα μόρια είναι λέξεις, συνήθως μονοσύλλαβες, που δεν ανήκουν σε άλλο μέρος του λόγου και τις χρησιμοποιούμε με διάφορες σημασίες (π.χ. δεν, θα, να).
- Τα επιφωνήματα είναι λέξεις που φανερώνουν κάποιο έντονο συναίσθημα, και συνήθως συνοδεύονται από θαυμαστικό, ερωτηματικό ή αποσιωπητικά (π.χ. αμάν!, ορίστε;).

5.2.6 Υπόλοιποι πίνακες

Για λόγους πληρότητας σχεδιάσαμε 3 ακόμα πίνακες, τους οποίους αναφέρουμε εδώ:

- Πίνακας αριθμών: περιλαμβάνει όλους τους αριθμούς από το μηδέν ως το εκατό, τις εκατοντάδες ως το χίλια, καθώς το(τα) εκατομμύριο(α). Με τον τρόπο αυτό, το λεξικό θα μπορεί να αναγνωρίζει αριθμούς όπως ο «πέντε εκατομμύρια χίλια εξακόσια πενήνταδυο».
- Πίνακας άρθρων: περιλαμβάνει μια πλήρη καταγραφή των άρθων, είτε είναι οριστικά (ο/η/το), είτε είναι αόριστα (ένας/μία/ένα), μαζί με τις κλίσεις τους.

- Πίνακας συντομογραφιών: περιλαμβάνει χαρακτηριστικές λέξεις, προερχόμενες από την ελληνική γλώσσα, που σε μηνύματα ή και αλλού, για χάριν ταχύτητας, συχνά αναφέρονται συντομευμένες, π.χ. δλδ, κλπ, τπτ ή φωτο.

5.2.7 Αποθήκευση λεξικού

Αφού ολοκληρώσαμε μια, όσο το δυνατό πληρέστερη, καταγραφή των λέξεων της ελληνικής γλώσσας, το επόμενο βήμα ήταν η αποθήκευσή τους με τρόπο που θα είναι αποδοτικός τόσο σε χώρο όσο και σε ταχύτητα χρήσης. Καταλήξαμε στη χρησιμοποίηση δύο βάσεων δεδομένων.

5.2.7.1 MySQL (έκδοση 5.5.40)

Χρησιμοποιήθηκε για την αποθήκευση των ολοκληρωμένων λεξικών της ελληνικής γλώσσας. Για το ελληνικό λεξικό, δημιουργήθηκαν πίνακες για όλα τα μέρη του λόγου (ουσιαστικά, επίθετα, ρήματα, μετοχές, άκλιτα, κλπ.). Οι συγκεκριμένοι πίνακες αποθηκεύουν το σύνολο των δυνατόν μορφών των λέξεων, οι οποίες προκύπτουν με τον τρόπο που αναλύθηκε στις προηγούμενες παραγράφους και καταγράφεται στα αντίστοιχα σχήματα.

Ο πίνακας 10 περιλαμβάνει κάποια ενδιαφέροντα στατιστικά στοιχεία για το σύνολο των πινάκων που χρησιμοποιήθηκαν για την αποθήκευση των λέξεων στη MySQL.

Περιεχόμενα πίνακα	**Πλήθος στηλών**	**Μοναδικές λέξεις**
Ουσιαστικά	7	28.137
Επίθετα	73	11.541
Ρήματα	77	5.569
Μετοχές κλιτές	121	4.707
Μετοχές άκλιτες	2	5.337
Άκλιτες λέξεις	4	4.091
Αριθμοί	2	109
Άρθρα	2	26
Συντομογραφίες	2	9

Πίνακας 10 Στατιστικά στοιχεία για το σύνολο των πινάκων που χρησιμοποιήθηκαν για την αποθήκευση των λέξεων στη MySQL

5.2.7.2 MongoDB (έκδοση 2.6.5, 64-bit)

Η MongoDB, το πιο γνωστό εργαλείο για κατασκευή NoSQL βάσεων δεδομένων επιλέχθηκε για την ταχύτητα που προσφέρει. Επίσης, επειδή στην παρούσα εργασία δεν χρειάζονται σχεσιακές βάσεις δεδομένων, καθώς δεν εκτελούνται ερωτήματα (queries) σε πάνω από έναν

πίνακα. Για κάθε λέξη, ανεξαρτήτως των μορφών που παίρνει, αποθηκεύονται τα εξής πεδία πλαίσιο 1:

```
{
"_id" : ObjectId("554dc383c9118c0eb39908bc"),
 "wordID" : "AO1",
 "ID" : [
 -1.334008824883531,
 -1.854454159887333,
        ……,
        ……,
        ……,
 -0.03915691369892169,
 -0.8225123058087737
 ],
 "startWord" : "αβαειο",
 "word" : "αβαείο",
 "noToneWord" : "αβαειο",
 "pos" : 1
}
```

Πλαίσιο 1 – Πεδία λεξικού MongoDB

Ειδικότερα:

- **_id:** είναι ένα μοναδικό id που ορίζεται αυτόματα από τη MongoDB.
- **wordID:** είναι το ID που έχουμε δώσει εμείς σε κάθε λέξη κατά την εισαγωγή της στη βάση δεδομένων MySQL. Ο πρώτος χαρακτήρας στο παράδειγμά μας το 'Α' είναι το γράμμα με το οποίο ξεκινάει η κάθε λέξη, Ο δεύτερος χαρακτήρας 'Ο' στο παράδειγμά μας είναι για το μέρος του λόγου δηλαδή Ο για ουσιαστικά, Ε για επίθετα, Ρ για ρήματα κτλ. Ο Αριθμός μετά είναι ένας αύξον αριθμός για την μοναδικότητα κάθε λέξης.
- **ID:** είναι μια διανυσματική αναπαράσταση 100 αριθμών με γκαουσιανή κατανομή, μοναδική για κάθε λέξης, δηλαδή μία λέξη θα έχει το ίδιο διάνυσμα αναπαράσταση σε όλες τις πτώσεις αν είναι ουσιαστικό, σε όλα τα πρόσωπα, χρόνους, εγκλίσεις αν είναι ρήμα κτλ. Η προσθήκη τις διανυσματική αναπαράστασης έγινε για μελλοντικές υλοποιήσεις.
- **startWord:** είναι η λέξη στην αρχική της μορφή Για τα ουσιαστικά ονομαστική ενικού, για τα επίθετα ονομαστική ενικού αρσενικό, για τα επίθετα ενεστώτας πρώτο πρόσωπο.
- **word:** είναι η λέξη στην κανονική της μορφή όπως θα υπήρχε σε κάποιο κείμενο.
- **noToneWord:** είναι η λέξη χωρίς τόνους. Για να μπορεί να γίνεται αναζήτηση λέξεων από κείμενα χρηστών.
- **pos:** συμβολίζει τη θέση της λέξης στη συνολική εγγραφή που την περιγράφει μέσα στη βάση δεδομένων MySQL, δίνοντάς μας πληροφορίες για τη μορφή της (πτώση,

γένος, χρόνος, φωνή κ.λπ.) με βάση το αντίστοιχο σχήμα. Π.χ. το ουσιαστικό «ανθρώπου» θα έχει pos=2 που υποδηλώνει ότι αντιστοιχεί σε γενική, ενικού αριθμού, ενώ το ρήμα «παίζαμε» θα έχει pos=10 που υποδηλώνει ότι αντιστοιχεί στο α' πρόσωπο, του πληθυντικού, στον παρατατικό χρόνο, της οριστικής στην ενεργητική φωνή.

Το συνολικό πλήθος λέξεων που περιέχονταν αρχικά στην βάση ήταν περισσότερες από 1.500.000. Για την χρήση όμως στην εργασίας μας μπορούμε να σβήσαμε λέξεις οι οποίες υπήρχαν διπλές και αναφέρονται στην ίδια αρχική λέξη. Για παράδειγμα η λέξη «*Σπίτι*» θα εμφανιζόταν στις μορφές του παρακάτω πίνακα.

Ενικός				**Πληθυντικός**			
Ονομαστική	**Γενική**	**Αιτιατική**	**Κλητική**	**Ονομαστική**	**Γενική**	**Αιτιατική**	**Κλητική**
σπίτι	σπιτιού	σπίτι	σπίτι	σπίτια	σπιτιών	σπίτια	σπίτια

γίνεται φανερό πώς δεν υπάρχει ανάγκη για τις εγγραφές στην αιτιατική και κλιτική ενικού όπως και αιτιατική και κλιτική πληθυντικού. Μετά την διαγραφή των παραπάνω λέξεων η βάση μας περιέχει 811.160 λέξεις.

Ο Πίνακας 11 περιλαμβάνει στοιχεία για την σύγκριση των λεξικών στις βάσεις δεδομένων MySQL και MongoDB.

	MongoDB	**MySQL**
Μέγεθος	5,95 GB	54,8 MB
Χρόνος για αναζήτηση	0,115 sec σε 811.160 λέξεις	0,035 sec σε 28.137
Πίνακες – Συλλογές	1 (Συλλογή)	9 (Πίνακες)
	Περιέχει διαφορετικές μορφές για τις λέξεις και κωδικοποιήσεις	Περιέχει τις λέξεις με τον τρόπο που κλείνονται

Πίνακας 11 Σύγκριση των λεξικών στις βάσεις δεδομένων MySQL και MongoDB.

5.3 Συλλογή κειμένων

Για την δημιουργία της συλλογής των κειμένων που τα χρησιμοποιήσαμε για την εκπαίδευση και την αξιολόγηση του μοντέλου μας συλλέξαμε σχόλια χρηστών για προϊόντα τεχνολογίας από γνωστή ιστοσελίδα σύγκρισής προϊόντων.

Σε πρώτη φάση αναπτύξαμε αλγόριθμο σε PHP με στόχο να συλλέξουμε τα σχόλια από την ιστοσελίδα και να τα αποθηκεύσουμε σε αρχεία με μορφή που θα είναι εύκολη η περεταίρω επεξεργασία τους. Αποκτήσαμε περισσότερα από 3000 σχόλια για κινητά τηλέφωνα σε μορφή "xml" με ετικέτες (tags) που περιέχουν πέρα από το σχόλιο την ιστοσελίδα του

προϊόντος και την βαθμολογία του. Η μορφή στην οποία αποθηκεύσαμε τα σχόλια φαίνεται στο πλαίσιο 2.

```
<File>
<Product>
<Url>http://www.skroutz.gr/s/2119391/Samsung-E1200.html</Url>
<Review>
<Id>sku_review_17047</Id>
<Rating>5</Rating>
<Comment>
ενα απλο κινητο με βασικες λειτουργιες για καθημερινη χρηση ..
</Comment>
</Review>
<Review>
<Id>sku_review_16879</Id>
<Rating>4</Rating>
<Comment>
Το κινητό αυτό το πήρα για το στρατό και κάνει απλά τη δουλεία του!
Μήνυματα-τηλέφωνα-ξυπνητήρι-ατελείωτη μπαταρία! Επίσης κορυφαία είναι και
η τιμή του!
</Comment>
</Review>
.
.
.
</Product>
</File>
```

Πλαίσιο 2 - Μορφή αρχείου σχολίων σε xml

Μετά από ανάλυση των σχολίων και της μορφής τους αποφασίσαμε πως για την εκπαίδευση του μοντέλου μας θα έπρεπε να τα χωρίσουμε σε προτάσεις και να μην κρατήσουμε τα σχόλια στην αρχική τους μορφή. Ο λόγος ήταν πως πολλά από τα σχόλια ήταν αρκετά μεγάλα και αποτελούνταν από αρκετές προτάσεις. Οι οποίες πολλές φορές είχαν και διαφορετική πολικότητα σε σχέση με το συναίσθημα. (παράδειγμα: σχόλιο με αρνητική βαθμολογία και αρνητικό σχόλιο για το ίδιο το προϊόν, το συγκρίνει με άλλο προϊόν για το οποίο οι προτάσεις μέσα στο σχόλιο είναι θετικές).

Αναπτύξαμε αλγόριθμο σε Java ο οποίος αρχικά διαβάζει τα “xml” αρχεία και εξάγει το κείμενο που βρίσκεται μέσα στα *tag* <*Comment*> και <*Rating*> και τα εγγράφει σε νέο αρχείο “.csv” το οποίο περιέχει το σχόλιο και την βαθμολογία χωρισμένα με κόμμα. Στην συνέχεια χωρίζουμε τα σχόλια σε προτάσεις με βάσει τα σημεία στίξης που περιείχαν (τελείς, κόμματα, θαυμαστικά, κτλ.) και τα εγγράφουμε σε νέα αρχεία όπου το κάθε αρχείο περιέχει από 250 προτάσεις. Συνολικά δημιουργήσαμε 63 αρχεία που στο σύνολο τους περιέχουν περίπου 16000 προτάσεις.

Στην συνέχεια μοιράσαμε κάποια από τα αρχεία σε ανθρώπους για να χαρακτηρίσουν την κάθε πρόταση με βάσει την συναισθηματική της πόλωση θετική, αρνητική ή ουδέτερη.

Καταλήξαμε να έχουμε μια συλλογή από συνολικά 5169 προτάσεις. Αναλυτικά η συλλογή μας περιέχει 2956 θετικές προτάσεις (στόχος = 1), 1426 αρνητικές (στόχος = -1) και 787 ουδέτερες (στόχος = 0).

Η μορφή που έχει η συλλογή μας φαίνεται στο πλαίσιο 3 όπου η πρόταση και ο χαρακτηρισμός της χωρίζονται με ερωτηματικό.

```
Μηνύματα, τηλέφωνα, ξυπνητήρι, ατελείωτη μπαταρία;1
Επίσης κορυφαία είναι και η τιμή του;1
ένα απλό κινητό;1
και λειπει και καπακι πισω;-1
ποτε δεν βγηκε η μπαταρια;0
```

Πλαίσιο 3 - Μορφή συλλογής Σχολίων

Επιλέξαμε να αποθηκεύσουμε την συλλογή μας σε μορφή ".csv" για να μπορούμε να την επεξεργαζόμαστε ευκολότερα με ένα πρόγραμμα υπολογιστικών φύλλων.

5.4 Διόρθωση ορθογραφίας

Το πρόβλημα που αντιμετωπίσαμε κατά την επεξεργασία της συλλογής μας από σχόλια χρηστών ήταν ότι πολλές λέξεις ήταν γραμμένες λάθος στο κείμενο (ορθογραφικά λάθη, έλλειψη τόνων, λάθος χρήση σημείων στίξης κτλ.).

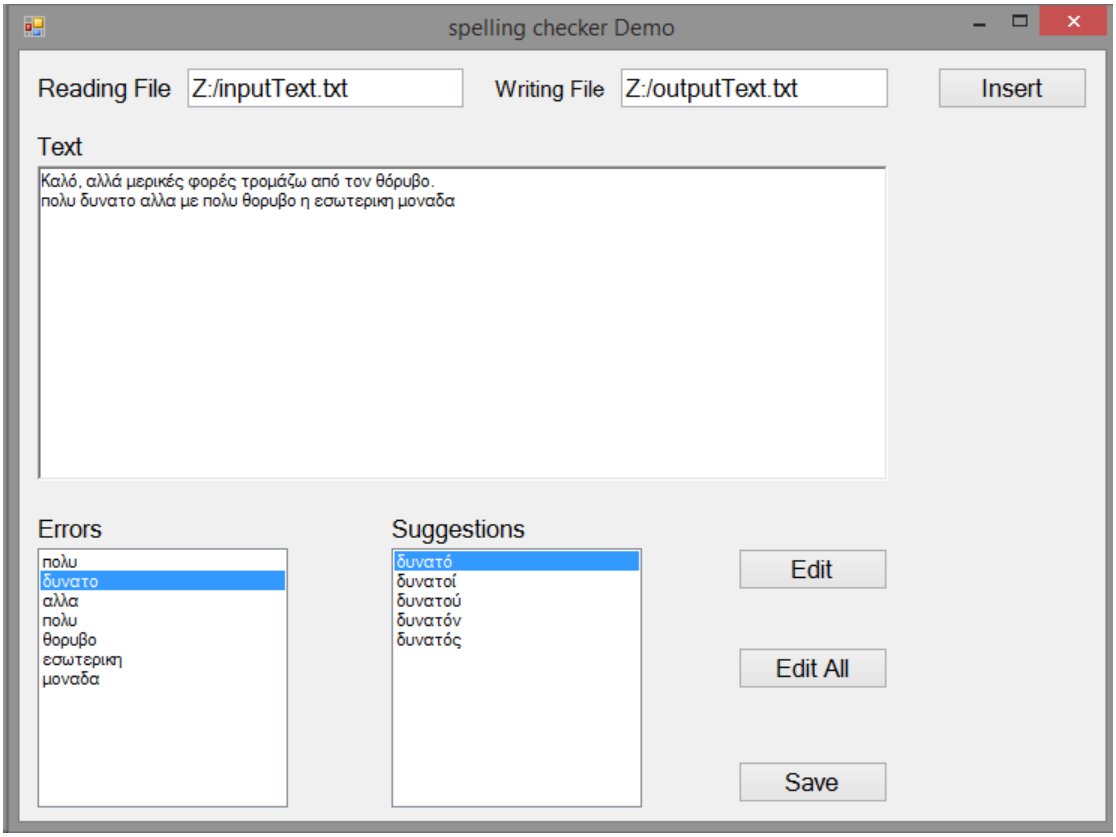
Μία λύση η οποία θα μπορούσε να εφαρμοστεί στο παραπάνω πρόβλημα θα ήταν η διόρθωση των κείμενων με ένα πρόγραμμα επεξεργασίας κειμένων το οποίο θα διαθέτει κάποιο εργαλείο για ορθογραφικό και γραμματικό έλεγχο.

Λόγω όμως του μεγάλου όγκου κειμένων που έχουμε και με την προοπτική ανάπτυξης ενός πλήρους αυτοματοποιημένου συστήματος αυτόματης εκτίμησης της διάθεσης αυτό δεν θα μπορούσε να γίνει με το χέρι. Αποφασίσαμε πως θα έπρεπε να αναπτύξουμε εφαρμογή η οποία θα μπορούσε να εντοπίζει και να διορθώνει αυτόματα τα κείμενα.

Αν και το μεγαλύτερο μέρος των εφαρμογών των οποίων κάνουμε χρήση στην παρούσα εργασία έχει αναπτυχθεί σε γλώσσα Java, την εφαρμογή αυτόματης διόρθωσης την αναπτύξαμε στην γλώσσα C#. Ο λόγος είναι ότι το Framework .net παρέχει βιβλιοθήκες με τις οποίες μας δίνεται πρόσβαση σε εργαλεία άλλων εφαρμογών της Microsoft όπως το MS Office Word για την περίπτωση μας.

Κατασκευάσαμε εφαρμογή η οποία κάνει χρήση του εργαλείου ορθογραφικού και γραμματικού ελέγχου του MS Office Word μέσω βιβλιοθήκης που παρέχεται.

Η εφαρμογή μας διαθέτει γραφικό περιβάλλον (βλ. εικόνα 11) από το οποίο μπορούμε να έχουμε πρόσβαση στις λειτουργίες της.



Εικόνα 11 Γραφικό περιβάλλον εφαρμογής ορθογραφικού ελέγχου

Το **Reding File** είναι η διαδρομή του αρχείου από το οποίο θα διαβάσει το αρχείο που περιέχει το αρχικό κείμενο.

Στο **Writing File** είναι η διαδρομή του αρχείου στο οποίο θα αποθηκευτεί το διορθωμένο κείμενο.

Με το κουμπί **Insert** διαβάζουμε το κείμενο από το αρχείο και το εμφανίζουμε στην περιοχή Text, και στην περιοχή Errors εμφανίζονται τα λάθη που περιέχει.

Στο **Text** εμφανίζεται αρχικά το κείμενο του αρχείου ανάγνωσης και μπορούμε να το επεξεργαστούμε.

Στο **Errors** εμφανίζονται οι λάθος λέξεις που έχουν εντοπιστεί στο κείμενο.

Επιλέγοντας μια λέξη από το errors στο **Suggestions** παρουσιάζονται οι προτάσεις διόρθωσης όπως δίνονται από το εργαλείο ορθογραφικού και γραμματικού ελέγχου του MS Office Word.

Επιλέγοντας μία λέξη από το Suggestions και πατώντας το κουμπί **Edit** η λάθος λέξη στο Text αντικαθίσταται με την επιλεγμένη λέξη.

Με το κουμπί **Edit All** διορθώνονται αυτόματα όλες οι λάθος λέξεις που περιέχει το κείμενο με την πρώτη λέξη από την λίστα Suggestions.

Τέλος για την αποθήκευση του κειμένου θα πρέπει να πατήσουμε το κουμπί **Save** με το οποίο θα αποθηκευτεί το κείμενο της περιοχής Text στο αρχείο της διαδρομής Writing File.

5.5 *Λημματοποίηση(Lemmatization)*

Για να αντιμετωπίσουμε το πρόβλημα των πολλαπλών μορφών μιας λέξεις μέσα στις προτάσεις αυτό που εφαρμόσαμε είναι η διαδικασία της λημματοποίησης.

Λημματοποίηση (Lemmatization) είναι η διαδικασία κατά την οποία ανάγουμε μια λέξη στην βασική της μορφή.

Για παράδειγμα:

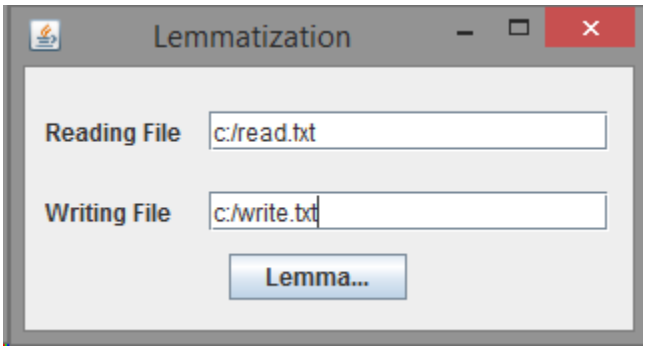
είμαι, είσαι, είναι => είμαι

καράβι, καράβια, καραβιού, καραβιών=> καράβι

Μια τέτοια αντιστοίχιση κειμένου σε μία πρόταση θα μας έδινε το παρακάτω αποτέλεσμα:

τα καράβια στο λιμάνι ξεφορτώνουν τα προϊόντα => το καράβι στο λιμάνι ξεφορτώνει το προϊόν.

Για την διαδικασία της λημματοποίησης αναπτύξαμε εφαρμογή σε java (βλ. εικόνα 12). Όπου δέχεται δύο παραμέτρους την διαδρομή του αρχείου από το οποίο θα διαβάσει τις προτάσεις και την διαδρομή του αρχείου που θα αποθηκεύσει τις λημματοποιημένες προτάσεις. Η εφαρμογή ξεκινά με το κουμπί lemmatize. Όταν ολοκληρωθεί η διαδικασία εμφανίζει σε παράθυρο μήνυμα ότι η διαδικασία ολοκληρώθηκε.



Εικόνα 12 Γραφικό περιβάλλον εφαρμογής λημματοποίησης

Η εφαρμογή διαβάζει τις προτάσεις από το αρχείο και τις χωρίζει στις λέξεις στους. Έπειτα αναζητεί στην βάση δεδομένων MongoDB στο λεξικό την κάθε λέξη (πεδίο word) και ζητάει την λέξη στην αρχική της μορφή (πεδίο startWord). Το λεξικό επιστρέφει την λέξη στην αρχική της μορφή. Στην συνέχεια οι προτάσεις που έχουν αντικατασταθεί οι λέξεις τους γράφονται σε νέο αρχείο το οποίο περιέχει προτάσεις μόνο με την αρχική μορφή των λέξεων.

6

Αξιολόγηση

Για την αξιολόγηση του μοντέλου των επαναληπτικών αυτοσυσχετιστών και των εργαλείων που αναπτύξαμε εκτελέσαμε μια σειρά από πειράματα στα οποία δοκιμάσαμε διαφορετικές παραμέτρους στο μοντέλο των RAE και τους εφαρμόσαμε σε διαφορετικές εκδόσεις της συλλογής σχολίων που δημιουργήσαμε. Για να μετρήσουμε την αποτελεσματικότητα τους καταγράψαμε διάφορες παραμέτρους από τις εξόδους των πειραμάτων μας.

6.1 Παράμετροι αξιολόγησης

Για την αξιολόγηση των πειραμάτων μας μετράμε τις τιμές από τέσσερεις παραμέτρους. Οι παράμετροι αυτοί είναι ακρίβεια (Precision), ανάκληση (Recall), Ορθότητα (Accuracy) και μέτρο F (F1 Score).

Ακρίβεια (Precision) P: είναι το κλάσμα των προτάσεων που ταξινομήθηκαν σωστά στην κατηγορία (θετικές /αρνητικές) προς το σύνολο των προτάσεων που ταξινομήθηκαν στην ίδια κατηγορία.

$$P = \frac{\#correct\ classification}{\#class\ total}$$

Ανάκληση (Recall) R: είναι το κλάσμα των προτάσεων που ταξινομήθηκαν σωστά σε μία κατηγορία προς το συνολικό πλήθος των προτάσεων που ανήκουν πραγματικά σε αυτή την κατηγορία.

$$R = \frac{\#correct\ classification}{\#total\ correct}$$

Μπορούμε να κατανοήσουμε καλύτερα τις παραπάνω έννοιες εξετάζοντας τον παρακάτω πίνακα ενδεχομένων:

	Θετικές προτάσεις	**Αρνητικές προτάσεις**
Θετικά ταξινομημένα	αληθή θετικά (true positives, tp)	ψευδή θετικά (false positives, fp)
Αρνητικά ταξινομημένα	ψευδή αρνητικά (false negatives, fn)	αληθή αρνητικά (true negatives, tn)

Τότε:

$$P = \frac{tp}{(tp + fp)}$$

$$R = \frac{tp}{(tp + fn)}$$

Ορθότητα (Accuracy) A: Είναι το κλάσμα των προτάσεων που ταξινομηθήκαν στην σωστή κλάση προς το σύνολο των προτάσεων της συλλογής. Με βάση τον παραπάνω πίνακα ενδεχομένων θα έχουμε:

$$A = \frac{(tp + tn)}{(tp + fp + fn + tn)}$$

Αυτό φαίνεται λογικό, επειδή υπάρχουν δύο κλάσεις, θετικά και αρνητικά. Έχουμε έναν ταξινομητή ο οποίος ταξινομεί τις προτάσεις σε δύο κλάσεις (θετικές, αρνητικές). Αυτό είναι το μέτρο αποτελεσματικότητας που χρησιμοποιείται πιο συχνά στην αξιολόγηση της ταξινόμησης μέσω μηχανικής μάθησης.

Μέτρο F (F1 Score) F1: Είναι το μέτρο για την αντιστάθμιση της ακρίβειας έναντι της ανάκλησης), δηλαδή ο σταθμισμένος αρμονικός μέσος ακρίβειας και ανάκλησης:

$$F1 = 2\,\frac{P\,R}{P + R}$$

Επίσης για τα πειράματα που εκτελέσαμε πήραμε μετρήσεις για τον χρόνο που χρειάστηκε το μοντέλο για να εκπαιδευτεί και για το μέγεθος του λεξικού που δημιουργούσε σε κάθε πείραμα.

6.2 Σύστημα αξιολόγησης - Οργάνωση πειραμάτων

Για την εκτέλεση των πειραμάτων αρχικά έπρεπε να διαμορφώσουμε την συλλογή μας ώστε τα αρχεία της να έχουν την κατάλληλη μορφή για χρησιμοποιηθούν από το μοντέλο των RAE. Αρχικά αφαιρέσαμε από την συλλογή μας τις προτάσεις που είχαν χαρακτηριστεί ως ουδέτερες. Τα πειράματα που θα εκτελέσουμε θα αξιολογήσουν το σύστημα μας ως προς την

δυνατότητα του να χαρακτηρίσουν κάποιο κείμενο θετικό ή αρνητικό. Κρατήσαμε μόνο τις προτάσεις που είχαν χαρακτηριστεί ως θετικές ή αρνητικές . Έπειτα χωρίσαμε τις προτάσεις σε δύο αρχεία με βάση την κατηγορία τους (θετικές και αρνητικές). Τα δύο αρχεία που προέκυψαν περιείχαν διαφορετικό πλήθος προτάσεων. Οι θετικές προτάσεις ήταν πολύ περισσότερες από τις αρνητικές . Για να έχουμε σωστότερη εκπαίδευση του μοντέλου και να αποφύγουμε λάθος αποτελέσματα λόγω διαφορετικού πλήθους θετικών, αρνητικών προτάσεων, εξισορροπούμε το πλήθος των προτάσεων αφαιρώντας θετικές προτάσεις από την συλλογή μας ώστε θετικές και αρνητικές να είναι ίσες. Η τελική μορφή της συλλογής μας με την οποία διεξήγαμε τα πειράματα περιέχει συνολικά 2800 προτάσεις (1400 θετικές και 1400 αρνητικές). Για την εκτέλεση των πειραμάτων χωρίσαμε το dataset σε "training" που χρησιμοποιείται για την εκπαίδευση του μοντέλου των επαναληπτικών αυτοσυσχετιστών και σε "test" με το οποίο θα ελέγχουμε την αποτελεσματικότητα του μοντέλου.

Τα χαρακτηριστικά της συλλογής μας που εκτελέσαμε τα πειράματα αποτελείται από τέσσερα αρχεία. «pos.txt, neg.txt test_pos.txt και test_neg.txt». Το "*pos.txt"* περιέχει 1260 θετικές προτάσεις και το "*neg.txt"* 1260 αρνητικές. Τα "*test_pos.txt"* και "*test_neg.txt"* περιέχουν από 140 προτάσεις θετικές και αρνητικές αντίστοιχα.

Στα πειράματα που θα κάνουμε θα χρησιμοποιήσουμε τρείς διαφορετικές μορφές της συλλογής μας.

- **raw**: Περιέχει τα σχόλια των χρηστών όπως τα αναρτήσανε στην ιστοσελίδα. Χωρίς να έχει γίνει κάποια επιπλέον επεξεργασία στο κείμενο τους.
- **spelling**: Περιέχει τα σχόλια των χρηστών διορθωμένα ως προς την ορθογραφία τους με την χρήση του εργαλείου που αναπτύξαμε.
- **lemmatiz**: Περιέχει τα σχόλια των χρηστών στα οποία έχει διορθωθεί αρχικά η ορθογραφία τους και στην συνέχεια έχει γίνει λημματοποίηση δηλαδή έχουν αντικατασταθεί όλες οι λέξεις στις προτάσεις με την αρχική τους (παράδειγμα η πρόταση «με εύκολη και άμεση αναζήτηση προσώπων από το μενού επαφών» έχει αλλάξει σε «με εύκολος και άμεσος αναζήτηση πρόσωπο από το μενού επαφή»).

Έχουμε αναπτύξαμε εφαρμογή σε java με ττην οποία μπορούμε να αυτοματοποιήσουμε την εκτέλεση των πειραμάτων. Η εφαρμογή μπορεί να ελέγξει τις παραμέτρους των πειραμάτων μέσα από γραφικό περιβάλλον και να καταγράψει τις παραμέτρους αξιολόγησης σε αρχείο.

Εκτός από τις διαφορετικές παραλλαγές της συλλογής μας με τις οποίες εκτελούμε τα πειράματα μας δοκιμάσαμε και διαφορετικές τιμές για τις παραμέτρους του μοντέλου. Οι παράμετροι του μοντέλου με τις οποίες θα πειραματιστούμε είναι :

- **MaxIterations:** Ο αριθμός των επαναλήψεων για την εκπαίδευση των RAE.

- **minCount:** Οι φορές που πρέπει να εμφανίζεται μια λέξη στη συλλογή εκπαίδευσης ώστε να προστεθεί στο λεξικό και να χρησιμοποιηθεί για την εκπαίδευση των RAE.
- **embeddingSize:** Οι RAE αναπαριστούν τις λέξεις του λεξικού σαν διανύσματα. Η συγκεκριμένη μεταβλητή είναι το μέγεθος του διανύσματος.

Κάθε πειράματα που διεξήγαμε το επαναλάβαμε τρεις φορές με τις ίδιες παραμέτρους. Οι τιμές των παραμέτρων αξιολόγησης που παρουσιάζονται είναι ο μέσος όρος από τις επαναλήψεις.

Τέλος όλα τα πειράματα έχουν εκτελεστεί στον ίδιο ηλεκτρονικό Υπολογιστή με τα χαρακτηριστικά που περιγράφονται στο κεφάλαιο 7.

6.3 Αποτελέσματα

Με βάση τις παραμέτρους του μοντέλου έχουμε εκτελέσει 8 βασικά πειράματα. Στα οποία δίνουμε διαφορετικές τιμές στα MaxIterations minCount και embeddingSize.

6.3.1 MaxIterations: 80 minCount: 5 embeddingSize: 50

Στο πρώτο πείραμα έχουμε δώσει τις εξής τιμές στις παραμέτρους του μοντέλου, MaxIterations: 80, minCount 5, και embeddingSize 50. Εκτελέσαμε το πείραμα στις τρείς παραλλαγές του dataset και τα αποτελέσματα φαίνονται στον πίνακα 12

Run	Dataset version	Iterations	Min word count *	embedding Size	vocabulary	function variables	Training time (in minutes)	Training	Testing	
1	raw	80	5	50	856	53001	90	0,797	0,679	Precision
								0,797	0,679	Recall
								0,797	0,679	Accuracy
								0,797	0,679	F1 Score
2	spelling	80	5	50	734	46901	86	0,900	0,811	Precision
								0,900	0,812	Recall
								0,900	0,811	Accuracy
								0,900	0,811	F1 Score
3	Lemmatiz	80	5	50	638	42101	83	0,885	0,813	Precision
								0,885	0,814	Recall
								0,885	0,813	Accuracy
								0,885	0,814	F1 Score

Πίνακας 12 Πείραμα 1 με παραμέτρους MaxIterations: 80, minCount 5, και embeddingSize 50

Από τα αποτελέσματα βλέπουμε πως οι παράμετροι αξιολόγησης βελτιώνονται εσθήτα μετά την επεξεργασία της συλλογής . Βλέπουμε επίσης ότι μειώνεται και ο χρόνιος εκπαίδευσης του συστήματος και το μέγεθος του λεξικού.

6.3.2 *MaxIterations: 80 minCount: 2 embeddingSize: 50*

Στο επόμενο πείραμα δώσαμε τις τιμές MaxIterations: 80, minCount 2, και embeddingSize 50 στις παραμέτρους του μοντέλου. Εκτελέσαμε το πείραμα στις τρείς παραλλαγές του dataset και τα αποτελέσματα παρουσιάζονται στον Πίνακα 13. Και σε αυτό το πείραμα βλέπουμε τις παράμετρους αξιολόγησης να βελτιώνονται στις συλλογές που επεξεργαστήκαμε. Παρατηρούμε επίσης ότι μειώνοντας την παράμετρο minCount αυξάνεται το πλήθος των λέξεων στο λεξικού, άρα το μοντέλο των REA χρησιμοποιεί περισσότερες λέξεις για την εκπαίδευση και πρόβλεψη του συναισθήματος. Επίσης παρατηρούμε βελτίωση και στα αποτελέσματα σε όλες της παραλλαγές της συλλογής μας σε σχέση με το προηγούμενο πείραμα.

Run	**Dataset version**	**Iterations**	**Min word count ***	**embedding Size**	**vocabulary**	**function variables**	**Training time** (in minutes)	**Training**	**Testing**	
4	raw	80	2	50	2181	119251	91	0,805	0,685	Precision
								0,805	0,687	Recall
								0,805	0,685	Accuracy
								0,805	0,686	F1 Score
5	spelling	80	2	50	1659	93151	87	0,893	0,831	Precision
								0,893	0,831	Recall
								0,893	0,831	Accuracy
								0,893	0,831	F1 Score
6	Lemmatiz	80	2	50	1214	70901	83	0,890	0,848	Precision
								0,890	0,848	Recall
								0,890	0,848	Accuracy
								0,890	0,848	F1 Score

Πίνακας 13 Πείραμα 2 με παραμέτρους MaxIterations: 80, minCount 2, και embeddingSize 50

6.3.3 *MaxIterations: 50 minCount: 5 embeddingSize: 50*

Στην συνέχεια εκτελούμε το πείραμα με τιμές παραμέτρων MaxIterations: 50, minCount 5, και embeddingSize 50 πινακας 14

Run	Dataset version	Iterations	Min word count *	embedding Size	vocabulary	function variables	Training time (in minutes)	Training	Testing	
7	raw	50	5	50	856	53001	88	0,825	0,754	Precision
								0,826	0,757	Recall
								0,825	0,754	Accuracy
								0,826	0,755	F1 Score
8	spelling	50	5	50	734	46901	85	0,871	0,833	Precision
								0,871	0,836	Recall
								0,871	0,833	Accuracy
								0,871	0,835	F1 Score
9	Lemmatiz	50	5	50	638	42101	82	0,866	0,848	Precision
								0,866	0,848	Recall
								0,866	0,848	Accuracy
								0,866	0,848	F1 Score

Πίνακας 14 Πείραμα 3 με παραμέτρους MaxIterations: 50, minCount 5, και embeddingSize 50

Και σε αυτό το πείραμα παρατηρούμε βελτίωση των παραμέτρων αξιολόγησης στις συλλογές που επεξεργαστήκαμε. Επίσης παρατηρούμε μια μικρή βελτίωση των αποτελεσμάτων σε σχέση με το πρώτο πείραμα που χρησιμοποιήσαμε περισσότερες εποχές εκπαίδευσης.

6.3.4 MaxIterations: 50 minCount: 2 embeddingSize: 50

Συνεχίζουμε με το επόμενο πείραμα το οποίο το εκτελούμε με παραμέτρους MaxIterations: 50, minCount 2, και embeddingSize 50 πίνακας 15

Run	Dataset version	Iterations	Min word count *	embedding Size	vocabulary	function variables	Training time (in minutes)	Training	Testing	
10	raw	50	2	50	2181	119251	95	0,779	0,721	Precision
								0,779	0,724	Recall
								0,779	0,721	Accuracy
								0,779	0,723	F1 Score
11	spelling	50	2	50	1659	93151	92	0,856	0,819	Precision
								0,856	0,820	Recall
								0,856	0,819	Accuracy
								0,856	0,819	F1 Score
12	Lemmatiz	50	2	50	1214	70901	86	0,882	0,849	Precision
								0,882	0,850	Recall
								0,882	0,849	Accuracy
								0,882	0,850	F1 Score

Πίνακας 15 Πείραμα 4 με παραμέτρους MaxIterations: 50, minCount 2, και embeddingSize 50

Και εδώ παρατηρούμε βελτίωση των παραμέτρων αξιολόγησης στις επεξεργασμένες συλλογές. Τα αποτελέσματα είναι παρόμοια με το δεύτερο πείραμα που είχαμε μεγαλύτερο αριθμό επαναλήψεων.

6.3.5 MaxIterations: 80 minCount: 5 embeddingSize: 20

Στα υπόλοιπα πειράματα θα μειώσουμε το embeddingSize σε 20. embeddingSize είναι ο αριθμός των διαστάσεων που θα χρησιμοποιούν οι RAE για την αναπαράσταση των προτάσεων.

Στο πίνακα 16 βλέπουμε τα αποτελέσματα από το πείραμα με παραμέτρους MaxIterations: 80, minCount 5, και embeddingSize 20 όπου και εδώ παρατηρούμε βελτίωση των παραμέτρων αξιολόγησης στις επεξεργασμένες συλλογές. Σε σχέση με το πρώτο πείραμα που είχαμε ίδιες παραμέτρους εκτός του embeddingSize δεν παρατηρούμε κάποια σημαντική αλλαγή στα αποτελέσματα. Αυτό που έγινε αισθητό με την αλλαγή του embeddingSize είναι ο χρόνος εκπαίδευσης του μοντέλου που έχει μειωθεί στο μισό.

Run	Dataset version	Iterations	Min word count *	embedding Size	vocabulary	function variables	Training time (in minutes)	Training	Testing	
13	raw	80	5	20	856	18801	40	0,870	0,753	Precision
								0,870	0,754	Recall
								0,870	0,753	Accuracy
								0,870	0,753	F1 Score
14	spelling	80	5	20	734	16361	35	0,897	0,808	Precision
								0,897	0,810	Recall
								0,897	0,808	Accuracy
								0,897	0,809	F1 Score
15	Lemmatiz	80	5	20	638	14441	30	0,891	0,824	Precision
								0,891	0,825	Recall
								0,891	0,824	Accuracy
								0,891	0,825	F1 Score

Πίνακας 16 Πείραμα 5 με παραμέτρους MaxIterations: 80, minCount 5, και embeddingSize 20

6.3.6 MaxIterations: 80 minCount: 2 embeddingSize: 20

Για τα αποτελέσματα του πίνακα 17 εκτελέσαμε τα πειράματα με παραμέτρους MaxIterations: 80, minCount 2, και embeddingSize 20.

Run	Dataset version	Iterations	Min word count *	embedding Size	vocabulary	function variables	Training time (in minutes)	Training	Testing	
16	raw	80	2	20	2181	45301	30	0,945	0,786	Precision
								0,945	0,787	Recall
								0,945	0,786	Accuracy
								0,945	0,786	F1 Score
17	spelling	80	2	20	1659	34861	30	0,949	0,828	Precision
								0,949	0,828	Recall
								0,949	0,828	Accuracy
								0,949	0,828	F1 Score
18	Lemmatiz	80	2	20	1214	25961	25	0,913	0,836	Precision
								0,913	0,836	Recall
								0,913	0,836	Accuracy
								0,913	0,836	F1 Score

Πίνακας 17 Πείραμα 6 με παραμέτρους MaxIterations: 80, minCount 2, και embeddingSize 20

Και εδώ βλέπουμε βελτίωση στις επεξεργασμένες συλλογές. Σε σχέση με το πείραμα 2 που είχαμε τις ίδιες παραμέτρους (εκτός του embeddingSize) παρατηρούμε μεγάλη βελτίωση στην συλλογή raw όπου το accuracy από 0,685 έφτασε στο 0,786. Σε σχέση με το πείραμα 5 που διαφέρουν ως προς το minCount δεν παρατηρούμε κάποια αισθητή αλλαγή.

6.3.7 MaxIterations: 50 minCount: 5 embeddingSize: 20

Στο επόμενο πείραμα έχουμε παραμέτρους MaxIterations: 80, minCount 2, και embeddingSize 20. Πίνακας 18

Run	Dataset version	Iterations	Min word count *	embedding Size	vocabulary	function variables	Training time (in minutes)	Training	Testing	
19	raw	50	5	20	856	18881	23	0,751	0,701	Precision
								0,752	0,701	Recall
								0,751	0,701	Accuracy
								0,752	0,701	F1 Score
20	spelling	50	5	20	734	16361	23	0,875	0,816	Precision
								0,875	0,817	Recall
								0,875	0,816	Accuracy
								0,875	0,816	F1 Score
21	Lemmatiz	50	5	20	638	14441	22	0,865	0,820	Precision
								0,865	0,821	Recall
								0,865	0,820	Accuracy
								0,865	0,820	F1 Score

Πίνακας 18 Πείραμα 7 με παραμέτρους MaxIterations: 50, minCount 5, και embeddingSize 20

Και εδώ βελτιώνονται τα αποτελέσματα σε σχέση με τις επεξεργασμένες συλλογές. Σε σχέση με το πείραμα 3 πού διαφέρουνε μόνο στο embeddingSize παρατηρούμε μια μικρή χειροτέρευση των αποτελεσμάτων σε όλες τις παραλλαγές τις συλλογής μας. Επίσης βλέπουμε και μια βελτίωση ως προς τον χρόνο εκπαίδευσης.

6.3.8 MaxIterations: 50 minCount: 2 embeddingSize: 20

Στο τελευταίο πείραμα το εκτελούμε με παραμέτρους MaxIterations: 50 minCount: 2 embeddingSize: 20 πίνακας 19. Όπου και εδώ παρατηρούμε βελτίωση στα αποτελέσματα των επεξεργασμένων συλλογών. Σε σχέση με το πείραμα 4 πού διαφέρουνε στο embeddingSize δεν παρατηρούμε κάποια αισθητή μεταβολή των αποτελεσμάτων όπως και στο πείραμα 7 που διαφέρουν ως προς το minCount.

Run	**Dataset version**	**Iterations**	**Min word count ***	**embedding Size**	**vocabulary**	**function variables**	**Training time** (in minutes)	**Training**	**Testing**	
22	raw	50	2	20	2181	45301	23	0,807	0,704	Precision
								0,808	0,704	Recall
								0,807	0,704	Accuracy
								0,807	0,704	F1 Score
23	spelling	50	2	20	1659	34861	23	0,886	0,821	Precision
								0,886	0,822	Recall
								0,886	0,821	Accuracy
								0,886	0,822	F1 Score
24	Lemmatiz	50	2	20	1214	25961	22	0,906	0,824	Precision
								0,906	0,825	Recall
								0,906	0,824	Accuracy
								0,906	0,824	F1 Score

Πίνακας 19 Πείραμα 8 με παραμέτρους MaxIterations: 50, minCount 2, και embeddingSize 20

6.4 Σύνοψη συμπερασμάτων αξιολόγησης

Σε όλα τα πειράματα παρατηρήσαμε σημαντική βελτίωση στις παραμέτρους αξιολόγησης για τις συλλογές που επεξεργαστήκαμε με τα εργαλεία.

Ανεξάρτητα από τις παραμέτρους του μοντέλου γίνεται φανερό πως με την επεξεργασία των προτάσεων μπορούμε να επιτύχουμε πολύ καλύτερα αποτελέσματα.

Σε σχέση τώρα με τις παραμέτρους του μοντέλου των RAE παρατηρήσαμε πως μια σημαντική παράμετρος για την εκπαίδευση του μοντέλου είναι το όριο που θα πρέπει να εμφανίζεται μια λέξη στην συλλογή για την προσθέσει στο λεξικό. Στα περισσότερα σετ πειραμάτων που διαφέρανε προς το minCount παρατηρήσαμε καλύτερα αποτελέσματα σε αυτό που είχε τον μικρότερο αριθμό.

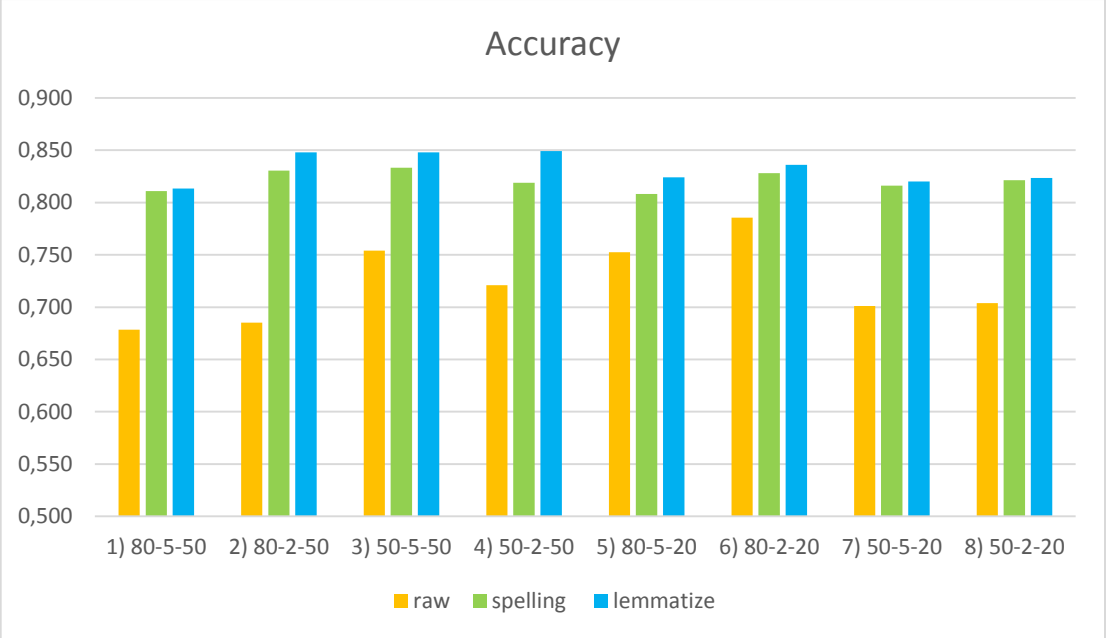
Μια άλλη παρατήρηση είναι η αισθητή μείωσή του χρόνου εκπαίδευσης όταν μειώνουμε το μέγεθος του embeddingSize, χωρίς να επηρεάζει σε μεγάλο βαθμό τις άλλες παραμέτρους αξιολόγησης.

Αναλυτικά:

- Ο μέσος όρος της ακρίβεια (Precision), σε όλα τα πειράματα με το αρχικό dataset (raw) είναι 0,723. Ο μέσος όρος για το dataset που έχει διορθωθεί η ορθογραφία (spelling) είναι 0,821 και με το lemmatize ο μέσος όρος έφτασε στο 0,833.
- Ο μέσος όρος της ανάκληση (Recall) σε όλα τα πειράματα με το αρχικό dataset (raw) είναι 0,724. Ο μέσος όρος για το dataset που έχει διορθωθεί η ορθογραφία (spelling) είναι 0,822 και με το lemmatize ο μέσος όρος έφτασε στο 0,833.
- Ο μέσος όρος της ορθότητας (Accuracy) σε όλα τα πειράματα με το αρχικό dataset (raw) είναι 0,723. Ο μέσος όρος για το dataset που έχει διορθωθεί η ορθογραφία (spelling) είναι 0,821 και με το lemmatize ο μέσος όρος έφτασε στο 0,833.
- Ο μέσος όρος το μέτρο F (F1 Score). σε όλα τα πειράματα με το αρχικό dataset (raw) είναι 0,723. Ο μέσος όρος για το dataset που έχει διορθωθεί η ορθογραφία (spelling) είναι 0,822 και με το lemmatize ο μέσος όρος έφτασε στο 0,833.
- Τα καλύτερα αποτελέσματα τα πετύχαμε στο πείραμα 4 με παραμέτρους μοντέλου MaxIterations: 50, minCount 2, και embeddingSize 50. Στην συλλογή που έγινε λημματοποίηση των σχολίων όπου είχαμε Precision:0,849 Recall:0,850 Accuracy:0,849 και F1Score: 0,850.

6.4.1 Τιμές Ορθότητας (Accuracy) στα πειράματα

Στο γράφημα 1 παρουσιάζεται η ορθότητα (Accuracy) για όλα τα πειράματα που διεξήγαμε. Ο κάθετος άξονας είναι η τιμή της ορθότητας και στον οριζόντιο εμφανίζονται τα πειράματα με την σειρά που τα εκτελέσαμε και τις παραμέτρους που είχαν (MaxIterations – minCount - embeddingSize).



Γράφημα 1 Ορθότητα (Accuracy) για όλα τα πειράματα

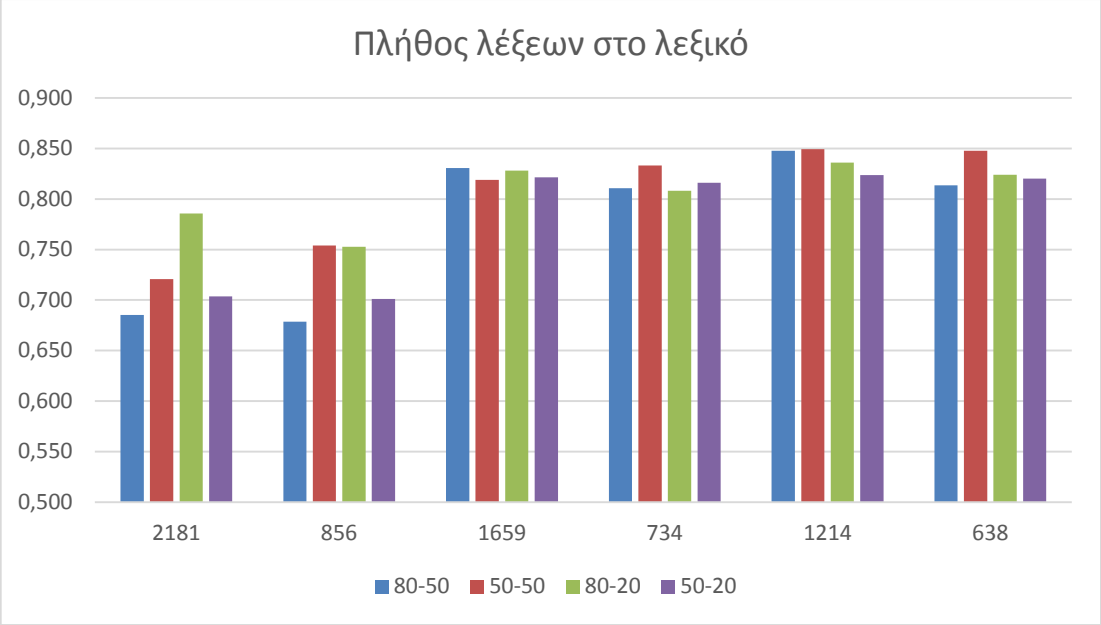
Το πρώτο που παρατηρούμε είναι η σημαντική βελτίωση τις τιμής της ορθότητας στις δύο επεξεργασμένες συλλογές.

Παρατηρούμε πως η ορθότητα των επεξεργασμένων συλλογών εμφανίζει πιο σταθερές τιμές στα πειράματα σε σχέση με την μη επεξεργασμένη συλλογή.

Επίσης βλέπουμε μια πτώση στην τιμή της ορθότητας όταν μειώνουμε το μέγεθος του διανύσματος αναπαράστασης των λέξεων (embeddingSize).

6.4.2 Ορθότητα (Accuracy) σε σχέση με το πλήθος των λέξεων

Στο επόμενο γράφημα 2 παρουσιάζεται η ορθότητα (Accuracy) σε σχέση με το πλήθος των λέξεων στο λεξικό εκπαίδευσης των RAE (κάθετος άξονας ορθότητα, οριζόντιο πλήθος λέξεων στο λεξικό). Οι διαφορές στον αριθμό των λέξεων προέρχονται α) από την συλλογή που χρησιμοποιήσαμε και β) από την τιμή του minCount (αριθμός που πρέπει να εμφανίζεται μια λέξη στην συλλογή για να προστεθεί στο λεξικό εκπαίδευσης.) Οι τέσσερεις στήλες που εμφανίζονται ομαδοποιημένες αφορούν την ίδια συλλογή σχολίων με αλλαγές στις παραμέτρους MaxIterations και embeddingSize. Ανά δύο οι ομάδες στηλών αναφέρονται στην ίδια συλλογή. (2181-856 συλλογή raw, 1659-734 συλλογή spelling και 1214-638 συλλογή lemmatize)



Γράφημα 2 Η ορθότητα (Accuracy) σε σχέση με το πλήθος των λέξεων στο λεξικό εκπαίδευσης των RAE

Τα συμπεράσματα που βγάζουμε από το γράφημα είναι πως η ορθότητα δεν επηρεάζεται σε πολύ μεγάλο βαθμό από το μέγεθος του λεξικού. Αναλυτικότερα αυτό που παρατηρούμε είναι πώς αν και ο αριθμός των λέξεων στο λεξικό διπλασιάζεται στα ζευγάρια γειτονικών ομάδων πειραμάτων που αναφέρονται στην ίδια συλλογή η μεταβολές στην ορθότητα είναι ελάχιστες. Καταλήγουμε στο συμπέρασμα πως έχει μεγαλύτερη σημασία να προστεθούν στο λεξικό εκείνες οι λέξεις που έχουν κάποια συναισθηματική πόλωση απ' ότι ένα μεγάλο πλήθος λέξεων.

6.4.3 Βελτίωση των αποτελεσμάτων στις συλλογές

Στον παρακάτω πίνακα 20 παρουσιάζονται οι τιμές της ορθότητα (Accuracy) σε όλα τα πειράματα μαζί με την η ποσοστιαία μεταβολή της σε σύγκριση με τις άλλες συλλογές. Στον πίνακα βλέπουμε:

Η πρώτη στήλη είναι ο αύξων αριθμός του πειράματος μαζί με τις παραμέτρους εκτέλεσης (MaxIterations – minCount - embeddingSize). Οι επόμενες τρείς στήλες είναι η τιμή της ορθότητας στα πειράματα και στις υπόλοιπες η διαφορά ανάμεσα στης παραλλαγές της συλλογής μας.

	raw	spelling	lemmatize	raw - spelling	raw - lemmatize	spelling - lemmatize
1) 80-5-50	0,679	0,811	0,813	19,49%	19,88%	0,33%
2) 80-2-50	0,685	0,831	0,848	21,24%	23,75%	2,07%
3) 50-5-50	0,754	0,833	0,848	10,53%	12,46%	1,75%
4) 50-2-50	0,721	0,819	0,849	13,58%	17,80%	3,72%
5) 80-5-20	0,753	0,808	0,824	7,38%	9,49%	1,96%
6) 80-2-20	0,786	0,828	0,836	5,39%	6,40%	0,96%
7) 50-5-20	0,701	0,816	0,820	16,42%	16,98%	0,49%
8) 50-2-20	0,704	0,821	0,824	16,73%	17,03%	0,26%
M.O.	0,723	0,821	0,833	13,84%	15,47%	1,44%

Πίνακας 20 Τιμές ορθότητας (Accuracy) για όλα τα πειράματα μαζί με την η ποσοστιαία μεταβολή της σε σύγκριση με τις συλλογές

Παρατηρούμε πως η μεταβολή της ορθότητας για τις συλλογές raw - spelling βελτιώνεται από 5,39% μέχρι και 21.24% με μέσο όρο 13,84%. Για τις συλλογές raw – lemmatize παρατηρούμε βελτίωση από 6,4% μέχρι και 23,75% με μέσο όρο 15,47%. Τέλος η βελτίωση ανάμεσα στην συλλογή spelling και lemmatize κυμαίνεται από 0,26% μέχρι και 3,72% με μέσο όρο 1,44%

7

Τεχνικές λεπτομέρειες

Για την κατασκευή των εργαλείων μας έχουμε αναπτύξει τις παρακάτω εφαρμογές - κλάσεις.

- Εφαρμογή SentimentAnalysisGR οποία παρέχει γραφικό περιβάλλον για την χρήση αναδρομικών αυτοσυσχετιστών για την αυτόματη εκτίμηση της διάθεσης.
- Κλάσεις σε java για την δημιουργία του λεξικού οι οποίες διαβάζουν από αρχεία λέξεις και παράγουν-δημιουργούν αρχεία που περιέχουν τις λέξεις σε διαφορετικές πτώσεις, κλείσεις, αριθμούς.
- Εφαρμογή σε java για την ανάγνωση των λέξεων από βάση δεδομένων MySQL και την αντιγραφή τους σε βάση δεδομένων MongoDB με προσθήκη επιπλέον πεδίων.
- Εφαρμογή σε java για την δημιουργία και την επεξεργασία της συλλογής σχολίων μας.
- Εφαρμογή σε C# για αυτόματη διόρθωση ορθογραφικών λαθών σε κείμενα.
- Κλάση σε java για την λημματοποίηση κειμένων.

7.1 Λεπτομέρειες υλοποίησης

7.1.1 Εφαρμογή αυτόματης εκτίμηση της διάθεσης κειμένων

Για την ανάπτυξη της εφαρμογής αυτόματης εκτίμηση της διάθεσης κειμένων κάναμε χρήση του κώδικα JRAE που προέρχεται από τον σύνδεσμο https://github.com/sancha/jrae

7.1.1.1 JRAE

Το project που υλοποίει τους RAE (jrae-master.zip) περιέχει:

- Φάκελο src στον οποίο βρίσκονται οι κλάσεις του project. Οι κλάσεις είναι χωρισμένες σε 7 packages ανάλογα με την χρήση τους. Το project ξεκινάει την εκτέλεση από την κλάση "FullRun.java" στην οποία η μέθοδος main() δέχεται ως ορίσματα τις παραμέτρους των RAE.
- Φάκελο data ο οποίος περιέχει α) Φάκελους mov και movie οι οποίοι περιέχουν σχόλια για ταινίες στην αγγλική γλώσσα με τα οποία μπορούμε να εκτελέσουμε δοκιμές. Ο φάκελος mov περιέχει και αρχείο *tunedTheta.rae* το οποίο είναι το εκπαιδευμένο μοντέλο των RAE. β) Φάκελο parsed ο οποίος περιέχει αρχεία από την ανάλυση της συλλογής του φακέλου mov.
- Φάκελο libs ο οποίος περιέχει τις βιβλιοθήκες που κάνει χρήση το project. (Πριν εκτελέσουμε την εφαρμογή θα πρέπει να εισάγουμε τις βιβλιοθήκες στο project).
- Αρχεία *README.md, run.sh* και USAGE τα οποία περιέχουν οδηγίες για την εκτέλεση του project. Το αρχείο README.md περιέχει πληροφορίες για το project. Το αρχείο run.sh περιέχει παραδείγματα εκτέλεσης του project με τις παραμέτρους των RAE και τέλος στο USAGE υπάρχουν επεξηγήσεις για τις παραμέτρους που δέχεται το project κατά την εκτέλεση του.

Η εκτέλεση των RAE ξεκινάει από την κλάση FullRun.java που βρίσκεται στο πακέτο main. Η κλάση "FullRun.java" αρχίζει να εκτελείται μέσω της μεθόδου main() η οποία δέχεται ως ορίσματα τις παραμέτρους εκτέλεσης των RAE.

7.1.1.2 Οι παράμετροι εκτέλεσης των RAE:

- DataDir DIR

  Είναι η διαδρομή φακέλων από την οποία θα διαβάσει τα δεδομένα και για εκπαίδευση και για τεστ. Τα αρχεία πρέπει να είναι οργανωμένα, έτσι ώστε όλες οι προτάσεις που ανήκουν σε μία κατηγορία να βρίσκονται στο ίδιο αρχείο με όνομα την κατηγορία. Κάθε γραμμή κειμένου είναι ξεχωριστή πρόταση. Τα δεδομένα ελέγχου μπορούν να βρίσκονται σε ένα αρχείο αν δεν είναι γνώστη η κατηγορία που ανήκουν. Αν γνωρίζουμε την κατηγορία τότε χρησιμοποιούμε το ίδιο όνομα με το πρόθεμα «test_». Για παράδειγμα οι προτάσεις εκπαίδευσης για τα θετικά σχόλια πάνε στο αρχείο με όνομα pos.txt και οι προτάσεις για τον έλεγχο στο αρχείο με όνομα «test_pos.txt» Αν οι κατηγορίες ελέγχου είναι γνωστές τότε στο τέλος της εκτέλεσης θα εμφανιστούν οι παράμετροι αξιολόγησης.

- minCount MINCOUNT

  Είναι πλήθος τον φορών που θα πρέπει να εμφανίζεται μια λέξη στο dataset για να προστεθεί στο λεξικό. Προεπιλεγμένη τιμή είναι το 5.

- TrainModel TRUE|FALSE

Δείχνει αν θα πρέπει να εκπαιδευτουν οι RAE ή αν θα κάνουν μόνο test.

- ModelFile FILE

  Αυτή η παράμετρος είναι προαιρετική. Και δείχνει σε ποιο αρχείο θα αποθηκευτούν τα δεδομένα εκπαίδευσης αν το TrainModel είναι True. Αν είναι False δείχνει το μοντέλο που θα χρησιμοποιήσει για να διεξάγει τα tests.

- ClassifierFile FILE

  Αυτή η παράμετρος είναι προαιρετική και δείχνει σε ποιο αρχείο θα αποθηκευτεί ο ταξινομητής αν το TrainModel είναι True. Αν είναι False δείχνει τον ταξινομητή που θα χρησιμοποιήθει για να διεξάγει τα tests

- FeaturesOutputFile FILE

  Αυτή η παράμετρος χρησιμοποιείται μόνο στο testing και είναι το αρχείο στο οποίο εξάγονται τα χαρακτηριστικά που χρησιμοποίησε το μοντέλο για το test. Το αρχείο δεν θα πρέπει να είναι .txt ή δεν θα πρέπει να δείχνει στον ίδιο φάκελο με το –DataDir.

- ProbabilitiesOutputFile FILE

  Αυτή η παράμετρος χρησιμοποιείται μόνο στο testing και είναι το αρχείο στο οποίο αποθηκεύονται οι ταξινομήσεις του μοντέλου.

- NumCores NUMCORES

  Ο αριθμός των παράλληλών νημάτων που θα χρησιμοποιηθούν για την επεξεργασία. Προεπιλεγμένη τιμή είναι ο αριθμός των πυρήνων του επεξεργαστή.

- MaxIterations MAXITERATIONS

  Ο αριθμός των επαναλήψεις για την εκπαίδευση των RAE. Προεπιλεγμένη τιμή 80.

- embeddingSize EMBEDDINGSIZE

  Ο αριθμός των διαστάσεων του διανύσματος που θα αναπαραστήστε η κάθε λέξη στο λεξικό. Προεπιλεγμένη τιμή είναι το 50.

- help

  Εμφανίζει την βοήθεια.

7.1.1.3 *Εφαρμογή SentimentAnalysisGR*

Στην εφαρμογή SentimentAnalysisGR κάνοντας χρήση του κώδικα των JRAE ο οποίος βρίσκεται στην κλάση FullRun.java αναπτύξαμε γραφικό περιβάλλον στην κλάση Start.java μέσω της οποίας εκτελούμε την εφαρμογή αυτόματης εκτίμηση της διάθεσης κειμένων.

Στην εφαρμογή μας διαβάζουμε τις παραμέτρους εκτέλεσης των RAE από τα πεδία της φόρμας και στην συνέχεια τις περνούμε ως παραμέτρους στους RAE πλαίσιο κώδικα 1.

```
String[] args = {"-DataDir", "data/raw", "-minCount", "5", "-TrainModel",
"True", "-ModelFile", "tunedTheta.rae", "-ClassifierFile", "Softmax.clf",
"-NumCores", "4", "-MaxIterations", "80", "-embeddingSize", "50",
"-ProbabilitiesOutputFile", "prob.out", "FeaturesOutputFile", "features.out",
"-numRep", "1", "-outputFile", "no"};

args[1] = Start.txtfld_datadir.getText();

if(!Start.txtfld_mincount.getText().isEmpty()){
	args[3] = Start.txtfld_mincount.getText().toString();
}
```

Πλαίσιο Κώδικα 1 - Παράμετροι RAE

Στην αρχή ορίζουμε μεταβλητή πίνακα string η οποία περιέχει τις παραμέτρους στις οποίες δίνουμε αρχικά τις προεπιλεγμένες τιμές τους και στην συνέχεια ελέγχουμε αν το αντίστοιχο textField της κάθε παραμέτρου έχει περιεχόμενο και αν έχει αλλάζουμε την τιμή της παραμέτρου με την τιμή του textField. Εκτός από τις παραμέτρους των RAE έχουμε προσθέσει και παραμέτρους για τον αριθμό των επαναλήψεων και για το αρχείο στο οποίο θα αποθηκευτούν τα αποτελέσματα των πειραμάτων ("-numRep", "1", "-outputFile", "no").

Η εκτέλεση των RAE γίνεται με το πάτημα του κουμπιού btn_Run πλαίσιο κώδικα 2 το οποίο ξεκινάει την εκτέλεση των RAE σε νέο νήμα (threat).

```
btn_Run.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		try {
			if(txtfld_datadir.getText().isEmpty()){
				txtpn_output.setText("DataDir is empty.\nGive
folder name to read training data");
			}else{
				Runnable runnable = new StartRunTrain();
				Thread thread = new Thread(runnable);
				thread.start();
			}
		} catch (Exception e1) {
			// TODO Auto-generated catch block
			e1.printStackTrace();
		}
	}
});
```

Πλαίσιο Κώδικα 2 – btn_Run

Η κλάση η οποία υλοποιεί το νήμα από το οποίο ξεκινάμε τους RAE πλαίσιο κώδικα 3 καλεί την μέθοδό fullRun() από την κλάση FullRun με της παραμέτρους που έχουμε ορίσει στα πεδία της φόρμας.

```
class StartRunTrain implements Runnable{
	@Override
	public void run() {
		fullRun(getArgs());
	}
	public static void fullRun(String[] args) {
		try {
			FullRun.fullRun(args);
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
```

Πλαίσιο Κώδικα 3 - Νήμα έναρξης των RAE

Στην κλάση FullRun.java έχουμε δημιουργήσει την μέθοδό fullRun() πλαίσιο κώδικα 4 η οποία δέχεται ως ορίσματα τις παραμέτρους από την κλάση Start.java.

```
public static void fullRun(String[] args) throws Exception {
	StringBuilder output = new StringBuilder();
	int repeatTimes = Integer.valueOf(args[21]);
	String saveFile = args[23];
	for (int i = 0; i < repeatTimes; i++) {
		output.append("Run " + i + "\n");
		output.append(runRAE(args));
		output.append("\n\n");
	}
	if(saveFile.equals("no")){
		JOptionPane.showMessageDialog(null, "Trainig is finish");
	}
	else{
		PrintWriter writer = new PrintWriter(saveFile, "UTF-8");
		writer.print(output.toString());
		writer.close();
		JOptionPane.showMessageDialog(null, "Results save at " +
saveFile);
		}
	}
```

Πλαίσιο Κώδικα 4 - μέθοδό fullRun()

Η κλάση δημιουργεί αντικείμενο StringBuilder στο οποίο αποθηκεύονται τα αποτελέσματα με σκοπό να τα αποθηκεύσει σε αρχείο σε περίπτωση που εκτελούμε κάποιο πείραμα με επαναλήψεις.

Έπειτα διαβάζει από τις παραμέτρους των τιμών για τις επαναλήψεις του πειράματος και το αρχείο στο οποίο θα σώσει τα αποτελέσματα. Στην συνέχεια καλεί την μέθοδο runREA() με τις παραμέτρους η οποία είναι η μέθοδος που τρέχει τους RAE. Αφού ολοκληρωθούν οι REA και είχαμε επιλέξει εκτέλεση train ή test μας εμφανίζει μήνυμα πως τελείωσε. Ενώ στην περίπτωση του πειράματος γράφει τα αποτελέσματα σε αρχείο και μας ενημερώνει για το πού αποθηκευτικέ το αρχείο.

Η εκτύπωση των αποτελεσμάτων των RAE στην περιοχή output της φόρμας γίνεται μέσω νήματος (threat) που καλείτε μετά από κάθε επανάληψη των RAE πλαίσιο κώδικα 5.

```
class PrintOutput implements Runnable{
	@Override
	public void run() {
		printOutput(getMsg());
	}

	public static void printOutput(String mgs) {
		try {
			Start.output(mgs);
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
	public static String getMsg(){
		String msg = FullRun.getoutput();
		return msg;
	}
}
```

Πλαίσιο Κώδικα 5 – Νήμα εκτύπωσης των αποτελεσμάτων των RAE

Το νήμα καλεί την μέθοδο output από την κλάση Start η οποία εμφανίζει τα αποτελέσματα. Τα αποτελέσματα στην κλάση FullRun αποθηκεύονται στο αντικείμενο printingOutput στο οποίο έχουμε πρόσβαση με την μέθοδο getoutput().

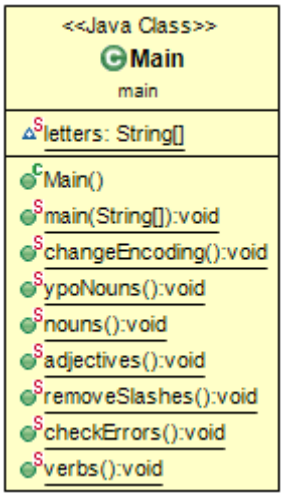
7.1.2 Κατασκευή λεξικού

Η κατασκευή του λεξικού γίνεται με το project GrDictionaryConstruct. Στο συγκεκριμένο project υπάρχει φάκελος data ο οποίος περιέχει τα αρχεία από τα οποία η εφαρμογή διαβάζει τις λέξεις. Ο φάκελος data περιέχει φάκελους στους οποίους είναι χωρισμένα τα αρχεία ανάλογα με το είδος των λέξεων που περιέχουν (ουσιαστικά - nouns, επίθετα – adjectives κτλ) το όνομα για κάθε αρχείο είναι «Χ_τύπος λέξεων» Δηλαδή το αρχείο που περιέχει τα ουσιαστικά που ξεκινάν από "Α" θα έχει τίτλο "A_nouns.csv".

Στο project γίνεται χρήση της βιβλιοθήκης common-lang3.jar για να μπορούμε να επεξεργαζόμαστε .csv αρχεία η βιβλιοθήκη βρίσκεται στο φάκελο libs. Οι κλάσης του project είναι στο φάκελο src όπου είναι χωρισμένες σε πακέτα ανάλογα με την χρήση τους.

Το project εκτελείτε από την κλάση main στο package main. Η κλάση main έχει έναν πίνακα letters που περιέχει τα γράμματα της αλφαβήτου με σκοπό να χρησιμοποιηθεί για να προσπελάσει τα αρχεία που περιέχουν τις λέξεις της κάθε κατηγορίας. Η κλάση main περιέχει μεθόδους που δημιουργούν στιγμιότυπα από της κλάσης για την δημιουργία των λέξεων.

Το διάγραμμά της κλάσης Main παρουσιάζεται στην εικόνα 13.



Εικόνα 13 Διάγραμμά κλάσης Main

7.1.2.1 Ουσιαστικά

Για την κατασκευή των ουσιαστικών καλούμε την μέθοδο nouns() από την κλάση Main. Η μέθοδος nouns() πλαίσιο κώδικα 6 κατασκευάζει στιγμιότυπο της κλάσης CombineNouns από το οποίο καλεί την μέθοδο combine()

```
public static void nouns() throws IOException{
                CombineNouns combineNouns = new CombineNouns();
                for(int i=0; i<letters.length; i++)
                        combineNouns.combine(letters[i]);
        }
```

Πλαίσιο Κώδικα 6 – κλήση μέθοδου nouns()

για όλα τα γράμματα από τον πίνακα letters πλαίσιο κώδικα 7

```
static String[] letters = {"Α", "Β", "Γ", "Δ", "Ε", "Ζ", "Η", "Θ", "Ι", "Κ",
"Λ", "Μ", "Ν", "Ξ", "Ο", "Π", "Ρ", "Σ", "Τ", "Υ", "Φ", "Χ", "Ψ", "Ω"};
```

Πλαίσιο Κώδικα 7 - τον πίνακα letters

Η κλάση CombineNouns.java χρησιμοποιείται για την δημιουργία των ουσιαστικών. Μέσο της κλάσης Read_Nouns.java διαβάζει τα αρχεία που περιέχουν τα ουσιαστικά και τις καταλήξεις τους και τα επιστρέφει ως πίνακα δύο διαστάσεων πλαίσιο κώδικα 8.

```
String[][] nouns = new String[high][width];
                while ((nextline = nounsReader.readNext()) != null) {
                String[] parts = nextline[0].split(";");
                        for (int j=0; j<parts.length; j++) {
                                nouns[i][j] = parts[j].trim();
                                }
                        i++;
                        }
                nounsReader.close();
                return nouns;
```

Πλαίσιο Κώδικα 8 - Ανάγνωση αρχείων ουσιαστικών και καταλήξεων τους

Οι μεταβλητές high και width είναι το ύψος και το μήκος του πίνακα.

Στην συνέχεια η κλάση CombineNouns.java αρχίζει να δημιουργεί τον πίνακα που περιέχει τις λέξεις σε όλες τις κλείσεις τους. Στο πλαίσιο κώδικα 9 παρουσιάζεται ο τρόπος δημιουργίας.

```
int counter = 0;
for (int i = 0; i < nounsCollection.length; i++) {
        boolean empty = true;
        String theme = nounsCollection[i][1];
        if(i==0){
                counter++;
        }
        elseif(!removeTone(nounsCollection[i][1]).equals(removeTone(nounsCollecti
on[i-1][1]))){
                counter++;
        }
        String line =
removeTone(String.valueOf(theme.charAt(0))).toUpperCase()+"O"+counter+ ";";
                        for (int j = 2; j < nounsCollection[i].length; j++) {
                                if ((!(nounsCollection[i][j]==null)) &&
(!nounsCollection[i][j].isEmpty())) {
                                        line = line +
theme+nounsCollection[i][j]+";";
                                        empty = false;
                                }
                                else{
                                        line = line + ";";
                                }
                        }
                        if (empty) {
                                line = line.replaceAll(";", "") + ";" + theme +
";;;;;;;;;;;;;;;;";
                        }
                        nounsList.add(line);
                }
                String[] outputArray = nounsList.toArray(new
String[nounsList.size()]);
                writer.writeNext(outputArray);
                writer.close();
                System.out.println(letter + " nouns completed");
        }
```

Πλαίσιο Κώδικα 9 - Δημιουργία πίνακα ουσιαστικών σε όλες τις πτώσεις

Αρχικά ελέγχει αν οι δύο γειτονικές λέξεις είναι ίδιες (ο έλεγχος γίνεται με την επιστροφή της μεθόδου removeTone() η οποία επιστρέφει την λέξη χωρίς τόνους). Αν οι λέξεις δεν είναι ίδιες αυξάνεται η μεταβλητή counter κατά 1. Στην συνεχεία δημιουργείται η εγγραφή για την λέξη. Για το id της λέξης παίρνουμε το πρώτο γράμμα της λέξης σε κεφαλαίο. Το γράμμα 'Ο' γιατί είναι ουσιαστικό και την τιμή του counter που είναι ένας αύξον αριθμός για κάθε λέξη.

Έπειτα κατασκευάζονται οι λέξεις όπου για κάθε στήλη του πίνακα ελέγχεται αν δεν είναι κενή. Στην γραμμή (line) προστίθεται το θέμα της λέξης με την κατάληξη και το ερωτηματικό ';' (το ερωτηματικό το χρησιμοποιούμε για τον διαχωρισμό των πεδίων στα .csv αρχεία που κατασκευάζουμε). Αν το πεδίο είναι κενό τότε προστίθεται μόνο το ερωτηματικό.

Τέλος η εγγραφή προστίθεται στο ArrayList nounsList. Αφού ολοκληρωθεί η διαδικασία για όλες τις λέξεις γράφονται σε αρχείο.

Για παράδειγμα αν είχαμε τις εγγραφές του πίνακα 21

O27	άδει	α		ας		α		α		ες				ες		ες
O27	αδει											ών				
O27	αδεί			ας												

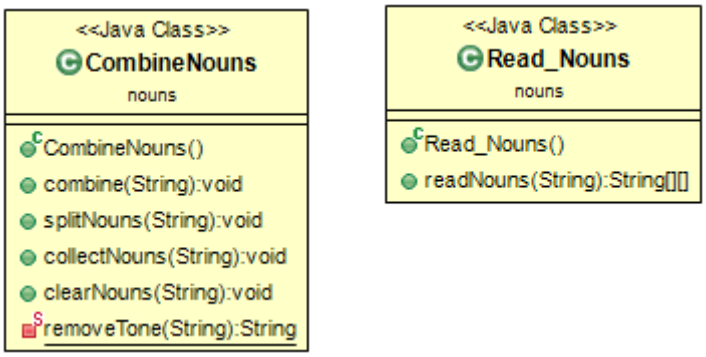
Πίνακας 21 Παράδειγμά θέματος ουσιαστικού με τις καταλήξεις

Στον έλεγχο των λέξεων (άδει, αδει, αδεί) είναι ίδιες γιατί τις ελέγχουμε χωρίς τόνο, άρα για όλες θα έχουμε τον ίδιο κωδικό. Για την κατασκευή των λέξεων θα περνάμε το θέμα 'αδεί' και θα το συνδυάζαμε με όλες τις καταλήξεις το τελικό αποτέλεσμα θα ήταν πίνακας 22.

AO304	άδεια	άδειας	άδεια	άδεια	άδειες	αδειών	άδειες	άδειες
AO304	αδειά	αδείας	αδειά	αδειά	αδειές	αδειών	αδειές	αδειές
AO304		αδειάς						

Πίνακας 22 Αποτέλεσμα κλήσης ουσιαστικού

Το διάγραμμά κλάσεων για την δημιουργία των ουσιαστικών παρουσιάζεται στην εικόνα 14

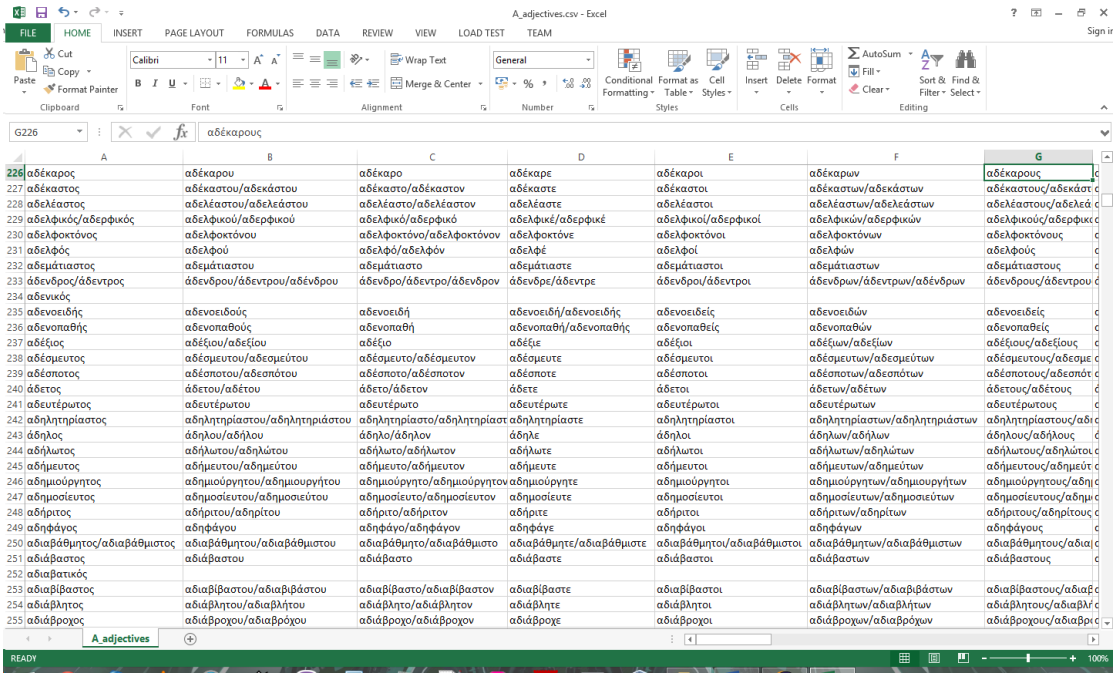


Εικόνα 14 Διάγραμμά κλάσεων δημιουργίας ουσιαστικών

7.1.2.2 Επίθετα

Για την κατασκευή των επιθέτων καλούμε την μέθοδο adjectives() από την κλάση Main.java. Η μέθοδος adjectives() κατασκευάζει στιγμιότυπο της κλάσης CombineAdjectives από το οποίο καλεί την μέθοδο combine() για όλα τα γράμματα από τον πίνακα letters όπως προηγούμενος και για τα ουσιαστικά.

Η μέθοδος combine() αρχικά διαβάζει αρχείο εικόνα 15 που περιέχει τα επίθετα στην αρχική τους μορφή,



Εικόνα 15 αρχείο επίθετων στην αρχική τους μορφή

μέσω της κλάσης Read_Adjectives (αντίστοιχη της Read_Nouns) και τα αποθηκεύει σε πίνακα δύο διαστάσεων adjectivesCollection. Πλαίσιο κώδικα 10

```
CSVWriter writer = new CSVWriter(new FileWriter("data/adjectives/" + letter +
"_adjectives_out.csv"), '\n');
Read_Adjectives readAdjects = new Read_Adjectives();
String[][] adjectivesCollection = readAdjects.readAdjectives("data/" + letter +
"_adjectives.csv");
```

Πλαίσιο Κώδικα 10 – Ανάγνωση επιθέτων μέσου της κλάσης Read_Adjectives

Αυτό που θέλουμε να κάνουμε είναι να χωρίσουμε τις λέξεις στα κελία που περιέχονται περισσότερες από 1 και να τις κωδικοποιήσουμε έτσι ώστε οι ίδιες λέξεις να έχουν τον ίδιο κωδικό.

Αρχικά διαβάζουμε με ένα for από τον πίνακα adjectivesCollection τις γραμμές του και αποθηκεύουμε σε μεταβλητές την λέξη τις πρώτης στήλης και το μήκος τις λέξης πλαίσιο κώδικα 11.

```
for (int i = 0; i < adjectivesCollection.length; i++) {
                        String word = adjectivesCollection[i][0];
                        int wordLength = word.length();
```

Πλαίσιο Κώδικα 11 – Ανάγνωση πίνακα adjectivesCollection

Στην συνέχεια δημιουργούμε το ID για την κάθε λέξη πλαίσιο κώδικα 12 από το πρώτο γράμμα τις λέξεις, το γράμμα Ε και ένα αύξων αριθμό για κάθε λέξη.

```
for (int j=0; j<adjectivesCollection[i].length; j++) {
	int ind=i+1;
	line1 = removeTone(String.valueOf(adjectivesCollection[i][0].charAt(0))).
toUpperCase()+"E"+ind+";";

	line2 = removeTone(String.valueOf(adjectivesCollection[i][0].charAt(0))).
toUpperCase()+"E"+ind+";";

	line3 = removeTone(String.valueOf(adjectivesCollection[i][0].charAt(0))).
toUpperCase()+"E"+ind+";";
```

Πλαίσιο Κώδικα 12 – Δημιουργία ID επιθέτων.

Έπειτα μετράμε τις καθέτους (/) μέσα στα κελία για την γραμμή που βρισκόμαστε στον πίνακα adjectivesCollection. Οι κάθετοι χρησιμοποιούνται στο αρχικό αρχείο που διαβάσαμε για να χωρίζουν διαφορετικές εκδοχές για την ίδια λέξη στην ίδια πτώση (πχ ακατάβρεκτου/ ακατάβρεχτου /ακαταβρέκτου) άρα βρίσκοντας τον μέγιστο αριθμό καθέτων που υπάρχουν σε μία γραμμή ξέρουμε πόσες γραμμές θα χρειαστούμε για να γράψουμε την λέξη πλαίσιο κώδικα 13. (Για την περίπτωση των επιθέτων ο μέγιστος αριθμός καθέτων για όλες τις λέξεις είναι 3).

```
try {
	count = StringUtils.countMatches(adjectivesCollection[i][j],"/");
} catch (Exception e) {
	adjectivesCollection[i][j] = "";
}
if (count>maxslashes)
	maxslashes=count;
```

Πλαίσιο Κώδικα 13 - Διαφορετικές εκδοχές για την ίδια λέξη επιθέτου

Στην συνεχεία ξέροντας των αριθμό των γραμμών που θα χρειαστούμε για να γράψουμε την λέξη επιλέγουμε με ένα case την κατάλληλη περίπτωση. Στο πλαίσιο κώδικα 14 παρουσιάζεται το case για την περίπτωση των δύο καθέτων. Παίρνουμε το περιεχόμενο του κάθε κελίου και βρίσκουμε την θέση της καθέτου και χωρίζει το String σε δυο μέρη το πρώτο πάει στο line1 και το δεύτερο στο line2 το line3 το συμπληρώνουμε μόνο με ερωτηματικά.

```
case 1:
	length = adjectivesCollection[i][j].length();
	pos1 = adjectivesCollection[i][j].indexOf("/");
	line1 = line1 + adjectivesCollection[i][j].substring(0, pos1)+ ";";
	line2 = line2+adjectivesCollection[i][j].substring(pos1+1, length)+ ";";
	line3=line3+";";
	break;
```

Πλαίσιο Κώδικα 14 – Επιλογή κατάλληλης περίπτωσης επιθέτου

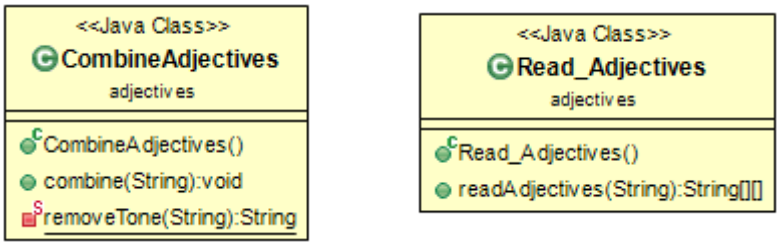
Στο επόμενο βήμα πλαίσιο κώδικα 15 προσθέτουμε στο ArrayList adjectivesList μόνο τις γραμμές που έχουν περιεχόμενο λέξεων. Στην περίπτωση των δύο καθέτων που περιγράψαμε παραπάνω το line11 και line22 θα είχαν μήκος μεγαλύτερο από 6 και θα προστίθετο στο adjectivesList ενώ το line33 το μήκος που θα είχε θα ήταν το μήκος του ID.

```
String line11=line1.replace(";","");
String line22=line2.replace(";","");
String line33=line3.replace(";","");
if (line11.length()>6)
       adjectivesList.add(line1);
if (line22.length()>6)
       adjectivesList.add(line2);
if (line33.length()>6)
       adjectivesList.add(line3);
```

Πλαίσιο Κώδικα 15 – Προσθήκη επιθέτων σε ArrayList

Τέλος το περιεχόμενο του adjectivesList γράφεται και αποθηκεύεται σε αρχείο.

Το διάγραμμά κλάσεων για την δημιουργία των επιθέτων παρουσιάζεται στην εικόνα 16



Εικόνα 16 - Διάγραμμά κλάσεων για την δημιουργία των επιθέτων

7.1.2.3 Ρήματα - Μετοχές

Για την κατασκευή των ρημάτων και των μετοχών καλούμε την μέθοδο verbs() και metoxes() αντίστοιχα από την κλάση Main.java. Η κατασκευή των ρημάτων και των μετοχών γίνεται από τις κλάσεις CombineVerbs.java και CombineMetoxes.java. Ο τρόπος που κατασκευάζονται είναι ο ίδιος με τα επίθετα με την μόνη διαφορά ότι έχουμε μεγαλύτερο αριθμό εκδοχών για την ίδια λέξη στην ίδια πτώση. Για την περίπτωση των ρημάτων έχουμε μέχρι και 9 διαφορετικές εκδοχές (αριθμός καθέτων σε ένα κελί). Ενώ για τις μετοχές μέχρι και 6.

Αφού ακολουθήσαμε την ίδια διαδικασία κατασκευής με τα επίθετα ας δούμε την περίπτωση για τα ρήματα στα οποία έχουμε 5 καθέτους για μια λέξη πλαίσιο κώδικα 16

```
case 4:
       length = verbsCollection[i][j].length();
       pos1 = verbsCollection[i][j].indexOf("/");
       pos2 = verbsCollection[i][j].indexOf("/",pos1+1);
       pos3 = verbsCollection[i][j].indexOf("/",pos2+1);
       pos4 = verbsCollection[i][j].indexOf("/",pos3+1);
       line1 = line1 + verbsCollection[i][j].substring(0, pos1)+ ";";
       line2 = line2 + verbsCollection[i][j].substring(pos1+1, pos2)+ ";";
       line3 = line3 + verbsCollection[i][j].substring(pos2+1, pos3)+ ";";
       line4 = line4 + verbsCollection[i][j].substring(pos3+1, pos4)+ ";";
       line5 = line5 + verbsCollection[i][j].substring(pos4+1, length)+ ";";
       line6=line6+";";
       line7=line7+";";
       line8=line8+";";
       line9=line9+";";
       break;
```

Πλαίσιο Κώδικα 16 – Κατασκευή ρήματων

Αρχικά βρίσκουμε το μέγεθος του string που υπάρχει στο κελί του πίνακα verbsCollection.

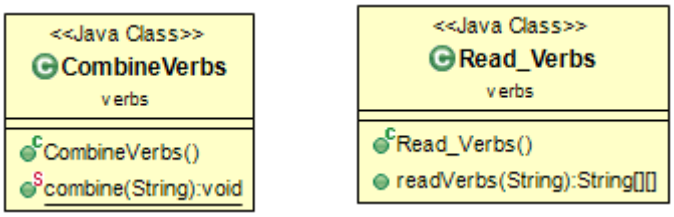
Στην συνέχεια βρίσκουμε τις θέσεις των τεσσάρων καθέτων μέσα στο string. Στην συνέχεια αρχίζουμε να δημιουργούμε τις γραμμές που θα γράψουμε στο αρχείο. Στην πρώτη γραμμή θα πάρουμε το κείμενο από την αρχή μέχρι την θέση της πρώτης καθέτου. Στην δεύτερη γραμμή από την θέση της πρώτης καθέτου μέχρι και την θέση της δεύτερης καθέτου. Με τον ίδιο τρόπο θα συνεχίσουμε μέχρι το τέλος του String. Τις υπόλοιπες γραμμές θα τις γεμίσουμε με ερωτηματικά.

Στην συνέχεια όπως και στα επίθετα θα αφαιρέσουμε τα ερωτηματικά και με βάση το μήκος του string θα αποφασίσουμε αν θα τα προσθέσουμε στο αρχείο πλαίσιο κώδικα 17.

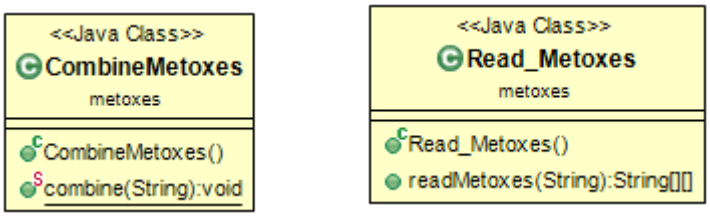
```
String line11=line1.replace(";","");
String line22=line2.replace(";","");
String line33=line3.replace(";","");
String line44=line4.replace(";","");
String line55=line5.replace(";","");
String line66=line6.replace(";","");
String line77=line7.replace(";","");
String line88=line8.replace(";","");
String line99=line9.replace(";","");

if (line11.length()>5)
        verbsList.add(line1);
if (line22.length()>5)
        verbsList.add(line2);
if (line33.length()>5)
        verbsList.add(line3);
if (line44.length()>5)
        verbsList.add(line4);
if (line55.length()>5)
        verbsList.add(line5);
if (line66.length()>5)
        verbsList.add(line6);
if (line77.length()>5)
        verbsList.add(line7);
if (line88.length()>5)
        verbsList.add(line8);
if (line99.length()>5)
        verbsList.add(line9);
```

Πλαίσιο Κώδικα 17 – Προσθήκη ρημάτων σε ArrayList

Τα διάγραμμά κλάσεων για την δημιουργία των ρημάτων και των μετοχών παρουσιάζεται στις εικόνες 17, 18.



Εικόνα 17 Διάγραμμά κλάσεων για την δημιουργία των ρημάτων



Εικόνα 18 Διάγραμμά κλάσεων για την δημιουργία των μετοχών

7.1.2.4 Μετοχές ενεργητικής φωνής

Για την κατασκευή των μετοχών ενεργητικής φωνής αναπτύξαμε την κλάση GenEMetoxes.java η οποία διαβάζει από αρχείο όλα τα ρήματα (το αρχείο έχει κατασκευαστεί προηγούμενος με την κλάση CombineVerbs.java)

Οι μετοχές ενεργητικής φωνής έχουν δύο διαφορετικές καταλήξεις (-οντας και -ώντας). Για την δημιουργία των μετοχών ελέγχουμε αν στο πρώτο πρόσωπο ενεστώτα τονίζεται το 'ω' αν τονίζεται παίρνει κατάληξη «–ώντας» διαφορετικά κατάληξη «– οντας». Πλαίσιο κώδικα 18.

```
if (lastc=='ω')
        metkat="οντας";
else
        metkat="ώντας";
metoxi=nextline[i].substring(0, nextline[i].length()-1)+metkat;
```

Πλαίσιο Κώδικα 18 – κατασκευή μετοχών ενεργητικής φωνής

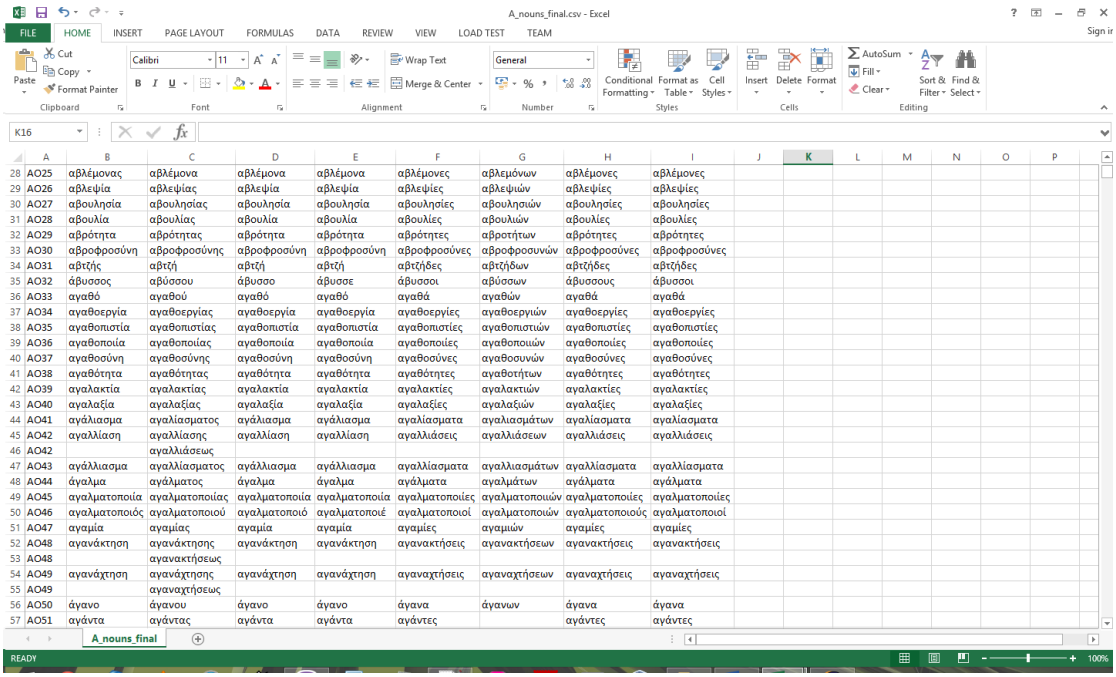
Στην συνέχεια αφαιρούμε τον τελευταίο χαρακτήρα από το string και προσθέτουμε την κατάληξη.

7.1.2.5 Υπόλοιπες λέξεις

Για την κατασκευή των υπολοίπων λέξεων (άκλητα, αριθμοί, άρθρα και συντομογραφίες) κάναμε απλή εισαγωγή των λέξεων σε αρχεία «.csv» και τα επεξεργαστήκαμε με την εφαρμογή υπολογιστικών φύλλων (excel)

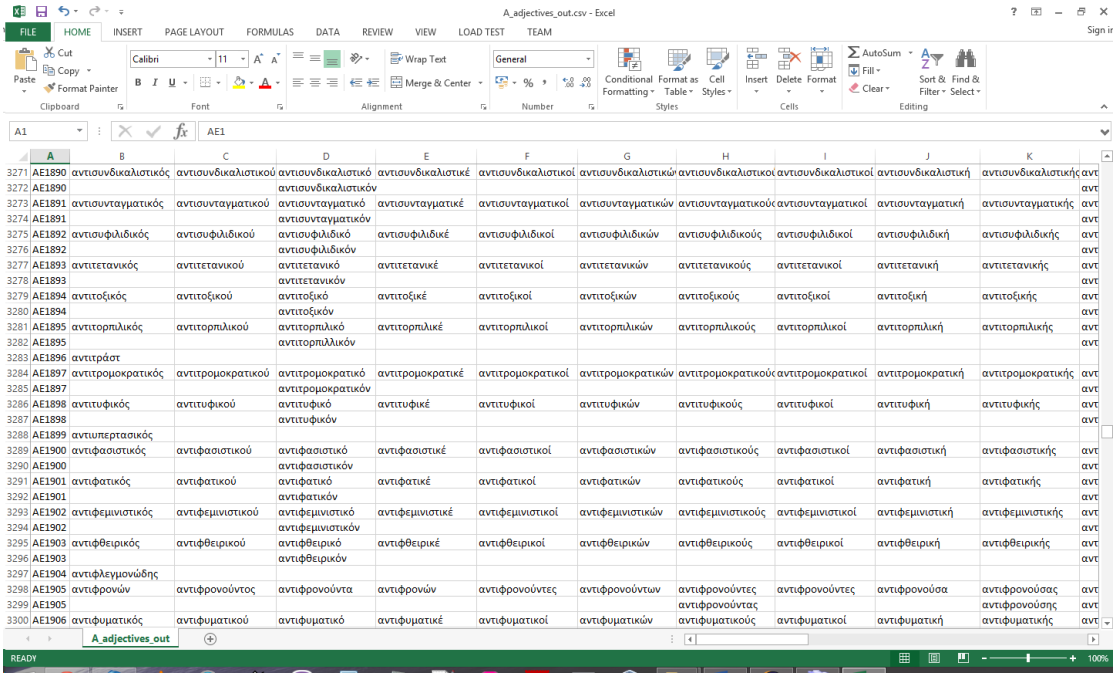
7.1.3 Αρχεία λεξικού – Λεξικό MySQL

Το αποτέλεσμα των παραπάνω κλάσεων είναι αρχεία «.csv» τα οποία περιέχουν τις λέξεις που δημιουργήσαμε. Η μορφή αυτών των αρχείων για τα Ουσιαστικά, επίθετα, ρήματα και μετοχές με το πρόγραμμα MS Excel φαίνεται στις εικόνες 19, 20, 21 και 22.



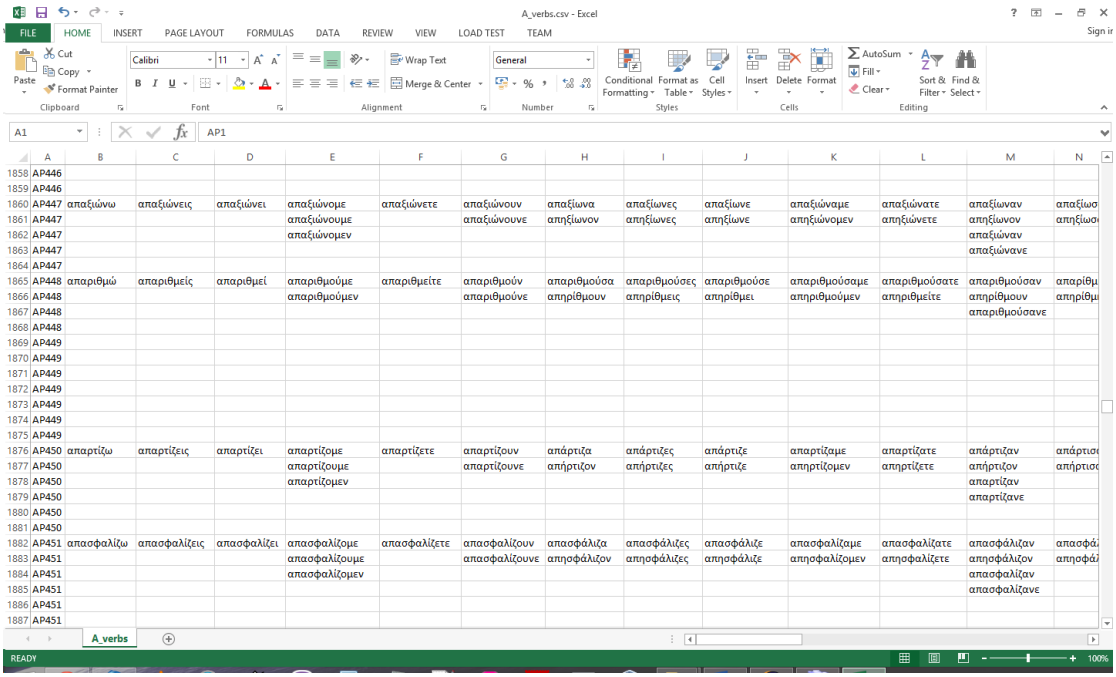
	A	B	C	D	E	F	G	H	I
28	AO25	αβλέμονας	αβλέμονα	αβλέμονα	αβλέμονα	αβλέμονες	αβλεμόνων	αβλέμονες	αβλέμονες
29	AO26	αβλεψία	αβλεψίας	αβλεψία	αβλεψία	αβλεψίες	αβλεψιών	αβλεψίες	αβλεψίες
30	AO27	αβουλησία	αβουλησίας	αβουλησία	αβουλησία	αβουλησίες	αβουλησιών	αβουλησίες	αβουλησίες
31	AO28	αβουλία	αβουλίας	αβουλία	αβουλία	αβουλίες	αβουλιών	αβουλίες	αβουλίες
32	AO29	αβρότητα	αβρότητας	αβρότητα	αβρότητα	αβρότητες	αβροτήτων	αβρότητες	αβρότητες
33	AO30	αβροφροσύνη	αβροφροσύνης	αβροφροσύνη	αβροφροσύνη	αβροφροσύνες	αβροφροσυνών	αβροφροσύνες	αβροφροσύνες
34	AO31	αβτζής	αβτζή	αβτζή	αβτζή	αβτζήδες	αβτζήδων	αβτζήδες	αβτζήδες
35	AO32	άβυσσος	αβύσσου	άβυσσο	άβυσσε	άβυσσοι	αβύσσων	άβυσσους	άβυσσοι
36	AO33	αγαθό	αγαθού	αγαθό	αγαθό	αγαθά	αγαθών	αγαθά	αγαθά
37	AO34	αγαθοεργία	αγαθοεργίας	αγαθοεργία	αγαθοεργία	αγαθοεργίες	αγαθοεργιών	αγαθοεργίες	αγαθοεργίες
38	AO35	αγαθοπιστία	αγαθοπιστίας	αγαθοπιστία	αγαθοπιστία	αγαθοπιστίες	αγαθοπιστιών	αγαθοπιστίες	αγαθοπιστίες
39	AO36	αγαθοποιία	αγαθοποιίας	αγαθοποιία	αγαθοποιία	αγαθοποιίες	αγαθοποιών	αγαθοποιίες	αγαθοποιίες
40	AO37	αγαθοσύνη	αγαθοσύνης	αγαθοσύνη	αγαθοσύνη	αγαθοσύνες	αγαθοσυνών	αγαθοσύνες	αγαθοσύνες
41	AO38	αγαθότητα	αγαθότητας	αγαθότητα	αγαθότητα	αγαθότητες	αγαθοτήτων	αγαθότητες	αγαθότητες
42	AO39	αγαλακτία	αγαλακτίας	αγαλακτία	αγαλακτία	αγαλακτίες	αγαλακτιών	αγαλακτίες	αγαλακτίες
43	AO40	αγαλαξία	αγαλαξίας	αγαλαξία	αγαλαξία	αγαλαξίες	αγαλαξιών	αγαλαξίες	αγαλαξίες
44	AO41	αγάλιασμα	αγαλίσματος	αγάλιασμα	αγάλιασμα	αγαλίσματα	αγαλιασμάτων	αγαλίσματα	αγαλίσματα
45	AO42	αγαλλίαση	αγαλλίασης	αγαλλίαση	αγαλλίαση	αγαλλιάσεις	αγαλλιάσεων	αγαλλιάσεις	αγαλλιάσεις
46	AO42		αγαλλιάσεως						
47	AO43	αγάλλιασμα	αγαλλίασματος	αγάλλιασμα	αγάλλιασμα	αγαλλίσματα	αγαλλιασμάτων	αγαλλίσματα	αγαλλίσματα
48	AO44	άγαλμα	αγάλματος	άγαλμα	άγαλμα	αγάλματα	αγαλμάτων	αγάλματα	αγάλματα
49	AO45	αγαλματοποιία	αγαλματοποιίας	αγαλματοποιία	αγαλματοποιία	αγαλματοποιίες	αγαλματοποιών	αγαλματοποιίες	αγαλματοποιίες
50	AO46	αγαλματοποιός	αγαλματοποιού	αγαλματοποιό	αγαλματοποιέ	αγαλματοποιοί	αγαλματοποιών	αγαλματοποιούς	αγαλματοποιοί
51	AO47	αγαμία	αγαμίας	αγαμία	αγαμία	αγαμίες	αγαμιών	αγαμίες	αγαμίες
52	AO48	αγανάκτηση	αγανάκτησης	αγανάκτηση	αγανάκτηση	αγανακτήσεις	αγανακτήσεων	αγανακτήσεις	αγανακτήσεις
53	AO48		αγανακτήσεως						
54	AO49	αγανάχτηση	αγανάχτησης	αγανάχτηση	αγανάχτηση	αγαναχτήσεις	αγαναχτήσεων	αγαναχτήσεις	αγαναχτήσεις
55	AO49		αγαναχτήσεως						
56	AO50	άγανο	άγανου	άγανο	άγανο	άγανα	άγανων	άγανα	άγανα
57	AO51	αγάντα	αγάντας	αγάντα	αγάντα	αγάντες		αγάντες	αγάντες

Εικόνα 19 Αρχείο .csv με τα ουσιαστικά



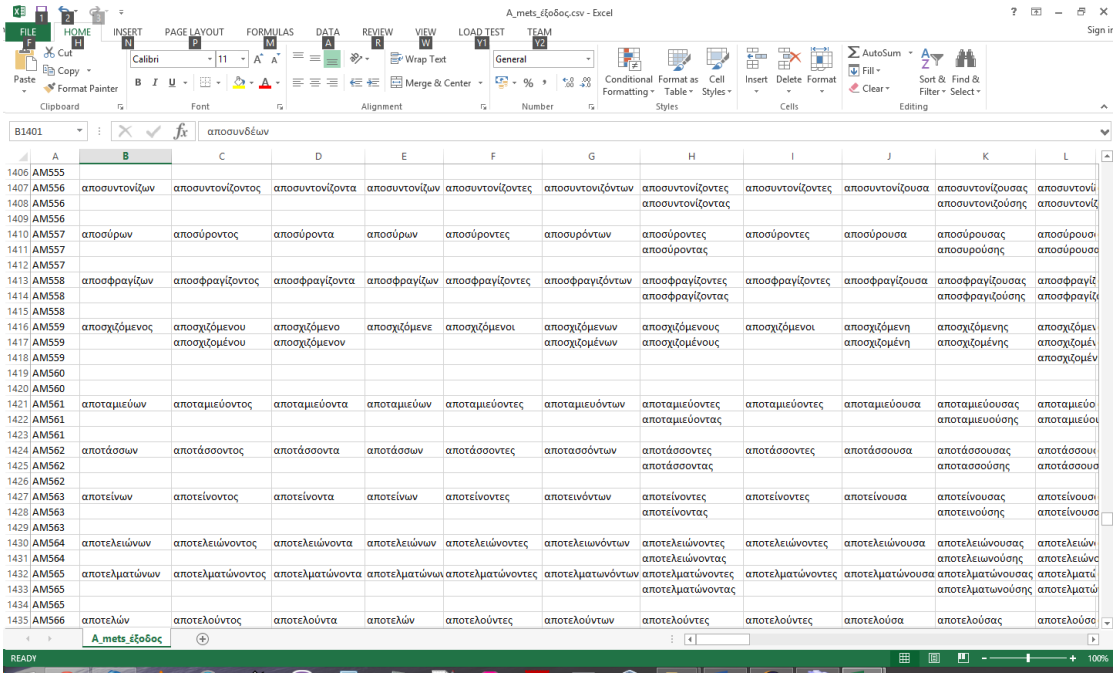
	A	B	C	D	E	F	G	H	I	J	K
3271	AE1890	αντισυνδικαλιστικός	αντισυνδικαλιστικού	αντισυνδικαλιστικό	αντισυνδικαλιστικέ	αντισυνδικαλιστικοί	αντισυνδικαλιστικών	αντισυνδικαλιστικούς	αντισυνδικαλιστικοί	αντισυνδικαλιστική	αντισυνδικαλιστικής
3272	AE1890			αντισυνδικαλιστικόν							
3273	AE1891	αντισυνταγματικός	αντισυνταγματικού	αντισυνταγματικό	αντισυνταγματικέ	αντισυνταγματικοί	αντισυνταγματικών	αντισυνταγματικούς	αντισυνταγματικοί	αντισυνταγματική	αντισυνταγματικής
3274	AE1891			αντισυνταγματικόν							
3275	AE1892	αντισυφιλιδικός	αντισυφιλιδικού	αντισυφιλιδικό	αντισυφιλιδικέ	αντισυφιλιδικοί	αντισυφιλιδικών	αντισυφιλιδικούς	αντισυφιλιδικοί	αντισυφιλιδική	αντισυφιλιδικής
3276	AE1892			αντισυφιλιδικόν							
3277	AE1893	αντιτετανικός	αντιτετανικού	αντιτετανικό	αντιτετανικέ	αντιτετανικοί	αντιτετανικών	αντιτετανικούς	αντιτετανικοί	αντιτετανική	αντιτετανικής
3278	AE1893			αντιτετανικόν							
3279	AE1894	αντιτοξικός	αντιτοξικού	αντιτοξικό	αντιτοξικέ	αντιτοξικοί	αντιτοξικών	αντιτοξικούς	αντιτοξικοί	αντιτοξική	αντιτοξικής
3280	AE1894			αντιτοξικόν							
3281	AE1895	αντιτορπιλικός	αντιτορπιλικού	αντιτορπιλικό	αντιτορπιλικέ	αντιτορπιλικοί	αντιτορπιλικών	αντιτορπιλικούς	αντιτορπιλικοί	αντιτορπιλική	αντιτορπιλικής
3282	AE1895			αντιτορπιλλικόν							
3283	AE1896	αντιτράστ									
3284	AE1897	αντιτρομοκρατικός	αντιτρομοκρατικού	αντιτρομοκρατικό	αντιτρομοκρατικέ	αντιτρομοκρατικοί	αντιτρομοκρατικών	αντιτρομοκρατικούς	αντιτρομοκρατικοί	αντιτρομοκρατική	αντιτρομοκρατικής
3285	AE1897			αντιτρομοκρατικόν							
3286	AE1898	αντιτυφικός	αντιτυφικού	αντιτυφικό	αντιτυφικέ	αντιτυφικοί	αντιτυφικών	αντιτυφικούς	αντιτυφικοί	αντιτυφική	αντιτυφικής
3287	AE1898			αντιτυφικόν							
3288	AE1899	αντιυπερτασικός									
3289	AE1900	αντιφασιστικός	αντιφασιστικού	αντιφασιστικό	αντιφασιστικέ	αντιφασιστικοί	αντιφασιστικών	αντιφασιστικούς	αντιφασιστικοί	αντιφασιστική	αντιφασιστικής
3290	AE1900			αντιφασιστικόν							
3291	AE1901	αντιφατικός	αντιφατικού	αντιφατικό	αντιφατικέ	αντιφατικοί	αντιφατικών	αντιφατικούς	αντιφατικοί	αντιφατική	αντιφατικής
3292	AE1901			αντιφατικόν							
3293	AE1902	αντιφεμινιστικός	αντιφεμινιστικού	αντιφεμινιστικό	αντιφεμινιστικέ	αντιφεμινιστικοί	αντιφεμινιστικών	αντιφεμινιστικούς	αντιφεμινιστικοί	αντιφεμινιστική	αντιφεμινιστικής
3294	AE1902			αντιφεμινιστικόν							
3295	AE1903	αντιφθειρικός	αντιφθειρικού	αντιφθειρικό	αντιφθειρικέ	αντιφθειρικοί	αντιφθειρικών	αντιφθειρικούς	αντιφθειρικοί	αντιφθειρική	αντιφθειρικής
3296	AE1903			αντιφθειρικόν							
3297	AE1904	αντιφλεγμονώδης									
3298	AE1905	αντιφρονών	αντιφρονούντος	αντιφρονούντα	αντιφρονών	αντιφρονούντες	αντιφρονούντων	αντιφρονούντες	αντιφρονούντες	αντιφρονούσα	αντιφρονούσας
3299	AE1905							αντιφρονούντας		αντιφρονούσης	
3300	AE1906	αντιφυματικός	αντιφυματικού	αντιφυματικό	αντιφυματικέ	αντιφυματικοί	αντιφυματικών	αντιφυματικούς	αντιφυματικοί	αντιφυματική	αντιφυματικής

Εικόνα 20 Αρχείο .csv με τα επίθετα



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1858	AP446													
1859	AP446													
1860	AP447	απαξιώνω	απαξιώνεις	απαξιώνει	απαξιώνομε	απαξιώνετε	απαξιώνουν	απαξίωνα	απαξίωνες	απαξίωνε	απαξιώναμε	απαξιώνατε	απαξίωναν	απαξίωσ
1861	AP447				απαξιώνουμε		απαξιώνουνε	απηξίωνον	απηξίωνες	απηξίωνε	απηξιώνομεν	απηξιώνετε	απηξίωνον	απηξίωσ
1862	AP447				απαξιώνομεν								απαξιώναν	
1863	AP447												απαξιώνανε	
1864	AP447													
1865	AP448	απαριθμώ	απαριθμείς	απαριθμεί	απαριθμούμε	απαριθμείτε	απαριθμούν	απαριθμούσα	απαριθμούσες	απαριθμούσε	απαριθμούσαμε	απαριθμούσατε	απαριθμούσαν	απαρίθμ
1866	AP448				απαριθμούμεν		απαριθμούνε	απηρίθμουν	απηρίθμεις	απηρίθμει	απηριθμούμεν	απηριθμείτε	απηρίθμουν	απηρίθμ
1867	AP448												απαριθμούσανε	
1868	AP448													
1869	AP449													
1870	AP449													
1871	AP449													
1872	AP449													
1873	AP449													
1874	AP449													
1875	AP449													
1876	AP450	απαρτίζω	απαρτίζεις	απαρτίζει	απαρτίζομε	απαρτίζετε	απαρτίζουν	απάρτιζα	απάρτιζες	απάρτιζε	απαρτίζαμε	απαρτίζατε	απάρτιζαν	απάρτισ
1877	AP450				απαρτίζουμε		απαρτίζουνε	απήρτιζον	απήρτιζες	απήρτιζε	απηρτίζομεν	απηρτίζετε	απήρτιζον	απήρτισ
1878	AP450				απαρτίζομεν								απαρτίζαν	
1879	AP450												απαρτίζανε	
1880	AP450													
1881	AP450													
1882	AP451	απασφαλίζω	απασφαλίζεις	απασφαλίζει	απασφαλίζομε	απασφαλίζετε	απασφαλίζουν	απασφάλιζα	απασφάλιζες	απασφάλιζε	απασφαλίζαμε	απασφαλίζατε	απασφάλιζαν	απασφά
1883	AP451				απασφαλίζουμε		απασφαλίζουνε	απησφάλιζον	απησφάλιζες	απησφάλιζε	απησφαλίζομεν	απησφαλίζετε	απησφάλιζον	απησφά
1884	AP451				απασφαλίζομεν								απασφαλίζαν	
1885	AP451												απασφαλίζανε	
1886	AP451													
1887	AP451													

Εικόνα 21 Αρχείο .csv με τα ρήματα



	A	B	C	D	E	F	G	H	I	J	K	L
1406	AM555											
1407	AM556	αποσυντονίζων	αποσυντονίζοντος	αποσυντονίζοντα	αποσυντονίζων	αποσυντονίζοντες	αποσυντονιζόντων	αποσυντονίζοντες	αποσυντονίζοντες	αποσυντονίζουσα	αποσυντονίζουσας	αποσυντονί
1408	AM556							αποσυντονίζοντας			αποσυντονιζούσης	αποσυντονίζ
1409	AM556											
1410	AM557	αποσύρων	αποσύροντος	αποσύροντα	αποσύρων	αποσύροντες	αποσυρόντων	αποσύροντες	αποσύροντες	αποσύρουσα	αποσύρουσας	αποσύρουσ
1411	AM557							αποσύροντας			αποσυρούσης	αποσύρουσα
1412	AM557											
1413	AM558	αποσφραγίζων	αποσφραγίζοντος	αποσφραγίζοντα	αποσφραγίζων	αποσφραγίζοντες	αποσφραγιζόντων	αποσφραγίζοντες	αποσφραγίζοντες	αποσφραγίζουσα	αποσφραγίζουσας	αποσφραγίζ
1414	AM558							αποσφραγίζοντας			αποσφραγιζούσης	αποσφραγίζ
1415	AM558											
1416	AM559	αποσχιζόμενος	αποσχιζόμενου	αποσχιζόμενο	αποσχιζόμενε	αποσχιζόμενοι	αποσχιζόμενων	αποσχιζόμενους	αποσχιζόμενοι	αποσχιζόμενη	αποσχιζόμενης	αποσχιζόμεν
1417	AM559		αποσχιζομένου	αποσχιζόμενον			αποσχιζομένων	αποσχιζομένους		αποσχιζομένη	αποσχιζομένης	αποσχιζομέν
1418	AM559											αποσχιζομέν
1419	AM560											
1420	AM560											
1421	AM561	αποταμιεύων	αποταμιεύοντος	αποταμιεύοντα	αποταμιεύων	αποταμιεύοντες	αποταμιευόντων	αποταμιεύοντες	αποταμιεύοντες	αποταμιεύουσα	αποταμιεύουσας	αποταμιεύο
1422	AM561							αποταμιεύοντας			αποταμιευούσης	αποταμιεύο
1423	AM561											
1424	AM562	αποτάσσων	αποτάσσοντος	αποτάσσοντα	αποτάσσων	αποτάσσοντες	αποτασσόντων	αποτάσσοντες	αποτάσσοντες	αποτάσσουσα	αποτάσσουσας	αποτάσσου
1425	AM562							αποτάσσοντας			αποτασσούσης	αποτάσσουσ
1426	AM562											
1427	AM563	αποτείνων	αποτείνοντος	αποτείνοντα	αποτείνων	αποτείνοντες	αποτεινόντων	αποτείνοντες	αποτείνοντες	αποτείνουσα	αποτείνουσας	αποτείνου
1428	AM563							αποτείνοντας			αποτεινούσης	αποτείνουσ
1429	AM563											
1430	AM564	αποτελειώνων	αποτελειώνοντος	αποτελειώνοντα	αποτελειώνων	αποτελειώνοντες	αποτελειωνόντων	αποτελειώνοντες	αποτελειώνοντες	αποτελειώνουσα	αποτελειώνουσας	αποτελειών
1431	AM564							αποτελειώνοντας			αποτελειωνούσης	αποτελειώνο
1432	AM565	αποτελματώνων	αποτελματώνοντος	αποτελματώνοντα	αποτελματώνων	αποτελματώνοντες	αποτελματωνόντων	αποτελματώνοντες	αποτελματώνοντες	αποτελματώνουσα	αποτελματώνουσας	αποτελματώ
1433	AM565							αποτελματώνοντας			αποτελματωνούσης	αποτελματώ
1434	AM565											
1435	AM566	αποτελών	αποτελούντος	αποτελούντα	αποτελών	αποτελούντες	αποτελούντων	αποτελούντες	αποτελούντες	αποτελούσα	αποτελούσας	αποτελούσα

Εικόνα 22 Αρχείο .csv με τις μετοχές

Για την εγγραφή των λέξεων στους πίνακες της βάση δεδομένων MySQL κάναμε εισαγωγή αυτών των αρχείων στην βάση.

7.1.4 Δημιουργία λεξικού σε MongoDB

Η δημιουργία του λεξικού στην MongoDB γίνεται μεταφέροντας τις εγγραφές της MySQL και προσθέτοντας κάποιες επιπλέον πληροφορίες.

Γίνεται με το project GrDictionaryDBs το οποίο περιέχει κλάσεις για την κάθε κατηγορία λέξεων (Adjectives.java, Nouns.java κτλ). Επίσης περιέχει δυο κλάσεις για την σύνδεση με τις βάσης δεδομένων (ConnectMongoDB.java για MongoDB και DBUtil.java για MySQL). Η εκτέλεση του προγράμματος γίνεται από την κλάση Main.java. Η κλάση Main.java καλεί τις μεθόδους insert από τις αντίστοιχες κλάσεις για να εισάγει τις λέξεις στην βάση.

Στην μέθοδο insertNouns() από την κλάση Nouns.java πλαίσιο κώδικα 19 αρχικά συνδέεται με την βάση δεδομένων MySQL μέσω της κλάσης DBUtil και εκτελεί ερώτημα που επιστρέφει όλα τα δεδομένα του πίνακα nouns της βάσης.

```
Connection conn = DBUtil.getConnection();
Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
ResultSet rs = stmt.executeQuery("SELECT * FROM nouns");
```

Πλαίσιο Κώδικα 19 - σύνδεση με την βάση δεδομένων MySQL

Στην συνέχεια καλεί την μέθοδο insertMongo της κλάσης ConnectMongoDB με ορίσματα τα δεδομένα που του επέστρεψε η βάση πλαίσιο κώδικα 20. Τα ορίσματα είναι το ID της κάθε λέξης, η λέξη στην ονομαστική ενικού και η λέξη στην πτώση που είμαστε κάθε φορά. Η κλήση της μεθόδου θα γίνεται για όλες τις εγγραφές που επέστρεψε το ερώτημα στην MySQL.

```
while (rs.next()) {
        ConnectMongoDB.insertMongo(rs.getString("ID"), rs.getString("Ονο_Εν"),
rs.getString("Ονο_Εν"), 1);
        ConnectMongoDB.insertMongo(rs.getString("ID"), rs.getString("Ονο_Εν"),
rs.getString("Γεν_Εν"), 2);
        ConnectMongoDB.insertMongo(rs.getString("ID"), rs.getString("Ονο_Εν"),
rs.getString("Αιτ_Εν"), 3);
        ConnectMongoDB.insertMongo(rs.getString("ID"), rs.getString("Ονο_Εν"),
rs.getString("Κλ_Εν"), 4);
        ConnectMongoDB.insertMongo(rs.getString("ID"), rs.getString("Ονο_Εν"),
rs.getString("Ονο_Πλ"), 5);
        ConnectMongoDB.insertMongo(rs.getString("ID"), rs.getString("Ονο_Εν"),
rs.getString("Γεν_Πλ"), 6);
        ConnectMongoDB.insertMongo(rs.getString("ID"), rs.getString("Ονο_Εν"),
rs.getString("Αιτ_Πλ"), 7);
        ConnectMongoDB.insertMongo(rs.getString("ID"), rs.getString("Ονο_Εν"),
rs.getString( "Κλ_Πλ").replace("\r", ""), 8);
```

Πλαίσιο Κώδικα 20 – Εισαγωγή ουσιαστικών στην MongoDB

Η κλάση ConnectMongoDB περιέχει την μέθοδο insertMongo που αρχικά δημιουργεί μια σύνδεση με την βάση πλαίσιο κώδικα 21

```
MongoClient mongoClient = new MongoClient( "localhost" , 27017 );
              DB db = mongoClient.getDB("gr_lem_dictionary");
              String collection = "grWords";
              DBCollection coll = db.getCollection(collection);
```

Πλαίσιο Κώδικα 21 - σύνδεση με την βάση δεδομένων MongoDB

Στην συνέχεια δημιουργεί τις διάφορες μορφές της λέξης πριν την εισάγει στην βάση πλαίσιο κώδικα 22

```
word = word.replaceAll("\\s","");
String noToneWord = removeTone(word);
startWord = removeTone(startWord);
```

Πλαίσιο Κώδικα 22 - Δημιουργία διαφόρων μορφές της λέξης

Αφαίρει όλους τους ιδικούς χαρακτήρες που μπορεί να υπάρχουν στην λέξη (κενά, αλλαγή γραμμής κτλ.) και δημιουργεί τις λέξεις χωρίς τόνους.

Έπειτα δημιουργεί στιγμιότυπο από το αντικείμενο BasicDBObject που περιέχει τα πεδία για την εισαγωγή στην βάση και τα εισάγει πλαίσιο κώδικα 23.

```
BasicDBObject    doc    =    new    BasicDBObject("wordID",    id).append("ID",
randomNum(id)).append("startWord",                    startWord).append("word",
word).append("noToneWord", noToneWord).append("pos", pos);
coll.insert(doc);
mongoClient.close();
```

Πλαίσιο Κώδικα 23 - Δημιουργεί στιγμιότυπου από το αντικείμενο BasicDBObject για εισαγωγή στην MongoDB

Η μέθοδος randomNum πλαίσιο κώδικα 24 επιστρέφει ένα διάνυσμα 100 τυχαίων αριθμών γκαουσιανής κατανομής. Κάνει έλεγχο αν το id της λέξης είναι ίδιο με το προηγούμενο wordID, αν είναι τότε επιστρέφει το προηγούμενο διάνυσμα. Διαφορετικά δημιουργεί νέο. Με αυτόν τον τρόπο οι ίδιες λέξεις έχουν ίδιο id.

```
private static double[] randomNum(String id) {
              if (id.equals(wordID)) {
                     return randomID;
              }
              else {
                     Random randomGenerator = new Random();

                     for (int i = 0; i < randomID.length; i++){
                            double randomNum = randomGenerator.nextGaussian();
                            randomID[i] = randomNum;
                     }
                     return randomID;
              }
       }
```

Πλαίσιο Κώδικα 24 – Δημιουργία διανύσματος 100 τυχαίων αριθμών γκαουσιανής κατανομής

Τέλος αφού εισαχθεί η εγγραφή στην βάση δεδομένων κλείνει η σύνδεση με τον server της βάσης.

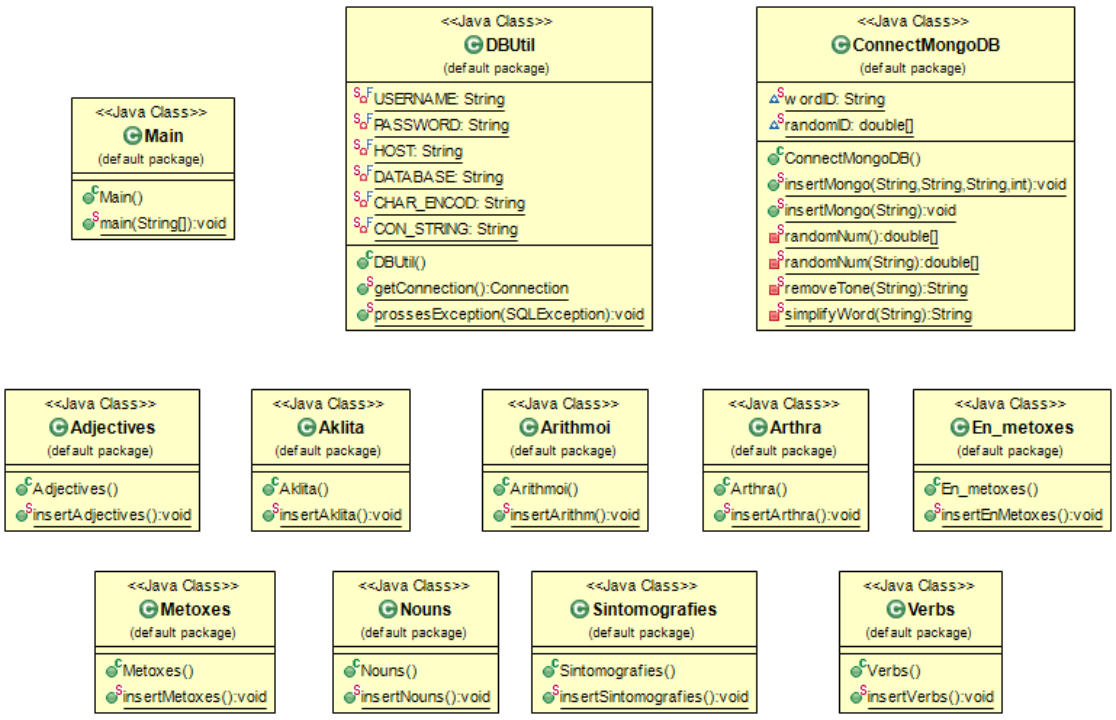
Για τα υπόλοιπά είδη λέξεων γίνεται η ίδια διαδικασία με την μόνη διαφορά πώς για κάθε είδος λέξης καλείτε η αντίστοιχη κλάση (Adjectives για τα επίθετα, Verbs για τα ρήματα

κτλ). Σε κάθε κλάση αυτό που αλλάζει είναι το πλήθος που θα καλέσουμε την μέθοδο ConnectMongoDB.insertMongo() ανάλογα με τις κλιτικές περιπτώσεις του κάθε είδους λέξης. (72 για επίθετα, 76 για τα ρήματα, 120 για μετοχές) στο πλαίσιο κώδικα 25 παρουσιάζεται ένα μέρος για την περίπτωση των μετοχών.

```
while (rs.next()) {
ConnectMongoDB.insertMongo(rs.getString("ID"),
rs.getString("Ενερ_Ενε_Αρσ_Ενι_ονο"), rs.getString("Ενερ_Ενε_Αρσ_Ενι_ονο"),1);
ConnectMongoDB.insertMongo(rs.getString("ID"),
rs.getString("Ενερ_Ενε_Αρσ_Ενι_ονο"), rs.getString("Ενερ_Ενε_Αρσ_Ενι_γεν"),2);
ConnectMongoDB.insertMongo(rs.getString("ID"),
rs.getString("Ενερ_Ενε_Αρσ_Ενι_ονο"), rs.getString("Ενερ_Ενε_Αρσ_Ενι_αιτ"),3);
.
.
.
ConnectMongoDB.insertMongo(rs.getString("ID"),
rs.getString("Ενερ_Ενε_Αρσ_Ενι_ονο"), rs.getString("Παθη_Παρ_Ουδ_Πλη_γεν"),118);
ConnectMongoDB.insertMongo(rs.getString("ID"),
rs.getString("Ενερ_Ενε_Αρσ_Ενι_ονο"), rs.getString("Παθη_Παρ_Ουδ_Πλη_αιτ"),119);
ConnectMongoDB.insertMongo(rs.getString("ID"),
rs.getString("Ενερ_Ενε_Αρσ_Ενι_ονο"), rs.getString("Παθη_Παρ_Ουδ_Πλη_κλη"),120);
                    }
```

Πλαίσιο Κώδικα 25 - Εισαγωγή μετοχών στην MongoDB

Το διάγραμμά κλάσεων του project GrDictionaryDBs παρουσιάζεται στην εικόνα 23



Εικόνα 23 Διάγραμμά κλάσεων του project GrDictionaryDBs

7.1.5 Κατασκευή Συλλογής σχολίων

Η κατασκευή του dataset γίνεται με το project GrDictionaryDataset το οποίο διαβάζει τα xml αρχεία που περιέχουν τα σχόλια των χρηστών και τα χωρίζει σε προτάσεις.

Η εκτέλεση του project γίνεται από την κλάση Main.java η οποία περιέχει πίνακα με τα αρχεία από τα οποία θα διαβάσει τα σχόλια. Στη συνέχεια δημιουργεί στιγμιότυπο από την κλάση GetReviews από την οποία καλεί την μέθοδο getComments με ορίσμα τα αρχεία .xml. πλαίσιο κώδικα 26

```
String[] reviews = {"reviews001_100", "reviews101_200", "reviews201_300",
"reviews301_400", "reviews401_500"};
		GetReviews getReviews = new GetReviews();
		for (int i=0; i<reviews.length; i++) {
			getReviews.getComments(reviews[i]);
		}
```

Πλαίσιο Κώδικα 26 – Ανάγνωση αρχείων xml με σχόλια

Η κλάση GetReviews χρησιμοποιεί την κλάση ReadXMLFile για να διαβάσει κείμενο από τα επιθυμητά tags από τα αρχεία .xml. Η κλάση ReadXMLFile πλαίσιο κώδικα 27 δέχεται δύο ορίσματα το όνομα του αρχείου και το όνομα του tag από το οποίο θα διαβάσει το κείμενο.

```
public ReadXMLFile(String review, String tagName) {
	try {
		File fXmlFile = new File("data/" + review +".xml");
		DocumentBuilderFactory dbFactory =
DocumenBuilderFactory.newInstance();
		DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
		Document doc = dBuilder.parse(fXmlFile);
		doc.getDocumentElement().normalize();
		nList = doc.getElementsByTagName(tagName);

	} catch (Exception e) {
		System.err.println(e.getMessage());
		System.err.println("Can't return comments list");
	}
```

Πλαίσιο Κώδικα 27 – Ανάγνωση από xml tags

Η μέθοδος getData() της κλάσης ReadXMLFile δέχεται σαν όρισμα το όνομα του tag και επιστρέφει σε ArrayList τα κείμενα που περιείχε το tag πλαίσιο κώδικα 28

```
ArrayList<String> dataList = new ArrayList<String>();
	for (int temp = 0; temp < nList.getLength(); temp++) {
		Node nNode = nList.item(temp);
		if (nNode.getNodeType() == Node.ELEMENT_NODE) {
			Element eElement = (Element) nNode;
	dataList.add(eElement.getElementsByTagName(tagName).item(0).getTex
tContent());
		}
	}
return dataList;
```

Πλαίσιο Κώδικα 28 - getData()

Στη συνέχεια η κλάση GetReviews πλαίσιο κώδικα 29 αποθηκεύει σε δύο ArrayList τα σχολεία και την βαθμολογία.

```
ArrayList<String> commentsList = readReview.getData("Comment");
ArrayList<String> ratingList = readReview.getData("Rating");

String[] commentsArray = commentsList.toArray(new String[commentsList.size()]);
	for (int i=0; i<commentsArray.length; i++) {
		commentsArray[i] = commentsArray[i].replace("\n", "");
		commentsArray[i] = commentsArray[i].replace("\r", "");
		commentsArray[i] = commentsArray[i].replace(";", "?");
	}
String[] ratingArray = ratingList.toArray(new String[ratingList.size()]);
for (int i=0; i<ratingArray.length; i++) {
	ratingArray[i] = ratingArray[i].replace("\n", "");
	ratingArray[i] = ratingArray[i].replace("\r", "");
}
String[] outputArray = new String[commentsList.size()];

for (int i=0; i<outputArray.length; i++) {
	outputArray[i] = commentsArray[i] + ";" + ratingArray[i];
}
```

Πλαίσιο Κώδικα 29 – Κλάση GetReviews αποθηκεύει σε ArrayList τα σχολεία και την βαθμολογία από σχόλια σε xml.

Επεξεργάζεται τα δύο ArrayList για να αφαιρέσει ιδικούς χαρακτήρες που μπορεί να περιέχουν. Δημιουργεί πίνακα outputArray στον οποίο προσθέτει σε γραμμές το σχόλιο και την βαθμολογία χωρισμένα με κόμμα.

Για τον διαχωρισμό των σχολείων σε προτάσεις χρησιμοποιούμε την κλάση ReadDataSet πλαίσιο κώδικα 30. Η κλάση ReadDataSet διαβάζει αρχείο που περιέχει σχόλια και με βάση τα σημεία στίξεις (τελείες, θαυμαστικά κτλ.) χωρίζει τα σχόλια σε προτάσεις και επιστρέφει τις προτάσεις σε ArrayList

```
String[] commentsArray = commentsList.toArray(new
String[commentsList.size()]);
ArrayList<String> sentencesList = new ArrayList<>();
 for (int i=0; i<commentsArray.length; i++) {
	commentsArray[i] = commentsArray[i].replace("!", ".");
	String[] sentences = commentsArray[i].split("\\.");
	for (int j=0; j<sentences.length; j++) {
		if (sentences[j].isEmpty() || sentences[j].length()<3) {
		}
		else{
		 sentencesList.add(sentences[j]);
		 }
	 }
	}
	return sentencesList;
```

Πλαίσιο Κώδικα 30 - Κλάση ReadDataSet διαχωρισμός σχολείων σε προτάσεις

7.1.6 Ορθογραφικός έλεγχος

Για την ανάπτυξη εργαλείου αυτόματου ορθογραφικού ελέγχου χρησιμοποιήσαμε το Visual Studio 2013 και η ανάπτυξη έγινε με την γλώσσα C#. Για τον ορθογραφικό έλεγχο των λέξεων κάνουμε χρήση του εργαλείου ορθογραφικού και γραμματικού ελέγχου του MS Office Word μέσω της βιβλιοθήκης “Microsoft Word 15.0 Object Library”.

Η εφαρμογή διαβάζει αρχείο που περιέχει το αρχικό κείμενο και αρχικά κάνει έναν διαχωρισμό των λέξεων μεταξύ τους όταν χωρίζονται με σημεία στίξης (είναι συχνό λάθος στα κείμενα να μην γίνεται σωστή χρήση των κενών με τα σημεία στίξης.)

Στην συνέχεια το ανοίγει με την εφαρμογή Word. Μέσω της κλάσης Microsoft.Office.Interop.Word.ProofreadingErrors δημιουργεί αντικείμενο SpellCollection στο οποίο αποθηκεύει τα λάθη πλαίσιο κώδικα 31

```
string inputDoc = System.IO.File.ReadAllText(txtbx_inputfile.Text);
inputDoc = inputDoc.Replace(".", ". ");
inputDoc = inputDoc.Replace(",", ", ");
while(inputDoc.Contains("  "))
{
       inputDoc = inputDoc.Replace("  ", " ");
}
inputDoc = inputDoc.Replace(" ,", ",");
inputDoc = inputDoc.Replace(" .", ".");
Microsoft.Office.Interop.Word.Range DRange;
WordApp.Documents.Add();
DRange = WordApp.ActiveDocument.Range();
DRange.InsertAfter(inputDoc);
rtxtbx_text.Text = inputDoc;
Microsoft.Office.Interop.Word.ProofreadingErrors SpellCollection =
DRange.SpellingErrors;
```

Πλαίσιο Κώδικα 31 – Ανάγνωση αρχείου, διόρθωση σημείων στίξης, δημιουργία αντικείμενο SpellCollection

Στην συνέχεια εμφανίζει τις λάθος λέξεις σε ListBox πλαίσιο κώδικα 32

```
int iword = 0;
string newWord = null;
for (iword = 1; iword <= SpellCollection.Count; iword++)
{
       newWord = SpellCollection[iword].Text;
       ls_errors.Items.Add(newWord);
}
```

Πλαίσιο Κώδικα 32 - Εμφάνιση λάθος λέξεων σε ListBox

Ι ια την διορθωση των λαθων εχουμε δυο επιλογες.

Α) μπορούμε να επιλέξουμε την λάθος λέξη και να μας εμφανίσει σε ListBox τις προτάσεις για διόρθωση πλαίσιο κώδικα 33.

```
for (iWord = 1; iWord <= CorrectionsCollection.Count; iWord++)
 {
 ls_suggest.Items.Add(CorrectionsCollection[iWord].Name);
 }
```

Πλαίσιο Κώδικα 33 – εμφάνιση προτάσεων λέξεων για διόρθωση

Και επιλεγοντας μια λεξη απο τις προτασεις για διορθωση να την αντικαταστει πλαισιο κώδικα 34

```
rtxtbx_text.Select(rtxtbx_text.Text.IndexOf(ls_errors.SelectedItem.ToString()),
ls_errors.SelectedItem.ToString().Length);
rtxtbx_text.SelectedText = ls_suggest.SelectedItem.ToString();
ls_errors.Items.Remove(ls_errors.SelectedItem);
ls_suggest.Items.Clear();
```

Πλαίσιο Κώδικα 34 – διόρθωση επιλεγμένης λέξης

Β) Μπορούμε να αντικαταστήσουμε όλες τις λέξεις με την πρώτη λέξη που μας δίνει σαν πρόταση διόρθωσης πλαίσιο κώδικα 35

```
rtxtbx_text.Select(rtxtbx_text.Text.IndexOf(er.ToString()),
er.ToString().ToString().Length);

rtxtbx_text.SelectedText = ls_suggest.Items[0].ToString();
```

Πλαίσιο Κώδικα 35 - Αντικατάσταση όλων των λάθος λέξεων

7.1.7 Λημματοποίηση

Με την λημματοποίηση αυτό που κάνουμε είναι να μετατρέπουμε όλες τις λέξεις μιας πρότασης στην αρχική τους μορφή. Αυτό γίνεται με το project Lemmatization το οποίο αποτελείται από δύο κλάσεις. Την κλάση Main.java και την κλάση ConnectMongoDB.java. Η κλάση Main.java πλαίσιο κώδικα 36 διαβάζει από αρχείο τις προτάσεις. Χωρίζει το κείμενο σε προτάσεις και για την κάθε πρόταση παίρνει τις λέξεις σε πίνακα words. Έπειτα με την μέθοδο getFromMongo() της κλάσης ConnectMongoDB αντικαθιστά την κάθε λέξη με την λέξη στην αρχική της μορφή. Στην συνέχεια προσθέτει τις λέξεις για να ξαναδημιουργήσει την πρόταση με όλες τις λέξεις στην αρχική τους μορφή και τις γράφει σε νέο αρχείο.

```
String readFile = "data/input/neg.txt";
String writeFile = "data/output/neg.txt";
BufferedReader in = new BufferedReader(new FileReader(readFile));
PrintWriter writer = new PrintWriter(writeFile, "UTF-8");
ArrayList<String> newTexts = new ArrayList<>();
String line;
StringBuffer newLine = new StringBuffer();
while((line = in.readLine()) != null){
        String[] words = getWords(line);
        for (String word : words) {
                word = removeTones(word);
                newLine.append(ConnectMongoDB.getFromMongo(word));
                newLine.append(" ");
        }
        newLine = stripSpecialCharacters(newLine);
        newLine.deleteCharAt(newLine.length()-1);
        newTexts.add(newLine.toString());
        newLine = new StringBuffer();
}
in.close();

for (String writeLine : newTexts) {
        writer.print(writeLine);
        writer.print("\n");
}
writer.close();
```

Πλαίσιο Κώδικα 36 – Λημματοποίηση

7.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Το μεγαλύτερο μέρος του κώδικα της εργασίας μας είναι σε java. Η έκδοση της java είναι η 1.8.0_25 και η ανάπτυξη του κώδικα έγινε με την χρήση του ολοκληρωμένου περιβάλλοντος ανάπτυξης (integrated development environment, IDE) Eclipse στην έκδοση Luna (4.4.0).

Η εφαρμογή για τον ορθογραφικό έλεγχο έγινε με την γλώσσα C# στο Visual Studio 2013 και έκδοση ms office 2013.

Για τις βάσης δεδομένων κάναμε χρήση της MySQL Version 5.5.40 και MongoDB Version 2.6.5 (64-bit). Για καλύτερη διαχείριση και ευκολότερη πρόσβαση στις βάσης δεδομένων χρησιμοποιήσαμε την εφαρμογή SQLyog και Robomongo για MySQL και MongoDB αντίστοιχα.

Τα χαρακτηριστικά του υπολογιστή στον οποίο αναπτύχθηκε η εφαρμογή και εκτελέστηκαν τα πειράματα φαίνεται στο πλαίσιο 4.

Επεξεργαστής: AMD FX(tm)-4100 Quad-Core Processor
Μνήμη RAM: 8GB
Λειτουργικό Σύστημα: Windows 8.1 Pro

Πλαίσιο 4 - Χαρακτηριστικά υπολογιστή όπου αναπτύχθηκε η εφαρμογή και εκτελέστηκαν τα πειράματα

7.3 Εγκατάσταση και εκτέλεση εφαρμογών

7.3.1 SentimentAnalysisGR

Για να ξεκινήσουμε την εφαρμογή SentimentAnalysisGR εκτελούμε το αρχείο SentimentAnalysisGR.jar το οποίο μας ανοίγει την εφαρμογή στην αρχική φόρμα επιλογών.

7.3.2 Λεξικό

7.3.2.1 Λεξικό MySQL

Για να εγκαταστήσω το λεξικό σε βάση δεδομένων MySQL θα πρέπει στον υπολογιστή να έχουμε εγκατεστημένη μια έκδοση MySQL (οδηγίες εδώ https://dev.mysql.com/doc/refman/5.1/en/installing.html) και να τρέχει σαν υπηρεσία. Στην συνέχεια εισάγω σστηνMySQL το αρχείο gr_dictionary.sql.

7.3.2.2 Λεξικό MongoDB

Για την εγκατάσταση του λεξικού σε υπολογιστή, αρχικά θα πρέπει να κάνουμε εγκατάσταση την MongoDB (οδηγίες εδώ http://docs.MongoDB.org/manual/installation/). Στην συνέχεια μεταφέρουμε το αρχείο gr_lem_dictionary.zip στο φάκελο data στην θέση που έγινε η εγκατάσταση της MongoDB και το κάνουμε αποσυμπιεστή. Ξεκινάμε την εκτέλεση της MongoDB (Εκτελούμε την εφαρμογή mongod.exe με παραμέτρους --dbpath τον φάκελο data στον οποίο έχουμε μεταφέρει τα αρχεία του λεξικού «mongod --dbpath ..\data»).

Για να ελέγξουμε ότι εκτελείται σωστά η MongoDB και το λεξικό εκτελούμε την εφαρμογή mongo.exe από το φάκελο bin της mongo. Αν η mongo τρέχει κανονικά μας εμφανίζει την έκδοση της mongo. Για να χρησιμοποιήσουμε την βάση του λεξικού δίνουμε την εντολή «use gr_lem_dictionary». Αν εμφανίσει το μήνυμα «switched to db gr_lem_dictionary» η βάση του

λεξικού δουλεύει κανονικά. Τέλος μπορώ να δώσω με την εντολή db.grWords.count() το πλήθος των εγγραφών στην συλλογή. Για το λεξικό θα πρέπει να είναι 811160.

7.3.3 Συλλογή σχολίων

Η πλήρης συλλογή σχολίων βρίσκεται στο αρχείο «All Sentences -5169.xlsx» η οποία περιέχει 5163 σχόλια χαρακτηρισμένα ως προς το συναίσθημα. Στην στήλη Α υπάρχει το κείμενο του σχολίου και στην στήλη Β υπάρχει ο χαρακτηρισμός του (1 θετικό, -1 αρνητικό και 0 για τα ουδέτερα σχόλια). Η συλλογή είναι σε αρχείο .xlsx το οποίο μπορούμε να επεξεργαστούμε πολύ εύκολα.

7.3.4 Ορθογραφικός έλεγχος

Η εκτέλεση τις εφαρμογής ορθογραφικού ελέγχου γίνεται από το Spelling.exe Για να εκτελεστεί σωστά σε ένα υπολογιστή θα πρέπει να υπάρχει εγκατεστημένο το MS Office word 2013 και το Microsoft .NET Framework 4.5

7.3.5 Λημματοποίηση

Την εφαρμογή της λημματοποίησης μπορούμε να την εκτελέσουμε από το αρχείο Lemmatization.jar. Για την λειτουργία της λημματοποίησης θα πρέπει να έχει γίνει εγκατάσταση του λεξικού σε MongoDB (παράγραφος 7.3.2.2). Η MongoDB θα πρέπει να εκτελείτε τοπικά στην προεπιλεγμένη πόρτα 27017 και χωρίς κωδικό πρόσβασης. Για διαφορετική περίπτωση θα πρέπει να αλλάξουν τα στοιχεία σύνδεσης στην κλάση ConnectMongoDB.java του project Lemmatization.

8

Επίλογος

8.1 Σύνοψη και συμπεράσματα

Στην παρούσα εργασία αναπτύξαμε εργαλεία, με σκοπό να τα κάνουμε χρήση σε εφαρμογή αυτόματης εκτίμηση της διάθεσης για κείμενα στην ελληνική γλώσσα.

Για την αυτόματη εκτίμηση της διάθεσης κάναμε χρήση μοντέλου ημι-επιβλεπόμενων αναδρομικών αυτοσυσχετιστών (Semi-Supervised Recursive Autoencoders) το οποίο, αποδίδει καλύτερα από άλλες προσεγγίσεις σε ευρέως χρησιμοποιημένες συλλογές. Οι αναδρομικοί αυτοσυσχετιστες είναι βαθιά νευρωνικά δίκτυα τα οποία έχουν ως στόχο την ανακατασκευή των διανυσμάτων εισόδου μέσο μιας αναπαράστασης σε κρυφό στρώμα μικρότερων διαστάσεων από την είσοδο. Οι ημι-επιβλεπόμενοι αναδρομικοί αυτοσυσχετιστές προσθέτουν και ένα στρώμα softmax σε κάθε κόμβο για την πρόβλεψη των κατανεμημένων κλάσεων.

Τα εργαλεία που αναπτύξαμε είναι:

- Λεξικό της ελληνικής γλώσσας για βάσεις δεδομένων MySQL και Mongodb. Το λεξικό περιέχει τις λέξεις σε όλες τις πτώσεις, αριθμούς, γένη, χρόνους κτλ. Στο λεξικό υπάρχουν περίπου 60.000 μοναδικές λέξεις και περισσότερες από 800.000 λέξεις στις διάφορες κλήσεις τους.
- Αρχείο .csv με συλλογή από 5196 σχόλια χρηστών για κινητά τηλέφωνα χαρακτηρισμένα ως προς το συναίσθημα.

- Εφαρμογή αυτόματης διόρθωσης ορθογραφικών λαθών που κάνει χρήση του εργαλείου ορθογραφικού ελέγχου του MS office word και δίνει τις δυνατότητες είτε να διορθωθεί η κάθε λέξη ξεχωριστά ή να αντικατασταθούν αυτόματα όλες οι λέξεις.
- Εφαρμογή λημματοποίησης κειμένων η οποία αντικαταστεί τις λέξεις ενός κειμένου στην αρχική τους μορφή κάνοντας χρήση του λεξικού που έχουμε αναπτύξει.
- Εφαρμογή εκπαίδευσης, ελέγχου και εκτέλεσης πειραμάτων για τους ημι-επιβλεπόμενους αναδρομικούς αυτοσυσχετιστές. Η εφαρμογή διαθέτει γραφικό περιβάλλον από το οποίο μπορούμε να αλλάξουμε τις παραμέτρους εκτέλεσης των RAE, να εκτελέσουμε πειράματα από τα οποία θα μας εμφανίσει τα αποτελέσματα ή μπορούμε να επιλέξουμε και να τα αποθηκεύσει σε αρχείο.

Τα συμπεράσματα που προκύπτουν από την εργασία μας είναι ότι:

- Για να εφαρμοστεί σωστά ένα σύστημα αυτόματης εκτίμησης διάθεσης κειμένων πολύ σημαντικό ρόλο παίζει η προ-επεξεργασία των δεδομένων για την εκπαίδευση του συστήματος. Στην περίπτωση της εργασίας μας καταφέραμε να βελτιώσουμε την σωστή πρόβλεψη του συναισθήματος σε ποσοστό που κυμαίνεται από 5,4% μέχρι και 21% με διόρθωση των ορθογραφικών λαθών στα κείμενα και με την διαδικασία της λημματοποίησης αυξήσαμε το ποσοστό αυτό από 0,3% μέχρι και 3,7%.
- Σε ένα σύστημα ημι-επιβλεπόμενων αναδρομικών αυτοσυσχετιστών για την αυτόματη εκτίμηση της διάθεσης κειμένων έχει σημασία το πλήθος των λέξεων που θα χρησιμοποιηθούν για την πρόβλεψη του συναισθήματος. Δεν θα πρέπει να χρησιμοποιούνται όλες οι λέξεις μιας πρότασης αλλά μόνο οι λέξεις οι οποίες εμφανίζονται συχνά στις προτάσεις και περιέχουν κάποια συναισθηματική πόλωση. Στα πειράματα της εργασία μας αλλάζοντας το πλήθος των λέξεων στο λεξικό παρατηρήσαμε πως όταν το λεξικό περιείχε πολλές λέξεις το μοντέλο πρόβλεψης γινόταν πολύ σύνθετο με αποτέλεσμα την μη σωστή πρόβλεψη.

8.2 Μελλοντικές επεκτάσεις

Ένας από τους στόχους της εργασίας μας είναι η χρήση των εργαλείων που αναπτύξαμε σε μελλοντικά έργα.

Το μοντέλο των RAE που κάναμε χρήση έχει την δυνατότητα να εκπαιδευτεί και να προβλέπει πολυδιάστατες κατανομές συναισθημάτων που αντικατοπτρίζουν καλύτερα τα σύνθετα ανθρώπινα συναισθήματα. Μία μελλοντική επέκταση θα ήταν να αναπτυχθεί συλλογή η οποία θα μπορούσε να εκπαιδεύσει τους RAE στο να προβλέπουν μια τέτοια κατανομή συναισθημάτων. Δηλαδή, να μην γίνεται χαρακτηρισμός μιας πρότασης απλά ως θετική ή αρνητική, αλλά να εμφανίζεται μια κατανομή πολλών συναισθήματα (θυμός, φόβος, χαρά, αγάπη, έκπληξη κτλ.) όπου μια πρόταση θα χαρακτηρίζετε με ποσοστά στα παραπάνω

συνθήματα. Για να γίνει αυτό θα πρέπει να αναπτυχθεί συλλογή στην οποία θα χαρακτηριστούν οι προτάσεις από ανθρώπους για τα παραπάνω συναισθήματα.

Οι δυνατότητες που έχουν οι RAE στην ταξινόμηση κειμένων δεν περιορίζονται μόνο στην εκτίμηση της διάθεσης, θα μπορούσαν να βρουν εφαρμογή και σε άλλες περιπτώσεις ταξινόμησης κείμενων. Για παράδειγμα θα μπορούσε να αναπτυχθεί σύστημα το οποίο θα ταξινομεί ειδησεογραφικά κείμενα σε κατηγορίες (αθλητικά, πολιτική, ψυχαγωγία κτλ.).

Το λεξικό που αναπτύξαμε χρησιμοποιεί ένα διάνυσμα για την αναπαράσταση τις κάθε λέξης. Αυτό το διάνυσμα δημιουργείται από τυχαίους αριθμούς κανονικής γκαουσιανής κατανομής. Μία επέκταση του λεξικού μας θα ήταν το διάνυσμα αναπαράστασης της κάθε λέξης να προέρχεται από έναν μηχανισμό word2vec. Το word2vec δημιουργεί διανύσματα αναπαράστασης των λέξεων τα οποία περιγράφουν την κάθε λέξη σε έναν n-διάστατο χώρο.

Τα διανύσματα των λέξεων που χρησιμοποιούνται με παρόμοιο τρόπο εμφανίζονται κοντά το ένα στο άλλο, δίνοντας μας την δυνατότητα να αναγνωρίζουμε παρόμοιες λέξεις. Το κάθε διάνυσμα σχετίζεται με την σημασιολογική έννοια της κάθε λέξης. Για να δημιουργήσουμε τα παραπάνω διανύσματα θα πρέπει να κάνουμε χρήση μιας πολύ μεγάλης συλλογής κείμενων (Για την συλλογή Google News dataset χρησιμοποιήθηκαν περίπου 100 δισεκατομμύρια λέξεις και αναπτύχθηκε συλλογή διανυσμάτων 300-διαστάσεων για 3 εκατομμύρια λέξεις και φράσεις). Μια λύση για την ελληνική γλώσσα θα ήταν να γίνει χρήση της συλλογής κειμένων από την Ελληνική Wikipedia.

Βιβλιογραφία

[1] P. D. Turney, “Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews,” p. 8, 2002.

[2] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? Sentiment Classification using Machine Learning Techniques,” *Conf. Empir. Methods Nat. Lang. Process. (EMNLP 2002)*, no. July, pp. 79–86, 2002.

[3] B. Pang and L. Lee, “Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales,” no. 1, 2005.

[4] B. Snyder, B. Snyder, R. Barzilay, and R. Barzilay, “Multiple aspect ranking using the good grief algorithm,” *Proc. NAACL HLT*, pp. 300–307, 2007.

[5] M. Thelwall, K. Buckley, G. Paltoglou, and D. Cai, “Sentiment Strength Detection in Short Informal Text,” *J. Am. Soc. Inf. Sci.*, vol. 61, no. 12, pp. 2544–2558, 2010.

[6] A. Thelwall, Mike; Buckley, Kevan; Paltoglou, Georgios; Cai, Di; Kappas, “Sentiment Strength Detection in Short Informal Text,” *J. Am. Soc. Inf. Sci. Technol.*, vol. 61, pp. 2544–2558, 2010.

[7] F. Su, F. Su, K. Markert, and K. Markert, “From words to senses: a case study of subjectivity recognition,” *Proc. 22nd Int. Conf. Comput. Linguist. 1*, pp. 825–832, 2008.

[8] B. Pang and L. Lee, “A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts,” 2004.

[9] B. Liu, “Sentiment Analysis and Subjectivity,” *Handb. Nat. Lang. Process.*, no. 1, pp. 1–38, 2010.

[10] E. Schinas, S. Papadopoulos, S. Diplaris, and J. Herzig, “EventSense : Capturing the Pulse of Large-scale Events by Mining Social Media Streams Categories and Subject Descriptors,” pp. 17–24.

[11] P. Agathangelou and I. Katakis, “Mining Domain-Specific Dictionaries of Opinion Words.”

[12] M. Osborne and V. Lavrenko, “Streaming First Story Detection with application to Twitter,” *Comput. Linguist.*, no. June, pp. 181–189, 2010.

[13] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, “Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions,” no. i, 2008.

[14] D. Mo, “A survey on deep learning : one small step toward AI,” pp. 1–16, 2012.

[15] Y. Bengio and Y. {LeCun}, “Scaling Learning Algorithms towards AI,” *Large Scale Kernel Mach.*, no. 1, pp. 321–360, 2007.

[16] Y. Bengio, *Learning Deep Architectures for AI*, vol. 2, no. 1. 2009.

[17] G. E. Hinton, S. Osindero, and Y.-W. Teh, “Communicated by Yann Le Cun A Fast Learning Algorithm for Deep Belief Nets 500 units 500 units,” *Neural Comput.*, vol. 18, pp. 1527–1554, 2006.

[18] I. Arel, D. C. Rose, and T. P. Karnowski, “Deep Machine Learning — A New Frontier in Artificial Intelligence Research,” no. November, pp. 13–18, 2010.

[19] “Convolutional Neural Networks (LeNet) — DeepLearning 0.1 documentation.” [Online]. Available: http://deeplearning.net/tutorial/lenet.html. [Accessed: 07-Jun-2015].

[20] D. Cireşan, U. Meier, and J. Schmidhuber, “Multi-column Deep Neural Networks for Image Classification,” *Int. Conf. Pattern Recognit.*, no. February, pp. 3642–3649, 2012.

[21] R. Sarikaya, G. E. Hinton, and A. Deoras, “Application of Deep Belief Networks for Natural Language Understanding,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 4, pp. 778–784, 2014.

[22] Π. Νούση, “Αλγόριθμοι Βαθειάς Μάθησης για Διακριτική Αυτοκωδικοποίηση,” Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, 2014.

[23] S. Ji, M. Yang, K. Yu, and W. Xu, “3D convolutional neural networks for human action recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–31, 2013.

[24] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” *Proc. 26th Annu. Int. Conf. Mach. Learn. ICML 09*, vol. 2008, pp. 1–8, 2009.

[25] J. B. Pollack, “Recursive Distributed Representations.”

[26] M. Jordan and J. Kleinberg, *Bishop - Pattern Recognition and Machine Learning*. .

[27] D. C. Liu and J. Nocedal, “On the limited memory BFGS method for large scale optimization,” *Math. Program.*, vol. 45, no. 1–3, pp. 503–528, 1989.

[28] T. Nakagawa, K. Inui, and S. Kurohashi, “Dependency Tree-based Sentiment Classification using CRFs with Hidden Variables,” *Comput. Linguist.*, no. June, pp. 786–794, 2010.