Πληροφορική

Πρακτική Εξάσκηση σε Θέματα Λογισμικού

Πτυχιακή Εργασία

Ανάπτυξη Εφαρμογής για την Ανίχνευση Ψευδών Ειδήσεων

Αριστοτέλης Πασαλίδης

Επιβλέπων Καθηγητής: Κυριάκος Σγάρμπας

Πάτρα, Ιούλιος 2018

Ανάπτυξη Εφαρμογής για την Ανίχνευση Ψευδών Ειδήσεων

Αριστοτέλης Πασαλίδης

Επιτροπή Επίβλεψης Πτυχιακής Εργασίας

Επιβλέπων Καθηγητής:	Συν-Επιβλέπων Καθηγητής:	Συν-Επιβλέπων Καθηγητής:
Κ. Σγάρμπας	**Ι. Τζήμας**	**Σ. Κωτσιαντής**
Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής	Λέκτορας
Τμήμα Ηλεκτρολόγων	Τμήμα Μηχανικών	Τμήμα Μαθηματικών
Μηχανικών & ΤΥ	Πληροφορικής ΤΕ	Πανεπιστήμιο Πατρών
Πανεπιστήμιο Πατρών	ΤΕΙ Δυτικής Ελλάδας	

Πάτρα, Ιούλιος 2018

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Σγάρμπα Κυριάκο για την υποστήριξη και την καθοδήγηση που μου προσέφερε καθ' όλη την διάρκεια της εργασίας.

Περίληψη

Με την ευρεία εξάπλωση των μέσων κοινωνικής δικτύωσης, αλλά και του διαδικτύου γενικότερα, οι ψευδείς ειδήσεις αποτελούν μία νέα μάστιγα, η οποία κατακλύζει το διαδίκτυο και διαδίδεται με ταχείς ρυθμούς. Οι ψευδείς ειδήσεις στο διαδίκτυο αποτελούν μία μορφή στοχευμένης παραπληροφόρησης, η οποία μπορεί να αποσκοπεί είτε σε κάποια προπαγάνδα, είτε στην αύξηση των θεάσεων μίας ιστοσελίδας και την εμπορική εκμετάλλευση αυτών. Η διάκριση της εγκυρότητας μιας είδησης δεν είναι πάντα μία εύκολη διαδικασία για τον μέσο χρήστη του διαδικτύου, ο οποίος δύναται να πειστεί για την εγκυρότητας της είδησης και με την σειρά του να συμβάλει στην εξάπλωσή της κοινοποιώντας την σε μέσα κοινωνικής δικτύωσης. Όσο πιο ακραία και πειστική είναι κάποια είδηση τόσο πιο γρήγορα εξαπλώνεται σαν πανδημία στο διαδίκτυο. Σε αυτό το χαοτικό περιβάλλον πληθώρας πληροφοριών, όπου η διασταύρωση πηγών και η εξακρίβωση γεγονότων καταλήγουν να είναι αρκετά χρονοβόρες διαδικασίες, γεννιέται η ανάγκη για αυτοματοποιημένων εργαλείων ανίχνευσης ψεύδους. Στόχος της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μίας εύχρηστης και ευκόλως εκπαιδεύσιμης εφαρμογής, η οποία θα μπορεί να διακρίνει στοιχεία ψεύδους μέσα σε ένα ελληνικό κείμενο ή μίας ελληνικής δημοσίευσης σε κάποιον ιστότοπο. Μέθοδοι υλοποίησης αυτής είναι η καταμέτρηση ποσοστού εμφάνισης λέξεων και μερών του λόγου, καθώς και η πιθανότητα εμφάνισης μίας αλληλουχίας αυτών, σε έγκυρες και μη ειδήσεις.

Λέξεις – Κλειδιά

Ασκός Λέξεων, Μέρη του Λόγου, Ανίχνευση Ψεύδους.

Περιεχόμενο

Κείμενο, πρόγραμμα σε γλώσσα Java.

Application Development for Fake News Detection

Aristotelis Pasalidis

Supervisor Professor:
K. Sgarbas
Associate Professor
Electrical & Computer Dept.
University of Patras

Co-Supervisor Professor:
I. Tzimas
Associate Professor
Computer & Informatics Engineering Dept.
TEI of Western Greece

Co-Supervisor Professor:
S. Kotsiantis
Lecturer
Mathematics Dept.
University of Patras

Patras, July 2018

Abstract

Fake news is a type of deliberate misinformation that spreads via the media to either influence the public's opinion or gain financially. Although fake news used primarily to spread through traditional media, the wide adaption of internet services over the past years has shifted the main source of misinformation. Internet has allowed information to propagate faster than ever before, but it has also enabled a completely new way to publish, share and consume news with very little to none regulation or editorial standards. User generated content far surpasses the amount of traditional media content resulting in an overload of the current online news environment. In such an environment, it has become extremely difficult for the average reader to evaluate the credibility of an article and, to some extent, to distinguish between content generated by users and content generated by journalists. The need of new technologies and automated tools arises, as fact checking and source verification becomes a time-consuming process. Various approaches of identifying deception in articles have shown very promising results. Patterns, like the frequency of a word's appearance or a phrase's syntactic structure, can give away the author's underlying intentions. The goal of this thesis is the development of an easy to use and train application, which will be able to check the veracity of either plain text or a webpage that contains a news article written, in both cases, in Greek.

Keywords

Bag of words, part of speech, deception detection.

Content

Text, Java program.

Περιεχόμενα

Κεφάλαιο 1: Ανάλυση

1.1 Ψευδείς Ειδήσεις

Η ευρεία διάδοση της διαδικτυακής πληροφόρησης έχει αλλάξει σημαντικά τον τρόπο με τον οποίο γίνεται η συγγραφή αλλά και η ανάγνωση ειδήσεων σε σχέση με τα παραδοσιακά μέσα μαζικής ενημέρωσης. Το διαδίκτυο έχει δώσει την δυνατότητα της ταχύτατης διάδοσης ειδήσεων από πηγές οι οποίες ποικίλλουν σε εγκυρότητα και ποιότητα. Οι χρήστες του διαδικτύου συναντάνε έναν καταιγισμό από πληροφορίες καθώς πλέον δεν είναι μόνο καταναλωτές αυτών, αλλά είναι και σε θέση να παράγουν και να διαδίδουν τα δικά τους άρθρα ειδήσεων.

Παρόλο που σύμφωνα με μελέτες οι καταναλωτές προσδίδουν μεγαλύτερη αξιοπιστία σε μία είδηση προερχόμενη από κάποιο δημοσιογραφικό γραφείο παρά σε κάποια η οποία προέρχεται από χρήστη, είναι αρκετά αμφίβολο το κατά πόσο ένας μέσος χρήστης διαδικτύου μπορεί να ξεχωρίσει με ακρίβεια τις δύο αυτές πηγές (Conroy, Chen & Rubin, 2015). Αυτό κυρίως οφείλεται στο ότι οι καταναλωτές παραδοσιακά αντιλαμβάνονταν την δημοτικότητα ενός δημοσιογραφικού γραφείου ως μία καλή ένδειξη εγκυρότητας, ωστόσο στο διαδίκτυο ένα άρθρο προερχόμενο από χρήστη ή ερασιτέχνη μπορεί να αποκτήσει ανάλογη δημοτικότητα ακόμα και αν είναι ψευδές. Αυτό επιτυγχάνεται δελεάζοντας τον χρήστη να κάνει click στον εν λόγω σύνδεσμο με κάποιο παραπλανητικό τρόπο, όπως μέσω μίας εντυπωσιακής επικεφαλίδας, το λεγόμενο clickbait. Ως κύριο στόχο έχει να αυξήσει τις προβολές (views) ενός συνδέσμου/άρθρου, κάτι που θα επιφέρει χρηματικό κέρδος στον αρθρογράφο μέσω διαφημίσεων που προβάλλονται στον ιστότοπό του.

Μέσα κοινωνικής δικτύωσης, όπως το Facebook και Twitter, προωθούν έμμεσα τέτοιες πρακτικές παραπλάνησης, δίνοντας προτεραιότητα εμφάνισης σε άρθρα με κύριο κριτήριο το πλήθος των προβολών και όχι της εγκυρότητας τους. Επιπλέον οι χρήστες έχουν την δυνατότητα να αναπαράγουν και να προωθήσουν ένα άρθρο στους φίλους τους, διαδίδοντας ακόμα περισσότερο μία πιθανά ψευδή είδηση. Όσο πιο ακραία και πιθανότατα ψευδή είναι μία είδηση, τόσο πιο γρήγορα αναπράγεται (Soroush, Deb & Sinan, 2018). Έτσι δημιουργείται ένα περιβάλλον όπου κάθε είδος πληροφορίας μπορεί να εξαπλωθεί ταχύτατα σε όλο το διαδίκτυο, ανεξαρτήτως της εγκυρότητας του (Mitchell & Page, 2015).

Η παραπληροφόρηση που επικρατεί στο διαδίκτυο θα μπορούσε να ελαττωθεί με την κατάλληλη ενημέρωση και εκπαίδευση των χρηστών, ωστόσο το πλήθος των πληροφοριών που δέχεται ένας μέσος χρήστης συνεχίζει να είναι μεγάλο και να αυξάνεται με το καιρό έχοντας ως επίπτωση την αναγκαιότητα ύπαρξης αυτόματων μέσων ανίχνευσης ψεύδους, που δεν θα αντικαταστήσουν αλλά θα ενισχύουν την ανθρώπινη κρίση.

1.2 Μέθοδοι Ανίχνευσης

1.2.1 Γλωσσική Ανάλυση

Οι περισσότεροι αρθρογράφοι ψευδών ειδήσεων κάνουν στρατηγική χρήση της γλώσσα για να αποφύγουν να γίνουν αντιληπτοί. Ωστόσο, παρόλο την προσπάθεια τους αυτή, μέσα από τα λεγόμενα τους αναδύονται κάποια μοτίβα τα όποια είναι δύσκολα να παρατηρηθούν με γυμνό μάτι, όπως είναι η συχνότητα εμφάνισης ορισμένων λέξεων, η χρήση χαρακτηριστικών επιθέτων και ρημάτων ή λέξεων που αποσκοπούν στην συναισθηματική φόρτιση του κειμένου (Feng & Hirst, 2013). Ο στόχος της γλωσσικής προσέγγισης είναι η εύρεση τέτοιων μοτίβων και στοιχείων εξαπάτησης.

Ορθογραφικός Έλεγχος: Πλέον σε κάθε σύγχρονη συσκευή υπάρχει δωρεάν πρόσβαση σε κάποιο είδος ορθογραφικού ελέγχου, κάτι που καθιστά την ύπαρξη μη ορθογραφημένων άρθρων πολύ σπάνια. Για αυτό ακριβώς τον λόγο η ορθογραφία αποτελεί μία ένδειξη αμέλειας του αρθρογράφου και ισχυρού κριτηρίου για την εγκυρότητα του άρθρου αυτού.

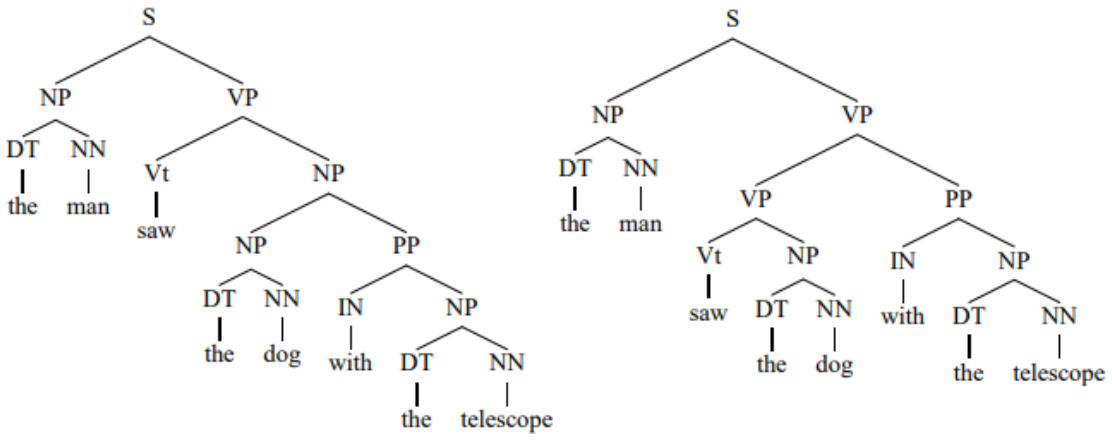
Ανίχνευση Συναισθημάτων: Μία κύρια τακτική των αρθρογράφων ψευδών ειδήσεων είναι να προκαλούν εντυπωσιασμό μέσα από συναισθηματικά φορτισμένες εκφράσεις. Κύρια στοιχεία αυτών τον εκφράσεων είναι η υπερβολική χρήση θαυμαστικών και η χρήση λέξεων αποτελούμενες αποκλειστικά από κεφαλαία γράμματα, όπως επίσης και η χρήση ποσοτικών ή εμφατικών επιρρημάτων. Τέτοια στοιχεία μπορούν να αποτελέσουν συμπληρωματικό κριτήριο για τον καθορισμό της εγκυρότητας ενός άρθρου.

Ασκός Λέξεων: Ο ασκός λέξεων (bag-of-words) αποτελεί την απλούστερη μέθοδο υλοποίησης της γλωσσικής προσέγγισης, ωστόσο είναι αρκετά αποτελεσματική σε σχέση με τις υπόλοιπες μεθόδους υλοποίησης (Feng & Banerjee, 2012). Η προσέγγιση αυτή βασίζεται στον υπολογισμό των συχνοτήτων εμφάνισης μεμονωμένων λέξεων ή ν-άδων γειτονικών λέξεων και την αντιστοίχιση τους με κάποιο ποσοστό εγκυρότητας. Ωστόσο λόγω της απλότητας αυτής της μεθόδου κάθε λέξη ή φράση (ν γειτονικές λέξεις) αποδεσμεύεται εντελώς από το ολικό νόημα της πρότασης, με αποτέλεσμα αμφίσημες λέξεις ή εκφράσεις να αντιστοιχίζονται στο ίδιο ποσοστό εγκυρότητας. Πολλοί ερευνητές θεωρούν αυτή την μέθοδο αρκετά χρήσιμη ως συμπληρωματική των υπόλοιπων προσεγγίσεων.

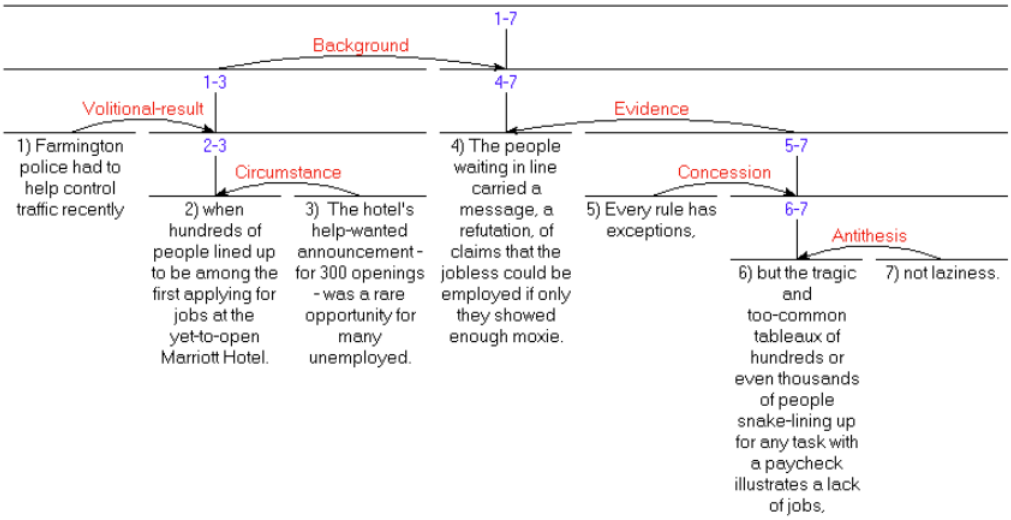
Πιθανολογική Ταξινόμηση: Με την χρήση απλού ταξινομητή Bayes, βασισμένο στην δομή του Ασκού Λέξεων και την κατάλληλη εκπαίδευση μαθηματικών μοντέλων του, μπορεί να γίνει διαχωρισμός των άρθρων σε μία από τις δύο κατηγορίες, ψευδές και αληθές. Η ακρίβεια αυτής μεθόδου ωστόσο είναι μικρότερη από τις υπόλοιπες.

Επιφανειακή Δομή: Βάσει ενός λεξικού ή προκαθορισμένων κανόνων γίνεται αντιστοίχιση της κάθε λέξης σε μία ετικέτα που περιγράφει τον ρόλο της λέξης στην πρόταση (part-of-speech tagging), όπως ρήμα, ουσιαστικό, άρθρο, αντωνυμία, κτλ. Για παράδειγμα, η πρόταση «Ο σκύλος κυνήγησε την γάτα» αντιστοιχεί στην αλληλουχία ετικετών «άρθρο ουσιαστικό ρήμα άρθρο ουσιαστικό». Έπειτα της αντιστοιχησης αυτής, η ανίχνευση γίνεται με τον ίδιο ακριβώς τρόπο όπως στην μέθοδο «Ασκός Λέξεων», με την διαφορά ότι πλέον γίνεται η καταμέτρηση της συχνότητας εμφάνισης των ετικετών ή γειτονικών ετικετών σε ένα άρθρο (bag-of-POS), αντί των λέξεων. Η μέθοδος αυτή αποφέρει ισάξια ακρίβεια σε σχέση της μεθόδου «Ασκός Λέξεων» (Appelgren, 2016).

Βαθεία Δομή: Η βαθεία δομή είναι ένα ακόμα βήμα πιο πέρα από την επιφανειακή, η οποία επιλύει πιθανολογικά την συντακτική και λεκτική αμφισημία που μπορεί να προκύψει στην Επιφανειακή Δομή και στον Ασκό Λέξεων. Χρησιμοποιώντας τις ετικέτες της προηγούμενης μεθόδου και μέσω της εφαρμογής Πιθανολογικής Ανεξάρτητης Συμφραζομένων Γραμματικής (Probabilistic Context Free Grammar – PCFG) δημιουργείται ένα δένδρο διάσχισης το όποιο έχει ως φύλλα τις λέξεις μίας πρότασης (Σχ. 1.1), ενώ δε κάθε κόμβος χαρακτηρίζεται από μία ετικέτα και μία πιθανότητα. Βάσει των πιθανοτήτων αυτών σχηματίζεται μία αθροιστική πιθανότητα για κάθε ενδεχόμενη συντακτική αναπαράσταση της πρότασης. Η αναπαράσταση αυτή συσχετίζεται με τιμές ψεύδους από όπου και αντλείται το ποσοστό εγκυρότητας της πρότασης. Αυτή η μέθοδος μπορεί να αποδώσει έως και 91% ακρίβεια στην ανίχνευση ψεύδους (Feng & Banerjee, 2012).

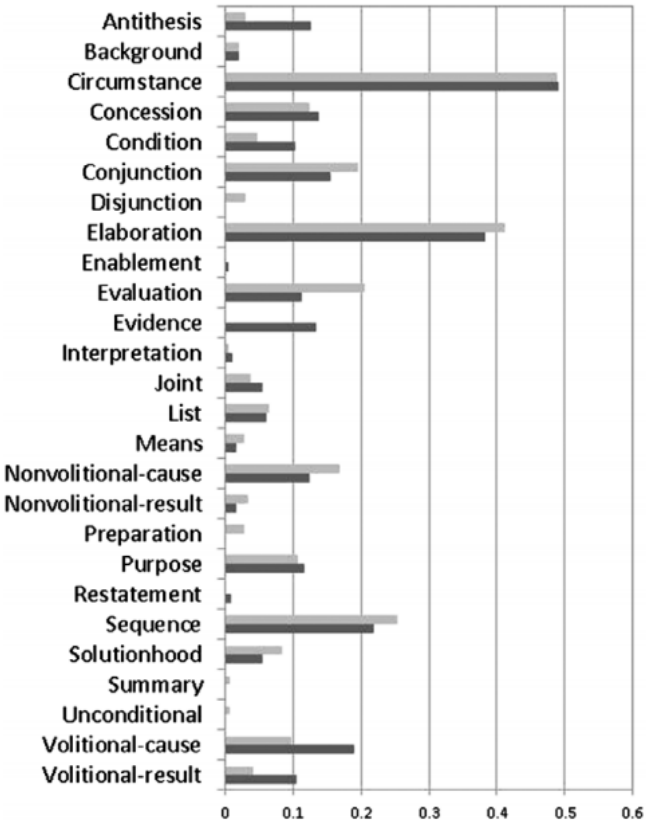


Σχήμα 1.1: Δύο παραλλαγές ενός δένδρου διάσχισης της ίδιας πρότασης πρότασης (Collins, 2011).



Σχήμα 1.2: Ανάλυση της Ρητορική Δομής ενός κειμένου.

Ρητορική Δομή: Σε αυτή την μέθοδο γίνεται μία προσπάθεια να αναγνωριστούν σε ένα κείμενο περιπλοκότερες δομές του λόγου και η συσχέτιση τους με τον πυρήνα του κειμένου (Σχ. 1.2). Αναφορικά κάποιες από αυτές τις συσχετίσεις είναι το κίνητρο, η αντίθεση, η προϋπόθεση, η συνένωση, το αιτιατό, το αποτέλεσμα, κτλ. Από την συχνότητα εμφάνισης τέτοιων δομών, όπως η συχνή εμφάνιση αντιθέσεων ή η χαρακτηριστική έλλειψη ενδείξεων/αποδείξεων (Σχ. 1.3), μπορούν να προκύψουν μοτίβα που θα αποδίδουν ένα ποσοστό εγκυρότητας στο κείμενο.



Σχήμα 1.3: Πίνακας με την συχνότητα εμφάνισης ρητορικών δομών σε ψευδείς και αληθείς δημοσιεύσεις ανοικτό και σκούρο χρώμα αντίστοιχα. (Rubin & Lukoianova, 2014)

1.2.2 Διαδικτυακή Διερεύνηση

Καθώς οι ειδήσεις δεν διαδίδονται μόνο μέσω σελίδων ενημέρωσης, αλλά και μέσω ιστοτόπων κοινωνικής δικτύωσης, όπως Twitter και Facebook, όπου το δημοσίευμα συνήθως είναι μικρό σε πλήθος λέξεων και χαλαρό σε ύφος, μία μη γλωσσική προσέγγιση αποδίδει καλύτερα αποτελέσματα στην ανίχνευση ψεύδους.

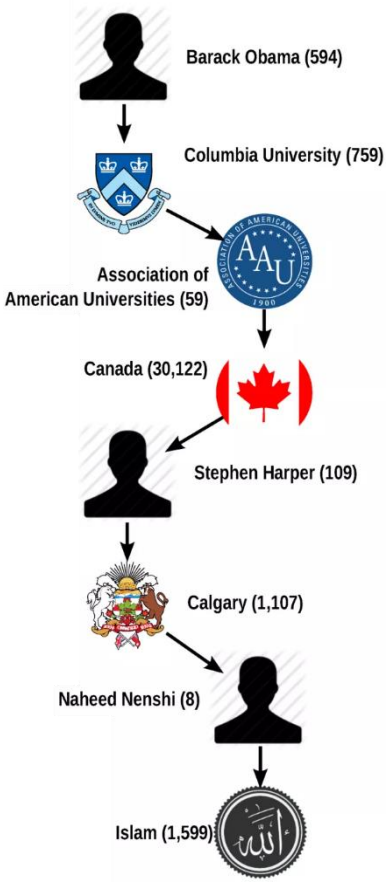
Πηγή Άρθρου: Μία ένδειξη της εγκυρότητας ενός άρθρου μπορεί να αποτελέσει η παρούσα πηγή στην οποία είναι δημοσιευμένο το άρθρο. Η πηγή μπορεί να βαθμολογηθεί από την δομή της διεύθυνσης (URL), το ποσοστό εγκυρότητας του μέσου στο οποίο δημοσιεύεται, αλλά και την εγκυρότητα του δημοσιογράφου που υπέγραψε το δημοσίευμα. Δύο κύριες προϋποθέσεις των τελευταίων χαρακτηριστικών είναι:

I. Η εξακρίβωση στοιχείων και ταυτοποίηση αρθρογράφου με κάποιο φυσικό πρόσωπο, κάτι το οποίο προσθέτει έναν βαθμό αξιοπιστίας στο δημοσίευμα αποτρέποντας την πλαστογραφία ή την σύγχυση των αναγνωστών. Το Twitter και εν μέρη το Facebook έχουν υιοθετήσει αυτή την μέθοδο (https://support.twitter.com/articles/119135).
II. Η ύπαρξη κάποιας δικτυακής βάσης δεδομένων στην οποία θα υπάρχει αντιστοίχιση μεταξύ ποσοστού εγκυρότητας και ενός αρθρογράφου ή μιας σελίδας δημοσίευσης.

Επίσης, παρόλο που η άμεση πηγή ενός δημοσιεύματος είναι ένα σημαντικό κριτήριο, η εύρεση της αρχικής πηγής του άρθρου, αν πρόκειται για αναδημοσίευση, είναι εξίσου σημαντική. Αυτό διότι, οι περισσότερες ψευδείς ειδήσεις προέρχονται από συγκεκριμένες και ελάχιστες, σε σχέση με το πλήθος των αναδημοσιεύσεων, αρχικές πηγές πριν εξαπλωθούν στο διαδίκτυο ως επιδημία. Η αναγνώριση αυτών βοηθάει στην περεταίρω βελτίωση της αξιολόγησης.

Ημερομηνία: Ένας καθοριστικός παράγοντας που συμβάλει στην ευρεία εξάπλωση κάποιων ψευδών ειδήσεων είναι η απουσία ημερομηνίας δημοσίευσης. Χωρίς αυτή την χρονική ένδειξη, ένα άρθρο μπορεί να διαδίδεται επ' άπειρον πείθοντας τους αναγνώστες ότι είναι επίκαιρο. Ωστόσο, συνήθως ένας αρθρογράφος ψευδών ειδήσεων, για να αποφύγει να γίνει τόσο εύκολα αντιληπτός, βαίνει σε περιοδική αναδημοσίευση των ειδήσεων με ανανεωμένη ημερομηνία. Στην περίπτωση αυτή χρειάζεται να γίνει διαδικτυακή έρευνα για παλαιότερες παραπλήσιες δημοσιεύσεις. Λόγω του ότι πολλές ψευδείς ειδήσεις αναπαράγονται σχεδόν αυτούσιες, ακόμα και αν ο αρχικός αρθρογράφος έχει διαγράψει την παλαιότερη δημοσίευσή του, δεν είναι τόσο δύσκολο να βρεθούν αναδημοσιεύσεις της ίδιας ψευδής ειδήσεως με διαφορετική ημερομηνία.

Συσχέτιση Δεδομένων: Με την χρήση δημόσια διαθέσιμων δομημένων βάσεων δεδομένων όπως το Google Relation Extraction Corpus (GREC) ή το DBpedia, μπορεί να ελεγχτεί κατά πόσο δύο οντότητες συσχετίζονται μεταξύ τους. Από την μία οντότητα προς την άλλη δημιουργούνται διάφορα μονοπάτια, με ενδιάμεσους κόμβους όπου ο καθένας σχετίζεται με τον γειτονικό του. Εξετάζοντας το συντομότερο από αυτά τα μονοπάτια, αλλά και τις ενδιάμεσες οντότητες, υπολογίζεται το ποσοστό εγκυρότητας μίας πρότασης. Για παράδειγμα, στο Σχήμα 1.3 βλέπουμε το συντομότερο μονοπάτι που βρέθηκε για την ψευδή πρόταση «Ο Μπαράκ Ομπάμα είναι μουσουλμάνος». Το μονοπάτι περνάει από υψηλόβαθμους κόμβους (οι αριθμοί μέσα στις παρενθέσεις), οι οποίοι αντιπροσωπεύουν γενικές οντότητες όπως είναι ο Καναδάς, και έτσι αποδίδεται μικρή τιμή αληθείας σε αυτό (Ciampaglia, Shiralkar, Rocha, Bollen, Menczer & Flammini, 2015). Το θετικό αυτής της μεθόδου είναι ότι μπορεί να εξεταστεί η εγκυρότητα ενός κειμένου ακόμα και αν σε αυτό δεν συναντιούνται σπουδαίες γλωσσικές ενδείξεις ψεύδους, δηλαδή έχει υψηλό ποσοστό εγκυρότητας βάσει των γλωσσικών προσεγγίσεων. Ωστόσο ως βασική προϋπόθεση της συσχέτισης δεδομένων είναι η εκ των προτέρων ύπαρξη των οντοτήτων στην βάση δεδομένων που χρησιμοποιείται, κάτι το οποίο δεν δύναται να συμβαίνει πάντα.



Σχήμα 1.4: Το συντομότερο μονοπάτι συσχέτισης δύο οντοτήτων.

1.2.3 Βαθεία Μηχανική Μάθηση

Μηχανική μάθηση ονομάζεται η δημιουργία μοντέλων ή προτύπων από ένα σύνολο δεδομένων σε ένα υπολογιστικό σύστημα. Με έναν ευρύτερο ορισμό θα μπορούσαν να θεωρηθούν οι προαναφερθείσες μέθοδοι της γλωσσικής ανάλυση ως μηχανική μάθηση, ωστόσο συνήθως προϋποθέτει την εκμάθηση μέσω ενός Τεχνικού Νευρωνικού Δικτύου (Artificial Neural Network - ANN) ή με την χρήση Μηχανών Διανυσμάτων Υποστήριξης (Support Vector Machines – SVM).

Μία υποκατηγορία της μηχανικής μάθησης αποτελεί η Βαθεία Μάθηση (Deep Learning), η οποία είναι ένας συνδυασμός κατηγοριοποίησης δεδομένων με SVM και ANN με πολλαπλά κρυφά επίπεδα (hidden layers). Η κατηγορία αυτή εκμάθησης είναι ιδιαίτερα χρήσιμη σε μέσα κοινωνικής δικτύωσης, όπου οι δημοσιεύσεις είναι μικρές σε πλήθος λέξεων. Σε αντίθεση με τις άλλες προσεγγίσεις δεν απαιτείται η ύπαρξη διαδικτύου και βάσης δεδομένων, ούτε και η επιλογή ή η ανάλυση κάποιας συγκεκριμένης μεθόδου μιας και τα σημαντικά χαρακτηριστικά ψεύδους αναδύονται από το νευρωνικό δίκτυο με την χρήση αλγορίθμων εκμάθησης πάνω σε ψευδείς δημοσιεύσεις ή άρθρα.

Η εκπαίδευση ενός νευρωνικού δικτύου, πέρα από τα κείμενα των δημοσιεύσεων, μπορεί να γίνει και πάνω στις απαντήσεις των χρηστών στο εκάστοτε δημοσίευμα. Τα δύο αυτά πεδία εκπαίδευσης σε συνδυασμό με την εκπαίδευση σε χαρακτηριστικά της πηγής, αν πρόκειται για αναδημοσίευση, έχει δειχθεί ότι μπορεί να αποφέρει καλύτερα αποτελέσματα από τις υπόλοιπες μεθόδους στην ανίχνευση ψεύδους σε κοινωνικά δίκτυα (Ruchansky, Sungyong & Liu, 2017).

1.2.4 Συνδυασμός Μεθόδων

Παρόλο της αποτελεσματικότητας της εκάστοτε μεθόδου, δεν υφίσταται κάποια μεμονωμένη προσέγγιση η οποία να είναι καλύτερη από κάθε άλλη σε όλους τους τύπους κειμένου. Κάθε μέθοδος έχει τις αδυναμίες της, όπως το χαμηλό πλήθος λέξεων σε ένα δημοσίευμα (Ασκός Λέξεων) ή την ελλιπή σε καταχωρίσεις βάση δεδομένων (Συσχέτιση Δεδομένων). Μία προσέγγιση είναι σε θέση να καλύψει τα κενά μίας άλλης και για αυτό το λόγο τα ακριβέστερα αποτελέσματα μπορούν να προκύψουν από τον συνδυασμό πολλαπλών μεθόδων.

1.3 Μέθοδοι Εφαρμογής

Ο στόχος της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μιας εύχρηστης και ευκόλως εκπαιδεύσιμης εφαρμογής για την ανίχνευση ψεύδους σε ελληνικό κείμενο. Η υλοποίηση της θα βασιστεί σε μεθόδους της γλωσσικής ανάλυσης ενός κειμένου, λόγω της αποτελεσματικότητας τους και της αμεσότητας στην εκπαίδευση τους. Συγκεκριμένα θα χρησιμοποιηθούν οι μέθοδοι «Ορθογραφικός Έλεγχος», «Ασκός Λέξεων» και «Επιφανειακή Δομή». Ο ορθογραφικός έλεγχος, με την προϋπόθεση ότι επαρκεί το λεξικό που θα χρησιμοποιηθεί, δεν απαιτεί κάποια περαιτέρω εκπαίδευση. Η χρήση των μεθόδων «Ασκός Λέξεων» και «Επιφανειακή Δομή» δίνουν την δυνατότητα στην εφαρμογή να βελτιώνεται σταδιακά με την αύξηση των πηγών εκμάθησης, διατηρώντας παράλληλα την ικανότητας της για ανίχνευση ψεύδους.

Η συντακτική δομή που πηγάζει από την μέθοδο «Επιφανειακή Δομή» περιορίζεται στο πλήθος των γειτονικών ετικετών (ΜτΛ) και αδυνατεί να αντλήσει την πιθανότητα εμφάνισης μιας πρότασης μεγαλύτερης από το πλήθος αυτό. Η αύξηση του πλήθος δεν αποτελεί λύση, διότι το πλήθος λέξεων είναι διαφορετικό από πρόταση σε πρόταση και, επίσης, οι προτάσεις με μικρότερο μήκος από το πλήθος γειτονικών ετικετών δεν θα μπορούν να καταγραφούν. Έτσι, για μία πιθανώς καλύτερη προσέγγιση της συντακτικής δομής μία πρότασης θα χρησιμοποιηθούν επιπροσθέτως Μαρκοβιανές αλυσίδες (βλ. 2.4.5), το αποτέλεσμα των οποίων θα εξεταστεί κατά τον έλεγχο της ακρίβειας της εφαρμογής (βλ. 3.2).

Κεφάλαιο 2: Σχεδίαση

2.1 Προδιαγραφές

2.1.1 Σύντομη Περιγραφή

Η εφαρμογή θα αναπτυχθεί σε γλώσσα προγραμματισμού Java και θα κάνει έναν συνδυασμό των μεθόδων που αναφέρονται στην γλωσσική προσέγγιση για την απόδοση ποσοστού εγκυρότητας σε ένα κείμενο το οποίο είτε θα εξάγεται από κάποιον ιστότοπο ή αρχείο κειμένου, είτε θα εισάγεται απευθείας στην εφαρμογή.

Επιπλέον, η εφαρμογή θα μεταφράζει το δοθέν κείμενο στα Αγγλικά εκτελώντας σε αυτό τις ίδιες μεθόδους για την απόδοσης εγκυρότητας. Το κατά πόσο θα είναι χρήσιμη αυτή η διαδικασία θα κριθεί από τα αποτελέσματα που θα προκύψουν (βλ. 3.2). Επίσης, η ύπαρξη του κειμένου στα Αγγλικά ενδέχεται να εξυπηρετήσει σε μελλοντικές βελτιώσεις της εφαρμογής, καθότι επιτρέπει την προσαρμογή εργαλείων σε αυτήν, τα οποία μπορεί να μην διατίθενται για την ελληνική γλώσσα.

2.1.2 Λειτουργικές Απαιτήσεις

Ο **χρήστης** πρέπει να έχει τις επιλογές:

i. Να δηλώσει αν το εισαγόμενο κείμενο (ή σύνδεσμος) είναι ψευδές ή αληθές και έτσι να εκπαιδεύσει με αυτό τον τρόπο την εφαρμογή.
ii. Να εκπαιδεύσει την εφαρμογή μαζικά από δύο αρχεία κειμένου τα οποία θα περιέχουν συνδέσμους ιστοσελίδων χωρισμένους ανά γραμμή. Στο ένα αρχείο οι σύνδεσμοι θα θεωρούνται έγκυροι και στο άλλο όχι.
iii. Να ζητήσει από την εφαρμογή να κρίνει την εγκυρότητα ενός κειμένου ή συνδέσμου με βάση την έως τώρα εκπαίδευση της και να λαμβάνει από την εφαρμογή αιτιολόγηση για την αξιολόγηση της.

Η **εφαρμογή** θα πρέπει να έχει την δυνατότητα:

i. Να επιτελεί ορθογραφικό έλεγχο στο εισαγόμενο κείμενο.
ii. Να κρίνει ένα κείμενο βάσει του ποσοστού εμφάνισης μίας λέξης ή ζευγών λέξεων (bag-of-words, unigram και bigram).
iii. Να κρίνει ένα κείμενο βάσει του ποσοστού εμφάνισης μερών του λόγου (part-of-speech) στο κείμενο αυτό.
iv. Να κρίνει ένα κείμενο βάσει της σύνταξης των προτάσεων του.
v. Να μεταφράζει ένα ελληνικό κείμενο στα Αγγλικά.

2.1.3 Μη Λειτουργικές Απαιτήσεις

i. Ο ορθογραφικός έλεγχος θα γίνεται στο κείμενο με την χρήση της δωρεάν υπηρεσίας LanguageTool (https://languagetool.org), η οποία προσφέρει API για Java (http://wiki.languagetool.org/java-api). Η εφαρμογή θα βρίσκει το πλήθος των ορθογραφικών λαθών και βάσει αυτού θα αξιολογεί την εγκυρότητα του άρθρου. Ωστόσο ενδέχεται διάφορα ακρώνυμα ή ονόματα να εκλαμβάνονται ως ορθογραφικά λάθη, κάτι που ίσως να επηρεάσει αρνητικά την ακρίβεια της μεθόδου αυτής. Το πλήθος λαθών θα αποθηκεύεται σε έναν πίνακα της βάσης δεδομένων της εφαρμογής ξεχωριστά για ψευδείς και αληθείς ειδήσεις.

ii. Η απόδοση ετικετών στις λέξεις κατηγοριοποιώντας τες στο ανάλογο συντακτικό ρόλο που έχουν στην πρόταση (part of speech tagging) θα γίνεται με την χρήση κάποιου επισημειωτή που υποστηρίζει Ελληνικά, όπως τα:
   - AUEB Greek PoS Tagger (http://nlp.cs.aueb.gr/software.html)
   - RDR PoS Tagger (https://goo.gl/4tFN72)

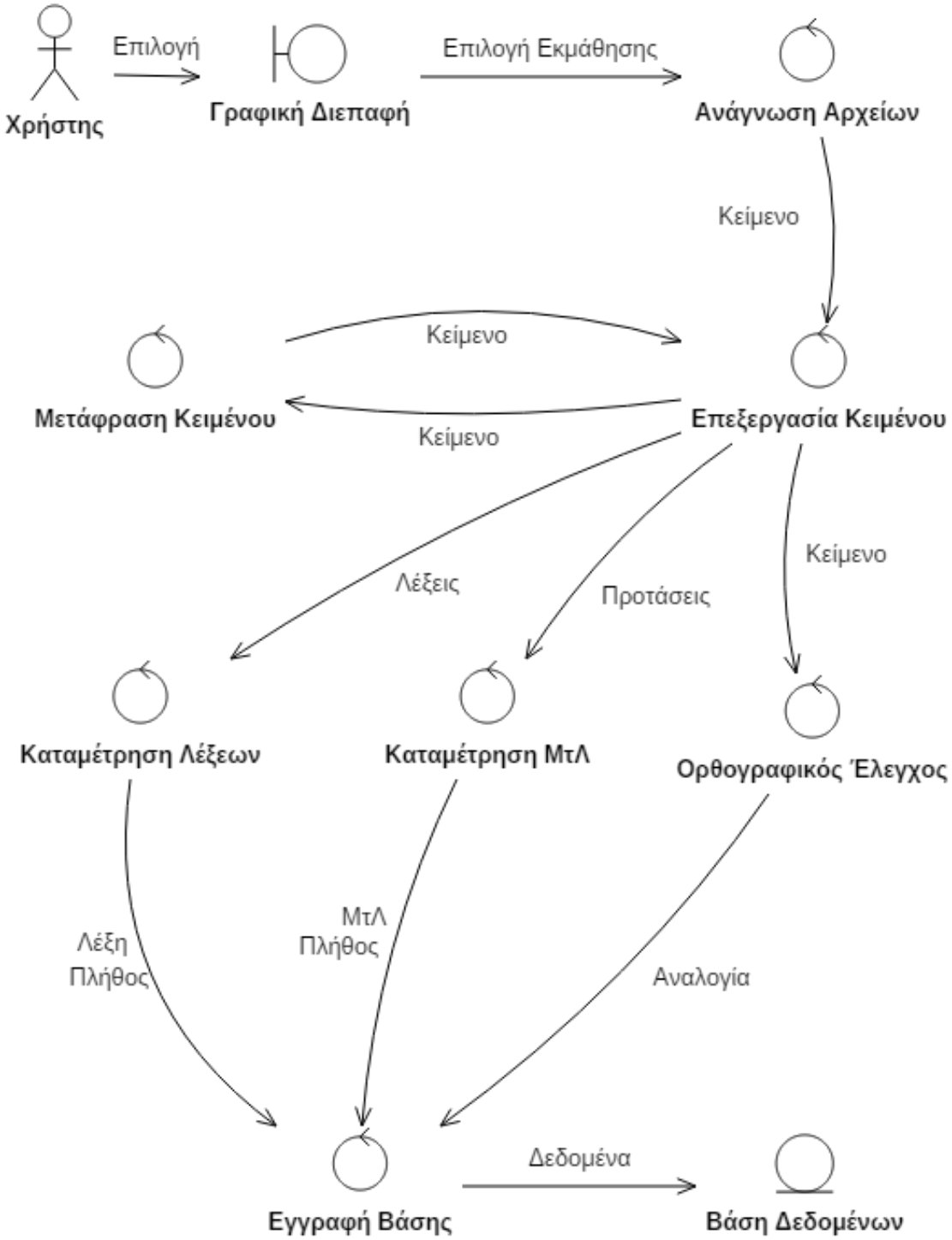
iii. Η μετάφραση του κειμένου θα γίνεται με την χρήση της υπηρεσίας Google Translate. Καθότι είναι δωρεάν η υπηρεσία που προσφέρει η Google, υποστηρίζεται η μετάφραση μόνο λίγων λέξεων για κάθε αίτημα. Έτσι, το κείμενο θα χωρίζεται σε προτάσεις οι οποίες κάθε μία θα μεταφράζεται ξεχωριστά και έπειτα θα ενώνονται πάλι σε ένα ενιαίο κείμενο.

iv. Ο συντακτικός έλεγχος θα γίνεται με την χρήση Μαρκοβιανών αλυσίδων (βλ. 2.4.5) στις ετικέτες των μερών του λόγου σε κάθε πρόταση. Οι αλυσίδες θα έχουν αρχικά μήκος δύο ετικέτες (bigram), ενώ στην συνέχεια, και εφόσον εξεταστεί το αν μπορεί επιτευχθεί μεγαλύτερη ακρίβεια με μεγαλύτερου μήκους αλυσίδες, θα αυξηθεί (βλ. 3.2.4).

v. Για την εκπαίδευση της εφαρμογής στο ποσοστό εμφάνισης κάθε λέξης, ζευγών λέξεων ή μερών του λόγου θα πρέπει να γίνεται καταμέτρηση της συχνότητας εμφάνισης αυτών σε κάθε κείμενο και αποθήκευση τους σε βάση δεδομένων ανάλογα με την εγκυρότητα του κειμένου. Η βάση θα αποτελείται από έναν πίνακα για κάθε μέθοδο ξεχωριστά. Ο κάθε πίνακας θα έχει τρεις στήλες, «word», «good» και «fake». Στην πρώτη στήλη αποθηκεύεται το χαρακτηριστικό αλφαριθμητικό της κάθε μεθόδου (λέξη, ζεύγος ή ΜτΛ), στην δεύτερη το πλήθος εμφάνισης του σε έγκυρα κείμενα και στην τρίτη το πλήθος εμφάνισης σε μη έγκυρα. Το ποσοστό για την κάθε κατηγορία εγκυρότητας θα προέρχεται από το πλήθος εμφάνισης προς το συνολικό πλήθος των αλφαριθμητικών για την συγκεκριμένη κατηγορία. Η αναλογία των δύο τελευταίων στηλών θα υποδεικνύει και το πόσο χρήσιμη θα είναι λέξη στον υπολογισμό της εγκυρότητας (Zhang, Fan, Zeng & Liu,2012).

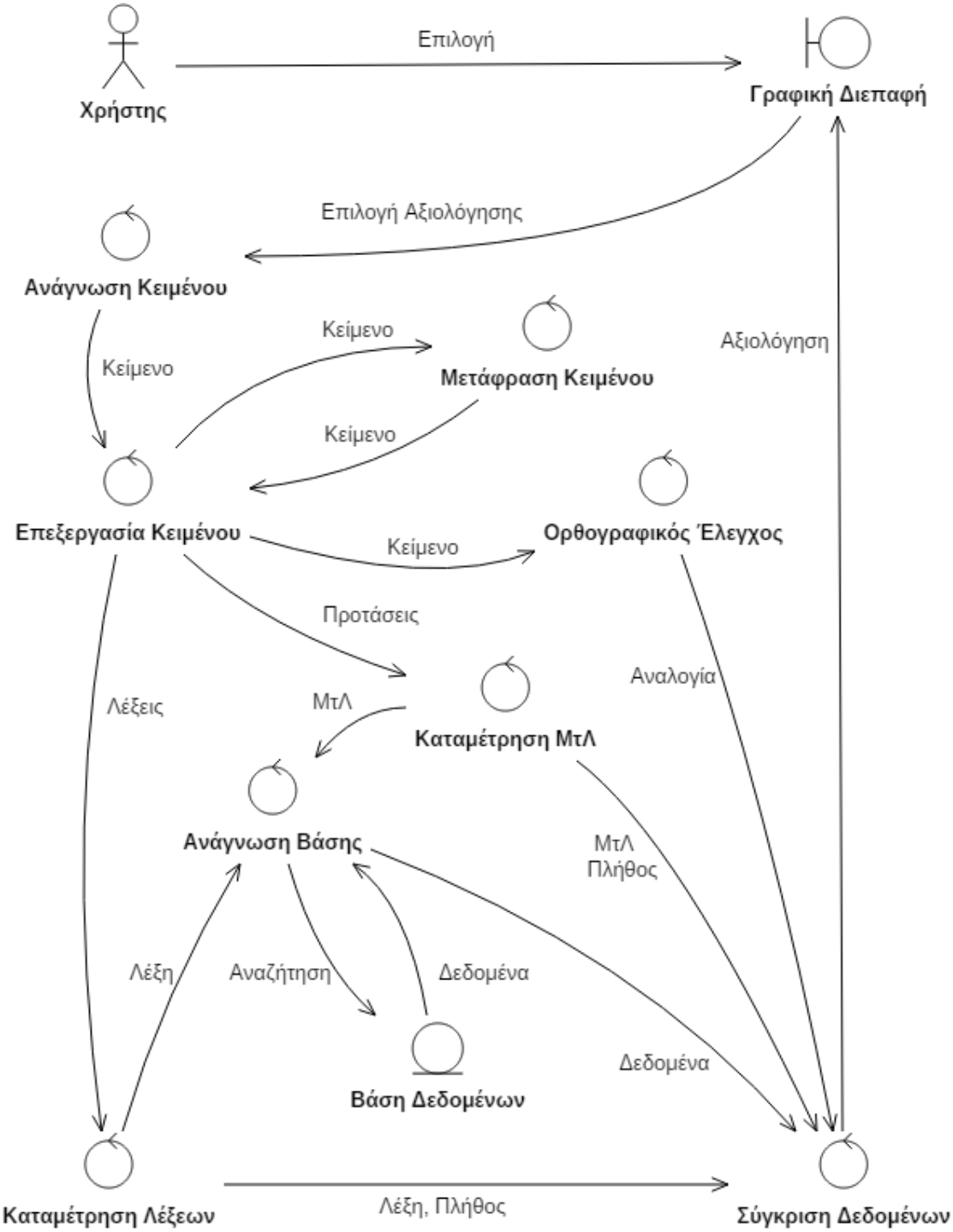
2.2 Περιπτώσεις Χρήσεως

2.2.1 Εκπαίδευση



Σχήμα 2.1: Διάγραμμα ευρωστίας για την ΠΧ Εκπαίδευση.

2.2.2 Αξιολόγηση



Σχήμα 2.2: Διάγραμμα ευρωστίας για την ΠΧ Αξιολόγηση.

2.2.3 Μονάδες Ελέγχου

Γραφική Διεπαφή: Το interface της εφαρμογής στο οποίο ο χρήστης δύναται να εισάγει κείμενο και να επιλέξει μία από τις περιπτώσεις χρήσης.

Ανάγνωση Αρχείων: Η μονάδα ελέγχου για την ανάγνωση αρχείων κειμένου εκμάθησης, τα οποία περιέχουν λίστα *URL* συνδέσμων.

Επεξεργασία Κειμένου: Η μονάδα ελέγχου για τον καθαρισμό του κειμένου από σημεία στίξης και ετικέτες *HTML*, καθώς και την εξόρυξη προτάσεων και λέξεων από το κείμενο.

Μετάφραση Κειμένου: Η μονάδα ελέγχου η οποία μεταφράζει το κείμενο στα Αγγλικά δημιουργώντας ένα νέο κείμενο προς επεξεργασία.

Καταμέτρηση Λέξεων: Η μονάδα ελέγχου που αντιστοιχεί σε ένα μέρος της μεθόδου bag of words, κάνει καταμέτρηση των λέξεων ενός κειμένου και αποστέλλει το πλήθος εμφάνισης της κάθε λέξης στην μονάδα εγγραφής για αποθήκευση στην βάση δεδομένων.

Καταμέτρηση ΜτΛ: Μονάδα ελέγχου καταμέτρησης μερών του λόγου (ΜτΛ), η οποία αντιστοιχεί στην μέθοδο bag of POS, κάνει καταμέτρηση και στέλνει τα αποτελέσματα στην μονάδα εγγραφής.

Ορθογραφικός Έλεγχος: Η μονάδα η οποία εκτελεί την καταμέτρηση των λαθών σε ένα κείμενο, υπολογίζει την αναλογία μεταξύ εσφαλμένων και συνολικού πλήθους λέξεων και την αποστέλλει προς εγγραφή στην αντίστοιχη μονάδα ελέγχου.

Εγγραφή Βάσης: Εκτελεί την εγγραφή στην βάση δεδομένων.

Ανάγνωση Κειμένου: Η μονάδα ελέγχου για την ανάγνωση κειμένου ή *URL* συνδέσμου από την διεπαφή**.**

Ανάγνωση Βάσης: Εκτελεί την ανάγνωση της βάση δεδομένων.

Σύγκριση Δεδομένων: Μονάδα ελέγχου σύγκρισης δεδομένων μεταξύ εισαγόμενου κειμένου και αποθηκευμένων στοιχείων.

2.3 Διευκρινίσεις

2.3.1 Ψευδοκώδικας

Παρόλο που η εργασία θα πραγματοποιηθεί σε γλώσσα προγραμματισμού Java, οι αλγόριθμοι θα περιγράφονται σε μία απλούστερη μορφή ψευδοκώδικα για να είναι ευανάγνωστη η διαδικασία που εκτελούν. Η σύνταξη του ψευδοκώδικα θα είναι indentation-based, όπου η εμφώλευση κάθε εντολής καθορίζεται από το πλήθος των εσοχών (tabs) που την προηγούνται. Επίσης, κάθε αλλαγή γραμμής θα θεωρείται ως τερματικό μίας εντολής.

```
String mem[] = getMembers();

for (int x = 0; x < 10; x++)
      if (mem[x] == “Helen”) {
            mem[x] = “Maria”;
            return true;
      }

return false;
```

```
mem[] = getMembers()

for (x = 0, x < 10, x++):
      if (mem[x] = “Helen”):
            mem[x] = “Maria”
            return(true)


return(false)
```

***Ψευδοκώδικας 2.1**: Ενδεικτική σύγκριση μεταξύ* Java *(αριστερά) και ψευδοκώδικα που θα χρησιμοποιθεί στην παρούσα πτυχιακή εργασία (δεξία).*

2.3.2 Ορολογία n-gram

Ο όρος n-gram αναφέρεται σε μια αλληλουχία n τεμαχίων, με το n να είναι φυσικός αριθμός. Το κάθε τεμάχιο, στην παρούσα πτυχιακή εργασία, μπορεί να είναι είτε μία λέξη, είτε μία ετικέτα μερών του λόγου. Για τιμές του n μεγαλύτερες του 4 δεν χρησιμοποιείται κάποια ονομασία και αναφέρονται ως 5-gram, 6-gram, 7-gram, κτλ. Ωστόσο, για τις αρχικές τιμές του n υπάρχουν κάποιες διαδεδομένες ονομάσιες, συγκεκριμά για:

- n = 1, unigram
- n = 2, bigram (ή digram)
- n = 3, trigram
- n = 4, quadgram (κάπως πιο σπάνια)

2.4 Μεθοδολογία

2.4.1 Ανάγνωση Κειμένου

Η διαδικασία ελέγχου ξεκινάει με την ανάγνωση ενός κειμένου, είτε αυτό είναι από αρχεία είτε από το πεδίο εισαγωγής της διεπαφής. Η διαδικασία έχει ως εξής:

1. Αν το κείμενο αρχίζει με "**http**" τότε αυτό αναγνωρίζεται ως σύνδεσμος, αν όχι τότε αναγνωρίζεται ως είδηση.
2. Στην περίπτωση συνδέσμου αντλείται όλο το κείμενο από την ιστοσελίδα και αποθηκεύεται ως αλφαριθμητικό στην μνήμη. Στην περίπτωση κειμένου η αποθήκευση γίνεται απευθείας.
3. Στην συνέχεια επεξεργάζεται με δύο διαφορετικούς τρόπους:
   i. Αφαιρούνται από το αλφαριθμητικό οποιοιδήποτε μη ελληνικοί χαρακτήρες και σύμβολα αλλαγής γραμμής, όπως Carriage Return και Line Feed, προετοιμάζοντας το κείμενο για επεξεργασία από τις υπόλοιπες μεθόδους.
   ii. Χωρίζεται σε προτάσεις, οι οποίες αποθηκεύονται σε έναν πίνακα έτσι ώστε μετέπειτα να μεταφραστεί η κάθε μία στα Αγγλικά.

```
if (startsWith("http", input)):
      input = getTextFromURL(input)
text = strip(input)
sentence[] = split(text, " ")
```

Ψευδοκώδικας 2.2: Αναγνώριση συνδέσμου από εισαγόμενο απλό κείμενο. Η μέθοδος startsWith *επιστρέφει* true *αν-ν το δεύτερο όρισμα της αρχίζει με το αλφαριθμιτικό της πρώτης παραμέτρου της. Η μέθοδος* getTextFromURL *επιστρέφει το κείμενο που υπάρχει στον σύνδεσμο, ενώ η* strip *το καθαρίζει από σύμβολα, άχρηστα προς την εφαρμογή. Τέλος, η* split *χωρίζει το αλφαριθμικό της πρώτης παραμέτρου σε λέξεις, τις οποίες τις επιστρέφει ως πίνακα.*

2.4.2 Μετάφραση Κειμένου

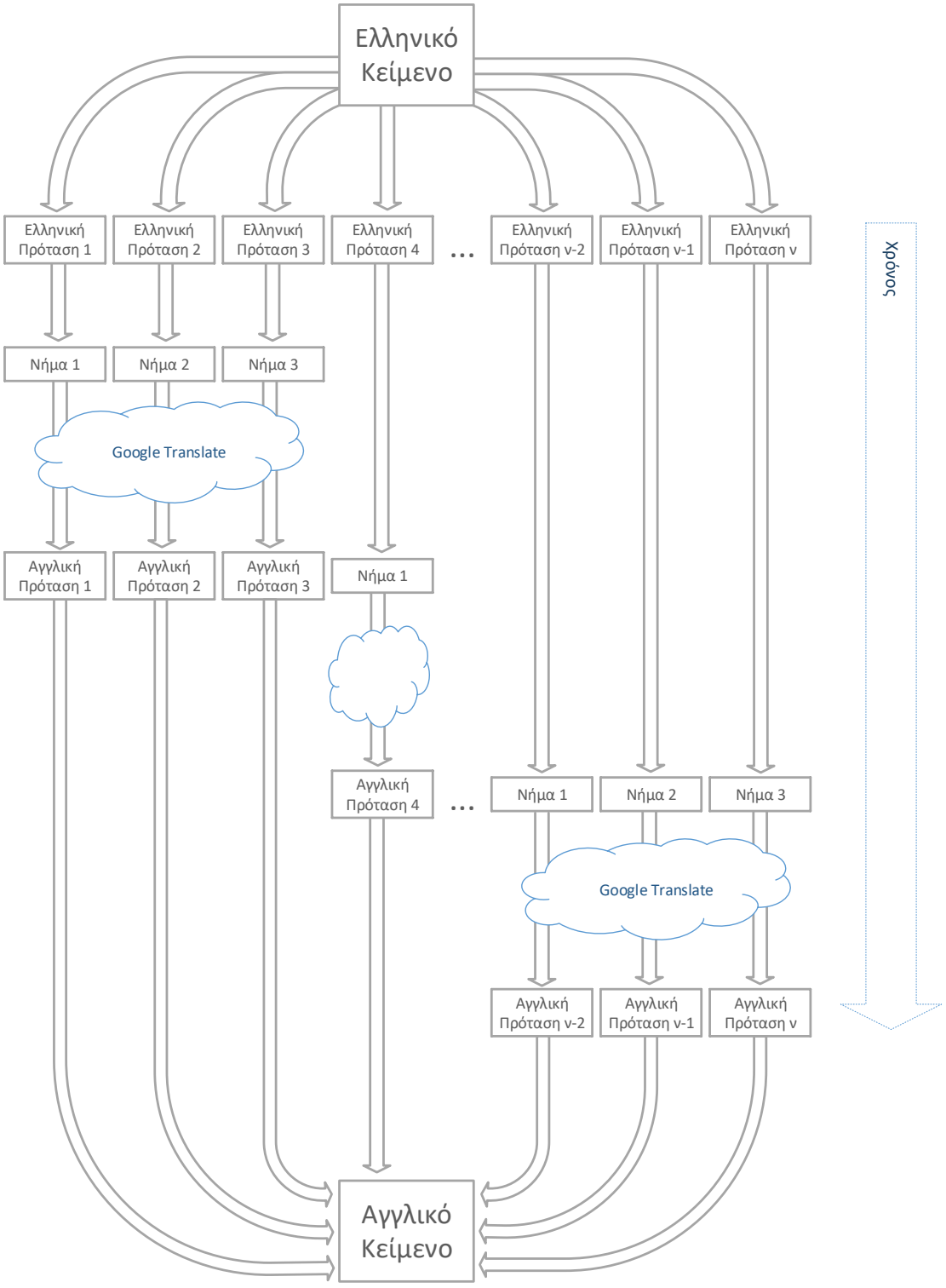
Για την μετάφραση του κειμένου, το κείμενο χωρίζεται στις προτάσεις του και κάθε πρόταση αποστέλλεται στο API μετάφρασης της Google το οποίο επιστρέφει την πρόταση μεταφρασμένη σε μορφή JSON. Λόγω του ότι υπάρχει καθυστέρηση μεταξύ αποστολής και λήψης αποτελέσματος, η οποία οφείλεται στην υπάρχουσα ταχύτητα σύνδεσης του υπολογιστή, για την επιτάχυνση της διαδικασίας δημιουργούνται 20 παράλληλα νήματα (threads) από τα οποία το καθένα αναλαμβάνει να στείλει και να λάβει το $\frac{1}{20}$ των προτάσεων, συν-πλην 1 για κάποια νήματα στην περίπτωση που το πλήθος των προτάσεων δεν είναι ακέραιο πολλαπλάσιο του 20. Όταν τελειώσουν όλα τα νήματα, οι προτάσεις ανασυγκροτούνται σε κείμενο το οποίο και ανατροφοδοτείται προς επεξεργασία από τις υπόλοιπες μεθόδους. Αν και το προκαθορισμένο πλήθος νημάτων είναι 20, μπορεί εύκολα να μεταβληθεί με την αλλαγή μίας σταθεράς στον πηγαίο κώδικα (βλ. 2.5.3, Translation). Στο σχήμα 2.3 αναπαριστάται η παραπάνω διαδικασία με 3 νήματα αντί για 20, για πρακτικούς λόγους.

```
sentence[] = getSentences(text)

for (id = 0, id < 20, id++):
      thread(sentence, id)

while (activeThreads()):
      wait()

return(sentence)
```

Ψευδοκώδικας 2.3: Διαχωρισμός του κειμένου σε προτάσεις και δημιουργία νημάτων με την μέθοδο thread. *Η* activeThreads *επιστρέφει true αν υπάρχουν νήματα που εκτελούν ακόμα μία διεργασία, ενώ η μέθοδος* wait *αναστέλει την λειτουργία του αλγορίθμου για μερικά χιλιοστά του δευτερολεύπτου. Στον Ψευδοκώδικα 2.4 περιγράφεται η λειτουργία του κάθε νήματος.*

```
for (c = 0, c < count(sentece), c++):
      if ( mod(c, 20) = id ):
            sentence[c] = translate(sentence[c])
```

Ψευδοκώδικας 2.4: Η λειτουργία ενός νήματος για την μετάφραση μίας πρότασης. Η μέθοδος mod *επιστρέφει το υπόλοιπο της διεραίσεως του πρώτου ορίσματος με το δεύτερο. Η μεταβλητή* id *δίδεται ως όρισμα κατά την δημιουργία του νήματος (Ψευδ. 2.3).*



***Σχήμα 2.3**: Σχηματική αναπαράσταση της μετάφρασης ενός κειμένου ν προτάσεων, με την χρήση τριών νημάτων. Στην συγκεκριμένη περίπτωση το ν είναι ακέραιο πολλαπλάσιο του 3.*

2.4.3 Βάση Δεδομένων

Η βάση δεδομένων υλοποιείται με SQLite3, η οποία προσφέρει φορητότητα σε σχέση με τις υπόλοιπες SQL με κόστος να είναι ελάχιστα πιο αργή. Στην βάση αποθηκεύεται ένας πίνακας για κάθε μέθοδο ο οποίος αποτελείται από τρία πεδία. Στο πρώτο πεδίο είναι το n-gram λέξεων ή ΜτΛ και στα άλλα δύο το πλήθος εμφάνισής του στην αντίστοιχη κατηγορία εγκυρότητας. Εύκολα μπορεί να υπολογιστεί το συνολικό πλήθος των λέξεων που έχουν αποθηκευτεί, και κατά συνέπεια να εξαχθεί το ποσοστό εμφάνισης κάθε λέξης ανά κατηγορία εγκυρότητας.

rowid	word	good	fake
1	αυτό	467	904
2	διασχίσει	2	0
3	εργαλεία	19	14
4	αποποίηση	2	3
5	επέπλεε	2	2
6	ένα	997	1253
7	μενού	18	30
8	σύλλογος	4	35
9	δωρεές	2	20
10	ωκεανό	7	12
11	γκιλγκαμές	1	0
12	αυτή	631	449
13	έγινε	85	212
14	άνθρωποι	103	97
15	περιστασιακά	1	1
16	κύρια	11	41
17	δίσκος	9	3
18	όλα	484	348
19	κολόμβου	1	0
20	είναι	3012	

rowid	word	good	fake
1	pron	13701	21165
2	cconj	8881	14564
3	aux	3909	4498
4	propn	69295	116388
5	sym	1828	6492
6	adj	39922	43357
7	num	15080	75065
8	sconj	3324	5294
9	adp	38398	47953
10	det	64610	76812
11	adv	15319	20844
12	punct	3477	16589
13	part	11288	15258
14	verb	33456	46903
15	x	15425	28249
16	noun	132042	182412

Σχήμα 2.4: Παράδειγμα πινάκων της βάσης δεδομένων για ελληνικό unigram (αριστερά) και ελληνικών ετικετών ΜτΛ (δεξιά).

2.4.4 Υπολογισμός Εγκυρότητας

Για τις μεθόδους bag-of-words και bag-of-PoS (βλ. 1.2.1), υπολογίζεται αρχικά το ποσοστό ύπαρξης μίας λέξης σε έγκυρο και μη έγκυρο κείμενο ξεχωριστά. Το ποσοστό ορίζεται ως το πλήθος εμφάνισης της λέξης ως προς το συνολικό πλήθος των λέξεων σε κάθε μία κατηγορία εγκυρότητας. Στην συνέχεια υπολογίζεται η ποσοστιαία διαφορά τους ως προς το άθροισμα τους, ως εξής:

Έστω X το ποσοστό ύπαρξης μίας λέξης σε έγκυρο κείμενο και Y το ποσοστό ύπαρξης σε μη έγκυρο, με $X, Y \in [0,1]$. Η ποσοστιαία διαφορά Q είναι:

$$Q = \begin{cases} \dfrac{X - Y}{X + Y}, & (X, Y) \neq (0,0) \\ 0, & (X, Y) = (0,0) \end{cases}$$

Έτσι, η ποσοστιαία διαφορά μπορεί να πάρει τιμές μεταξύ -1 και 1 και το ποσοστό εγκυρότητας της κάθε λέξης υπολογίζεται ως:

$$P = \frac{1 + Q}{2}, \qquad Q \in [-1,1]$$

Τέλος υπολογίζεται το ποσοστό εγκυρότητας του κειμένου, το οποίο είναι ο μέσος όρος των ποσοστών εγκυρότητας των λέξεων του κειμένου:

$$Z = \sum_{i=1}^{n} \frac{P_i}{n}, \qquad \text{όπου } n \text{ το πλήθος λέξεων}.$$

Για $Z < 0.5$ η είδηση θεωρείται μη έγκυρη, ειδάλλως έγκυρη.

```
words = getWordsWithCount(text)
total = getNumberOfWords(text)
veracity = 0

for (w = 0, w < count(words), w++):
	word = getKey(words, w)
	count = getValue(words, word)
	goodPercentage = getGoodPercentage(word)
	fakePercentage = getFakePercentage(word)

	percentage = 0
	if (goodPercentage != 0 || fakePercentage != 0):
		difPercentage = goodPercentage – fakePercentage
		sumPercentage = goodPercentage + fakePercentage
		percentage = difPercentage / sumPercentage

	veracity += (1 + percentage / 2) * count / total

return(veracity)
```

Ψευδοκώδικας 2.5*: Υπολογισμός της εκγυρότητας ενός κειμένου. Η μέθοδος getWordsWithCount επιστρέφει μία αντιστοίχηση λέξεων (κλειδί) και ποσοστού εμφάνισης (τιμή) από το κείμενο του ορίσματος της. Ο αλγόριθμος είναι παρόμοιος και για τις δύο προσεγγήσεις, Ασκός Λέξεων και Ασκός Μερών του Λόγου, με την διαφορά ότι η προαναφερθείσα μέθοδος επιστρέφει ετικέτες μερών του λογού στην δεύτερη περίπτωση αντί για λέξεις. Η μέθοδος getNumberOfWords επιστρέφει το πλήθος λέξεων του ορίσματος της. Η μέθοδος getGoodPercentage επιστρέφει το καταχωριμένο από την εκμάθηση ποσοστό εμφάνισης του ορίσματος της σε έγκυρες ειδήσεις, ενώ η getFakePercentage επιστρέφει το ποσοστό εμφάνισης σε μη έγκυρες. Τέλος, οι μέθοδοι getKey και getValue αντλούν το κλειδί και την αντίστοιχη τιμή από το πρώτο όρισμα τους.*

2.4.5 Συντακτική Εγκυρότητα

Ο έλεγχος της συντακτικής εγκυρότητας γίνεται ανά πρόταση με βάση το μοντέλο του Markov, με το οποίο μπορεί να εξαχθεί ένα ποσοστό από αλληλουχία ετικετών. Αρχικώς, καταμετρείται το ποσοστό εμφάνισης τετράδων γειτονικών ετικετών μερών του λόγου, κάτι που ούτως ή άλλως γίνεται στην «Καταμέτρηση ΜτΛ». Έχοντας την πιθανότητα εμφάνισης κάθε πιθανής τετράδας μπορεί να υπολογιστεί η πιθανότητα εμφάνισης της πρότασης ως εξής:

Έστω $P(q_i)$ η πιθανότητα εμφάνισης μίας τετράδας (quadgram) ετικετών q_i σε κάποιο κείμενο, i η σειρά εμφάνισης σε μία πρόταση και m το πλήθος λέξεων της πρότασης αυτής.

$$\text{Τότε } P(\text{πρότασης}) = P(q_{m-3}|q_{m-4}, q_{m-5}, \dots, q_3, q_2, q_1)$$
$$= P(q_{m-3})P(q_{m-4}|q_{m-5}, q_{m-6}, \dots, q_3, q_2, q_1) = \cdots$$
$$= P(q_{m-3})P(q_{m-4})P(q_{m-5}) \dots P(q_3)P(q_2)P(q_1)$$

Ο υπολογισμός της πιθανότητας εμφάνισης μίας πρότασης γίνεται δύο φορές, μία με την $P(q_i)$ να αντιστοιχεί στην πιθανότητα εμφάνισης του q_i σε μη έγκυρο κείμενο και μία σε έγκυρο. Έτσι, έχοντας τις πιθανότητες εμφάνισης της πρότασης για κάθε κατηγορία εγκυρότητας, μπορεί πλέον να γίνει η σύγκριση μεταξύ αυτών των δύο. Η διαδικασία είναι παρόμοια με τον Υπολογισμό Εγκυρότητας (2.4.4), όπου υπολογίζεται η ποσοστιαία διαφορά των πιθανοτήτων αυτών. Συγκεκριμένα:

Έστω $X = P(\text{πρόταση σε έγκυρο κείμενο})$ και $Y = P(\text{πρόταση σε μη έγκυρο κείμενο})$, τότε η ποσοστιαία διαφορά τους είναι:

$$Q = \begin{cases} \dfrac{X - Y}{X + Y}, & (X, Y) \neq (0,0) \\ 0, & (X, Y) = (0,0) \end{cases}$$

Έτσι, το ποσοστό εγκυρότητας της πρότασης είναι:

$$P = \frac{1 + Q}{2}, \qquad Q \in [-1,1]$$

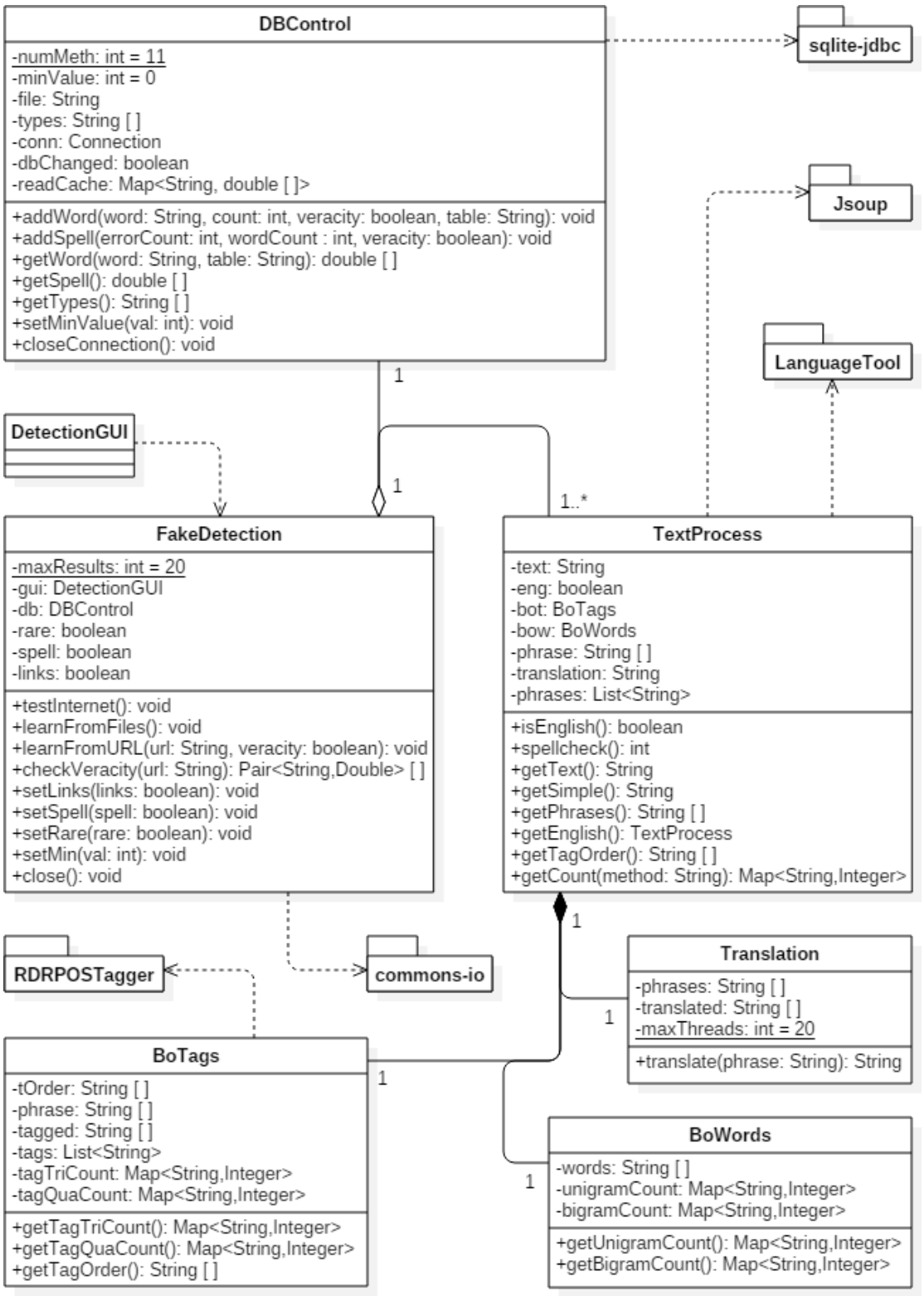
Και συνεπώς το ποσοστό εγκυρότητας του κειμένου είναι:

$$Z = \sum_{i=1}^{n} \frac{P_i}{n}, \qquad \text{όπου } n \text{ το πλήθος των προτάσεων.}$$

Για $Z < 0.5$ η είδηση θεωρείται μη έγκυρη, ειδάλλως έγκυρη.

2.5 Δομή Προγράμματος

2.5.1 Διάγραμμα Κλάσεων



Σχήμα 2.5: Διάγραμμα κλάσεων εφαρμογής ανίχνευσης ψευδών ειδήσεων.

2.5.2 Βιβλιοθήκες

RDRPOSTagger: Επισημειώνει το κείμενο με μέρη του λόγου βάσει προεκπαιδευμένων μοντέλων με επαρκή ακρίβεια (Nguyen & Pham, 2005). Έχει μοντέλα για 42 γλώσσες και στην παρούσα εφαρμογή θα χρησιμοποιηθεί για επισημείωση στο ελληνικό, αλλά και στο αγγλικό κείμενο. Στους παρακάτω πίνακες (Πίν. 2.1 & 2.2) παρουσιάζονται τα ΜτΛ, τα οποία δύναται να αναγνωριστούν σε ένα κείμενο.

Σύνδεσμος βιβλιοθήκης: https://github.com/bnosac/RDRPOSTagger.

Ετικέτα	**Περιγραφή**
ADJ	Επίθετο
ADP	Πρόθεση
ADV	Επίρρημα
AUX	Βοηθητικό Ρήμα
CCONJ	Συμπλεκτικός Σύνδεσμος
DET	Άρθρο
NOUN	Ουσιαστικό
NUM	Αριθμητικό

Ετικέτα	**Περιγραφή**
PART	Μόριο
PRON	Αντωνυμία
PROPN	Όνομα
PUNCT	Σημείο Στίξης
SCONJ	Μη Συμπλεκτικός Σύνδεσμος
SYM	Σύμβολο
VERB	Ρήμα
X	Οτιδήποτε Άλλο

Πίνακας 2.1: Μέρη του λόγου αναγνωρίσιμα σε ένα ελληνικό κείμενο.

SQLite JDBC: Η βιβλιοθήκη αυτή επιτρέπει την σύνδεση της εφαρμογής σε μία βάση δεδομένων SQLite. Η SQLite είναι ένας τύπος SQL βάσης δεδομένων, η οποία σε αντίθεση με τις συνηθισμένες SQL βάσεις μπορεί να περιέχεται σε ένα μόνο αρχείο και η πρόσβαση σε αυτή να γίνεται δίχως την μεσολάβηση κάποιου εξυπηρετητή. Αν και χαμηλότερη σε ταχύτητα απάντησης αιτημάτων (SQL queries) από τις υπόλοιπες, επιλέχτηκε για την φορητότητα της και την ευκολία δημιουργίας αντιγράφων ασφαλείας (backups).

Σύνδεσμος βιβλιοθήκης: https://github.com/xerial/sqlite-jdbc.

Jsoup: Η βιβλιοθήκη αυτή προσφέρει API για την εξόρυξη κειμένου και αντικειμένων από ένα HTML αρχείο. Με την Jsoup μπορεί να απομονωθεί κατά μεγάλο βαθμό το καθαρό κείμενο μία ιστοσελίδας από τα υπόλοιπα μη σημαντικά για την παρούσα εφαρμογή στοιχεία. Ωστόσο, επειδή δεν ακολουθούν όλες οι σελίδες την ίδια δομή και πόσο μάλλον όταν ο λόγος γίνεται για σελίδες που παρέχουν μη έγκυρες ειδήσεις, η διαρροή περιττών στοιχείων σε αυτό που θεωρείται καθαρό κείμενο είναι αναπόφευκτη.

Σύνδεσμος βιβλιοθήκης: https://jsoup.org/.

Commons IO: Παρέχει εργαλεία για συνήθεις λειτουργίες εισόδου και εξόδου που μπορεί να χρειαστεί μία εφαρμογή. Στην παρούσα εφαρμογή χρησιμοποιείται για την ανάγνωση αρχείων κειμένου τα οποία θα περιέχουν συνδέσμους ειδήσεων.

Σύνδεσμος βιβλιοθήκης: https://commons.apache.org/proper/commons-io/.

LanguageTool: Προσφέρει ορθογραφικό έλεγχο σε 29 γλώσσες εκ τις οποίες και Ελληνικά. Επίσης προσφέρει υποδείξεις για την διόρθωση των ορθογραφικών και επισήμανση συντακτικών λαθών, τα οποία ωστόσο δεν θα χρησιμοποιηθούν στην παρούσα εφαρμογή.

Σύνδεσμος βιβλιοθήκης: http://wiki.languagetool.org/java-api.

Ετικέτα	**Περιγραφή**	**Παράδειγμα**
CC	conjunction, coordinating	*and, or, but*
CD	cardinal number	*five, three, 13%*
DT	determiner	*the, a, these*
EX	existential there	*there were six boys*
FW	foreign word	*mais*
IN	conjunction, subordinating or preposition	*of, on, before, unless*
JJ	adjective	*nice, easy*
JJR	adjective, comparative	*nicer, easier*
JJS	adjective, superlative	*nicest, easiest*
LS	list item marker	
MD	verb, modal auxillary	*may, should*
NN	noun, singular or mass	*tiger, chair, laughter*
NNS	noun, plural	*tigers, chairs, insects*
NNP	noun, proper singular	*Germany, God, Alice*
NNPS	noun, proper plural	*we met two Christmases ago*
PDT	predeterminer	*both his children*
POS	possessive ending	*'s*
PRP	pronoun, personal	*me, you, it*
PRP$	pronoun, possessive	*my, your, our*
RB	adverb	*extremely, loudly, hard*
RBR	adverb, comparative	*better*
RBS	adverb, superlative	*best*
RP	adverb, particle	*about, off, up*
SYM	symbol	*%*
TO	infinitival to	*what to do?*
UH	interjection	*oh, oops, gosh*
VB	verb, base form	*think*
VBZ	verb, 3rd person singular present	*she thinks*
VBP	verb, non-3rd person singular present	*I think*
VBD	verb, past tense	*they thought*
VBN	verb, past participle	*a sunken ship*
VBG	verb, gerund or present participle	*thinking is fun*
WDT	*wh*-determiner	*which, whatever, whichever*
WP	*wh*-pronoun, personal	*what, who, whom*
WP$	*wh*-pronoun, possessive	*whose, whosever*
WRB	*wh*-adverb	*where, when*

Πίνακας 2.2: Μέρη του λόγου αναγνωρίσιμα σε ένα αγγλικό κείμενο. Κάποια σημεία στίξης που αναγνωρίζονται ξεχωριστά από τα υπόλοιπα, παραλείπονται από τον πίνακα.

2.5.3 Κλάσεις

TextProcess: Είναι υπεύθυνη για την εξαγωγή κειμένου από URL διευθύνσεις ή την λήψη κειμένου και τον καθαρισμό του από σημεία στίξεις, επιπλέον κενά και *HTML* ετικέτες χρησιμοποιώντας την βιβλιοθήκη *Jsoup*. Απαρτίζεται από τις κλάσεις *BoWords*, *BoTags* και *Translation*, από τις οποίες αντλεί λέξεις, προτάσεις και μέρη του λόγου από το κείμενο. Χρησιμοποιεί επίσης την βιβλιοθήκη *LanguageTool* για ορθογραφικό έλεγχο, υλοποιώντας έτσι τις μονάδες ελέγχου «*Επεξεργασία Κειμένου*» και «*Ορθογραφικός Έλεγχος*».

DBControl: Υπεύθυνη κλάση για την ενημέρωση και την ανάγνωση και εγγραφή της βάσης δεδομένων με την χρήση της βιβλιοθήκης *SQLite JDBC*. Υλοποιεί, δηλαδή, τις μονάδες «Εγγραφή Βάσης» και «Ανάγνωση Βάσης» και οποιαδήποτε ενέργεια γίνεται στην βάση δεδομένων, γίνεται αποκλειστικά μέσω αυτής της κλάσης. Επίσης, εφόσον η κλάση θα πρέπει να δημιουργεί κατά την αρχικοποίηση έναν πίνακα για κάθε μέθοδο που θα χρησιμοποιηθεί, σε αυτή την κλάση ορίζεται και το πλήθος των μεθόδων ως σταθερά ονόματι *numMeth*.

BoWords: Η υλοποίηση της μεθόδου bag-of-words και της μονάδας ελέγχου «Καταμέτρηση Λέξεων», η οποία με την λήψη κάποιου κειμένου απαριθμεί το πλήθος λέξεων και δυάδων γειτονικών λέξεων (unigram και bigram). Οι κύριες μεταβλητές που χρειάζεται είναι ένας πίνακας όλων των λέξεων κατά σειρά εμφάνισης και το πλήθος εμφάνισης μεμονωμένων και δυάδων λέξεων, τα οποία αντιστοιχούν στις μεταβλητές *words*, *unigramCount* και *bigramCount*. Οι δύο τελευταίες αναλύονται πιο λεπτομερώς στις μεθόδους $getUnigramCount()$ και $getBigramCount()$ που τις επιστρέφουν (βλ. 2.5.4).

```
word[] = split(text, “ “)
for (c = 0, c < count(word), c++):
       increase(unigramCount, word[c])
       if (c < count(word) - 1):
              bigram = word[c].” “.word[c+1]
              increase(bigramCount, bigram)
```

***Ψευδοκώδικας 2.6**: Αλγόριθμος καταμέτρησης των* unigrams *και* bigrams *ενός κειμένου. Η μέθοδος increase αυξάνει τον μετρητή της λέξεως που αντιστοιχεί στο δεύτερο όρισμα κατά 1 στην μεταβλητή της πρώτης παραμέτρου.*

BoTags: Η υλοποίηση της μεθόδου bag-of-POS, η οποία με την χρήση της βιβλιοθήκης επισημειωτή *RDRPoSTagger* πραγματοποιεί καταμέτρηση των μερών του λόγου του κειμένου που της δόθηκε. Παρόμοια με την κλάση *BoWords*, χρησιμοποιεί τις μεταβλητές *tags*, *tagTriCount* και *tagQuaCount*, για την σειρά εμφάνισης των ετικετών, το πλήθος των τριάδων γειτονικών ετικετών και το πλήθος των τετράδων γειτονικών ετικετών αντίστοιχα. Κατά συνέπεια υλοποιεί την μονάδα ελέγχου «Καταμέτρηση ΜτΛ».

```
tag[] = getTags(text)
for (t = 0, t < count(tag) - 2, t++):
      trigram = tag[t].” “.tag[t+1].” “.tag[t+2]
      increase(tagTriCount, trigram)
      if (t < count(tag) - 3):
            quadgram = tag[t].” “.tag[t+1].” “.tag[t+2].” “.tag[t+3]
            increase(tagQuaCount, quadgram)
```

Ψευδοκώδικας 2.7: Αλγόριθμος καταμέτρησης των trigrams *και* quadgrams *ετικετών ενός κειμένου. Η μέθοδος* increase *αυξάνει τον μετρητή, στην μεταβλητή της πρώτης παραμέτρους, της λέξεως που αντιστοιχεί στο δεύτερο όρισμα κατά 1. Η μέθοδος* getTags *αντλύει τις ετικέτες από το κείμενο και τις επιστρέφει σε έναν μονοδιάστατο πίνακα (array) διατηρώντας την σειρά εμφάνισης τους στο κείμενο.*

FakeDetection: Η κεντρική κλάση της εφαρμογής η οποία είναι υπεύθυνη για την ανάγνωση κειμένου ή *URL* συνδέσμου από το interface ή από αρχείο με την χρήση της βιβλιοθήκης *Commons IO,* υλοποιώντας τις μονάδες ελέγχου «*Ανάγνωση Αρχείων*» και «*Ανάγνωση Κειμένου*». Επίσης, υλοποιεί την μονάδα «*Σύγκριση Δεδομένων*» εκδίδοντας κάποιο αποτέλεσμα για τον έλεγχο εγκυρότητας του κειμένου. Η σταθερά *maxResults* καθορίζει τα το πλήθος των αποτελεσμάτων που θα επιστρέφονται από την μέθοδο $checkVeracity(String\ url)$ (βλ. 2.5.4), ενώ οι μεταβλητές *rare*, *spell* και *links* αποτελούν διακόπτες για την καταμέτρηση σπάνιων λέξεων, για την εκτέλεση ορθογραφικού ελέγχου και για την διαγραφή μη προσβάσιμων ιστοτόπων. Επιπροσθέτως, η κλάση αυτή, παρέχει και *API* για την διευκόλυνση της προσαρμογής της εφαρμογής σε κάποια τρίτη (βλ. 4.2.5).

Translation: Μία εκτελέσιμη (callable) κλάση, η οποία αναλαμβάνει την μετάφραση του κειμένου που της δίνεται από τα Ελληνικά στα Αγγλικά, επιστρέφοντας ένα νέο αντικείμενο *TextProcess* με το αγγλικό κείμενο. Η μετάφραση εκτελείται με την αποδόμηση του κείμενου σε προτάσεις και με την χρήση πολλαπλών νημάτων στέλνει αίτημα μετάφρασης στην υπηρεσία Translate της Google για κάθε πρόταση (βλ. 2.4.2). Το πλήθος των νημάτων ορίζεται από την σταθερά *maxThreads* και είναι προκαθορισμένο στα 20. Ολόκληρη η λειτουργία της κλάσης αντιστοιχεί στην μονάδα ελέγχου «Μετάφραση Κειμένου».

DetectionGUI: Το interface της εφαρμογής, για εισαγωγή κειμένου και εμφάνισης αποτελεσμάτων. Υλοποιεί την μονάδα ελέγχου «*Γραφική Διεπαφή*» χρησιμοποιώντας την κλάση *FakeDetection* για το κύριο μέρος των λειτουργιών της.

2.5.4 Μέθοδοι

Για την κλάση ***BoWords***:

$getUnigramCount()$: Επιστρέφει την μεταβλητή *unigramCount*, η οποία είναι ένα αντικείμενο Map, δηλαδή μία λίστα δυάδων αντικειμένων άμεσα συνδεδεμένων. Στην παρούσα περίπτωση, σε κάθε εγγραφή του Map έχουμε ένα αντικείμενο String συνδεδεμένο με ένα αντικείμενο Integer και πιο συγκεκριμένα, έχουμε αντιστοίχιση κάθε μεμονωμένης λέξης (unigram) με έναν ακέραιο ο οποίος αντιπροσωπεύει το πλήθος εμφανίσεων της λέξης αυτή στο κείμενο. Κάθε λέξη είναι μοναδική (κλειδί) μέσα στο Map.

$getBigramCount()$: Όμοια με την προηγούμενη μέθοδο, επιστρέφει ένα αντικείμενο Map<String,Integer>, την μεταβλητή *bigramCount*. Εν αντιθέσει με την προηγούμενη μέθοδο ωστόσο, αυτή την φορά αντί για μεμονωμένη λέξη στο αντικείμενο String, περιέχεται μία δυάδα γειτονικών λέξεων (bigram).

Για την κλάση ***BoTags***:

$getTagTriCount()$: Επιστρέφει ένα αντικείμενο Map<String,Integer>, το οποίο στο αντικείμενο String περιέχει μία τριάδα γειτονικών ετικετών (ΜτΛ, trigram). Αρχικώς υπήρχε μέθοδος η οποία επέστρεφε μεμονωμένη ετικέτα, ωστόσο αυτό άλλαξε κατά την ολοκλήρωση της εφαρμογής (βλ. 3.2.4).

$getTagQuaCount()$: Όμοια με την προηγούμενη μέθοδο, αρχικώς επέστρεφε δυάδα ετικετών, ωστόσο παρατηρήθηκε στις δοκιμές ότι η τετράδα γειτονικών ετικετών (quadgram) ήταν περισσότερο ακριβής.

Για την κλάση **DBControl**:

$addWord(String\ word, int\ count, boolean\ veracity, String\ table)$:
Ενημερώνει την καταχώρηση *word* στον πίνακα *table* στην βάση δεδομένων, αν αυτή υπάρχει ήδη, προσθέτοντας το πλήθος εμφάνισης της count στο ήδη υπάρχον ανάλογα με την εγκυρότητα (*veracity*) που έχει δηλωθεί. Αν δεν υπάρχει η καταχώρηση *word*, την εισάγει με αρχική τιμή το *count*. Το *table* πρέπει να αντιστοιχεί σε κάποια από τις μεθόδους που έχουν δηλωθεί στον πίνακα *types*.

$addSpell(int\ errorCount, int\ wordCount, boolean\ veracity)$:
Ενημερώνει τον πίνακα spell της βάσης δεδομένων προσθέτοντας το πλήθος λαθών και το πλήθος λέξεων που ελέγχθηκαν στην στήλη με την ανάλογη εγκυρότητα (*veracity*).

$getWord(String\ word, String\ table)$: Επιστρέφει την αναλογία του πλήθους εμφάνισης της λέξης *word* ως προς το συνολικό πλήθος εγγραφών στον πίνακα *table* για κάθε κατηγορία εγκυρότητας ξεχωριστά σε μορφή πίνακα. Επίσης εδώ γίνεται η χρήση της μεταβλητής readCache η οποία αποτελεί μία προσωρινή μνήμη για τις λέξεις που ζητούνται με αυτή την μέθοδο (Ψευδ. 2.8). Δηλαδή, αν η λέξη έχει ζητηθεί ξανά στο παρελθόν και δεν έχουν γίνει αλλαγές στην βάση δεδομένων, τότε δεν αποστέλλεται αίτημα (SQL query) στην βάση δεδομένων, αλλά ανακτάται άμεσα το ποσοστό εγκυρότητας από την μεταβλητή. Αυτό επιταχύνει αρκετά τον έλεγχο εγκυρότητας, ειδικά όταν η βάση δεδομένων είναι μεγάλη.

$setMinValue(int\ val)$: Θέτει την μεταβλητή $minValue$ σε val. Η $minValue$ χρησιμοποιείται ως η αρχική τιμή σε πλήθος εμφανίσεων με την οποία μπορεί να αποθηκευτεί στην βάση δεδομένων μία λέξη. Αν τεθεί πάνω από μηδέν, τότε βοηθάει στην εξομάλυνση των αποτελεσμάτων σε περιπτώσεις που μία λέξη έχει βρεθεί κατά την εκπαίδευση μόνο σε μία από τις δύο κατηγορίες εγκυρότητας.

$getTypes()$: Επιστρέφει τις κωδικές ονομασίες των μεθόδων που χρησιμοποιούνται, οι οποίες είναι δηλωμένες στην μεταβλητή $types$.

$getSpell()$: Επιστρέφει το ποσοστό λάθους για κάθε κατηγορία εγκυρότητας.

$closeConnection()$: Ελευθερώνει την σύνδεση με την βάση δεδομένων άμεσα, αντί για την προκαθορισμένη λήξη της (timeout).

```
percentage[] = get(readCache, word)

if (percentage = NULL):
      percentage[] = getFromDB(table, word)
      put(readCache, word, percentage)

return(percentage)
```

Ψευδοκώδικας 2.8: Ο αλγόριθμος της μεθόδου getWords *σε ψευδοκώδικα. Η* readCache *είναι μία μεταβλητή που περιέχει μία αντιστοιχία ενός κλειδιού (λέξη) και μίας τιμής (ποσοστό). Η μέθοδος* get *επιστρέφει την τιμή της λέξεως (δεύτερο όρισμα) από την μεταβλητή* readCache *(πρώτο όρισμα), ενώ η μέθοδος put εκτελεί το αντίθετο βάζοντας το κλειδί (δεύτερο όρισμα) και την τιμή (τρίτο όρισμα) στην μεταβλητή* readCache*. Η μέθοδος* getFromDB *επιστρέφει το ποσοστό εμφάνισης της λέξης (δεύτερο όρισμα) από το ζητούμενο πίνακα (πρώτο όρισμα). Το ποσοστό που επιστρέφεται είναι σε μορφή πίνακα καθότι περιέχει δύο τιμές, μία για το ποσοστό στις έγκυρες ειδήσεις και μία για τις μη έγκυρες.*

Για την κλάση ***FakeDetection***:

$learnFromURL(String\ url, boolean\ veracity)$: Εκτελεί τις μεθόδους εκμάθησης για το αλφαριθμητικό *url*, το οποίο μπορεί να είναι είτε κείμενο είτε σύνδεσμος κάποιου ιστοτόπου. Το veracity καθορίζει το αν θα αποθηκευτούν τα αποτελέσματα των μεθόδων ως έγκυρα ή μη, για τις τιμές true ή false αντίστοιχα.

$checkVeracity(String\ url)$: Ελέγχει το δοθέν σύνδεσμο ή κείμενο για την εγκυρότητα του και επιστρέφει έναν πίνακα ζευγών (Pair) αντικειμένων String και Double, όπου στο πρώτο αναγράφεται η αιτιολόγηση της αξιολόγησης του συνδέσμου και στο δεύτερο το ποσοστό εγκυρότητα του. Το ποσοστό εγκυρότητας είναι μεταξύ 0 και 1, με 1 να είναι το πιο έγκυρο.

$learnFromFiles()$: Εκτελεί την $learnFromURL(url, veracity)$ για κάθε γραμμή του των αρχείων κειμένου *good.txt* και *fake.txt* (βλ. 3.1.5), με *veracity* true και false αντίστοιχα.

$testInterent()$: Εξετάζει την ύπαρξη συνδεσιμότητας στο διαδύκτιοο δοκιμάζοντας να συνδεθεί στα domain google.com, amazon.com και yahoo.com. Εκτελείται όταν ένας ιστότοπος δεν είναι προσβάσιμος για να καθοριστεί με ακρίβεια το αν φταίει ο ιστότοπος ή σύνδεση του υπολογιστή.

$setLinks(boolean\ links)$: Ενεργοποιεί ή απενεργοποιεί την διαγραφή μη προσβάσιμων συνδέσμων από τα αρχεία κατά την εκτέλεση της εκμάθησης. Χρησιμοποιείται η μέθοδος $testInternet()$ για την εξακρίβωση της προσβασιμότητας του ιστοτόπου.

$setSpell(boolean\ spell)$: Ενεργοποιεί ή απενεργοποιεί τον ορθογραφικό έλεγχο κατά την διάρκεια της εκμάθησης και του ελέγχου εγκυρότητας.

$setRare(boolean\ rare)$: Διακόπτης για την αγνόηση ή μη σπάνια εμφανιζόμενων λέξεων σε κείμενα για την εξομάλυνση των αποτελεσμάτων του ελέγχου εγκυρότητας (βλ. 3.1.3, Ignore Rare).

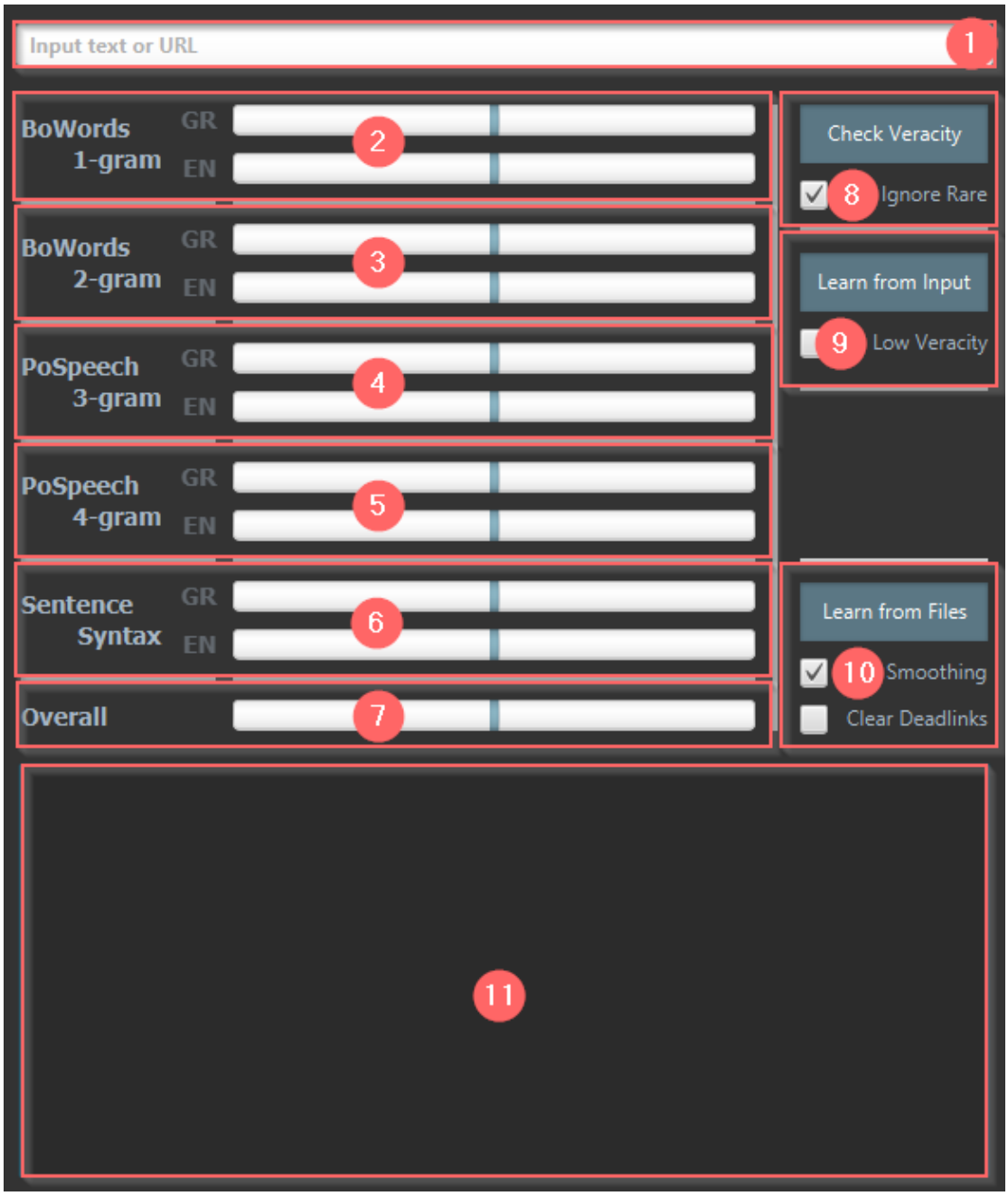
$setMin(int\ val)$: Θέτει το minValue της DBControl (βλ. $setMinValue()$).

$close()$: Εκτελεί την μέθοδο $closeConnection()$ της *DBControl*, τερματίζει και απελευθερώνει οποιοδήποτε άλλο πόρο δεσμευμένο από την *FakeDetection*.

Κεφάλαιο 3: Εφαρμογή

3.1 Χρήση Διεπαφής

3.1.1 Επισκόπηση



***Σχήμα 3.1**: Γραφική διεπαφή εφαρμογής ανίχνευσης ψευδών ειδήσεων.*

Περιγραφή Ετικετών (Σχ. 3.1)

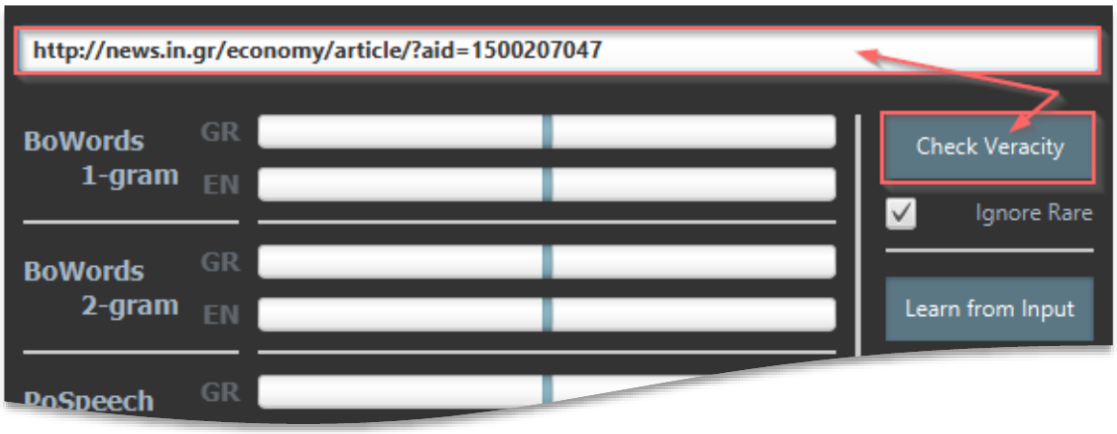
1. Πεδίο εισαγωγής κειμένου ή διεύθυνσης URL, η οποία περιέχει κάποια είδηση.
2. Ποσοστό εγκυρότητας για μεμονωμένες λέξεις του κειμένου.
3. Ποσοστό εγκυρότητας για ζευγάρια γειτονικών λέξεων.
4. Ποσοστό εγκυρότητας τριάδων ετικετών μερών του λόγου.
5. Ποσοστό εγκυρότητας τετράδων ετικετών μερών του λόγου.
6. Ποσοστό συντακτικής εγκυρότητας προτάσεων κειμένου.
7. Ποσοστό συνολικής εγκυρότητας ειδήσεως.
8. Λειτουργίες ανίχνευσης εγκυρότητας σε εισαγόμενο κείμενο/URL.
9. Λειτουργίες εκμάθησης από εισαγόμενο κείμενο/URL.
10. Λειτουργίες μαζικής εκμάθησης από αρχεία κειμένων.
11. Πλαίσιο εκτύπωσης αποτελεσμάτων.

3.1.2 Εισαγωγή Κειμένου

Στο πεδίο εισαγωγής (1, Σχ. 3.1) δίδεται η δυνατότητα στον χρήστη να γράψει ή επικολλήσει ένα κείμενο ή έναν σύνδεσμο ιστοτόπου. Ο διαχωρισμός αυτών των δύο γίνεται αυτόματα από την εφαρμογή όπως περιγράφεται στην μεθοδολογία (βλ. 2.4.1). Ωστόσο χρειάζεται προσοχή από τον χρήστη κατά την επικόλληση κειμένου, καθότι το συγκεκριμένο πεδίο απογυμνώνει το κείμενο από χαρακτήρες αλλαγής γραμμής μετατρέποντας το κείμενο σε μία γραμμή. Αυτό έχει ως αποτέλεσμα οι λέξεις στις οποίες μεσολαβούσε ο χαρακτήρας αλλαγής γραμμής να κολλάνε μεταξύ τους, κάτι που δύναται να προκαλέσει μικρή αύξηση της αστοχίας της εφαρμογής για τον συγκεκριμένο έλεγχο. Προς αποφυγήν αυτού, το ιδανικό είναι να αντικαθίστανται οι χαρακτήρες αλλαγής γραμμής από τον χαρακτήρα του κενού (space) πριν την επικόλληση του κειμένου.

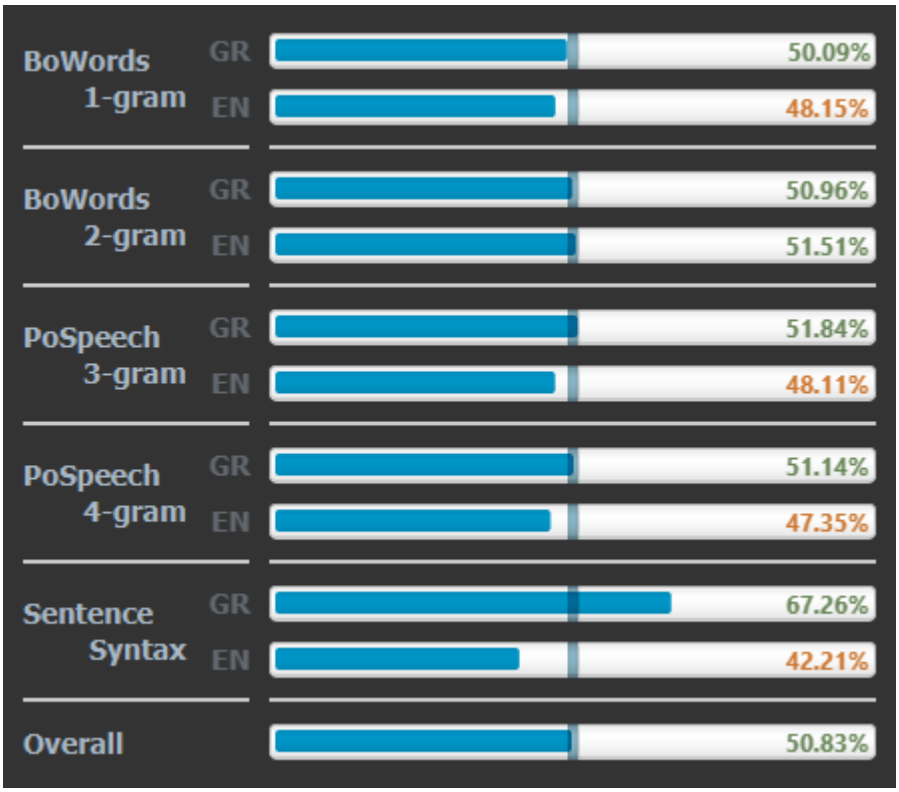
3.1.3 Έλεγχος Εγκυρότητας

Η διαδικασία ελέγχου εγκυρότητας ενός κειμένου ξεκινάει με τον χρήστη να συμπληρώνει το πεδίο εισαγωγής με έναν σύνδεσμο ή κείμενο και στην συνέχεια να πατάει το κουμπί έναρξης της διαδικασίας, «**Check Veracity**» (Σχ. 3.2). Η καταχώρηση αρχίζει να ελέγχεται σειριακά για κάθε μία από της μεθόδους ανίχνευσης, με τα αποτελέσματα να εμφανίζονται σταδιακά με το πέρας κάθε ελέγχου.



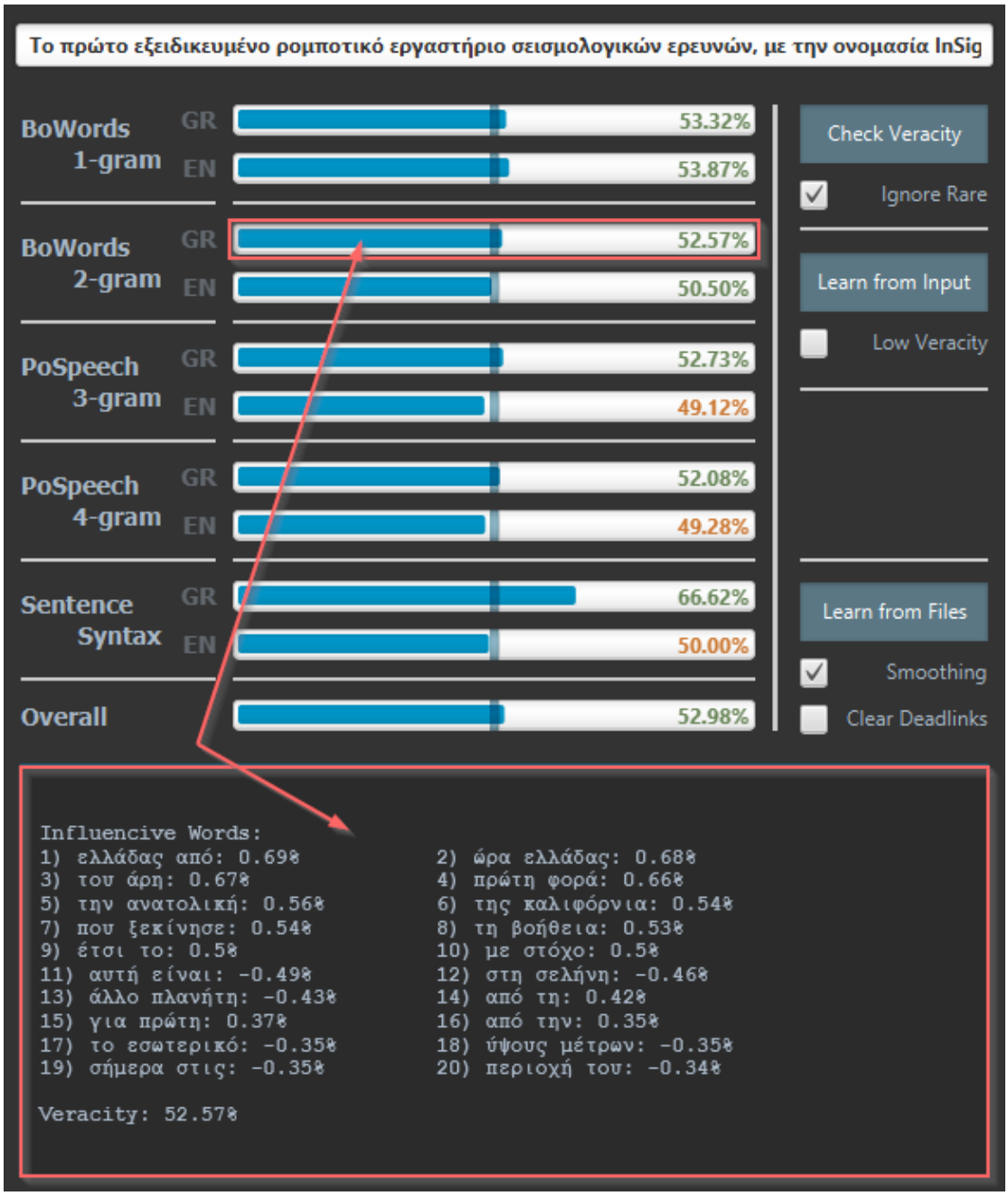
Σχήμα 3.2: Έλεγχος εγκυρότητας ενός συνδέσμου ειδήσεως.

Με την ολοκλήρωση του ελέγχου ο χρήστης μπορεί να δει τα αποτελέσματα στις μπάρες κάθε μεθόδου. Αν το ποσοστό είναι πάνω από το 50% η είδηση θεωρείται έγκυρη. Κάθε μέθοδος έχει μία πιθανότητα αστοχίας και συνεπώς δεν συμφωνούν πάντοτε τα αποτελέσματα μεταξύ τους.



Σχήμα 3.3: Αποτελέσματα ελέγχου ενός συνδέσμου ειδήσεως.

Ο χρήστης έπειτα έχει την επιλογή, πατώντας σε μία από τις μπάρες να δει τον λόγο για τον οποίο βγήκε το αποτέλεσμα αυτό. Όπως φαίνεται στο επόμενο screenshot (Σχ. 3.4), με το πάτημα μίας μπάρας εμφανίζονται στην κονσόλα οι λέξεις, τα ζευγάρια λέξεων ή τα μέρη του λόγου που επηρέασαν περισσότερο το αποτέλεσμα είτε αρνητικά, είτε θετικά. Το αρνητικό πρόσημο δηλώνει ότι οι αντίστοιχες λέξεις έχουν ευρεθεί κατά μεγαλύτερο ποσοστό σε μη έγκυρα κείμενα, με συνέπεια να επηρεάζουν αρνητικά την εγκυρότητας του κειμένου.



Σχήμα 3.4: Αιτιολόγηση αποτελεσμάτων ελέγχου ενός συνδέσμου ειδήσεως..

Με την ενεργοποίηση του «**Ignore Rare**», αγνοούνται εντελώς κάποιες σπάνιες λέξεις ή ζευγάρια λέξεων που έχουν καταγραφεί μόνο από έγκυρη είδηση και όχι από είδηση χαμηλής εγκυρότητας, ή το αντίστροφο. Η λειτουργία του, ουσιαστικά, είναι η εξομάλυνση των αποτελεσμάτων από την έντονη επιρροή μονόπλευρα καταγραμμένων λέξεων. Θα πρέπει να χρησιμοποιείται κυρίως αν η βάση δεδομένων αποτελείται ακόμα από μικρό πλήθος πηγών.

3.1.4 Εκπαίδευση από Μεμονωμένη Πηγή

Ο χρήστης έχει την επιλογή να εισάγει ολόκληρο κείμενο ή κάποιο σύνδεσμο στο πεδίο κειμένου, το οποίο θα αναλυθεί σύμφωνα με τις μεθόδους που χρησιμοποιούνται και τα αποτελέσματα θα καταχωρηθούν στην βάση δεδομένων. Είναι σημαντικό να οριστεί η εγκυρότητα της εισαγόμενης είδησης, επιλέγοντας ή μη το checkbox «**Low Veracity**» αν πρόκειται για ψευδή ή έγκυρη είδηση αντίστοιχα.

3.1.5 Εκπαίδευση από Πολλαπλές Πηγές

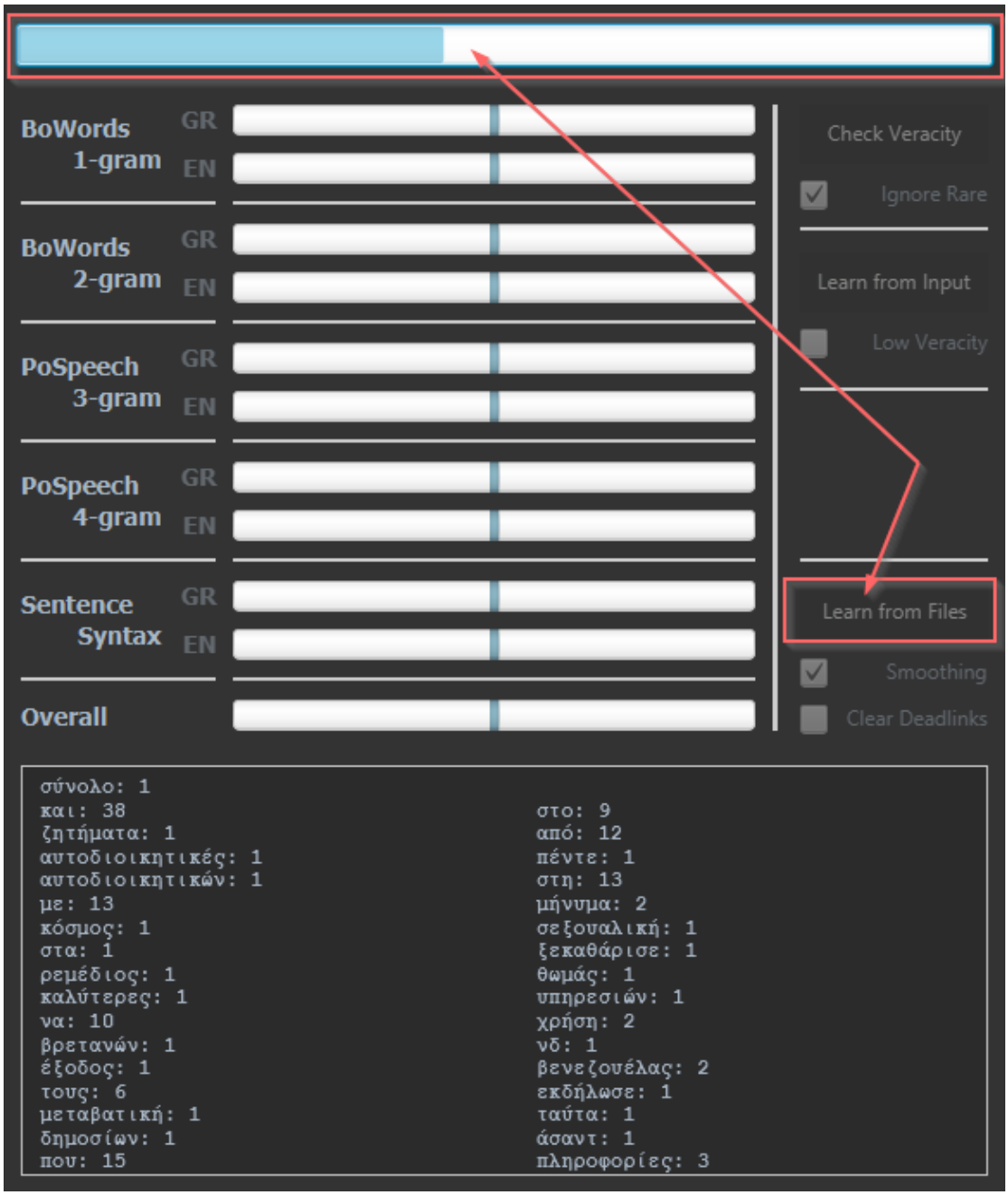
Η εφαρμογή έχει την δυνατότητα να εκπαιδευτεί μαζικά από δύο αρχεία κειμένου τα οποία θα πρέπει να βρίσκονται μέσα στον φάκελο "*<φάκελος_εφαρμογής>\News*". Το κάθε αρχείο αντιπροσωπεύει και την εγκυρότητα των πηγών του και θα πρέπει να ονομαστεί κατάλληλα για να αναγνωριστεί από την εφαρμογή, ***fake.txt*** και ***good.txt*** για ψευδείς και έγκυρες πηγές αντίστοιχα. Στην δομή των αρχείων αυτών κάθε σειρά αντιπροσωπεύει μία είδηση, είτε αυτή είναι σε μορφή συνδέσμου είτε σε μορφή κειμένου. Το κείμενο θα ήταν ιδανικό να βρίσκεται εξολοκλήρου σε μία γραμμή. Πρακτικά ωστόσο, θα μπορούσε να είναι σε πολλαπλές γραμμές χωρίς να επηρεάζεται η εκμάθηση, με μοναδική προϋπόθεση να μην χωρίζεται κάποια πρόταση σε γραμμές. Σε αυτή την περίπτωση είναι απαραίτητο να αντικατασταθούν οι χαρακτήρες αλλαγής γραμμής με τον χαρακτήρα του κενού, ειδάλλως θα χάνεται η συνοχή των προτάσεων.

Η έναρξη της μαζικής εκπαίδευσης γίνεται με το πάτημα του κουμπιού «**Learn from Files**» και μπορεί να κρατήσει αρκετά λεπτά. Ο χρόνος εκτέλεσης εξαρτάται άμεσα από τις ικανότητες του υπολογιστή στον οποίον γίνεται και την ταχύτητα σύνδεσης του στο διαδίκτυο. Το ποσοστό ολοκλήρωσης της εκμάθησης είναι εμφανές στο φόντο του πεδίου εισαγωγής, στο επάνω μέρος της διεπαφής (Σχ. 3.5).

Ο χρήστης έχει επίσης, πριν την έναρξη της εκπαίδευσης, τις επιλογές «**Smoothing**» και «**Clear Deadlinks**»:

- Με την επιλογή «Clear Deadlinks», η εφαρμογή αφαιρεί όλους τους μη προσβάσιμους συνδέσμους που συναντάει κατά την εκπαίδευση από τα αρχεία κειμένων στον φάκελο *News*.

- Με την επισήμανση της πρώτης επιλογής, smoothing, η ελάχιστη τιμή που παίρνει μία πρωτοεισαγόμενη λέξη στην βάση δεδομένων γίνεται ένα (1) αντί για μηδέν (0). Η τακτική αυτή εξομαλύνει ακραίες διαφορές μεταξύ έγκυρων και μη ποσοστών, και όπως φαίνεται από τις δοκιμές (βλ. 3.2.5) αυξάνει την ακρίβεια της εφαρμογής στον εντοπισμό ψεύδους. Ωστόσο, όταν μία βάση έχει εκπαιδευτεί με smoothing, αχρηστεύεται η επιλογή «Ignore Rare» (βλ. 3.1.3) η οποία βασιζόταν στις λέξεις καταχωρημένες με τιμή μηδέν.



Σχήμα 3.5: Διαδικασία μαζικής εκμάθησης και μπάρα προόδου εκμάθησης.

3.2 Έλεγχος Ακρίβειας

3.2.1 Πρόγραμμα Ελέγχου

Για τον έλεγχο της εφαρμογής που υλοποιήθηκε, χρειάστηκε η δημιουργία ενός μικρού παράπλευρου προγράμματος το οποίο εξετάζει την ακρίβειά της στην ταυτοποίηση της εγκυρότητας προκατηγοριοποιημένων συνδέσμων. Το πρόγραμμα αυτό υλοποιείται από την κλάση **Tester** (Παρ. Α), η οποία σε κάθε κύκλο εκτέλεσης διαλέγει τυχαία το 20% των πηγών εκμάθησης για έλεγχο της εγκυρότητας τους ενώ εκπαιδεύει την εφαρμογή σε μία νέα βάση δεδομένων βάσει του υπολοίπου 80% των πηγών. Ο παρακάτω Ψευδοκώδικας 3.1 περιγράφει την εν λόγω διαδικασία.

```
for (c = 0, c < countLines("fake.txt"), c++):
      line = getLine("fake.txt", c)
      if (random(0,1) > .2):
            learnFromURL(line, false)
      else: append(fakeList, line)

for (c = 0, c < countLines("good.txt"), c++):
      line = getLine("good.txt", c)
      if (random(0,1) > .2):
            learnFromURL(line, true)
      else: append(goodList, line)

for (i = 0, i < count(fakeList), i++):
      if (checkVeracity(fakeList[i]) = true):
            fakeAccuracy++

for (i = 0, i < count(goodList), i++):
      if (checkVeracity(goodList[i]) = true):
            goodAccuracy++

accuracy = goodAccuracy + fakeAccuracy
total = count(goodList) + count(fakeList)

return(  accuracy / total )
```

Ψευδοκώδικας 3.1*: Κύρια λειτουργία της κλάσης* Tester. *Τα* checkVeracity *και* learnFromURL *αντιστοιχούν στις πραγματικές μεθόδους που χρησιμοποιήθηκαν, οι οποίες περιγράφονται στην ενότητα 2.5.4. Το* random *θεωρείται ότι είναι μία μέθοδος η οποία επιστρέφει έναν τυχαίο πραγματικό αριθμό από το 0 έως το 1, δεδομένου ότι αυτά είναι τα ορίσματα της. Τα fake.txt και good.txt είναι αρχεία κειμένου τα οποία περιέχουν μία είδηση ανά γραμμή (βλ. 3.1.5). Η μέθοδος* countLine *κάνει καταμέτρηση γραμμών ενός αρχείου και τέλος, η* append προσθέτει ένα νέο στοιχείο (2^{ο} όρισμα) στον πίνακα του πρώτου ορίσματος της.

3.2.2 Πηγές Εκμάθησης

Ως πηγές εκμάθησης για την λειτουργία της εφαρμογής, αλλά και τις δοκιμές, χρησιμοποιήθηκαν συνολικά 1141 σύνδεσμοι ιστοτόπων εκ των οποίων οι 620 θεωρήθηκαν έγκυροι και οι 521 μη έγκυροι. Όσο μεγαλύτερο είναι το πλήθος τους, τόσο πιο ακριβής γίνεται και η εφαρμογή στην ανίχνευση ψεύδους. Για τον καθορισμό και την κατηγοριοποίηση της εγκυρότητας των πηγών έγινα οι παρακάτω παραδοχές ανά κατηγορία.

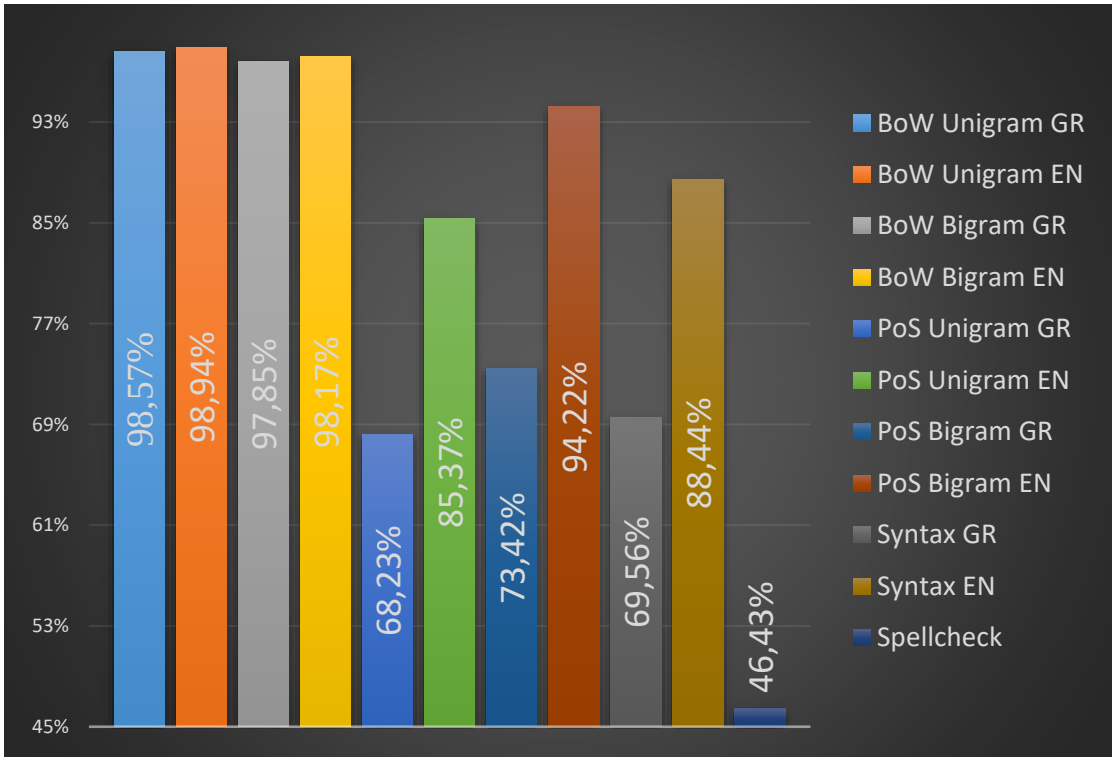
Έγκυρες Πηγές: Σύνδεσμοι που προέρχονταν από ιστοτόπους μεγάλων μέσων μαζικής ενημέρωσης θεωρήθηκαν έγκυροι, διότι ως επί το πλείστον οι ειδήσεις σε αυτούς δημοσιεύονται από δημοσιογράφους οι οποίοι διασταυρώνουν τις πληροφορίες που αναπαράγουν, αλλά και για το λόγο ότι σε μεγάλο βαθμό αποφεύγεται το clickbaiting στα άρθρα τους. Τα άρθρα που χρησιμοποιήθηκαν για την εκμάθηση κυρίως αντλήθηκαν από τους ιστοτόπους *kathimeri.gr, cnn.gr* και σε μικρότερο βαθμό από το *in.gr*. Η βάση δεδομένων που δημιουργείται ενδέχεται να είναι ελαφρώς προκατειλημμένη προς αυτούς τους ιστοτόπους, καθότι στοιχεία (χαρακτηριστικές λέξεις) του ιστοτόπου, αν και ελάχιστα, διαρρέουν αναπόφευκτα στην γνώση της εφαρμογής.

Ανέγκυρες Πηγές: Κύρια πηγή ψευδών ειδήσεων αποτέλεσε ο ιστότοπος *ellinikahoaxes.gr*. Η κεντρική λειτουργία του ιστοτόπου αυτού είναι η κατάρριψη ψευδών ειδήσεων, μύθων και αναληθών δημοσιευμάτων στο διαδίκτυο με εκτενή έρευνα και συλλογή στοιχείων από αξιόπιστες πηγές. Το προτέρημα του ιστοτόπου είναι ότι διατηρεί τις αρχικές πηγές ψεύδους, οι οποίες μπορούν να χρησιμοποιηθούν για την εκμάθηση της παρούσας εφαρμογής. Οι πηγές ποικίλουν αρκετά σε αντίθεση με τις έγκυρες πηγές που χρησιμοποιήθηκαν, ωστόσο κάποιες ψευδείς ειδήσεις είναι συγγενικές μεταξύ τους, κάτι που μπορεί να προκαλέσει την προκατάληψη της εφαρμογής ως προς την θεματολογία.

3.2.3 Πρώτη Δοκιμή

Από τον πρώτο κύκλο δοκιμών διαπιστώθηκε ότι στον ορθογραφικό έλεγχο δεν επαρκεί το λεξικό της βιβλιοθήκης LanguageTool, με αποτέλεσμα να βρίσκονται πολλά λάθη σε έγκυρες ειδήσεις. Η ακρίβεια της εφαρμογής σε αυτή την μέθοδο φθάνει το 46,43% (Σχ. 3.6) και, παρόλο που κυρίως αυτές οι εσφαλμένες αναγνωρίσεις προέρχονται από ονόματα ή ακρωνύμια, δεν ήταν εφικτό να απομονωθούν σε ικανοποιητικό βαθμό. Για αυτό τον λόγο ο ορθογραφικός έλεγχος αφαιρέθηκε από την τελική έκδοση της γραφικής διεπαφής της εφαρμογής. Ωστόσο ο κώδικας παραμένει ακόμα διαθέσιμος προς χρήση μέσω του API και μπορεί να ενεργοποιηθεί με την μέθοδο $setSpell(boolean\ spell)$. Σε μελλοντική έκδοση της εφαρμογής, η προσέγγιση του ορθογραφικού ελέγχου δύναται να βελτιωθεί σχετικά εύκολα (βλ. 4.2.1).

Επίσης, διαπιστώθηκε ότι οι μέθοδοι βασιζόμενοι στο unigram και bigram ετικετών ΜτΛ είχαν αρκετά χαμηλότερη ακρίβεια από την μέθοδο bag-of-words. Για αυτό τον λόγο στον επόμενο κύκλο δοκιμάστηκε η ακρίβεια των ετικετών σε trigram και quadgram. Η χαμηλή ακρίβεια διαπιστώθηκε κυρίως στα Ελληνικά, καθότι υπάρχει μικρότερο πλήθος ΜτΛ.

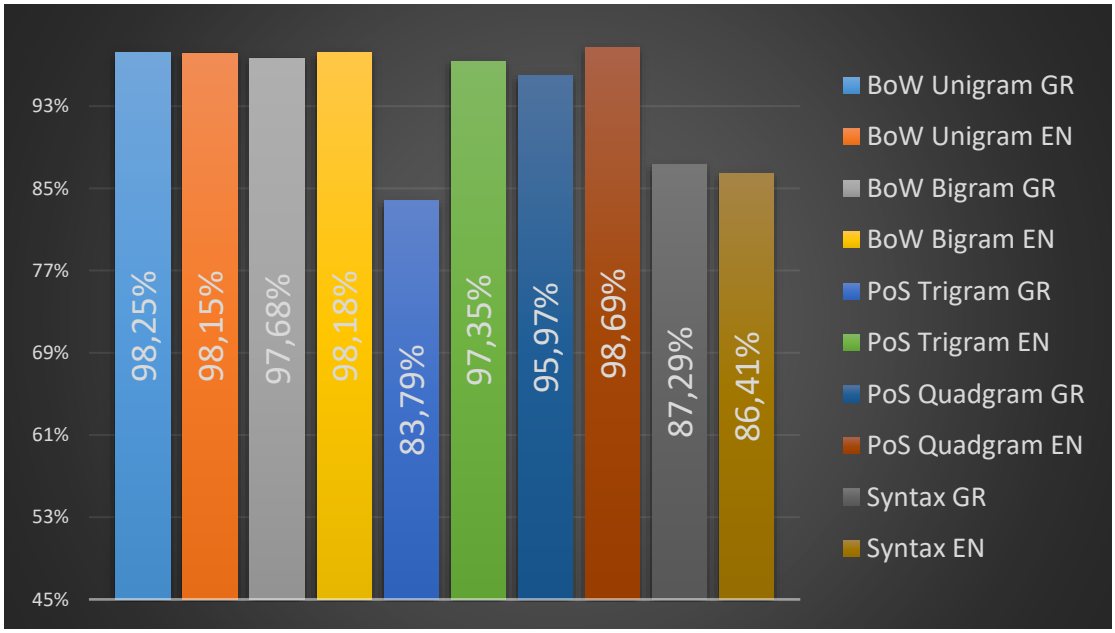


***Σχήμα 3.6**: Αποτελέσματα πρώτων δοκιμών με «Ignore Rare» και χωρίς «Smoothing».*

3.2.4 Δεύτερη Δοκιμή

Στον δεύτερο κύκλο ελέγχου αντικαταστάθηκε το unigram και bigram ετικετών μερών του λόγου από trigram και quadgram αντίστοιχα, με αποτέλεσμα η ακρίβεια να βελτιωθεί σημαντικά. Το trigram ελληνικών ετικετών εξακολουθεί να έχει χαμηλή ακρίβεια σε σχέση με τα υπόλοιπα και αυτό οφείλεται στο ότι οι ετικέτες σε πλήθος είναι λιγότερες από ότι στα Αγγλικά, 45 στο μεν και 16 στο δε.

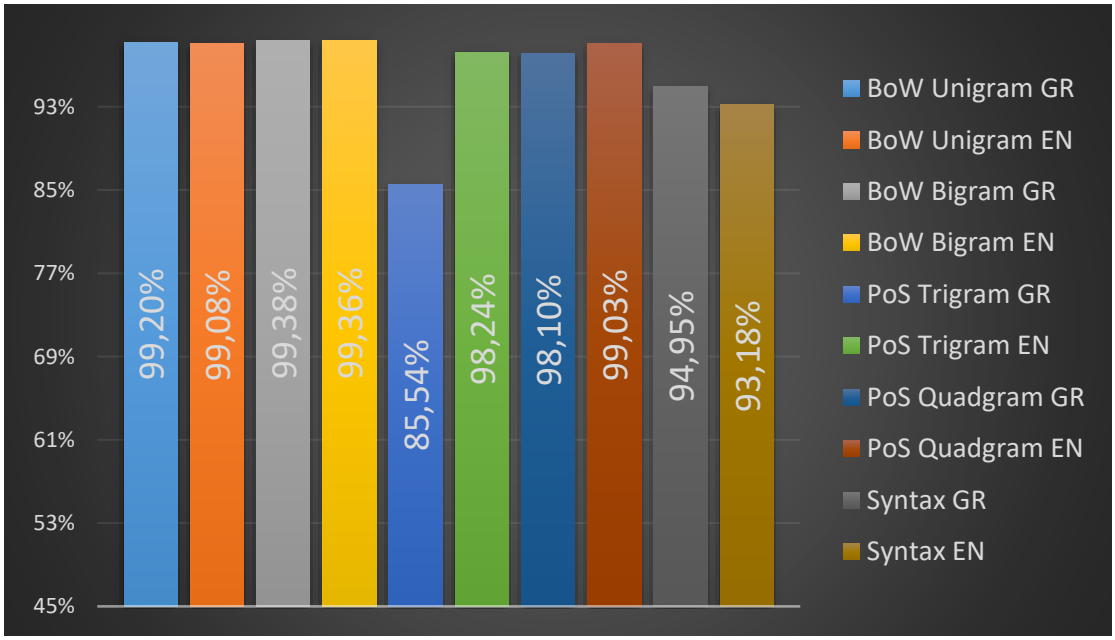
Η μέθοδος της συντακτικής εγκυρότητας, σε αυτόν τον έλεγχο, βασίστηκε στο quadgram και παρουσίασε επίσης αρκετή βελτίωση στα Ελληνικά, ενώ μία μικρή μείωση στα Αγγλικά. Η βελτίωση ήταν αναμενόμενη εφόσον βελτιώθηκε η ακρίβεια στις μεθόδους ετικετών ΜτΛ.



Σχήμα 3.7: Αποτελέσματα δεύτερου κύκλου δοκιμών, ίδιες ρυθμίσεις με τον πρώτο.

3.2.5 Τρίτη Δοκιμή

Ενώ στις προηγούμενες δοκιμές η εξομάλυνση (*Smoothing*, βλ. 3.1.5) ήταν απενεργοποιημένη και η αγνόηση σπανίων λέξεων (*Ignore Rare*, βλ. 3.1.3) ενεργοποιημένη, σε αυτή την δοκιμή έγινε το αντίθετο. Αν και είναι προφανές ότι υπάρχει κάποιου είδους προκατάληψης μιας και η ακρίβεια της εφαρμογής είναι υψηλότερη από το αναμενόμενο, η βελτίωση που παρουσιάζεται με την εξομάλυνση είναι πολύ σημαντική. Όπως φαίνεται και στο Σχ. 3.8, η μέθοδος της συντακτικής εγκυρότητας βελτιώθηκε κατά 8,78% και 7,83% για τα Ελληνικά και Αγγλικά αντίστοιχα.



Σχήμα 3.8: Αποτελέσματα τρίτου κύκλου δοκιμών, με «Smoothing».

Κεφάλαιο 4: Συμπεράσματα

4.1 Ανασκόπηση

Το πρόβλημα των ψευδών ειδήσεων, και της παραπληροφόρισης γενικότερα, αποκτά μεγάλες διαστάσεις με την άνθηση του διαδικτύου, καθιστόντας απαραίτητη την αυτοματοποίηση της ανίχνευσης ψεύδους. Ήδη διακρίνονται τεχνολογίες και εργαλεία για την αντιμετώπιση του προβλήματος, τα οποία μελετήθηκαν και αναλύθηκαν στην παρούσα εργασία. Ένα μέρος αυτών επιλέκτηκε προς εφαρμογή.

Βάσει των μεθόδων γλωσσικής ανάλυσης έγινε η σχεδίαση ενός προγράμματος, το οποίο υλοποιήθηκε επιτυχώς. Από τις δοκιμές που πραγματοποιήθηκαν δείχθηκε η καλή λειτουργεία του και η αρκετά ικανοποιητική ακρίβεια στην απόδοση εγκυρότητας σε ελληνικά κείμενα. Ωστόσο δύναται να δεχτεί βελτιώσεις ή και επεκτάσεις, οι οποίες παρουσιάζονται στην επόμενη ενότητα.

4.2 Μελλοντικές Βελτιώσεις

4.2.1 Ορθογραφικός Έλεγχος

Το λεξικό του ορθογραφικού ελέγχου αν και αρκετά ικανοποιητικό σε πλήθος λέξεων, ποτέ δεν θα είναι αρκετό για να καλύψει ακρώνυμα, ονόματα ή και νέες ορολογίες που προκύπτουν με την πάροδο του χρόνου. Μια σημαντική βελτίωση στην εφαρμογή θα αποτελούσε η διαδικτυακή διερεύνηση των λέξεων που αναγνωρίζονται ως λανθασμένες, έτσι ώστε να προσεγγίζεται αν όντως πρόκειται για ορθογραφικό λάθος ή όχι. Δύο ευκόλως υλοποιήσιμες μέθοδοι με την χρήση της μηχανής αναζήτησης Google είναι οι παρακάτω.

Πλήθος αποτελεσμάτων: Με την προϋπόθεση ότι η αναζήτηση έχει γίνει για ακριβής αντιστοίχιση (exact match) της υποψήφια λανθασμένης λέξης, όσο μεγαλύτερο το πλήθος αποτελεσμάτων που επιστρέφει η αναζήτηση τόσο το πιθανότερο είναι η λέξη να υπάρχει.

«**Μήπως εννοείτε**»: Για αναζήτηση με μη ακριβή αντιστοίχιση, σε περίπτωση άγνωστης λέξεως το Google επιστρέφει μία δημοφιλή παραπλήσια λέξη με την ετικέτα «Μήπως εννοείτε». Έτσι, αναζητώντας την υποψήφια λανθασμένη λέξη, η μη επιστροφή της ετικέτας «Μήπως εννοείτε» είναι μία καλή ένδειξη ότι η λέξη είναι ορθογραφικά σωστή.

4.2.2 Πηγές Εκμάθησης

Για την αποφυγή της δημιουργίας προκατειλημμένης βάσης δεδομένων είναι απαραίτητη η πληθώρα ειδήσεων με όσο το δυνατό λιγότερο κοινά χαρακτηριστικά ανά κατηγορία εγκυρότητας. Για την ύπαρξη αυτής της ανομοιογένειας ειδήσεων δεν απαιτούνται μόνο διαφορετικές πήγες, αλλά και η αποκόμιση αυτών από διαφορετικές χρονικές περιόδους. Αυτό διότι η επικαιρότητα συμβάλει σε μεγάλο βαθμό στην ομοιογένεια των ειδήσεων ασχέτως της διαφορετικής προελεύσεως της εκάστης είδησης.

4.2.3 Επιπρόσθετες Μέθοδοι

Η παρούσα εφαρμογή δύναται να βελτιωθεί με την ανάπτυξη επιπρόσθετων μεθόδων, όπως είναι η Διαδικτυακή Διερεύνηση (βλ. 1.2.2) και η Βαθεία Μηχανική Μάθηση (βλ. 1.2.3). Για παράδειγμα, η συσχέτιση δεδομένων της μεθόδου Διαδικτυακής Διερεύνησης δεν απαιτεί πάνω από μία πρόταση για την ανίχνευση ψεύδους, εν αντιθέσει ο Ασκός Λέξεως είναι ακριβέστερος όταν υπάρχουν και αρκετές λέξεις στο εξεταζόμενο κείμενο. Ο συνδυασμός πολλαπλών μεθόδων αυξάνει την ακρίβεια της εφαρμογής σε διαφορετικούς τύπους κειμένου.

4.2.4 Κατηγοροποίηση Ειδήσεων

Με την συνεχή βελτιώση των εργαλείων ανίχνευσης ψεύδους και των αποτελεσμάτων που προσφέρουν, αναδύεται ένα πιο περίπλοκο πρόβλημα. Το πρόβλημα αυτό είναι η κατηγοριοποίηση των ειδήσεων για την εκπαίδευση του κάθε εργαλείου, καθότι η εγκυρότητα κάθε εργαλείου θα εξαρτάται πάντα και εξολοκλήρου από την υποκειμενικότητα των ατόμων που το εκπαιδεύουν. Μία καλή προσέγγηση για την επίλυση του προβλήματος αυτού αποτελεί η χρήση πληθοπορισμού (crowdsourcing) για την κατηγοριοποίηση των ειδήσεων. Πρόκειται για μία αρκετά αντικειμενική λύση, μιας και βασίζεται σε εθελοντική και μη αμειβόμενη εργασία, παρόμοια με τον τρόπο που δουλεύει ο ιστότοτοπος Wikipedia.

Παρόλο που προς το παρόν είναι στην αγγλική γλώσσα και δεν μπορεί να χρησιμοποιηθεί στην παρούσα εφαρμογή, αξίζει να αναφερθεί η πλατφόρμα WikiTribune (https://www.wikitribune.com/) η οποία λειτουργεί με τον τρόπο αυτό για την δημοσίευση έκγυρων ειδήσεων.

4.2.5 Προσαρμογή σε Άλλη Εφαρμογή

Εν κατακλείδι, η παρούσα εφαρμογή αντί να επεκταθεί η ίδια, δύναται να επεκτείνει μία άλλη εφαρμογή ανίχνευσης ψεύδους. Παράλληλα με την δημιουργία της εφαρμογής, δημιουργήθηκε και ένα API που επιτρέπει τις λειτουργίες της εφαρμογής να προσαρμοστούν σε κώδικα κάποιας άλλης με την προσάρτηση των κλάσεων και την δημιουργία ενός νέου instance του FakeDetection.

Για παράδειγμα για ένα αντικείμενο ονόματι detection, έχουμε:

$$FakeDetection\ detection = new\ FakeDetection()$$

Στο αντικείμενο αυτό υπάρχουν τρεις κύριοι μέθοδοι:

1. $learnFromFiles()$, με την οποία γίνεται μαζική εκμάθηση από τα αρχεία που αναφέρονται στην ενότητα «Εκπαίδευση από Πολλαπλές Πηγές» (3.1.5).
2. $learnFromURL(String\ url, boolean\ veracity)$, με την οποία μέθοδο γίνεται εκμάθηση από κάποιον σύνδεσμο (ή κείμενο) δηλώνοντας ως δεύτερη παράμετρο true ή false για το αν είναι έγκυρη η είδηση ή όχι αντίστοιχα.
3. $checkVeracity(String\ url)$, η οποία δέχεται ως όρισμα έναν σύνδεσμο ή κείμενο και επιστρέφει ένα πίνακα *Pair<String, Double>.* Ο πίνακας σε κάθε του θέση περιέχει ένα ζευγάρι αιτιολόγησης (*String*) και ποσοστού εγκυρότητας (*Double*) το οποίο αντιστοιχεί σε μία από τις μεθόδους. Οι θέσεις του πίνακα έχουν ως εξής:

   1η θέση: Ελληνικό Unigram.
   2η θέση: Αγγλικό Unigram.
   3η θέση: Ελληνικό Bigram.
   4η θέση: Αγγλικό Bigram.
   5η θέση: Ελληνικό Tag Trigram.
   6η θέση: Αγγλικό Tag Trigram.
   7η θέση: Ελληνικό Tag Quadgram.
   8η θέση: Αγγλικό Tag Quadgram.
   9η θέση: Ελληνικό Syntax.
   10η θέση: Αγγλικό Syntax.

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Αλγορίθμων

Κατάλογος Ακρώνυμων

ANN – Artificial Neural Network

API – Application Programming Interface

BoT – Bag of Tags

BoW – Bag of Words

CR – Carriage Return

DB – Database

GREC – Google Relation Etaction Corpus

HTTP – HyperText Transfer Protocol

IO – Input/Output

JSON – JavaScript Object Notation (δομημένη μορφή κειμένου)

LF – Line Feed

PCFG – Probabilistic Context Free Grammar

PoS – Part of Speech

RDR – Ripple Down Rules

SQL – Structured Query Language (γλώσσα αιτημάτων σε βάση δεδομένων)

SVM – Support Vector Machines

URL – Uniform Resource Locator (διεύθυνση διαδυκτιακού τόπου)

ΑΛ – Ασκός Λέξεων

ΕΔ – Επιφανειακή Δομή

ΜτΛ – Μέρη του Λόγου

ΟΕ – Ορθογραφικός Έλεγχος

ΠΧ – Περίπτωση Χρήσεως

ΣΔ – Συσχέτηση Δεδομένων

Βιβλιογραφία

- Appelgren, M. (2016). Detecting Deception using Natural Language.
- Chen, Y., Conroy, N. J., & Rubin, V. L. (2015). News in an Online World: The Need for an “Automatic Crap Detector”.
- Ciampaglia, G., Shiralkar, P., Rocha, L., Bollen, J. Menczer, F. & Flammini, A. (2015). Computational fact checking from knowledge networks.
- Collins, M. (2011). Probabilistic Context-Free Grammars (PCFGs). Course.
- Conroy, N. J., Chen, Y., & Rubin, V. L. (2015). Automatic Deception Detection: Methods for Finding Fake News.
- Dzieza, J. (2014, Oct 22). Fake News Sites Are Using Facebook to Spread Ebola Panic. The Verge.
- Feng, S., Banerjee, R. & Choi, Y. (2012). Syntactic Stylometry for Deception Detection.
- Feng, V. & Hirst, G. (2013) Detecting deceptive opinion with profile compatibility.
- Nguyen, D. Q., Nguyen, D. Q., Pham, D. D. & Pham, S. B. (2005). RDRPOSTagger: A Ripple Down Rules-based Part-Of-Speech Tagger.
- Rubin, V. & Lukoianova, T. (2014). Truth and deception at the rhetorical structure level.
- Ruchansky, N., Sungyong S. & Liu, Y. (2017). CSI: A Hybrid Deep Model for Fake News Detection.
- Soroush V., Deb R. & Sinan A. (2018). The Spread of True and False News Online.
- Zhang, H., Fan, Z., Zeng, J. & Liu, Q. (2012). An Improving Deception Detection Method in Computer-Mediated Communication.

Παράρτημα Α: Πηγαίος Κώδικας

FakeDetection.java

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import static java.lang.Double.min;
import static java.lang.Integer.min;
import static java.lang.Math.abs;
import static java.lang.Math.round;
import static java.lang.Thread.sleep;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map.Entry;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.stream.Stream;
import javafx.util.Pair;

public class FakeDetection {
    private final Charset utf8 = StandardCharsets.UTF_8;
    private List<String> deadlinks;
    private final int numMeth = 11;
    private int maxResults = 20;
    private DetectionGUI gui;
    private String[] types;
    private DBControl db;
    private boolean rare;
    private boolean spell;
    private boolean links;


    public FakeDetection(String dbase)
            throws SQLException {
        System.setProperty("http.agent", "Chrome");
        deadlinks = new ArrayList<>();
        db = new DBControl(dbase);
        types = db.getTypes();
        db.setMinValue(1);
        links = false;
        spell = false;
        rare = true;
    }
```

```
public FakeDetection(DetectionGUI gui) throws SQLException {
    this("edu.db");
    this.gui = gui;
}

public void learnFromFiles() {
    int learnSum = 0;

    if (gui != null) {
        try (Stream<String> lines = Files.lines(
                Paths.get("News/fake.txt"))) {
            learnSum += lines.count(); } catch (IOException ex) {
            output("The file 'fake.txt' in "
                    + "News folder was not found."); }

        try (Stream<String> lines = Files.lines(
                Paths.get("News/good.txt"))) {
            learnSum += lines.count(); } catch (IOException ex) {
            output("The file 'good.txt' in "
                    + "News folder was not found."); }

        gui.setLearnSum(learnSum);
    }

    if (learnFromFile("good"))
        if (learnFromFile("fake"))
            output("\n\n\n\n\n\n\n\n\n\n"
                    + "Learning Completed!");

    if (links) {
        clearLinks("good");
        clearLinks("fake");
    }
}

private boolean learnFromFile(String file) {
    boolean veracity = true;
    if (file.equals("fake"))
        veracity = false;

    try(BufferedReader br = Files.newBufferedReader(
            Paths.get("News/" + file + ".txt"), utf8)) {

        String url = br.readLine();
        while (url != null) {

            learnFromURL(url, veracity);

            url = br.readLine();
            if (gui != null) gui.updateLearn();
        }

        return true;
    } catch (Exception ex) {
        output("The file '" + file + ".txt' in "
```

```
                + "News folder was not found.");
        Logger.getLogger(FakeDetection.class.getName())
                .log(Level.SEVERE, null, ex);
        return false;
    }
}

public void learnFromURL(String url, boolean veracity) {
    try {
        TextProcess tpGr = new TextProcess(url);
        TextProcess tpEn = tpGr.getEnglish();

        if (!tpGr.getSimple().equals("")
                && !tpEn.getSimple().equals("")) {

            if (spell) learnSpell(tpGr, veracity);
            for (int t=types.length-4; t>-1; t--) {
                learnWords(tpEn, veracity, types[t--]);
                learnWords(tpGr, veracity, types[t]);
            }
        }

        System.gc();
    } catch (IOException ex) {
        if (testInternet()) {
            if (links)
                deadlinks.add(url);
            errorLog(url);
            Logger.getLogger(FakeDetection.class.getName())
                .log(Level.SEVERE, null, ex);
        }
        else {
            try { sleep(10000); }
            catch (Exception iex) {}
            finally { learnFromURL(url, veracity); }
        }
    }
}

private void learnWords(TextProcess tp,
        boolean veracity, String method) {
    tp.getCount(method)
            .forEach((word,value)->{
        if (word != null)
            db.addWord(word, value, veracity, method);
        outputln(word+": "+value);
    });
}

private void learnSpell(TextProcess tp, boolean veracity)
        throws IOException {
    db.addSpell(tp.spellcheck(),
            tp.getSimple().split(" ").length, veracity);
}

public Pair<String, Double>[] checkVeracity(String url) {
```

```
        Pair<String, Double>[] reasoning = null;
        try {
            TextProcess tpGr = new TextProcess(url);
            TextProcess tpEn = tpGr.getEnglish();

            if (!tpGr.getSimple().equals("")
                    && !tpEn.getSimple().equals("")) {

                reasoning = new Pair[numMeth];
                if (spell)
                    reasoning[numMeth-1] = checkSpell(tpGr);

                for (int t=0; t<types.length-3; t++) {
                    reasoning[t] = checkFor(types[t++], tpGr);
                    reasoning[t] = checkFor(types[t], tpEn);
                }

                reasoning[numMeth-3] = checkSyntax(tpGr);
                reasoning[numMeth-2] = checkSyntax(tpEn);
            }

        } catch (IOException ex) {
            if (testInternet())
                errorLog(url);
            else {
                try { sleep(10000); }
                catch (Exception iex) { }
                finally { checkVeracity(url); }
            }
        }

        if (gui != null) {
            gui.btnDisabled(false);
            gui.cls();
        }

        return reasoning;
    }

    private String roundPercent(double num) {
        return (double)round(num*10000)/100+"%";
    }

    private void output(String text) {
        if (gui != null) gui.print(text);
        else System.out.print(text);
    }

    private void outputln(String text) {
        if (gui != null) gui.println(text);
        else System.out.print(text + ", ");
    }

    private Pair<String, Double> checkSpell(TextProcess tp)
            throws IOException {
        double[] ratio = db.getSpell();
```

```
        int errorCount = tp.spellcheck();
        int wordCount = tp.getSimple().split(" ").length;
        double percent = (double)errorCount/wordCount;
        double progress = 0;

        String reason = "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n"
                + "Error ratio: "+roundPercent(percent) +
                "\nThere were found " + errorCount +
                " unknown words in " + wordCount +
                " words.";
        progress = min(percent,(ratio[0]+ratio[1]))/(ratio[0]+ratio[1]);

        if (ratio[1]>ratio[0])
            progress = 1-progress;
        return new Pair(reason, progress);
    }

    private Pair<String, Double> checkFor(String method,
                                          TextProcess tp) {

        String reasonWord[] = new String[maxResults+1];
        double reasonInfl[] = new double[maxResults+1];

        int sum = 0;
        double veracity = 0;

        // Getting the learnt ratio of every word, depending
        // on the method, of the text in TextProcess.
        for (Entry<String,Integer> entry :
                  tp.getCount(method).entrySet()) {
            double[] ratio = db.getWord(entry.getKey(), method);

            double percent = 0;
            if (ratio[0] != 0 || ratio[1] != 0)
                percent = (ratio[0]-ratio[1]) / (ratio[0]+ratio[1]);

            if (abs(percent) == 1 && !rare) continue;
            outputln(entry.getKey()+": "+roundPercent(percent));

            veracity += (1+percent)/2 * entry.getValue();

            reasonInfl[maxResults] = percent * entry.getValue();
            reasonWord[maxResults] = entry.getKey();

            for (int c = maxResults-1; c >= 0; c--)
                if (abs(reasonInfl[c]) < abs(reasonInfl[c+1])) {
                    double tempPercent = reasonInfl[c+1];
                    reasonInfl[c+1] = reasonInfl[c];
                    reasonInfl[c] = tempPercent;
                    String tempWord = reasonWord[c+1];
                    reasonWord[c+1] = reasonWord[c];
                    reasonWord[c] = tempWord;
                }

            sum += entry.getValue();
```

```
            double progress = veracity/sum;

            if (gui!=null) {
                gui.updateProg(method,progress);
                try { sleep(10); }
                 catch (InterruptedException ex) {}
            }
        }

        int strlen = 0;
        String reason = "\n\nInfluencive Words:\n";
        for (int c = 0; c < maxResults; c++)
            if (reasonWord[c] != null && c%2==0) {
                String len = c+1+") "+reasonWord[c]+": "
                        +roundPercent(reasonInfl[c]/sum);
                if (strlen < len.length())
                    strlen = len.length();
            }

        for (int c = 0; c < maxResults; c++)
            if (reasonWord[c] != null) {
                reason += String.format("%-" +
                        min(32,strlen) + "s", c+1+") " +
                        reasonWord[c] + ": " +
                        roundPercent(reasonInfl[c]/sum));

                if (c%2==1) reason += "\n";
                else reason += "\t";
            }
        reason += "\nVeracity: "+roundPercent(veracity/sum);
        return new Pair(reason,veracity/sum);
    }

    private Pair<String, Double> checkSyntax(TextProcess tp) {

        int length = maxResults/2;
        String reasonWord[] = new String[length];
        double reasonInfl[] = new double[length];

        int sum = 0;
        double veracity = 0;

        String[] tagged = tp.getTagOrder();
        String[] phrase = tp.getPhrases();
        String method = tp.isEnglish()? "tqen" : "tqgr";

        for (int p = 0; p < tagged.length; p++) {
            String[] tag = tagged[p].split(" ");

            double[] phRatio = {1, 1};
            for (int c = 0; c < tag.length - 3; c++) {
                String qTag = tag[c]+" "+tag[c+1]+
                            " "+tag[c+2]+" "+tag[c+3];

                double[] ratio = db.getWord(qTag, method);
                phRatio[0] *= ratio[0]==0? Double.MIN_VALUE:ratio[0];
```

```
                phRatio[1] *= ratio[1]==0? Double.MIN_VALUE:ratio[1];
            }

            double percent = 0;
            if (phRatio[0] != 0 || phRatio[1] != 0)
                percent = (phRatio[0]-phRatio[1])
                        / (phRatio[0]+phRatio[1]);

            if (abs(percent) == 1 && !rare) continue;

            veracity += (1+percent)/2;

            String reasonPh = phrase[p];
            reasonInfl[length-1] = percent;
            reasonWord[length-1] = reasonPh
                    .substring(0, min(64,
                            reasonPh.length())) + "...";

            for (int c = length-2; c >= 0; c--)
                if (abs(reasonInfl[c]) < abs(reasonInfl[c+1])) {
                    double tempPercent = reasonInfl[c+1];
                    reasonInfl[c+1] = reasonInfl[c];
                    reasonInfl[c] = tempPercent;
                    String tempWord = reasonWord[c+1];
                    reasonWord[c+1] = reasonWord[c];
                    reasonWord[c] = tempWord;
                }

            sum++;
            double progress = veracity/sum;
            outputln(sum + ") Veracity: " + roundPercent(percent)+
                    " | " + phrase[p]);
            System.out.println(sum + ") Veracity: " +
                        roundPercent(percent) +
                    " | " + phrase[p]);

            if (gui!=null) {
                gui.updateProg(tp.isEnglish()?
                        "synen" : "syngr", progress);
                try { sleep(40); }
                catch (InterruptedException ex) {}
            }
        }

        String reason = "\n\nInfluencive Phrases:\n";
        for (int c = 0; c < length; c++)
            if (reasonWord[c] != null) {
                reason += (c+1)+") " +
                        roundPercent(reasonInfl[c]/sum)
                        + ": " + reasonWord[c] + "\n";
            }
        reason += "\nVeracity: "+roundPercent(veracity/sum);
        return new Pair(reason,veracity/sum);
    }

    public void close() {
```

```
        try {
            db.closeConnection();
        } catch (SQLException ex) {
            Logger.getLogger(FakeDetection.class.getName())
                    .log(Level.SEVERE, null, ex);
        }
    }

    public void setRare(boolean rare) {
       this.rare = rare;
    }

    public void setMin(int val) {
        db.setMinValue(val);
    }

    public void setSpell(boolean spell) {
        this.spell = spell;
    }

    public void setLinks(boolean links) {
        this.links = links;
    }

    private void errorLog(String url) {
        try(FileWriter fw = new FileWriter("SiteError.log", true);
                BufferedWriter bw = new BufferedWriter(fw);
                PrintWriter pw = new PrintWriter(bw)) {
            output("\nSite Unavailable: "+url);
            pw.append("\nSite Unavailable: "+url);
        } catch (IOException ex) {
            Logger.getLogger(FakeDetection.class.getName())
                    .log(Level.SEVERE, null, ex);
        }
    }

    private boolean testSite(String site) {
        Socket sock = new Socket();
        InetSocketAddress addr =
                new InetSocketAddress(site,80);

        try {
            sock.connect(addr,3000);
            return true;
        } catch (IOException e) {
            return false;
        } finally {
            try { sock.close(); }
            catch (IOException e) {}
        }
    }

    public boolean testInternet() {
        if (testSite("google.com")
                && testSite("amazon.com")
                && testSite("yahoo.com"))
```

```
            return true;
        return false;
    }

    private void clearLinks(String file) {
        try {
            File input = new File("News/" + file + ".txt");
            File temp = new File("News/tmp" + file + ".txt");

            BufferedReader reader = Files
                .newBufferedReader(input.toPath(), utf8);
            BufferedWriter writer = Files
                .newBufferedWriter(temp.toPath(), utf8);

            String line;
            while ((line = reader.readLine()) != null)
                if (!deadlinks.contains(line))
                    writer.write(line + System.lineSeparator());

            writer.close();
            reader.close();
            while(!temp.renameTo(input))
                sleep(5000);

        }   catch (Exception ex) {
            Logger.getLogger(FakeDetection.class.getName())
                    .log(Level.SEVERE, null, ex);
        }   finally { deadlinks.clear(); }
    }
}
```

DetectionGUI.java

```
import java.io.IOException;
import static java.lang.Integer.min;
import static java.lang.Math.round;
import static java.lang.Thread.sleep;
import java.net.URL;
import java.sql.SQLException;
import java.util.Locale;
import java.util.ResourceBundle;
import java.util.concurrent.CompletableFuture;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.prefs.Preferences;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.CheckBox;
import javafx.scene.control.Label;
import javafx.scene.control.ProgressBar;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.input.MouseEvent;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.scene.text.TextFlow;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
import javafx.stage.WindowEvent;
import javafx.util.Pair;

public class DetectionGUI extends Application implements Initializable {
    final int numMeth = 11;

    private CompletableFuture<Pair<String,Double>[]> results;
    private Pair<String,Double>[] reason = new Pair[numMeth];
    private boolean resultChanged = true;
    private ProgressBar[] bars;
    private FakeDetection fd;
    private Preferences pref;
    private int learningSum;
    private int learningCur;
    private int strlen = 0;
    private int line = 1;
    private final int fixed = 597;
    @FXML private TextFlow console;
    @FXML private ScrollPane scroll;
```

```
    @FXML private CheckBox chkVera;
    @FXML private CheckBox chkRare;
    @FXML private CheckBox chkSmoo;
    @FXML private CheckBox chkLink;
    @FXML private Button btnCheck;
    @FXML private Button btnFiles;
    @FXML private Button btnInput;
    @FXML private TextField input;
    @FXML private ProgressBar progUniGr;
    @FXML private ProgressBar progBiGr;
    @FXML private ProgressBar progTagTriGr;
    @FXML private ProgressBar progTagQuaGr;
    @FXML private ProgressBar progSynGr;
    @FXML private ProgressBar progUniEn;
    @FXML private ProgressBar progBiEn;
    @FXML private ProgressBar progTagTriEn;
    @FXML private ProgressBar progTagQuaEn;
    @FXML private ProgressBar progSynEn;
    @FXML private ProgressBar progOverall;
    @FXML private ProgressBar progLearn;
    @FXML private Label txtUniGr;
    @FXML private Label txtBiGr;
    @FXML private Label txtTagTriGr;
    @FXML private Label txtTagQuaGr;
    @FXML private Label txtSynGr;
    @FXML private Label txtUniEn;
    @FXML private Label txtBiEn;
    @FXML private Label txtTagTriEn;
    @FXML private Label txtTagQuaEn;
    @FXML private Label txtSynEn;
    @FXML private Label txtOverall;

    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader loader = new FXMLLoader(getClass()
                .getResource("DetectionGUI.fxml"));
        Parent root = loader.load();
        Scene scene = new Scene(root);
        stage.setTitle("Fake Detection");
        stage.getIcons().add(new Image(DetectionGUI.class
                .getResourceAsStream("/Assets/icon.png")));

        stage.initStyle(StageStyle.UNIFIED);
        stage.setMinHeight(fixed+130);
        stage.setMaxHeight(fixed+140);
        stage.setMaxWidth(fixed+100);
        stage.setMinWidth(fixed);
        stage.setHeight(fixed+130);
        stage.setWidth(fixed);
        stage.setScene(scene);
        stage.show();

        stage.setOnCloseRequest(new EventHandler<WindowEvent>() {
            @Override public void handle(WindowEvent t) {
                if(fd != null) fd.close();
                Platform.exit();
```

```
                System.exit(0);
            }
        });
    }

    public static void main(String[] args) {
        Locale.setDefault(new Locale("el", "GR"));
        System.setProperty("file.encoding", "UTF-8");
        launch(args);
    }

        public void print(String text) {
        Platform.runLater(()->{
                Text line = new Text(text);
                line.setFont(Font.font("Monospaced",
                        FontWeight.BOLD, 12));
                line.setFill(Color.web("#a9b7c6"));
                if (console.getChildren().size()>100)
                    console.getChildren().remove(0);
                console.getChildren().add(line);
                scroll.layout();
                scroll.setVvalue(scroll.getVmax());
        });
    }

    public void println(String text) {
        if (strlen < text.length())
            strlen = text.length();

        text = String.format("%-" +
                min(32,strlen) + "s", text);

        if (line > 256) line = 1;
        if (line++%2==0) text += "\n";
        else text += "\t";

        print(text);
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        try {
            pref = Preferences
                    .userNodeForPackage(DetectionGUI.class);
            fd = new FakeDetection(this);
            fd.setLinks(pref.getBoolean("link", false));
            fd.setRare(pref.getBoolean("rare", true));
            fd.setMin(pref.getInt("min", 1));
            fd.setSpell(false);

            bars = new ProgressBar[] { progUniGr,
                progUniEn, progBiGr, progBiEn, progTagTriGr,
                progTagTriEn, progTagQuaGr, progTagQuaEn,
                progSynGr, progSynEn, progOverall };

            chkRare.setSelected(!pref.getBoolean("rare", true));
```

```
            chkRare.setOnMouseClicked(new EventHandler<MouseEvent>() {
            @Override public void handle(MouseEvent event) {
                pref.putBoolean("rare", !chkRare.isSelected());
                fd.setRare(!chkRare.isSelected()); }});

            chkSmoo.setSelected(pref.getInt("min", 1)==0? false:true);
            chkSmoo.setOnMouseClicked(new EventHandler<MouseEvent>() {
            @Override public void handle(MouseEvent event) {
                pref.putInt("min", chkSmoo.isSelected()? 1:0);
                fd.setMin((chkSmoo.isSelected())? 1:0); }});

            chkLink.setSelected(pref.getBoolean("link", false));
            chkLink.setOnMouseClicked(new EventHandler<MouseEvent>() {
            @Override public void handle(MouseEvent event) {
                pref.putBoolean("link", chkLink.isSelected());
                fd.setLinks(chkLink.isSelected()); }});

            for (int b = 0; b < bars.length; b++) {
                int bar = b;
                bars[b].setOnMouseClicked(
                        new EventHandler<MouseEvent>() {
                    @Override public void handle(MouseEvent event) {
                        if (!btnCheck.isDisabled())
                            try {
                                if (results != null && resultChanged)
                                    reason = results.get();
                                resultChanged = false;
                            } catch (Exception ex) {
                               Logger.getLogger(DetectionGUI.class.getName())
                                    .log(Level.SEVERE, null, ex);
                            }

                        cls();
                        if (reason[bar] != null) {
                            print(reason[bar].getKey());
                               System.out.println(reason[bar].getKey());
                        }
                    }});}
        } catch (SQLException ex) {
            System.out.println("Database is in use! "
                    + "Retrying in few seconds...");
            try { sleep(5000);
                initialize(location, resources);
            } catch (InterruptedException ex1) { }
        }

    }

    @FXML
    public void learnFromFiles() {
        btnDisabled(true);
        Thread th = new Thread(()->{
            fd.learnFromFiles();
            btnDisabled(false);});
        th.setPriority(Thread.MAX_PRIORITY);
        th.start();
```

```
    }

    @FXML
    public void checkVeracity() {
        btnDisabled(true);
        results = CompletableFuture.supplyAsync(()-> {
                Pair<String,Double>[] result = new Pair[numMeth];
                if (checkInput())
                    result = fd.checkVeracity(input.getText());
                btnDisabled(false);
                return result;
        });
        resultChanged = true;
    }

    @FXML
    public void learnFromInput() {
        btnDisabled(true);
        if (checkInput()) {
            Thread th = new Thread(()->{
                    fd.learnFromURL(input.getText(),
                            !chkVera.isSelected());
                    btnDisabled(false); });
            th.setPriority(Thread.MIN_PRIORITY);
            th.start();
        }
    }

    private boolean checkInput() {
        if (input.getText().equals("")) {
            println("Input is empty!");
            return false;
        } return true;

    }

    public void updateProg(String select, double progress) {
        ProgressBar prog;
        Label label;

        switch (select) {
            case "unigr": prog = progUniGr; label = txtUniGr; break;
            case "bigr": prog = progBiGr; label = txtBiGr; break;
            case "ttgr": prog = progTagTriGr; label = txtTagTriGr; break;
            case "tqgr": prog = progTagQuaGr; label = txtTagQuaGr; break;
            case "syngr": prog = progSynGr; label = txtSynGr; break;
            case "unien": prog = progUniEn; label = txtUniEn; break;
            case "bien": prog = progBiEn; label = txtBiEn; break;
            case "tten": prog = progTagTriEn; label = txtTagTriEn; break;
            case "tqen": prog = progTagQuaEn; label = txtTagQuaEn; break;
            case "synen": prog = progSynEn; label = txtSynEn; break;
            default: prog = null; label = null;
        }

        Platform.runLater(()->{
            if (prog != null) {
```

```
                prog.setProgress(progress);
                updateLabel(label, progress);
                updateOverall();
            }
        });
    }

    private void updateLabel(Label lab, double prog) {
                String text = (double)round(prog*10000)/100+"";
                if (text.length()<5)
                    text += "0%";
                else text += "%";
                lab.setText(text);

                if (prog > 0.5)
                    lab.setTextFill(Color.rgb(106, 135, 89));
                else lab.setTextFill(Color.rgb(204, 120, 50));
    }

    private void updateOverall() {
        double prog =
              ( .9825 * progOf(progUniGr)
              + .9815 * progOf(progUniEn)
              + .9768 * progOf(progBiGr)
              + .9818 * progOf(progBiEn)
              + .8379 * progOf(progTagTriGr)
              + .9735 * progOf(progTagTriEn)
              + .9597 * progOf(progTagQuaGr)
              + .9869 * progOf(progTagQuaEn)
              + .8729 * progOf(progSynGr)
              + .8641 * progOf(progSynEn))
              /(.9825 + .9815 + .9768 + .9818 +
                .8379 + .9735 + .9597 + .9869 +
                .8729 + .8641);

        progOverall.setProgress(prog);
        updateLabel(txtOverall, prog);
    }

    private double progOf(ProgressBar bar) {
        if (bar.getProgress()==.0) return .5;
        return bar.getProgress();
    }

    public void setLearnSum(int learnSum) {
        learningSum = learnSum;
        learningCur = 0;
        btnDisabled(true);
    }

    public void updateLearn() {
        Platform.runLater(()->{
            progLearn.setProgress((double)learningCur++/learningSum);
            if (learningCur == learningSum) {
                progLearn.setProgress(0);
                btnDisabled(false);
```

```
            }
        });
    }

    public void btnDisabled(boolean wuh) {
        Platform.runLater(()->{
            if (wuh) progLearn.setProgress(-1);
            else progLearn.setProgress(0);
            btnCheck.setDisable(wuh);
            btnFiles.setDisable(wuh);
            btnInput.setDisable(wuh);
            chkVera.setDisable(wuh);
            chkRare.setDisable(wuh);
            chkSmoo.setDisable(wuh);
            chkLink.setDisable(wuh);
            input.setEditable(!wuh);
            strlen = 0;
        });
    }

    public void cls() {
        Platform.runLater(()->{
            console.getChildren().clear();
        });
    }
}
```

TextProcess.java

```
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.text.BreakIterator;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.jsoup.Jsoup;
import org.languagetool.JLanguageTool;
import org.languagetool.language.Greek;
import org.languagetool.rules.RuleMatch;

public class TextProcess {
    private boolean eng;
    private BoTags bot;
    private BoWords bow;
    private String text;
    private String[] phrase;
    private List<String> phrases;
    private String translation = "";

    public TextProcess(String t) throws IOException {
        this(t, false);
    }

    public TextProcess(String t, boolean eng) throws IOException {
        if (t.startsWith("http")) {
            InputStream is = new URL(t).openStream();
            text = Jsoup.parse(is,"UTF-8",t).body().text();
        } else text = t;

        phrases = new ArrayList();
        text = text.replace("<br>", " ")
                .replaceAll("\\s{2,}", " ");
        this.eng = eng;

        Locale loc = new Locale("el","GR");
        if (eng) loc = new Locale("en","US");

        BreakIterator sentenceIterator =
            BreakIterator.getSentenceInstance(loc);

        int prev = 0;
        sentenceIterator.setText(text);
        int next = Math.max(sentenceIterator.next(),0);
        do {
            phrases.add(text.substring(prev, next));
            prev = next;
```

```
            next = sentenceIterator.next();
        } while (next > BreakIterator.DONE);
    }

    public String getText() {
        return text;
    }

    public String[] getPhrases() {
        if (phrase == null)
            phrase = phrases.toArray(
                    new String[phrases.size()]);
        return phrase;
    }

    public String getSimple() {
        if (!this.isEnglish())
            return text     // Remove any non-Greek character.
                    .replaceAll("[^\\p{InGreek}]+", " ")
                    .replaceAll("\\s{2,}", " ")
                    .trim().toLowerCase();

        return text     // Remove punctuation.
                .replaceAll("\\p{Punct}", " ")
                .replace("\n"," ").replace("\r", " ")
                .replace("\t", " ").replace("·"," ")
                .replace("«"," ").replace("»"," ")
                .replace("–", " ").replace("•", " ")
                .replace("("," ").replace(")"," ")
                .replace("|", " ")
                .replaceAll("\\d+"," ")
                .replaceAll("\\s{2,}", " ")
                .trim().toLowerCase();
    }

    public TextProcess getEnglish() throws IOException {
        if (getSimple().equals("")
                || isEnglish())
            return this;

        if (translation.equals("")) {
            try {
                ExecutorService service
                        = Executors.newFixedThreadPool(1);
                Future<String> translate = service.submit(
                        new Translation(getPhrases()));
                translation = translate.get();
                service.shutdownNow();
            } catch (Exception ex) {
                Logger.getLogger(TextProcess.class.getName())
                        .log(Level.SEVERE, null, ex); }}
        return new TextProcess(translation, true);
    }

    public Map<String, Integer> getCount(String method) {
        switch (method) {
```

```
            case "tqgr":
            case "tqen":
                if (bot == null) bot = new BoTags(this);
                return bot.getTagQuaCount();
            case "ttgr":
            case "tten":
                if (bot == null) bot = new BoTags(this);
                return bot.getTagTriCount();
            case "bigr":
            case "bien":
                if (bow == null) bow = new BoWords(this);
                return bow.getBigramCount();
            case "unigr":
            case "unien":
                if (bow == null) bow = new BoWords(this);
                return bow.getUnigramCount();
            default: return null;
        }
    }

    public int spellcheck() throws IOException {
        String text = this.text
                .replaceAll("[^\\p{InGreek}]+", " ")
                .replaceAll("\\s{2,}", " ").trim();

        JLanguageTool tool = new JLanguageTool(new Greek());
        List<RuleMatch> matches = tool.check(text);

        if (matches.size()>0)
            System.out.println("Erroneous Words: ");

        for (RuleMatch match : matches) {
            String error = text.substring(match.getFromPos(),
                    match.getToPos());
            System.out.print(error + " ");
        }
        System.out.println("\nNumber of Errors: "+matches.size());
        return matches.size();
    }

    public String[] getTagOrder() {
        if (bot == null) bot = new BoTags(this);
        return bot.getTagOrder();
    }

    public boolean isEnglish() {
        return eng;
    }
}
```

Translation.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import static java.lang.Thread.sleep;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.concurrent.Callable;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONArray;

public class Translation implements Callable<String> {
    private int maxThreads = 20;
    private String[] translated;
    private String[] phrases;

    public Translation(String[] p) {
        translated = new String[p.length];
        phrases = p;
    }

    @Override
    public String call() throws Exception {
        Thread[] threads = new Thread[maxThreads];
        String translation = "";

        for (int t=0;t<maxThreads;t++)
            threads[t] = createThread(t);


        boolean activeThreads = true;
        while (activeThreads) {
            activeThreads = false;
            for (Thread t : threads)
                if (t.isAlive()) activeThreads = true;
            sleep(50); }

        for (int p=0; p<translated.length; p++)
            translation += translated[p] + " ";
        return translation;
    }

    private Thread createThread(int t) {
        Thread thread = new Thread() {
            @Override public void run() { try {
               for (int c=0;c<phrases.length;c++)
                   if (c%maxThreads==t) {
                       translated[c] = translate(phrases[c]);
                       System.out.println(c+" "+phrases[c]);
                       System.out.println(c+" "+translated[c]);
                   }

               sleep(50);
               } catch (Exception ex) { Logger.getLogger(
```

```
                        Translation.class.getName())
                        .log(Level.SEVERE, null, ex);
                }}};

        thread.setPriority(Thread.MAX_PRIORITY);
        thread.start();
        return thread;
    }

    private String translate(String phrase) {
        try {
            String url = "https://translate.googleapis.com/"
                    + "translate_a/single?client=gtx"
                    + "&sl=el&tl=en&dt=t&q="
                    + URLEncoder.encode(phrase, "UTF-8");

            URL obj = new URL(url);
            HttpURLConnection con = (HttpURLConnection)
                    obj.openConnection();
            con.setRequestProperty("User-Agent", "Mozilla/5.0");

            BufferedReader in = new BufferedReader(
                    new InputStreamReader(con.getInputStream()));
            String inputLine;

            StringBuffer response = new StringBuffer();
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine); }
            in.close();

            JSONArray jsonArray =
                    new JSONArray(response.toString());
            JSONArray jsonArray2 = (JSONArray) jsonArray.get(0);
            JSONArray jsonArray3 = (JSONArray) jsonArray2.get(0);
            return jsonArray3.get(0).toString();

        } catch (Exception ex) {
            System.out.println("Translation of '"
                    + phrase + "' unavailable!");
            return "";
        }
    }
}
```

DBControl.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

// Database controlling class
public class DBControl {
    private Map<String, double[]> readCache = new HashMap<>();
    private String sqlSelect = "SELECT * FROM $ WHERE word = ?";
    private String sqlInsert = "INSERT OR IGNORE INTO $ VALUES (?,?,?)";
    private String sqlUpdate = "UPDATE $ SET good = good + ?,"
                + " fake = fake + ? WHERE word IS ?";

    private String[] types = { "unigr", "unien", "bigr",
            "bien", "ttgr", "tten", "tqgr", "tqen",
            "syngr", "synen", "spell" };

    private final int numMeth = types.length;
    private int sumGood[] = new int[numMeth];
    private int sumFake[] = new int[numMeth];
    private boolean dbChanged = true;
    private final String file;
    private int minValue = 0;
    private Connection conn;

    public DBControl(String file) throws SQLException {
        this.file = file;   // Connecting to the database
        conn = DriverManager.getConnection("jdbc:sqlite:"+file);
        Statement query = conn.createStatement();

        // The first batch of queries make accessing and
        // editing the database a lot faster at the cost
        // of potential data loss of unsaved changes in
        // a case of system failure.

        // main.case_size sets database's cache to 100MiB,
        // in case of low system memory it should be changed.
        query.execute("PRAGMA main.cache_size = -100000");

        // locking_mode grants exclusive access to this
        // connection, the database can't be accessed twice.
        query.execute("PRAGMA locking_mode = EXCLUSIVE");

        // journal_mode is set to memory, changes made to
        // the database are saved temporary on RAM which
        // saves a lot of HDD reads and writes.
        query.execute("PRAGMA journal_mode = MEMORY");
```

```
        // synchronous set to OFF makes commits a lot
        // faster at the cost of data corruption in
        // case of operating system failure.
        query.execute("PRAGMA synchronous = OFF");

        // Initialization of tables that are used by the
        // application, if they don't already exist.
        for (int t=0; t<types.length-3; t++)
            query.execute("CREATE TABLE IF NOT EXISTS " + types[t] +
                    " (word VARCHAR(255) UNIQUE, good INT, fake INT)");
        query.execute("CREATE TABLE IF NOT EXISTS spell "
                + "(wgood INT, wfake INT, good INT, fake INT)");
        query.executeUpdate("INSERT INTO spell VALUES (0,0,0,0)");
        query.close();

        // Getting the total number of words and PoS.
        updateMax();
    }

    public void addWord(String word, int count,
            boolean veracity, String table) {

        int good = 0;
        int fake = 0;
        dbChanged = true;

        try {
            word = word.toLowerCase();
            if (veracity) good += count;
            else fake += count;

            updateQuery(table,word,good,fake);

        } catch (SQLException ex) {
            Logger.getLogger(DBControl.class.getName())
                    .log(Level.SEVERE, null, ex);
        }
    }

    public void addSpell(int errorCount,
            int wordCount, boolean veracity) {
        int good = 0;
        int fake = 0;
        int wgood = 0;
        int wfake = 0;

        try (Statement query = conn.createStatement()) {

        if (veracity) {
            good += errorCount;
            wgood += wordCount;
        } else {
            fake += errorCount;
            wfake += wordCount;
        }
```

```
        query.executeUpdate("UPDATE spell SET" +
                " wgood = wgood + " + wgood +
                ", wfake = wfake + " + wfake +
                ", fake = fake + " + fake +
                ", good = good + " + good);

        } catch (SQLException ex) {
            Logger.getLogger(DBControl.class.getName())
                    .log(Level.SEVERE, null, ex);
        }
    }

    public double[] getSpell() {
        int wgood = 0;
        int good = 0;
        int fake = 0;
        int wfake = 0;

        try (Statement query = conn.createStatement()){

            ResultSet res = query
                    .executeQuery("SELECT * FROM spell");

            fake = res.getInt("fake");
            good = res.getInt("good");
            wgood = res.getInt("wgood");
            wfake = res.getInt("wfake");
            query.close();

            return new double[] {
                (double)good/wgood,
                (double)fake/wfake
            };
        } catch (SQLException ex) {
            Logger.getLogger(DBControl.class.getName())
                    .log(Level.SEVERE, null, ex);
            return null;
        }

    }

    public double[] getWord(String word, String table) {
        int type = Arrays.asList(types).indexOf(table);
        double[] result;
        int good = 0;
        int fake = 0;

        try {

            word = word.toLowerCase();
            if (dbChanged) updateMax();

            result = readCache.get(word);
            if (result != null) return result;
```

```
            PreparedStatement pstat;
            pstat = conn.prepareStatement(
                    sqlSelect.replace("$",table));
            pstat.setString(1, word);

            ResultSet res = pstat.executeQuery();
            if (res.isBeforeFirst()) {
                fake = res.getInt("fake");
                good = res.getInt("good");
            } pstat.close();

            result = new double[] {
                (double)good/sumGood[type],
                (double)fake/sumFake[type]
            };

            readCache.putIfAbsent(word, result);
            return result;

        } catch (SQLException ex) {
            Logger.getLogger(DBControl.class.getName())
                    .log(Level.SEVERE, null, ex);
            return null;
        }
    }

    private void updateMax() throws SQLException {
        for (int i=0; i<types.length-3; i++) {
            Statement query = conn.createStatement();
            ResultSet res = query.executeQuery(
                    "SELECT SUM(good), SUM(fake)"
                            + " FROM "+ types[i]);
            sumGood[i] = res.getInt("SUM(good)");
            sumFake[i] = res.getInt("SUM(fake)");
            query.close();
        }
        readCache.clear();
        dbChanged = false;
    }

    private void insertQuery(String table,
            String word) throws SQLException {
        PreparedStatement pstat;
        pstat = conn.prepareStatement(
                sqlInsert.replace("$",table));
        pstat.setString(1, word);
        pstat.setInt(2, minValue);
        pstat.setInt(3, minValue);
        pstat.executeUpdate();
        pstat.close();
    }

    private void updateQuery(String table,
            String word, int good, int fake)
            throws SQLException {
```

```
        conn.setAutoCommit(false);
        insertQuery(table,word);

        PreparedStatement pstat;
        pstat = conn.prepareStatement(
                sqlUpdate.replace("$", table));
        pstat.setInt(1, good);
        pstat.setInt(2, fake);
        pstat.setString(3, word);
        pstat.executeUpdate();
        pstat.close();

        conn.commit();
        conn.setAutoCommit(true);
    }

    public void closeConnection()
            throws SQLException {
        conn.close();
    }

    public void setMinValue(int val) {
        minValue = val;
    }

    public String[] getTypes() {
        return types;
    }
}
```

BoWords.java

```
import java.util.HashMap;
import java.util.Map;

// Bag of Words class.
public class BoWords {

    private String[] words;
    private Map<String, Integer> unigramCount
            = new HashMap<String, Integer>();
    private Map<String, Integer> bigramCount
            = new HashMap<String, Integer>();

    // Constructor.
    public BoWords(TextProcess text) {

        // Extracting words from text.
        words = text.getSimple().split(" ");

        // Counting bigrams.
        for (int c = 0; c < words.length - 1; c++) {
            String comb = words[c]+" "+words[c+1];
            bigramCount.putIfAbsent(comb, 0);
            bigramCount.put(comb, bigramCount.get(comb)+ 1);
        }

        // Counting unigrams.
        for (int c = 0; c < words.length; c++) {
            unigramCount.putIfAbsent(words[c], 0);
            unigramCount.put(words[c],
                    unigramCount.get(words[c])+ 1);
        }
    }

    public Map<String, Integer> getBigramCount() {
        return bigramCount;
```

```
    }

    public Map<String, Integer> getUnigramCount() {
        return unigramCount;
    }
}
```

BoTags.java

```
import POStagger.RDRPOSTagger;
import POStagger.Utils;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

// Bag of Part of Speech class.
public class BoTags {
    private Map<String, Integer> tagTriCount = new HashMap<>();
    private Map<String, Integer> tagQuaCount = new HashMap<>();
    private List<String> tags;
    private String[] phrase;
    private String[] tagged;
    private String[] tOrder;

    // Constructor.
    public BoTags(TextProcess tp) {
        phrase = tp.getPhrases();
        tagged = new String[phrase.length];
        tOrder = new String[phrase.length];
        tags = new ArrayList<>();

        try {   // PoS-tagging.
            RDRPOSTagger tree = new RDRPOSTagger();
            HashMap<String, String> FREQDICT;

            if (tp.isEnglish()) {
                tree.constructTreeFromRulesFile(
                        "Models/POS/English.RDR");
                FREQDICT = Utils.getDictionary(
                        "Models/POS/English.DICT");
            } else {
                tree.constructTreeFromRulesFile(
                        "Models/UniPOS/UD_Greek/el-upos.RDR");
                FREQDICT = Utils.getDictionary(
                        "Models/UniPOS/UD_Greek/el-upos.DICT");
            }

            // PoS tagging each phrase(sentence) of text.
            for (int p = 0; p < phrase.length; p++) {
                tagged[p] = tree.tagSentence(FREQDICT,phrase[p]);
                System.out.println(tagged[p]);
                tOrder[p] = "$start$ ";

                // Splitting senteces in to words and tags.
                String[] wordWithTags = tagged[p].split(" ");
                for (String wordWithTag : wordWithTags) {
                    String[] split = wordWithTag.split("/");
                    String tag = split[split.length-1];
                    tOrder[p] += tag + " ";
                    tags.add(tag);
                }
```

```
                tOrder[p] += "$$end$$";
            }


            // Counting tags.
            for (String phrase : tOrder) {
                String[] tag = phrase.split(" ");

                // Trigram.
                for (int c = 0; c < tag.length - 2; c++) {
                    String comb = tag[c]+" "
                                  +tag[c+1]+" "+tag[c+2];
                    tagTriCount.putIfAbsent(comb, 0);
                    tagTriCount.put(comb,
                                  tagTriCount.get(comb) + 1);
                }

                // Quadgram.
                for (int c = 0; c < tag.length - 3; c++) {
                    String comb = tag[c]+" "+tag[c+1]+
                            " "+tag[c+2]+" "+tag[c+3];
                    tagQuaCount.putIfAbsent(comb, 0);
                    tagQuaCount.put(comb,
                                  tagQuaCount.get(comb) + 1);
                }
            }

        } catch (IOException ex) {
            System.out.println("Greek UPoS are missing!");
        }
    }

    public Map<String, Integer> getTagTriCount() {
        return tagTriCount;
    }

    public Map<String, Integer> getTagQuaCount() {
        return tagQuaCount;
    }

    public String[] getTagOrder() {
        return tOrder;
    }
}
```

Tester.java

```
import FakeDetection.DBControl;
import FakeDetection.FakeDetection;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import static java.lang.Math.round;
import static java.lang.System.out;
import static java.lang.Thread.sleep;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.util.Pair;

public class Tester {
    static Pair<String, Double>[] result;
    final static int numMeth = 11;
    static List<String> good;
    static List<String> fake;
    static FakeDetection fd;
    static int falseGood[];
    static int falseFake[];
    static PrintWriter pw;
    static Random rand;

    public static void main (String args[]) {
        int reps = 1;
        if (args.length > 0)
            reps = Integer.parseInt(args[0]);
        for (int r=0; r<reps; r++)
            testAccuracy();
    }

    private static void testAccuracy() {
        try {
            Files.deleteIfExists(Paths.get("test.db"));
            fd = new FakeDetection("test.db");
            fd.setLinks(false);
            fd.setSpell(false);
            fd.setRare(true);
            fd.setMin(1);

            falseGood = new int[numMeth];
            falseFake = new int[numMeth];
            good = new ArrayList();
            fake = new ArrayList();
            rand = new Random();
```

```
            learnFrom(true);
            learnFrom(false);

            checkAccuracy(true);
            checkAccuracy(false);

            printResults();
            fd.close();

        } catch (Exception ex) {
            Logger.getLogger(DBControl.class.getName())
                    .log(Level.SEVERE, null, ex);
            System.out.println("Database is in use! "
                    + "Retrying in few seconds...");
            try { sleep(50000); }
            catch (InterruptedException eex) { }
            finally { testAccuracy(); }
        }
    }

    private static void learnFrom(boolean vera) {
        List<String> pool = new ArrayList();
        String file;

        if (vera) file = "good";
        else file = "fake";

        try(BufferedReader br = new BufferedReader(
                new FileReader("News/" + file + ".txt"))) {

            String url = br.readLine();
            while (url != null) {
                if(rand.nextFloat()>.2)
                    fd.learnFromURL(url, vera);
                else pool.add(url);
                url = br.readLine();
            }

        } catch (IOException ex) {
            Logger.getLogger(Tester.class.getName())
                    .log(Level.SEVERE, null, ex);
            System.out.println("The file '" + file +
                    ".txt' in News folder was not found.");
        }

        if (vera) good.addAll(pool);
        else fake.addAll(pool);
    }

    private static void checkAccuracy(boolean vera) {
        List<String> pool = new ArrayList();
        int falsePool[] = new int[numMeth];
        int ineq = 1;

        if (vera) pool.addAll(good);
        else {
```

```
            pool.addAll(fake);
            ineq *= -1;
        }

        for (String url : pool) {
            result = fd.checkVeracity(url);
            if (result != null)
                for (int c=0; c<result.length; c++)
                    if (result[c] != null)
                        if (ineq*result[c].getValue()<.5*ineq)
                            falsePool[c]++;
        }

        for (int c=0; c<falsePool.length; c++)
            if (vera) falseGood[c] += falsePool[c];
            else falseFake[c] += falsePool[c];
    }

    private static void printResults() {
        try(FileWriter fw
                = new FileWriter("TestResults.txt", true);
                BufferedWriter bw = new BufferedWriter(fw)) {

            pw = new PrintWriter(bw);
            String label[] = { "UniGr",  "UniEn", "BiGr",
                "BiEn", "TTGr", "TTEn", "TQGr", "TQEn",
                "SynGr", "SynEn" };

            println("\n");
            for(int c=0; c<label.length; c++)
                println(label[c] + ": \t" + falseGood[c]
                        + ", \t" + falseFake[c] + ", \t"
                        + (falseGood[c] + falseFake[c])+ ", \t"
                        + roundPercent(1 - ((double)(falseGood[c]
                        + falseFake[c])) /
                                (good.size() + fake.size())));
            println("Total: \t" + good.size() + ", \t"
                    + fake.size() + ", \t" + (good.size()
                    + fake.size()) + "\n");

        } catch (IOException ex) {
            Logger.getLogger(Tester.class.getName())
                    .log(Level.SEVERE, null, ex);
        }
    }

    private static void println(String s) {
        out.println(s);
        pw.println(s);
    }

    private static String roundPercent(double num) {
        return (double)round(num*10000)/100+"%";
    }
}
```

Παράρτημα Β: Στατιστικά Δοκιμών

Στατιστικά 1ης Δοκιμής

Bag of Words				Bag of PoS				Syntax		Spell
Unigram		Bigram		Unigram		Bigram				
GR	EN	GR	EN	GR	EN	GR	EN	GR	EN	
98,08%	99,52%	99,04%	98,56%	71,15%	87,02%	74,52%	97,60%	72,60%	93,75%	49,52%
100,00%	100,00%	96,73%	98,13%	70,09%	88,32%	79,91%	96,73%	74,30%	94,86%	42,99%
98,08%	98,08%	96,63%	96,63%	65,38%	84,13%	71,15%	93,75%	69,23%	91,35%	44,23%
98,54%	99,51%	98,06%	98,54%	67,48%	84,95%	71,84%	95,15%	70,39%	93,20%	45,63%
99,11%	99,56%	96,89%	97,33%	63,11%	88,00%	72,00%	94,22%	66,67%	91,11%	36,89%
97,93%	97,93%	97,41%	97,41%	68,91%	82,90%	82,54%	92,75%	72,02%	91,19%	46,63%
99,01%	98,52%	98,52%	97,04%	64,04%	86,21%	71,43%	93,10%	64,04%	91,63%	34,98%
98,00%	99,00%	97,00%	98,50%	65,50%	84,00%	70,50%	94,50%	67,00%	89,50%	42,00%
99,11%	99,11%	98,21%	98,21%	66,52%	85,71%	70,98%	93,75%	68,75%	91,07%	44,64%
97,91%	98,95%	96,86%	97,91%	68,59%	84,82%	71,73%	95,81%	68,59%	88,48%	43,98%
99,50%	99,50%	98,51%	99,50%	68,66%	88,56%	74,13%	97,01%	70,15%	89,55%	38,81%
96,94%	98,47%	98,47%	97,96%	73,47%	89,29%	76,53%	94,90%	74,49%	87,24%	42,86%
99,00%	98,51%	98,01%	98,01%	72,14%	85,07%	76,62%	91,54%	71,64%	91,54%	36,32%
99,00%	99,50%	98,00%	98,00%	68,00%	90,00%	71,50%	95,00%	67,00%	94,00%	39,50%
98,61%	99,54%	98,61%	97,69%	68,52%	87,04%	73,15%	93,52%	66,67%	93,52%	38,43%
97,63%	98,10%	98,10%	97,16%	70,14%	87,68%	75,36%	94,31%	68,25%	91,47%	38,39%
97,95%	99,49%	96,92%	98,97%	68,21%	89,74%	87,82%	94,36%	70,26%	89,74%	41,54%
98,54%	99,02%	97,56%	98,05%	66,34%	89,27%	68,78%	93,17%	67,80%	86,34%	46,83%
98,64%	99,55%	98,64%	96,36%	68,64%	86,82%	73,64%	93,64%	71,82%	88,18%	40,45%
98,57%	99,05%	98,57%	99,05%	68,10%	82,38%	70,48%	92,86%	70,00%	88,10%	39,05%
98,49%	98,49%	98,49%	98,49%	63,82%	84,92%	69,85%	93,47%	68,34%	87,94%	44,22%
97,38%	98,43%	97,38%	97,91%	67,54%	82,72%	70,16%	91,62%	67,02%	84,29%	40,84%
98,58%	98,58%	97,64%	98,11%	63,21%	87,74%	72,64%	94,81%	69,81%	94,34%	36,32%
97,17%	97,17%	96,23%	97,17%	66,04%	79,72%	67,45%	90,57%	65,57%	79,72%	67,92%
98,12%	99,06%	98,59%	99,06%	70,89%	83,57%	74,18%	93,90%	73,24%	69,01%	64,79%
99,51%	98,53%	99,51%	98,04%	75,49%	79,41%	74,51%	94,12%	65,20%	77,45%	66,18%
99,51%	98,52%	98,03%	99,01%	74,88%	84,73%	73,40%	97,54%	66,01%	81,77%	69,95%
98,91%	99,46%	98,37%	100,00%	72,83%	86,96%	75,00%	95,11%	71,20%	80,43%	64,13%
99,52%	99,52%	99,04%	99,04%	80,29%	88,46%	81,25%	96,15%	82,67%	86,54%	39,90%
97,57%	98,06%	96,12%	97,57%	66,02%	83,01%	64,56%	92,72%	63,59%	88,35%	35,92%
99,02%	99,02%	98,53%	99,02%	63,73%	83,82%	69,12%	92,65%	64,22%	83,82%	67,65%
99,05%	99,53%	98,58%	99,05%	72,04%	87,68%	74,41%	96,21%	69,67%	87,20%	72,51%
99,49%	100,00%	97,44%	97,95%	60,51%	84,10%	62,05%	94,36%	64,10%	88,72%	69,23%
98,36%	98,91%	97,27%	96,72%	64,48%	87,43%	80,87%	91,80%	75,96%	86,89%	34,97%
99,10%	99,55%	98,21%	99,10%	72,20%	81,17%	76,68%	95,96%	73,54%	91,93%	43,50%

99,09%	99,09%	98,18%	98,64%	68,18%	85,45%	75,45%	96,82%	73,64%	91,36%	43,64%
97,62%	98,10%	98,10%	98,10%	72,38%	82,38%	74,29%	93,33%	70,00%	90,95%	39,05%
98,95%	98,42%	98,42%	98,95%	64,21%	82,63%	67,37%	95,26%	65,26%	91,05%	42,63%
99,48%	99,48%	98,44%	98,44%	67,71%	86,46%	78,13%	92,19%	73,96%	86,98%	42,71%
99,05%	99,52%	96,19%	97,62%	58,57%	85,24%	68,10%	93,81%	65,71%	89,52%	39,52%
97,13%	98,09%	96,17%	98,09%	69,38%	80,86%	76,08%	92,82%	71,77%	91,87%	44,50%
98,57%	**98,94%**	**97,85%**	**98,17%**	**68,23%**	**85,37%**	**73,42%**	**94,22%**	**69,56%**	**88,44%**	**46,43%**

Στατιστικά 2ης Δοκιμής

Bag of Words				Bag of PoS				Syntax	
Unigram		Bigram		Trigram		Quadgram			
GR	EN	GR	EN	GR	EN	GR	EN	GR	EN
99,08%	99,08%	97,70%	99,08%	88,48%	99,54%	97,70%	99,54%	87,56%	82,95%
98,76%	98,34%	99,17%	99,59%	78,42%	97,51%	95,44%	99,17%	82,99%	89,21%
99,58%	98,31%	97,88%	99,15%	74,58%	94,49%	94,49%	98,73%	85,59%	90,68%
97,10%	96,27%	96,27%	97,51%	83,82%	96,68%	95,85%	97,93%	88,38%	90,46%
97,07%	98,54%	98,05%	99,02%	82,93%	97,56%	96,59%	98,54%	85,85%	86,34%
95,97%	97,58%	95,97%	98,39%	84,27%	95,16%	96,77%	98,79%	85,08%	87,90%
95,78%	97,47%	95,78%	98,31%	74,26%	96,20%	88,61%	98,31%	81,86%	88,19%
98,29%	97,44%	97,01%	97,44%	78,63%	96,15%	94,02%	99,15%	89,32%	91,03%
98,10%	99,05%	98,58%	98,58%	79,15%	98,58%	95,73%	98,58%	85,31%	91,00%
97,85%	98,28%	97,85%	98,28%	87,12%	97,85%	97,85%	99,14%	91,42%	91,42%
98,20%	99,55%	96,85%	98,20%	87,39%	98,20%	96,85%	99,55%	87,39%	89,19%
98,66%	98,66%	97,32%	97,77%	84,82%	98,21%	98,21%	98,66%	86,16%	92,41%
96,96%	99,13%	97,83%	99,57%	82,61%	99,13%	95,22%	99,57%	86,09%	89,57%
96,38%	98,19%	95,48%	96,83%	82,81%	99,10%	89,59%	98,19%	79,19%	84,62%
97,64%	98,11%	95,75%	97,17%	85,85%	96,70%	96,23%	98,58%	86,32%	89,15%
97,27%	95,45%	96,82%	97,73%	79,55%	94,09%	94,09%	93,64%	84,09%	89,55%
97,52%	97,93%	96,28%	98,35%	74,79%	95,87%	92,98%	98,35%	81,82%	89,67%
98,67%	98,22%	97,78%	99,56%	80,00%	96,44%	96,44%	98,67%	87,11%	86,67%
98,06%	96,12%	97,57%	97,09%	89,32%	94,66%	96,12%	96,60%	87,86%	87,86%
99,11%	99,56%	98,22%	99,11%	88,89%	97,78%	98,22%	98,67%	91,11%	91,56%
98,74%	99,58%	98,74%	100,00%	87,03%	97,07%	96,65%	98,74%	88,70%	87,03%
98,24%	97,80%	96,92%	98,24%	80,62%	95,59%	93,83%	97,80%	79,74%	85,46%
99,07%	98,61%	99,54%	99,07%	87,04%	98,61%	96,76%	99,54%	92,13%	87,50%
98,48%	97,97%	98,48%	98,48%	80,20%	98,48%	97,46%	98,98%	87,82%	85,28%
97,44%	97,01%	97,01%	98,29%	84,19%	97,44%	96,58%	98,72%	86,32%	84,62%
99,07%	96,74%	98,14%	97,67%	84,19%	93,49%	95,81%	96,28%	86,51%	89,77%
98,28%	98,28%	99,14%	97,85%	83,69%	97,42%	96,57%	99,57%	87,12%	88,41%
97,46%	97,03%	97,88%	98,31%	86,86%	95,76%	97,46%	98,73%	90,68%	90,25%
99,09%	96,80%	98,63%	99,09%	82,65%	94,98%	94,98%	97,26%	86,76%	88,58%
98,78%	98,37%	96,75%	98,78%	87,40%	97,15%	97,56%	99,19%	89,84%	85,77%
99,07%	98,15%	98,61%	99,07%	88,43%	98,15%	99,07%	99,07%	90,28%	91,20%
98,62%	99,08%	97,71%	99,08%	92,66%	98,17%	99,08%	99,54%	93,58%	94,04%
98,13%	98,13%	96,73%	98,13%	84,58%	98,60%	95,33%	99,53%	85,98%	94,39%
98,73%	99,16%	97,89%	97,89%	85,23%	97,89%	97,47%	99,16%	87,76%	92,83%
97,58%	97,98%	97,58%	97,58%	86,29%	97,58%	95,97%	98,39%	87,50%	90,32%
99,11%	98,21%	98,21%	95,54%	83,93%	96,43%	95,54%	98,21%	86,16%	92,41%
98,24%	98,24%	98,24%	96,92%	77,97%	97,36%	92,95%	98,68%	85,46%	89,87%

99,12%	99,56%	97,79%	99,56%	84,51%	97,35%	96,90%	99,56%	86,28%	91,59%
99,07%	99,07%	97,66%	98,13%	85,98%	96,73%	95,79%	98,13%	84,11%	90,65%
98,73%	97,47%	97,47%	98,31%	86,08%	97,47%	96,20%	99,16%	88,19%	91,14%
98,68%	96,48%	98,68%	97,80%	86,78%	96,48%	96,92%	98,68%	90,75%	89,43%
98,73%	97,47%	97,05%	97,05%	79,75%	94,51%	94,09%	98,31%	84,81%	90,72%
97,71%	98,62%	97,71%	97,71%	83,03%	97,25%	97,71%	98,62%	87,16%	91,74%
98,37%	99,18%	97,96%	97,96%	82,45%	98,37%	95,92%	98,78%	85,71%	90,20%
97,57%	98,38%	96,36%	95,95%	85,02%	98,38%	97,17%	98,79%	87,45%	91,09%
98,06%	98,06%	97,09%	97,09%	81,07%	98,06%	95,15%	99,03%	83,98%	94,17%
98,70%	97,83%	96,96%	99,13%	83,04%	96,96%	95,65%	99,57%	87,83%	90,87%
98,60%	99,07%	98,60%	98,60%	86,45%	100,00%	97,20%	99,53%	92,99%	92,52%
97,84%	99,14%	97,41%	99,57%	81,90%	97,41%	92,24%	99,57%	82,33%	93,53%
98,70%	98,27%	96,97%	99,13%	85,28%	97,40%	96,54%	99,13%	88,31%	86,58%
97,33%	97,33%	97,33%	98,67%	82,22%	96,00%	95,11%	99,11%	90,22%	84,44%
99,11%	98,67%	99,11%	98,67%	84,89%	98,67%	98,22%	99,56%	91,56%	86,67%
99,08%	96,79%	99,08%	96,33%	80,28%	94,50%	93,12%	97,25%	87,16%	84,40%
99,07%	99,07%	99,07%	98,61%	85,19%	97,69%	98,15%	100,00%	92,13%	82,41%
98,33%	97,50%	96,25%	97,50%	89,58%	96,25%	98,75%	98,75%	87,50%	83,33%
99,18%	99,18%	97,55%	99,18%	86,12%	98,37%	99,18%	100,00%	92,24%	82,86%
96,22%	97,06%	96,22%	95,80%	86,55%	97,90%	96,22%	98,32%	89,08%	80,25%
98,35%	99,18%	97,94%	97,12%	89,71%	99,18%	97,94%	99,59%	89,30%	86,83%
95,28%	96,70%	94,81%	96,23%	80,66%	91,51%	94,34%	97,64%	84,91%	81,13%
97,56%	99,02%	99,02%	98,54%	80,00%	98,54%	95,12%	99,02%	83,90%	80,49%
98,26%	97,83%	96,96%	95,65%	86,96%	97,39%	96,52%	98,70%	88,70%	83,04%
97,37%	98,68%	98,25%	98,25%	85,09%	98,68%	96,49%	99,56%	91,67%	82,46%
98,65%	99,10%	99,10%	99,10%	87,44%	98,65%	95,96%	99,55%	91,93%	87,44%
96,26%	99,07%	96,73%	97,20%	81,78%	98,13%	94,39%	99,07%	88,79%	85,51%
99,13%	98,70%	98,27%	97,84%	88,74%	96,97%	98,70%	98,70%	87,45%	80,09%
99,57%	100,00%	99,57%	98,70%	80,00%	99,57%	93,48%	100,00%	82,61%	80,87%
98,63%	99,09%	98,63%	98,17%	87,21%	98,63%	95,89%	98,63%	91,32%	87,21%
99,13%	99,13%	98,70%	99,13%	87,45%	99,13%	96,97%	99,57%	88,31%	81,82%
98,00%	98,80%	98,80%	98,80%	82,80%	98,40%	96,80%	99,20%	87,20%	84,00%
96,33%	99,08%	96,33%	97,71%	83,94%	98,62%	94,50%	99,08%	84,40%	87,16%
98,68%	99,56%	98,25%	98,25%	89,47%	98,68%	97,81%	99,12%	92,54%	82,46%
97,79%	99,12%	97,35%	96,46%	91,15%	99,56%	98,67%	100,00%	90,27%	85,84%
97,79%	99,56%	97,79%	99,12%	79,65%	99,12%	94,69%	99,56%	83,19%	85,40%
97,33%	98,67%	98,67%	99,11%	77,33%	98,67%	94,22%	99,11%	84,89%	85,33%
98,68%	98,68%	97,37%	97,81%	89,91%	96,49%	97,37%	99,56%	91,67%	89,04%
97,14%	88,57%	98,57%	99,05%	82,38%	86,19%	94,29%	89,05%	85,24%	89,05%
98,60%	99,07%	99,07%	100,00%	84,65%	97,67%	93,49%	100,00%	91,16%	89,77%
98,01%	98,80%	95,22%	98,01%	81,27%	98,41%	91,63%	99,20%	81,27%	88,84%
98,35%	97,94%	98,77%	99,18%	83,54%	97,53%	93,42%	98,77%	87,24%	89,30%

98,18%	99,55%	98,18%	97,73%	85,91%	99,55%	95,00%	100,00%	87,73%	87,73%
97,75%	96,85%	95,95%	96,85%	83,78%	96,40%	93,24%	99,10%	84,23%	87,84%
99,13%	98,70%	97,40%	97,84%	84,42%	98,70%	95,67%	99,57%	90,04%	89,61%
99,56%	100,00%	98,22%	99,56%	86,22%	98,67%	97,33%	100,00%	91,11%	90,22%
98,73%	99,58%	98,73%	99,16%	81,86%	100,00%	96,20%	100,00%	83,54%	94,09%
99,09%	97,26%	97,26%	97,26%	83,56%	96,35%	93,61%	98,63%	83,56%	87,21%
97,72%	98,17%	97,72%	98,17%	85,39%	98,17%	97,26%	98,63%	89,95%	86,76%
99,06%	99,53%	99,06%	98,58%	83,02%	99,53%	94,81%	99,06%	86,32%	83,96%
97,93%	99,17%	97,93%	97,93%	87,19%	97,52%	97,93%	99,17%	90,08%	83,88%
96,62%	97,10%	96,14%	96,62%	88,41%	96,62%	97,10%	98,07%	86,96%	80,68%
97,38%	89,08%	97,38%	96,94%	81,66%	87,34%	94,32%	90,39%	83,84%	82,53%
100,00%	99,54%	99,08%	99,54%	84,79%	97,24%	96,77%	99,08%	88,48%	81,57%
99,06%	99,53%	98,12%	98,12%	81,22%	96,71%	96,71%	99,06%	90,61%	77,00%
99,53%	99,53%	99,07%	99,53%	88,37%	99,07%	99,07%	100,00%	89,77%	80,93%
96,17%	98,30%	97,02%	97,87%	75,74%	97,02%	93,19%	97,87%	81,70%	82,98%
98,21%	98,21%	97,76%	98,21%	78,03%	98,65%	95,52%	98,65%	83,86%	83,86%
99,05%	100,00%	98,57%	99,52%	85,24%	98,10%	98,10%	99,52%	90,00%	86,19%
99,11%	98,67%	98,67%	98,67%	82,22%	97,33%	94,67%	99,11%	88,44%	82,22%
99,16%	89,03%	97,89%	98,31%	85,23%	89,45%	97,89%	90,30%	91,56%	78,90%
99,58%	98,73%	97,05%	98,73%	86,50%	98,31%	98,31%	99,58%	88,61%	83,97%
99,56%	98,25%	98,68%	97,37%	86,84%	98,68%	96,49%	99,56%	91,67%	80,26%
98,63%	99,09%	98,17%	98,17%	87,67%	97,72%	98,17%	99,54%	89,95%	85,39%
99,16%	99,58%	97,89%	98,31%	84,81%	98,73%	97,89%	99,58%	86,50%	86,08%
97,29%	98,19%	95,93%	97,29%	76,47%	97,74%	90,05%	97,74%	76,47%	78,73%
98,25%	98,68%	97,81%	97,81%	84,65%	98,68%	97,37%	99,12%	86,40%	79,39%
99,55%	98,21%	97,76%	98,65%	80,27%	98,21%	95,96%	99,55%	87,89%	82,96%
98,59%	99,06%	98,12%	98,59%	81,22%	98,12%	96,71%	98,59%	84,51%	81,69%
97,96%	98,78%	95,51%	97,96%	87,76%	97,55%	95,92%	98,37%	89,39%	80,00%
98,54%	99,02%	99,02%	99,02%	80,98%	99,51%	93,17%	99,02%	81,95%	84,39%
98,68%	98,25%	97,81%	99,56%	86,40%	99,12%	98,25%	100,00%	85,53%	82,89%
98,69%	99,13%	97,82%	98,69%	86,03%	98,25%	98,25%	99,13%	89,96%	84,72%
99,54%	99,54%	98,16%	99,08%	89,40%	99,54%	97,70%	99,54%	93,09%	82,95%
99,56%	98,67%	97,78%	99,56%	87,56%	98,67%	96,89%	100,00%	90,67%	85,33%
97,50%	98,75%	95,83%	97,92%	80,00%	99,17%	93,33%	99,58%	87,08%	82,92%
99,10%	98,21%	98,21%	98,21%	87,89%	97,76%	97,76%	98,65%	92,38%	79,37%
99,10%	99,55%	98,65%	99,10%	84,75%	98,21%	100,00%	99,55%	91,48%	86,55%
97,89%	99,16%	98,73%	99,16%	82,70%	98,31%	94,94%	99,58%	82,28%	84,81%
98,55%	97,58%	97,58%	98,07%	84,06%	98,07%	96,62%	99,52%	85,51%	89,37%
98,58%	98,11%	98,11%	98,58%	86,79%	96,23%	98,58%	99,53%	91,04%	84,91%
98,56%	90,91%	98,56%	98,56%	79,90%	89,95%	90,43%	90,43%	81,34%	82,78%
96,58%	98,72%	96,15%	97,44%	81,20%	98,29%	92,74%	98,72%	85,04%	87,18%
99,57%	100,00%	98,70%	99,57%	80,52%	100,00%	92,21%	100,00%	85,71%	84,85%

97,69%	98,61%	96,76%	98,61%	89,35%	97,69%	98,15%	99,07%	89,35%	83,80%
96,98%	93,10%	95,26%	96,12%	83,19%	91,81%	94,83%	93,53%	84,91%	88,36%
96,62%	97,89%	94,94%	95,78%	81,86%	97,47%	96,20%	98,31%	87,76%	75,95%
97,92%	99,17%	96,25%	98,33%	75,83%	100,00%	90,83%	100,00%	84,17%	83,75%
98,18%	99,09%	96,82%	98,18%	81,36%	98,64%	97,27%	99,55%	85,91%	81,82%
99,16%	99,58%	99,16%	98,74%	85,36%	98,74%	99,58%	100,00%	89,12%	82,01%
97,02%	98,72%	96,60%	97,87%	81,70%	94,89%	96,60%	98,30%	85,96%	83,83%
98,72%	97,86%	97,44%	99,15%	81,20%	97,44%	95,30%	99,15%	88,03%	84,62%
97,91%	98,33%	97,49%	97,49%	79,50%	97,91%	94,14%	99,16%	87,87%	86,61%
97,30%	96,40%	97,30%	96,40%	86,49%	96,40%	99,10%	98,65%	88,74%	81,08%
98,73%	98,31%	98,73%	97,88%	80,51%	97,46%	95,76%	99,58%	84,75%	91,95%
98,68%	96,49%	98,25%	96,93%	77,63%	97,81%	93,86%	98,68%	83,77%	90,79%
97,03%	97,88%	97,03%	97,03%	82,20%	97,46%	96,61%	99,15%	84,75%	88,98%
96,15%	98,29%	96,15%	97,44%	82,48%	96,58%	94,02%	98,72%	88,46%	89,74%
99,56%	98,68%	99,12%	98,24%	85,90%	99,12%	99,12%	100,00%	90,31%	92,51%
98,31%	98,31%	97,89%	97,89%	79,75%	97,89%	97,89%	98,31%	90,30%	85,23%
98,73%	99,58%	99,15%	98,73%	84,75%	98,31%	97,03%	99,15%	85,17%	85,17%
98,25%	**98,15%**	**97,68%**	**98,18%**	**83,79%**	**97,35%**	**95,97%**	**98,69%**	**87,29%**	**86,41%**

Στατιστικά 3ης Δοκιμής

Bag of Words				Bag of PoS				Syntax	
Unigram		Bigram		Trigram		Quadgram			
GR	EN	GR	EN	GR	EN	GR	EN	GR	EN
98,56%	95,22%	99,04%	99,04%	85,17%	94,74%	97,61%	95,22%	95,69%	92,34%
98,65%	99,55%	100,00%	99,55%	86,94%	98,65%	98,65%	99,55%	94,14%	95,05%
99,16%	98,74%	99,16%	99,16%	88,24%	97,48%	97,06%	98,32%	93,70%	92,44%
99,17%	99,17%	99,59%	99,17%	83,40%	97,93%	96,27%	99,17%	94,61%	94,19%
100,00%	100,00%	100,00%	100,00%	86,81%	98,30%	99,15%	99,57%	94,47%	94,47%
99,56%	99,13%	99,56%	99,56%	82,97%	97,82%	96,94%	99,13%	94,76%	94,76%
99,10%	100,00%	99,10%	99,55%	81,00%	99,10%	98,19%	100,00%	94,57%	91,40%
99,05%	99,05%	99,05%	99,05%	85,31%	98,10%	97,16%	98,58%	93,36%	92,42%
100,00%	99,53%	100,00%	100,00%	87,74%	98,58%	98,58%	99,06%	95,75%	94,81%
98,31%	98,31%	99,15%	98,73%	92,37%	98,31%	98,73%	98,31%	96,19%	94,92%
98,44%	98,83%	98,83%	99,22%	82,88%	98,44%	97,67%	98,83%	94,16%	94,55%
98,74%	98,74%	99,16%	98,74%	90,76%	99,16%	98,32%	98,74%	96,64%	90,34%
98,82%	99,21%	99,21%	99,21%	95,28%	100,00%	99,21%	99,61%	97,64%	95,67%
99,13%	98,70%	98,70%	99,13%	86,15%	97,84%	98,70%	98,70%	96,10%	93,07%
100,00%	100,00%	100,00%	100,00%	88,70%	99,58%	98,74%	99,58%	94,98%	95,40%
99,22%	99,22%	99,61%	99,22%	87,98%	97,67%	98,45%	98,84%	95,35%	93,02%
98,63%	98,63%	99,54%	99,09%	88,13%	98,17%	98,63%	98,63%	94,52%	89,50%
99,07%	99,07%	99,54%	99,54%	82,41%	99,07%	99,07%	99,07%	94,44%	93,06%
99,59%	99,18%	99,18%	99,59%	90,20%	97,55%	97,96%	98,78%	95,92%	92,65%
100,00%	99,55%	100,00%	99,55%	83,04%	98,21%	97,32%	99,55%	95,98%	93,30%
99,11%	98,21%	99,11%	99,55%	76,34%	97,77%	97,32%	99,55%	91,07%	94,64%
99,15%	99,15%	99,15%	98,72%	82,13%	97,87%	97,87%	98,30%	95,32%	94,89%
99,55%	99,55%	99,55%	99,55%	81,53%	98,65%	98,20%	100,00%	94,14%	95,05%
99,56%	99,12%	99,56%	99,56%	91,19%	99,12%	99,56%	99,12%	96,92%	92,07%
100,00%	100,00%	100,00%	100,00%	75,53%	99,58%	96,20%	100,00%	90,30%	96,20%
98,82%	99,61%	99,22%	99,61%	84,31%	98,04%	99,61%	99,61%	96,47%	88,24%
98,80%	98,80%	99,20%	98,80%	90,44%	98,41%	99,20%	99,20%	95,62%	91,24%
100,00%	100,00%	100,00%	99,59%	85,31%	97,96%	99,59%	99,59%	96,33%	93,88%
100,00%	99,53%	100,00%	100,00%	91,98%	98,58%	100,00%	99,53%	96,70%	90,57%
98,72%	99,57%	98,72%	99,15%	82,48%	98,72%	97,86%	99,57%	94,87%	90,60%
99,54%	99,08%	99,08%	99,54%	79,72%	97,70%	96,77%	99,54%	93,55%	94,47%
98,72%	98,72%	99,15%	98,72%	76,07%	97,86%	95,30%	98,72%	92,74%	89,74%
99,11%	99,11%	99,11%	99,55%	84,38%	98,66%	98,21%	98,66%	96,43%	93,75%
98,74%	98,32%	98,74%	99,16%	92,44%	97,90%	97,90%	98,32%	94,54%	92,44%
99,12%	99,12%	99,12%	99,12%	84,58%	96,92%	97,36%	99,12%	95,15%	96,04%
99,20%	**99,08%**	**99,38%**	**99,36%**	**85,54%**	**98,24%**	**98,10%**	**99,03%**	**94,95%**	**93,18%**