ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

<< Προθεματοποιητές που βασίζονται σε κανόνες έναντι Προθεματοποιητών που δημιουργούνται με μηχανική μάθηση >>

<<ΜΑΤΕΟΥΣ ΖΥΒΤΣΑΚ>>
Α.Μ. 151011

Εισηγητής: << καθηγητής ΝΙΚΗΤΑΣ Ν. ΚΑΡΑΝΙΚΟΛΑΣ>>

Nikitas Karanikolas
Digitally signed by Nikitas Karanikolas
Date: 2024.08.02 01:20:42 +03'00'

Ioannis Voyiatzis
Digitally signed by Ioannis Voyiatzis
Date: 2024.08.02 08:40:39 +03'00'

Paris Mastorokostas
Digitally signed by Paris Mastorokostas
Date: 2024.08.02 07:09:59 +03'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

«Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

Żywczak Mateusz

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά θα ήθελα να ευχαριστήσω τους γονείς μου για τη συμπαράσταση κατά τη διάρκεια των σπουδών μου.
Θα ήθελα επίσης να ευχαριστήσω το επιβλέπων καθηγητή, τον κύριο Νικήτα Καρανικόλα για την βοήθειά του, την διαθεσιμότητα του για οποιαδήποτε απορία μου και την εξαιρετική συνεργασία μαζί του για την εκπόνηση της εργασίας.

ΠΕΡΙΛΗΨΗ

Για την τεχνολογία των μηχανών αναζήτησης απαιτούνται Stemmers. Οι stemmers εμφανίζονται και παλαιότερα στην τεχνολογία «Information retrieval». Οι stemmers (προθεματοποιητές) χρησιμοποιούνται για να ομαδοποιούν ένα σύνολο λέξεων με παρόμοια σημασία σε μια μοναδική (συνήθως) κοινή ρίζα της ομάδας των λέξεων που αντιπροσωπεύουν. Αυτό διευκολύνει στην εύρεση και ανάκτηση κειμένων που μπορεί να έχουν τις αναζητούμενες λέξεις παραλλαγμένες με μικρή γραμματική διαφοροποίηση (από γένος, αριθμό, πτώση, κλπ.). Πιο πρόσφατα έχουν γίνει προσπάθειες για μηχανική δημιουργία stemmers από συλλογές κειμένων και συλλογές καταλήξεων που μπορεί να υπάρχουν σε μια γλώσσα. Στόχος αυτών των προσπάθειών είναι να μην απαιτούνται άνθρωποι που θα γράφουν τους κανόνες εύρεσης ριζών (stems) και αυτό να γίνεται (η παραγωγή κανόνων) με μηχανική μάθηση επιβλεπόμενη από ειδικούς (experts). Στόχος της παρούσας εργασίας είναι να συγκρίνουμε έναν μηχανικά παραγμένο stemmer με έναν χειροκίνητα παραγμένο stemmer (με κανόνες δημιουργημένους από ειδικούς). Στην εργασία αυτή θα χρησιμοποιήσουμε την Πολωνική γλώσσα ως πεδίο εφαρμογής της σύγκρισης που θέλουμε να κάνουμε. Δηλαδή θα χρησιμοποιήσουμε έναν stemmer builder για μηχανική και επιβλεπομένη δημιουργία stemmer με βάση λέξεις και καταλήξεις της Πολωνικής γλώσσας. Από αυτό θα παράγουμε τον *μηχανικά παραγμένο stemmer* (στο εξής *ΜΠΣ*). Στην συνέχεια θα αναζητήσουμε και θα επιλέξουμε κάποιον έτοιμο stemmer (επιθυμητό να είναι σε πηγαία γλώσσα προγραμματισμού) της ιδίας γλώσσας (της Πολωνικής) που βασίζεται σε *χειροποίητα φτιαγμένους κανόνες stemming* (*ΧΠΣ*). Στο τέλος θα συγκρίνουμε τα αποτελέσματα των δυο stemmers (ΜΠS versus ΧΠS) και θα αξιολογήσουμε κατά πόσο καλά τα καταφέρνει ο μηχανικά δημιουργημένος stemmer (ΜΠS).

ABSTRACT

Stemmers are a mandatory part of search engines. Earlier they were used as well in various Information Retrieval technologies. Stemmers are used to group a set of words with similar meanings into a single (usually) common root of the group of words they represent. This makes it easier to find and retrieve texts that may have the searched words varied with little grammatical variation (by gender, number, case, etc.). More recently, attempts have been made to mechanically generate stemmers from collections of texts and collections of word endings that may exist in a language. The goal of these efforts is to eliminate the need for humans to write the stem finding rules and to do the rule generation by machine learning supervised by experts. The goal of this paper is to compare a machine-generated stemmer with a manually generated stemmer (with rules generated by experts). In this paper we will use the Polish language as the scope of the comparison we want to make. That is, we will use a stemmer builder for mechanical and supervised stemmer creation based on Polish words and word endings. From this we will generate the mechanically generated stemmer. Then we will search for and select a ready-made stemmer (desirable to be in source code) of the same language (Polish) based on hand-made stemming rules. At the end we will compare the results of the two stemmers and evaluate how well the mechanically generated stemmer performs.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1: Stemming

ΕΙΣΑΓΩΓΗ

Η ανάκτηση πληροφορίας (IR) είναι η εύρεση υλικού (συνήθως εγγράφου) μη δομημένης φύσης (συνήθως κείμενο) το οποίο ικανοποιεί μια ανάγκη πληροφορίας μέσα από μεγάλες συλλογές (συνήθως αποθηκευμένες σε υπολογιστές). Μια εφαρμογή της ανάκτησης πληροφορίας είναι οι μηχανές αναζήτησης όπως πχ. η Google. Πως είναι δυνατόν μια τέτοια μηχανή να παράγει εκατοντάδες χιλιάδες αποτελέσματα σε διάστημα κλασμάτων δευτερολέπτου από την στιγμή που εισάγουμε την ερώτησή μας; Μια από τις τεχνικές που χρησιμοποιούνται είναι το stemming. Στο κεφάλαιο αυτό θα μελετήσουμε την έννοια του stemming (στελεχοποίηση). Τι είναι; Ποια είναι η χρήση του; Ποιες είναι οι κατηγορίες των αλγορίθμων stemming και πως υλοποιήθηκαν στον πραγματικό κόσμο; Επίσης θα μελετήσουμε την έννοια του lemmatization (λεματοποίηση) και το πως διαφέρει με το stemming.

1.1 Τι είναι το Stemming και το Lemmatization

Στην επεξεργασία της φυσικής γλώσσας κάποιες φορές, λόγο απαιτήσεων χρόνου και χώρου, θέλουμε να μειώσουμε τον όγκο του κειμένου χωρίς να χάσουμε την πληροφορία. Στην φυσική γλώσσα μια λέξη έχει πολλές διαφορετικές μορφές, πχ. ένα επίθετο έχει τρία γένη (αρσενικό, θηλυκό, ουδέτερο) και μπορεί να βρίσκεται είτε σε ενικό είτε σε πληθυντικό αριθμό (πχ. ‘ωραίος’, ‘ωραίοι’). Καταλαβαίνουμε λοιπόν ότι παρόλο που κάποιες λέξεις είναι διαφορετικές έχουν την ίδια ή τουλάχιστον παρόμοια σημασιολογική έννοια.

Οι αλγόριθμοι stemming στοχεύουν στην μείωση του όγκου του κειμένου μετατρέποντας (κόβοντας) τις λέξεις σε ρίζες λέξεων (word stems). Μια ρίζα λέξεως (stem) δεν είναι απαραίτητα η μορφολογική ρίζα βασισμένη στο λεξικό, αλλά μπορεί να είναι η ίδια ή μια μικρότερη μορφή της λέξης (πχ οι λέξεις ‘άνθρωπος’ και ‘ανθρώπινος’ μπορούν να κοπούν σε ‘ανθρωπ’).

Οι αλγόριθμοι stemming παρουσιάζουν δυο αρκετά σημαντικά προβλήματα, το over stemming και το under stemming.

Over stemming έχουμε όταν κόβεται από μια λέξη περισσότερο από ότι πρέπει. Αυτό έχει σαν αποτέλεσμα η λέξη ρίζα να χάσει εντελώς το νόημα της αρχικής λέξης ή δύο μη σχετικές μεταξύ τους λέξεις παράγουν την ίδια λέξη ρίζα. Για παράδειγμα αν πάρουμε τις λέξεις ‘καθόλου’ και ‘καθολικός’ ένας stemming αλγόριθμος μπορεί να βγάλει από αυτές την ρίζα ‘καθολ’. Αυτό είναι ένα παράδειγμα over stemming και μπορεί να λυθεί αντιστοιχίζοντας την λέξη ‘καθόλου’ σε ‘καθόλ’ και ‘καθολικός’ σε ‘καθολικ’, αυτή η αλλαγή όμως μπορεί να παράγει διαφορετικά προβλήματα στον αλγόριθμο (πχ. under stemming).

Αντίθετα με το over stemming, το under stemming έχουμε όταν μια λέξη δεν κόβεται αρκετά. Αυτό έχει σαν συνέπεια δύο λέξεις με ίδια ή παρόμοια σημασιολογική έννοια να παράγουν διαφορετικές λέξεις ρίζες. Για παράδειγμα οι λέξεις ‘θρησκεία’ και ‘θρησκευτικά’ μπορούν να παράγουν τις ρίζες ‘θρησκ’ και ‘θρησκευτ’ ενώ θα μπορούσαν να έχουν την ίδια ρίζα ‘θρησκ’.

Το lemmatization είναι η διαδικασία η οποία προσπαθεί να μετατρέψει κλιτικές μορφές λέξεων στις σωστές μορφές λεξικού τους, ή αλλιώς λήμμα (lemma). Έτσι το lemmatization εξαρτάται από την σωστή εύρεση του μέρους του λόγου της λέξης, καθώς και την σημασία της λέξης από τα συμφραζόμενα. Εάν για παράδειγμα η φράση ‘πήγα ταξίδι’ περάσει από ένα lemmatizer θα μας δώσει τα λήμματα ‘πάω’ ή ‘πηγαίνω’ και ‘ταξίδι’, διότι αυτές είναι οι μορφές λεξικού της φράσης. Αυτό το αποτέλεσμα είναι πολύ διαφορετικό με τα αποτελέσματα ενός stemmer το οποίο θα επέστρεφε δύο ρίζες πχ. ‘πηγ ταξίδ’. Παρατηρούμε ότι η χρήση των lemmatizers οδηγεί στην χρήση λιγότερων λέξεων, αφού χρησιμοποιούμε μόνο το λήμμα. Αυτό το χαρακτηριστικό είναι αρκετά ωφέλιμο για γλώσσες οι οποίες είναι κλιτές σε υψηλό βαθμό, όπως πχ. τα Πολωνικά στα οποία τα ουσιαστικά και επίθετα έχουν επτά πτώσεις.

Η δημιουργία ενός lemmatizer είναι αρκετά πιο δύσκολη από την δημιουργία ενός stemmer διότι χρειάζονται πολύ πιο βαθιές γνώσεις της γλώσσας την οποία επεξεργαζόμαστε.

1.2 Βασικές κατηγορίες των stemmers

Μπορούμε να κατηγοριοποιήσουμε τους stemming αλγόριθμους σε τρείς βασικές κατηγορίες:

- Brute force αλγόριθμους,
- Rule based αλγόριθμους, και
- Στοχαστικούς (stochastic) αλγόριθμους.

Οι brute force αλγόριθμοι υλοποιούνται με έναν απλό πίνακα (lookup table). Ο πίνακας περιέχει λέξεις και τις κλιτικές μορφές τους μαζί με την ρίζα τους, έτσι ο αλγόριθμος απλά ψάχνει την λέξη στον πίνακα και την αντικαθιστά με την ρίζα της. Οι αλγόριθμοι αυτοί είναι απλοί στην υλοποίηση και μπορούν εύκολα να χειριστούν τις εξαιρέσεις, αφού εμείς (άνθρωποι) φτιάχνουμε τον πίνακα. Παρουσιάζονται προβλήματα όμως όταν βρεθούν λέξεις οι οποίες δεν υπάρχουν στον πίνακα, αυτές θα αγνοηθούν κατά την διάρκεια του stemming. Οι brute force αλγόριθμοι έχουν μεγαλύτερη απόδοση σε μορφολογικά απλές γλώσσες (πχ. Αγγλικά) από τις γλώσσες οι οποίες είναι κλιτές σε υψηλό βαθμό (πχ. Πολωνικά). Στις γλώσσες αυτές μια λέξη μπορεί να έχει δεκάδες παραλλαγές (κλιτικές μορφές), πράγμα το οποίο αυξάνει το μέγεθος του πίνακα σε μεγάλο βαθμό.

Οι rule based αλγόριθμοι, όπως υποδηλώνει το όνομα, χρησιμοποιούν κάποιο σύνολο κανόνων για να μετατρέψουν μια λέξη στην ρίζα της. Μια από τις κατηγορίες των rule based αλγόριθμων είναι οι suffix stripping αλγόριθμοι (αλγόριθμοι αφαίρεσης επιθέματος). Οι αλγόριθμοι αυτοί διαθέτουν μια ομάδα κανόνων με τους οποίους αφαιρούν τα επιθέματα λέξεων για να παράγουν την ρίζα. Ένα παράδειγμα τέτοιου κανόνα μπορεί να είναι ¨εάν η λέξη τελειώνει σε ‘ω’ → αφαίρεσε το ‘ω’”. Μπορούν να υπάρχουν λέξεις για τις οποίες ισχύουν δύο ή περισσότεροι κανόνες, στις περιπτώσεις αυτές πρέπει να δώσουμε με κάποιο κριτήριο προτεραιότητα σε κάποιον κανόνα. Ο κανόνας “αφαίρεσε το ‘ω’” σε αρκετές λέξεις μπορεί να ισχύει μαζί με τον κανόνα “αφαίρεσέ το ‘αω’” στην περίπτωση αυτή καλή ιδέα θα ήταν να δώσουμε προτεραιότητα στον δεύτερο κανόνα, αφού αφαιρείται μεγαλύτερο κομμάτι. Σημειώνεται ότι οι ρίζες οι οποίες παράγουν αυτοί οι αλγόριθμοι συνήθως δεν είναι πραγματικές λέξεις, συνεπώς η μεθοδολογία αυτή μπορεί να μην είναι η βέλτιστη σε κάποιες εφαρμογές.

Οι suffix substitution αλγόριθμοι (αλγόριθμοι αλλαγής επιθέματος) είναι μια παραλλαγή των suffix stripping αλγόριθμων. Αντίθετα με τους suffix stripping αλγόριθμους οι suffix substitution αλγόριθμοι δεν αφαιρούν απλά το επίθεμα αλλά το αλλάζουν σε ένα άλλο. Στα ρήματα για παράδειγμα μπορεί να θέλουμε να τα γυρίσουμε στην μορφή α' ενικού, έτσι μπορούμε να φτιάξουμε κανόνα τύπου "εάν η λέξη τελειώνει σε 'ιζουμε' → μετέτρεψε το 'ιζω'". Οι ρίζες συνεπώς που παράγονται από αυτήν την κατηγορία είναι πραγματικές λέξεις, κάτι που μπορεί να είναι χρήσιμο σε ορισμένες εφαρμογές,

Συχνά οι κανόνες suffix stripping και suffix substitution χρησιμοποιούνται μαζί στην ίδια εφαρμογή. Μια συνήθης χρήση αυτών των ομάδων κανόνων σε μια λέξη είναι πρώτα να γίνει suffix substitution και ύστερα να εφαρμοστεί suffix stripping. Αυτές οι δύο ενέργειες μπορούν να επαναληφθούν μέχρι να μας οδηγήσουν σε αποτέλεσμα το οποίο θέλουμε.

Επιπλέον υπάρχουν και οι affix removal αλγόριθμοι (αλγόριθμοι αφαίρεσης προσφύματος). Το affix (πρόσφυμα) είναι το prefix (πρόθεμα) και το suffix (επίθεμα). Αλγόριθμοι αυτής της κατηγορίας αφαιρούν και το πρόθεμα και το επίθεμα στοχεύοντας να παράγουν την ρίζα της λέξης από την οποία παράγεται η αρχική λέξη.

Για να δημιουργήσουμε rule based αλγόριθμους χρειαζόμαστε αρκετά καλή γνώση της γλώσσας πάνω στην οποία δουλεύουμε, διότι αλλιώς θα ήταν δύσκολο να δημιουργήσουμε τους κανόνες.

Τέλος οι στοχαστικοί αλγόριθμοι παράγονται μέσω μοντέλων μηχανικής μάθησης (machine learning). Τον στοχαστικό αλγόριθμο τον εκπαιδεύουμε δίνοντάς του λέξεις μαζί με τις σωστές ρίζες τους. Αυτό προσπαθεί με βάση τις εισόδους (λέξεις) και τις εξόδους (σωστές ρίζες) να δημιουργήσει ένα στοχαστικό μοντέλο το οποίο με την είσοδο νέων λέξεων θα παράγει τις σωστές ρίζες τους. Μπορούμε να πούμε υπό μια έννοια ότι ο στοχαστικός αλγόριθμος μαθαίνει τους κανόνες με τους οποίους θα μετατρέψει τις λέξεις στις ρίζες τους.

1.3 Γνωστοί stemmers

Επειδή η έρευνα στους stemmers άρχισε από την αγγλική γλώσσα, οι πιο γνωστοί stemming αλγόριθμοι βασίζονται σε αυτήν. Οι πιο γνωστοί αλγόριθμοι stemming είναι οι:

- Lovins stemmer
- Porter stemmer
- Lancaster stemmer

Ο πρώτος stemmer προτάθηκε από την Julie Beth Lovins το 1968 για την αγγλική γλώσσα. Το paper για τα χρόνια εκείνα ήταν αρκετά πρωτοπόρο, καθώς η ανάκτηση πληροφορίας (IR) τότε χρησιμοποιούταν μόνο σε τεχνικά κείμενα και σε ερευνητικές εργασίες, έτσι το stemming θεωρούταν μη πρακτικό και σπατάλη πόρων. Η μελέτη της Lovins επηρέασε σημαντικά τις μελλοντικές εργασίες πάνω στον τομέα του stemming.

Ο Lovins stemmer αποτελείται από 294 επιθέματα, 29 συνθήκες και 35 κανόνες μεταμόρφωσης. Για κάθε επίθεμα αντιστοιχεί και μια συνθήκη. Οι συνθήκες αποτελούνται από τις ακόλουθες υπό συνθήκες, κάποιον συνδυασμό τους (1 με 2 και 1 με 3), ή καμία απολύτως συνθήκη:

1. Ελάχιστο μήκος της ρίζας (σε χαρακτήρες)
2. Αφαίρεσε το επίθεμα μόνο όταν το επίθεμα έπεται συγκεκριμένου γράμματος
3. Άφησε το επίθεμα μόνο όταν το επίθεμα έπεται συγκεκριμένου γράμματος

Οι κανόνες μεταμόρφωσης αφαιρούν κάποια διπλά γράμματα και χειρίζονται κάποιες ιδιαιτερότητες της αγγλικής γλώσσας όπως πχ. ανώμαλες μορφές πληθυντικού.

Ο αλγόριθμος ψάχνει το μεγαλύτερο επίθεμα από την λίστα, το οποίο ικανοποιεί και την συνθήκη. Άσχετα με το αν βρέθηκε το επίθεμα ή όχι, εφαρμόζεται πρώτα ο κανόνας αφαίρεσης διπλών γραμμάτων και ύστερα οι υπόλοιποι 34 κανόνες.

Ο stemmer της Lovins φτιάχτηκε κυρίως για τα επιστημονικά έγγραφα. Δίνει έμφαση σε επιστημονικό λεξιλόγιο λόγο ύπαρξης επιθεμάτων και κανόνων τα οποία θα εμφανιζόταν μόνο σε επιστημονικά έγγραφα (πχ. το επίθεμα -oidal). Επίσης υπάρχει έλλειψη επιθεμάτων του πληθυντικού αριθμού, επειδή τα θέματα των επιστημονικών εγγράφων είναι γραμμένα κυρίως στον ενικό αριθμό και συνεπώς και οι όροι

αναζήτησης. Τέλος το πολύ μικρό μέγεθος της μικρότερης ρίζας, 2, δεν προκαλεί προβλήματα στο επιστημονικό λεξιλόγιο το οποίο χρησιμοποιεί αρκετά μεγάλες λέξεις.

Το 1980 προτάθηκε μια παραλλαγή του Lovins stemmer, από τον Martin Porter. Ο Porter λόγο της απλής υλοποίησης του stemmer του, έδωσε έμπνευση για πολλούς νεότερους αλγόριθμους για διάφορες γλώσσες πέρα της Αγγλικής.

Ο Porter stemmer είναι ένας απλός rule-based suffix stripping και suffix substitution αλγόριθμος με περίπου 60 επιθέματα. Η κύρια λογική του αλγόριθμου είναι να εφαρμοστούν οι κανόνες σε πέντε στάδια. Το πρώτο στάδιο χειρίζεται τα επιθέματα τα οποία προκύπτουν από την κλίση όπως ο πληθυντικός ή το παρελθόν. Τα τρία επόμενα (2, 3, 4) στάδια χειρίζονται επιθέματα από τα οποία προκύπτουν παράγωγες λέξεις πχ. ‘thoughtful’ γίνεται ‘thought’. Το τελευταίο, πέμπτο στάδιο είναι ένα στάδιο κωδικοποίησης το οποίο αφαιρεί υπό ορισμένες περιπτώσεις το ‘e’ και κάποια διπλά γράμματα.

Ο αλγόριθμος αυτός αντί να χρησιμοποιεί στους κανόνες του τον ελάχιστο αριθμό χαρακτήρων της ρίζας που προκύπτει, χρησιμοποιεί τον ελάχιστο συνδυασμό φωνήεν-σύμφωνο (ή measure (m)) της ρίζας που προκύπτει. Μπορούμε να συγκρίνουμε το measure με μια συλλαβή. Τέλος αν υπάρχει σύμπτωση μεταξύ δύο κανόνων όπως για παράδειγμα “SSES -> SS” και “S -> ε” έχει προτεραιότητα ο κανόνας με το μεγαλύτερο πρόθεμα (όπου ‘ε’ είναι το κενό).

Αν και είναι απλός ο Porter stemmer, είναι αρκετά αποτελεσματικός και γρήγορος. Παρά την πάροδο των χρόνων ο αλγόριθμος αυτός χρησιμοποιείται ακόμα ως βάση για πολλούς άλλους stemmers για γλώσσες πολύ πιο περίπλοκες από την Αγγλική.

Τέλος ο Lancaster stemmer είναι όμοιος με αυτόν του Porter με την έννοια ότι είναι επίσης rule-based suffix stripping και suffix substitution αλγόριθμος. Χρησιμοποιεί πάνω από 100 κανόνες για την παραγωγή ρίζων. Ο αλγόριθμος αυτός έχει την αρνητική τάση να κάνει over stemming σε πολλές λέξεις. Για την καταπολέμηση αυτού χρησιμοποιεί επιπλέον δυο συνθήκες:

- Αν η λέξη αρχίζει από φωνήεν, τότε η ρίζα πρέπει να περιέχει τουλάχιστον δυο χαρακτήρες (πχ. το “eat” δεν μπορεί να γίνει “e”)

- Αν η λέξη αρχίζει από σύμφωνο, τότε η ρίζα πρέπει να έχει τουλάχιστον τρεις χαρακτήρες εκ των οποίον τουλάχιστον ένα είναι φωνήεν (πχ. το “strategy” δεν μπορεί να γίνει “str”)

Ο αλγόριθμος του Lancaster σε σύγκριση με τον αλγόριθμο του Porter παρέχει μεγαλύτερο επίπεδο μείωσης των διαστάσεων των κειμένων. Αυτό το καταφέρνει επειδή είναι αρκετά επιθετικός και οι ρίζες που παράγονται από αυτόν είναι μικρές. Λόγω αυτού όμως έχει τάση για over stemming και συνεπώς τα αποτελέσματά του είναι λιγότερο ακριβής από αυτά του Porter.

Συνοψίζοντας, οι stemming αλγόριθμοι στοχεύουν στην μείωση κειμένων με την μετατροπή λέξεων σε λέξεις ρίζες, ενώ οι lemmatizers στοχεύουν στην μείωση κειμένων με την μετατροπή λέξεων στις βασικές μορφές λεξικού τους (λήμματα). Οι stemming αλγόριθμοι χωρίζονται στους brute force, rule based και στοχαστικούς αλγορίθμους. Οι πιο γνωστοί stemmers Lovins, Porter και Lancaster είναι όλοι rule based, με τον Porter να βρίσκει την περισσότερη χρήση λόγο της απλότητας και ακρίβειάς του.

ΚΕΦΑΛΑΙΟ 2: Πολωνικά

ΕΙΣΑΓΩΓΗ

Η επίσημη γλώσσα της Πολωνίας, τα πολωνικά, ανήκει στην δυτική-σλαβική ομάδα των ινδοευρωπαϊκών γλωσσών. Την μιλάνε περίπου 45 εκατομμύρια άτομα, καθώς ο πληθυσμός της Πολωνίας βρίσκεται στα 38 εκατομμύρια άτομα και μετριέται ότι 12 – 15 εκατομμύρια πολωνοί μετανάστες υπάρχουν σε διάφορα μέρη του κόσμου, όπως ΗΠΑ, Γερμανία. Αυτό την κάνει να είναι δεύτερη (ή τρίτη, με δεύτερη τα ουκρανικά ανάλογος τις πηγές) πιο κοινά ομιλούμενη σλαβική γλώσσα, με πρώτη να είναι τα ρωσικά. Η τυπική πολωνική γλώσσα σήμερα βασίζεται σε διαλέκτους τριών περιοχών, των Wielkopolska, Małopolska και Mazowsze. Τοπικά, κυρίως σε χωριά, χρησιμοποιούνται σε ομιλούμενη μορφή μέχρι και σήμερα διάλεκτοι όπως, małopolski, mazowiecki, wielkopolski, śląski και kaszubski. Η διάλεκτος kaszubski θεωρείται διάλεκτος της πολωνικής γλώσσας αλλά και ξεχωριστή γλώσσα. Στο κεφάλαιο αυτό θα δούμε την γραμματική της πολωνικής γλώσσας και κάποιους κανόνες της, ώστε να καταλάβουμε γιατί η δημιουργία stemmers για τα πολωνικά είναι δύσκολο έργο.

2.1 Πολωνική αλφάβητος

Η πολωνική αλφάβητος αποτελείται από 32 γράμματα, από τα οποία τα 23 προέρχονται από την λατινική αλφάβητο. Τα υπόλοιπα 9 γράμματα είναι διακριτικοί χαρακτήρες, δηλαδή σε κάποιους χαρακτήρες που υπάρχουν έχει προστεθεί τελεία, τόνος (γραμμή) ή ουρίτσα με τον σκοπό να αναπαρασταθούν νέοι ήχοι (φθόγγοι).

Aa	Ąą	Bb	Cc	Ćć	Dd	Ee	Ęę
Ff	Gg	Hh	Ii	Jj	Kk	Ll	Łł
Mm	Nn	Ńń	Oo	Óó	Pp	Rr	Ss
Śś	Tt	Uu	Ww	Yy	Zz	Źź	Żż

Επίσης στην πολωνική γλώσσα υπάρχουν 7 δίψηφα γράμματα. Αυτά τα γράμματα έχουν τον σκοπό να αποτυπώσουν έναν ήχο (φθόγγο) στον οποίο δεν αντιστοιχεί σε κανένα γράμμα της αλφαβήτου.

CZ	RZ	SZ	DZ	DŹ	DŻ	CH

Τέλος στην πολωνική γλώσσα χρησιμοποιούνται τα ακόλουθα 3 γράμματα, τα οποία υπάρχουν σε ξένες λέξεις οι οποίες δεν έχουν αλλάξει την ορθογραφία τους.

Qq	Vv	Xx

Παραδείγματα ξένων λέξεων που δεν έχουν αλλάξει την ορθογραφία τους: quad, quiz, quasi, VAT, VIP, veto, matrix, xero.

2.2 Προφορά

Ο φθόγγος είναι η μικρότερη, αδιαίρετη ηχητική μονάδα στην οποία μπορεί να αναλυθεί μια λέξη. Το γράμμα σε ένα αλφαβητικό σύστημα γραφής είναι η γραφική αναπαράσταση ενός φθόγγου. Συνεπώς τους φθόγγους τους ακούμε και τους προφέρουμε ενώ τα γράμματα τα γράφουμε. Τις δύο έννοιες αυτές συχνά τις θεωρούμε συνώνυμες ή ίδιες, κάτι που είναι λάθος.

Συχνά ο αριθμός των φθόγγων σε μια λέξη αντιστοιχεί στον αριθμό των γραμμάτων, για παράδειγμα:

- D-O-M (σπίτι), 3 γράμματα, 3 φθόγγοι
- B-A-N-A-N (μπανάνα), 5 γράμματα, 5 φθόγγοι
- K-O-R-U-P-C-J-A (διαφθορά, δωροδοκία), 8 γράμματα, 8 φθόγγοι

Αυτό όμως δεν ισχύει πάντα, και ο αριθμός των φθόγγων και γραμμάτων μπορεί να διαφέρει, για παράδειγμα:

- K-O-SZ (καλάθι), 4 γράμματα, 3 φθόγγοι
- L-U-DZI-E (άνθρωποι), 6 γράμματα, 4 φθόγγοι
- CH-RZ-Ą-SZ-CZ (σκαθάρι), 9 γράμματα, 5 φθόγγοι

Οι φθόγγοι διαχωρίζονται σε 4 κατηγορίες ανάλογα με την διάταξη των φωνητικών οργάνων (πχ. χείλη, δόντια, ουρανίσκος, γλώσσας, κλπ.):

- σύμφωνα και φωνήεντα
- άηχους και ηχηρούς
- λαρυγγικούς και ρινικούς
- μαλακά και σκληρά (μόνο για σύμφωνα)

Στην πολωνική γλώσσα υπάρχουν 8 φωνήεντα (φθόγγοι): a, ą, e, ę, i, y, u - ó. Αξίζει να αναφέρουμε ότι ο φθόγγος y είναι γνωστός ως igrek. Τα u και ó επειδή προφέρονται το ίδιο, τα μετράμε σαν έναν φθόγγο άρα και ένα φωνήεν. Κατά την διάρκεια της προφοράς των φωνηέντων, ακούγεται μόνο ένας φθόγγος, ο οποίος μιλιέται με ελαφρά τεντωμένο τρόπο, πχ. a-aaaaa, i-iiiiiii. Επίσης άλλο ένα χαρακτηριστικό των φωνηέντων είναι να μπορούν μόνα τους να σχηματίζουν μια συλλαβή (πχ. όνομα O-la).

Ως σύμφωνο χαρακτηρίζουμε όποιο γράμμα δεν είναι φωνήεν, δηλαδή: b, c, ć, d, f, g, h, j, k, l, ł, m, n, ń, p, r, s, ś, t, w, z, ź, ż. Κατά την διάρκεια της προφοράς των συμφώνων δεν ακούμε μόνο έναν φθόγγο, αλλά παραπάνω, πχ. τον φθόγγο “t” μπορούμε να τον προφέρουμε ως “t-e” ή „t-y”. Επίσης τα σύμφωνα δεν μπορούν να φτιάξουν από μόνα τους συλλαβές και χρειάζονται να συνοδεύονται από κάποιο φωνήεν.

Τα δίψηφα γράμματα CZ, RZ, SZ, DZ, DŹ, DŻ, CH όπως αναφέραμε και παραπάνω χρησιμοποιούνται για την αναπαράσταση ενός φθόγγου.

Το “ Ii ” σε συνδυασμό με ορισμένα σύμφωνα (C, N, S, Z) μαλακώνει την προφορά τους. Αυτοί οι συνδυασμοί (CI, NI, SI, ZI) έχουν ακριβώς την ίδια προφορά με το αν θα βάζαμε τόνο πάνω από τα αντίστοιχα σύμφωνα (Ć, Ń, Ś, Ź). Δηλαδή στα πολωνικά ο τόνος μαλακώνει, και όχι σκληραίνει τα σήμφωνα.

Είναι σημαντικό να αναφέρουμε ότι υπάρχουν και ζευγάρια γραμμάτων τα οποία αποτυπώνουν τον ίδιο ακριβώς φθόγγο. Αυτά τα ζευγάρια είναι:

- U και Ó, τα οποία προφέρονται σαν “ου”
- CH και H, τα οποία προφέρονται σαν “χ”
- RZ και Ż, τα οποία προφέρονται σαν “j” του “je” στο γαλλικό “je m’appellee”

Τέλος ο τονισμός στην Πολωνική γλώσσα πέφτει σχεδόν πάντα στην παραλήγουσα, δηλαδή στην προτελευταία συλλαβή της λέξης. Υπάρχουν φυσικά εξαιρέσεις σε αυτόν τον κανόνα και κάποιες λέξεις τονίζονται στην προπαραλήγουσα, την τρίτη από το τέλος συλλαβή. Αυτό συμβαίνει σε:

- Ορισμένες λέξεις ο οποίες προέρχονται από ξένες λέξεις
- Αριθμούς οι οποίοι προέρχονται από άλλους πολυσύλλαβους αριθμούς
- Ουσιαστικά τα οποία τονίζονται παραδοσιακά στην προπαραλήγουσα
- Ορισμένες μορφές ρημάτων

Επιπλέων υπάρχουν και μορφές ρημάτων στα οποία ο τόνος πέφτει στην τέταρτη από το τέλος συλλαβή.

2.3 Ουσιαστικά

Τα ουσιαστικά αναπαριστούν αντικείμενα, ανθρώπους, ζώα, τόπους, καιρικά φαινόμενα και αφηρημένες έννοιες (πχ. αγάπη – miłość). Στην πολωνική γραμματική τα ουσιαστικά πάντα απαντάνε στην ερώτηση “Kto? Co?” δηλαδή “Ποιος? Ποια? Ποιο? Τι?”.

Ομοίως με την ελληνική γλώσσα τα ουσιαστικά στην πολωνική γλώσσα κλίνονται ως προς αριθμό και πτώση. Οι αριθμοί είναι δυο όπως και στα ελληνικά, ενικός και πληθυντικός. Η διαφορά βρίσκεται στον αριθμό των πτώσεων μεταξύ των δυο γλωσσών. Στα ελληνικά έχουμε τέσσερεις πτώσεις την ονομαστική, την γενική, την αιτιατική και την κλιτική. Ενώ στα πολωνικά έχουμε επτά πτώσεις mianownik (ονομαστική), dopełniacz (γενική), celownik (δοτική), biernik (αιτιατική), narzędnik (οργανική), miejscownik (τοπική), wołacz (κλιτική).

Για να βρούμε τα ουσιαστικά σε ποια πτώση είναι, πρέπει να βρούμε σε ποια ερώτηση απαντάνε. Και πιο συγκεκριμένα:

- Mianownik (ονομαστική) απαντάει στο Kto? Co? (jest) – Ποιος/α/ο? Τι? (είναι αυτός). Η πτώση αυτή είναι η βασική μορφή όλων των ουσιαστικών. Συνήθως σε μια πρόταση έχει τον ρόλο του υποκειμένου.

- Dopełniacz (γενική) απαντάει στο Kogo? Czego? (nie ma) - Ποιος/α/ο? Τι? (λείπει). Την πτώση αυτή την χρησιμοποιούμε σε αρνήσεις και όταν θέλουμε να δείξουμε σε ποιόν κάτι ανήκει
- Celownik (δοτική) απαντάει στο Komu? Czemu? (przyglądam się) - Ποιον/α/ο? Τι? (παρατηρώ). Η δοτική πτώση χρησιμοποιείται συχνά δίπλα από ρήματα.
- Biernik (αιτιατική) απαντάει στο Kogo? Co? (widzę) - Ποιον/α/ο? Τι? (βλέπω). Όμοια με τα ελληνικά η αιτιατική πτώση χρησιμοποιείται όταν δηλώνουμε αντικείμενο κάποιου ρήματος
- Narzędnik (οργανική) απαντάει στο Z kim? Z czym? (idę) – Με ποιον/α/ο? Με τι? (περπατάω). Η πτώση αυτή προσδιορίζει ένα μέσο μεταφοράς, ένα εργαλείο, ποιος είμαστε ή τι μας ενδιαφέρει
- Miejscownik (τοπική) απαντάει στο O kim? O czym? (myślę) - Ποιον/α/ο? Τι? (σκέφτομαι). Η πτώση αυτή χρησιμοποιείται όταν θέλουμε να περιγράψουμε που (τον τόπο) όπου κάτι βρίσκεται.
- Wołacz (κλιτική) απαντάει στο ο! – ω!. Η μόνη πτώση που δεν απαντάει σε ερώτημα. Χρησιμοποιείται όταν καλούμε κάποιον.

Παρόμοια με τα ελληνικά, στα πολωνικά τα ουσιαστικά κατηγοριοποιούνται σε γένη και ταυτόχρονα δεν κλίνονται ως προς αυτά. Στον ενικό αριθμό στα πολωνικά έχουμε αρσενικό, θηλυκό και ουδέτερο γένος (męski, żeński, nijaki), ενώ στον πληθυντικό αριθμό τα ουσιαστικά χωρίζονται σε αρσενικό και μη αρσενικό γένος (męskooosobowy, niemęskooosobowy). Στην ομάδα του πληθυντικού αρσενικού γένους εντάσσονται όλα τα ουσιαστικά τα οποία δηλώνουν ένα αρσενικό πρόσωπο ή μια ομάδα η οποία περιλαμβάνει ένα αρσενικό μέλος. Ενώ στην ομάδα του πληθυντικού μη αρσενικού γένους εντάσσονται όλα τα υπόλοιπα ουσιαστικά, δηλαδή αυτά που είναι σε θηλυκό και ουδέτερο γένος αλλά και όλα τα υπόλοιπα αρσενικά (ενικού) ουσιαστικά που δεν εντάσσονται στην ομάδα του αρσενικού (πληθυντικού). Άρα δεν δηλώνουν ούτε αρσενικό πρόσωπο, ούτε μια ομάδα η οποία περιλαμβάνει ένα αρσενικό μέλος και συνήθως η ομάδα αυτή περιέχει αντικείμενα αρσενικού (ενικού) γένους. Παράδειγμα:
Pisarz (ποιητής) – Ονομαστική ενικός αρσενικό -> Pisarze – Ονομαστική πληθυντικό αρσενικό

Pisarka (ποιήτρια) - Ονομαστική ενικός θηλυκό -> Pisarki – Ονομαστική πληθυντικό μη αρσενικό
Obraz (εικόνα) - Ονομαστική ενικός αρσενικό -> Obrazy – Ονομαστική πληθυντικό μη αρσενικό

Πτώση	Ερώτημα	Ενικός	Πληθυντικός
Mianownik (ονομαστική)	Kto? Co?	pies	psy
Dopełniacz (γενική)	Kogo? Czego?	psa	psów
Celownik (δοτική)	Komu? Czemu?	psu	psom
Biernik (αιτιατική)	Kogo? Co?	psa	psy
Narzędnik (οργανική)	Z kim? Z czym?	psem	psami
Miejscownik (τοπική)	O kim? O czym?	psie	psach
Wołacz (κλιτική)	o!	psie	psy

Πίνακας 2.1: Παράδειγμα κλίσης της λέξης pies (σκυλί) στα πολωνικά

Πτώση	Ενικός	Πληθυντικός
Ονομαστική	το σκυλί	τα σκυλιά
Γενική	του σκυλιού	των σκυλιών
Αιτιατική	το σκυλί	τα σκυλιά
Κλητική	σκυλί	σκυλιά

Πίνακας 2.2: Παράδειγμα κλίσης της λέξης pies (σκυλί) στα ελληνικά

Στα πολωνικά υπάρχουν ουσιαστικά που είναι εξαίρεση και δεν κλίνονται ως προς τον αριθμό. Υπάρχουν ουσιαστικά που έχουν μόνο τον ενικό αριθμό, αυτά για παράδειγμα

είναι κάποιες αφηρημένες έννοιες (πχ. αγάπη – miłość) ή κάποια αμέτρητα αντικείμενα (πχ. ryż - ρύζι). Αντίστοιχα υπάρχουν και ουσιαστικά τα οποία δεν έχουν ενικό αριθμό και αυτά είναι για παράδειγμα μεγάλες γεωγραφικές περιοχές (πχ. Bieszczady) ή κάποια αμέτρητα αντικείμενα (πχ. fusy – αυτό που μένει μετά από έναν καφέ).

2.4 Επίθετα

Τα επίθετα είναι ένα μέρος του λόγου τα οποία χρησιμοποιούμε για να περιγράψουμε τα διάφορα χαρακτηρίστηκα και τις διάφορες ιδιότητες των ουσιαστικών. Στην πολωνική γραμματική τα επίθετα πάντα απαντάνε στις ερωτήσεις Jaki? Jaka? Jakie? Który? Która? Które? Czyj? Czyja? Czyje? (Πως είναι αυτός? Πως είναι αυτή? Πως είναι αυτό?, Ποιος? Ποια? Ποιο? Ποιανού? Ποιανής? Ποιανού?).

Τα επίθετα κλίνονται ως προς την πτώση, τον αριθμό και το γένος. Η πτώση, ο αριθμός και το γένος ενός επιθέτου, είναι ακριβώς το ίδιο με αυτά του ουσιαστικού το οποίο χαρακτηρίζεται από αυτό. Αυτό συμβαίνει και στην ελληνική γλώσσα.

Συνεπώς στα πολωνικά ο αριθμός πτώσεων των επιθέτων είναι ακριβώς ίδιος με αυτών των ουσιαστικών. Άρα τα επίθετα έχουν επτά πτώσεις τις mianownik (ονομαστική), dopełniacz (γενική), celownik (δοτική), biernik (αιτιατική), narzędnik (οργανική), miejscownik (τοπική), wołacz (κλιτική) οι οποίες απαντάνε στις ακριβώς ίδιες ερωτήσεις με τα ουσιαστικά.

Αντιθέτως με τα ουσιαστικά τα επίθετα κλίνονται ως προς το γένος. Όμως η κατηγοριοποίηση είναι ακριβώς η ίδια, δηλαδή έχουμε αρσενικό, θηλυκό και ουδέτερο γένος στον ενικό αριθμό και αρσενικό και μη αρσενικό στον πληθυντικό αριθμό.

Αντίστοιχα, στα ελληνικά τα επίθετα κλίνονται ως προς πτώση (Ονομαστική, Γενική, Αιτιατική, Κλιτική), αριθμό (ενικό, πληθυντικό), γένος (αρσενικό, θηλυκό, ουδέτερο σε ενικό και πληθυντικό).

Πτώση	Ερώτημα	Ενικός Αρσενικού	Ενικός Θηλυκού	Ενικός Ουδέτερου
Mianownik (ονομαστική)	Kto? Co?	duży	duża	duże
Dopełniacz (γενική)	Kogo? Czego?	dużego	dużej	dużego
Celownik (δοτική)	Komu? Czemu?	dużemu	dużej	dużemu
Biernik (αιτιατική)	Kogo? Co?	dużego	dużą	duże
Narzędnik (οργανική)	Z kim? Z czym?	dużym	dużą	dużym
Miejscownik (τοπική)	O kim? O czym?	dużym	dużej	dużym
Wołacz (κλιτική)	o!	duży	duża	duże

Πίνακας 2.3: Παράδειγμα κλίσης της λέξης duży (μεγάλος) στα πολωνικά (ενικός αριθμός)

Πτώση	Ερώτημα	Πληθυντικός Αρσενικού	Πληθυντικός Μη Αρσενικού
Mianownik (ονομαστική)	Kto? Co?	duzi	duże
Dopełniacz (γενική)	Kogo? Czego?	dużych	dużych
Celownik (δοτική)	Komu? Czemu?	dużym	dużym
Biernik (αιτιατική)	Kogo? Co?	dużych	duże
Narzędnik (οργανική)	Z kim? Z czym?	dużymi	dużymi
Miejscownik (τοπική)	O kim? O czym?	dużych	dużych

Wołacz (κλιτική)	o!	duzi	duże

Πίνακας 2.4: Παράδειγμα κλίσης της λέξης duży (μεγάλος) στα πολωνικά (πληθ. αριθμός)

Πτώση	Ενικός Αρσενικού	Ενικός Θηλυκού	Ενικός Ουδέτερου
Ονομαστική	ο μεγάλος	ο μεγάλη	το μεγάλο
Γενική	του μεγάλου	της μεγάλης	του μεγάλου
Αιτιατική	τον μεγάλο	την μεγάλη	το μεγάλο
Κλητική	μεγάλε	μεγάλη	μεγάλο

Πίνακας 2.5: Παράδειγμα κλίσης της λέξης duży (μεγάλος) στα ελληνικά (ενικός αριθμός)

Πτώση	Πληθυντικός Αρσενικού	Πληθυντικός Θηλυκού	Πληθυντικός Ουδέτερου
Ονομαστική	οι μεγάλοι	οι μεγάλες	τα μεγάλα
Γενική	των μεγάλων	των μεγάλων	των μεγάλων
Αιτιατική	τους μεγάλους	τις μεγάλες	τα μεγάλα
Κλητική	μεγάλοι	μεγάλες	μεγάλα

Πίνακας 2.6: Παράδειγμα κλίσης της λέξης duży (μεγάλος) στα ελληνικά (πληθ. αριθμός)

Τα επίθετα τα οποία δεν δηλώνουν κάποιο αμετάβλητο χαρακτηριστικό (πχ. μεταλλικό), κλιμακώνονται ως προς την ένταση ενός συγκεκριμένου χαρακτηριστικού (πχ στα ελληνικά ψηλός -> ψηλότερος -> υψηλότατος). Οι τρεις τύποι κλιμάκωσης είναι απλή, ακανόνιστη, περιγραφική (Stopniowanie Proste, Nieregularne, Opisowe) και οι τρεις βαθμοί κλιμάκωσης είναι ίσος, υψηλότερος, υψηλότατος (Stopien Równy, Wyższy, Najwższy) (για ευκολία θα ονομάσουμε τους βαθμούς πρώτο, δεύτερο και τρίτο). Πιο συγκεκριμένα για τους τρεις τύπους κλιμάκωσης:

- Στην απλή κλιμάκωση για να μετατρέψουμε μια λέξη από τον πρώτο βαθμό στον δεύτερο αρκεί να αλλάξουμε την κατάληξή της (συνήθως σε “-szy”, αντίστοιχο του ελληνικού “μεγάλος” – “μεγαλύτερος”) και για να πάμε στον τρίτο βαθμό πρέπει να προσθέσουμε στην αρχή του δεύτερου βαθμού naj- το οποίο σημαίνει “το περισσότερο”.

- Στην ακανόνιστη κλιμάκωση η μορφή του επιθέτου στον δεύτερο βαθμό αλλάζει από αυτήν στον πρώτο βαθμό (γίνεται διαφορετική λέξη, αντίστοιχο του ελληνικού “κακός” – “χειρότερος”). Για να μετατρέψουμε το επίθετο στον τρίτο βαθμό αρκεί να προσθέσουμε το naj- στον δεύτερο βαθμό.
- Στην περιγραφική κλιμάκωση προσθέτουμε την λέξη “bardziej” (πιο πολύ) για να πάμε από τον πρώτο στον δεύτερο βαθμό και προσθέτουμε “najbardziej” (το περισσότερο) στο επίθετο πρώτου βαθμού για να φτιάξουμε τον τρίτο βαθμό.

Κλιμάκωση	Ίση	Υψηλότερη	Υψηλότατη
Απλή	silny	silniejsy	najsilniejszy
Ακανόνιστη	duży	większy	największy
Περιγραφική	grecki	bardziej grecki	najbardziej grecki

Πίνακας 2.7: Παράδειγμα κλιμάκωσης της λέξης silny (δυνατός), duży (μεγάλος), grecki (ελληνικός)

Τέλος όταν προσθέσουμε την λέξη “nie” (όχι) στην αρχή επιθέτου του πρώτου βαθμού, παράγεται ένα νέο επίθετο με έννοια ακριβώς αντίθετη της αρχικής. Στον δεύτερο και τρίτο βαθμό η λέξη “nie” γράφεται χωριστά και έχουμε δυο λέξεις. Για παράδειγμα η λέξη πρώτου βαθμού “duży” (μεγάλος) με το “nie” γίνεται “nieduży” (όχι μεγάλος, συνώνυμο με το μικρός), ενώ η μορφή του “duży” στον δεύτερο βαθμό “większy” μαζί με το “nie” γράφεται “nie większy” (όχι μεγαλύτερος).

2.5 Ρήματα

Ως ρήμα χαρακτηρίζουμε το κλιτό μέρος του λόγου το οποίο δηλώνει μια ενέργεια η μια κατάσταση. Στην πολωνική γραμματική τα ρήματα πάντα απαντάνε στις ερωτήσεις Co robi? Co się z nim dzieje? (Τι κάνει? Τι γίνεται με αυτόν?).

Αρχικά πρέπει να αναφέρουμε ότι τα ρήματα μπορούν να πάρουν προσωπικές ή απρόσωπες μορφές. Τα ρήματα στην προσωπική μορφή περιέχουν την πληροφορία για το πρόσωπο που εκτελεί την ενέργεια (πχ. gra (on/ona/ono) – παίζει (αυτός/αυτή/αυτό)). Αντίθετα οι απρόσωπες μορφές δεν περιέχουν την πληροφορία για το πρόσωπο που εκτελεί την ενέργεια. Σε αυτά ανήκουν:

- Απαρέμφατα, τα οποία τελειώνουν σε c, ć, ś, ź (πχ. grać)

- Ρήματα που τελειώνουν σε no, to (πχ. grano)
- Μετοχές οι οποίες τελειώνουν σε ąc, wszy, łszy, na, ny, ne, ty, ta, te, ony, ona, one (πχ. grając)
- Συγκεκριμένα ρήματα στον ενεστώτα με την αντωνυμία “się” (πχ. mówi się)

Στα ελληνικά τα ρήματα κλίνονται ως προς το πρόσωπο, τον αριθμό, τον χρόνο, την φωνή, την έγκλιση και την όψη (ποιόν ενέργειας - μη συνοπτικός, συνοπτικός, συντελεσμένος). Στην πολωνική γλώσσα τα ρήματα επιπλέον κλίνονται και ως προς το γένος.

Τα πολωνικά ρήματα έχουν δύο αριθμούς (ενικό και πληθυντικό) και ο κάθε αριθμός έχει τρία πρόσωπα (πρώτο, δεύτερο και τρίτο). Ο αριθμός δείχνει εάν το υποκείμενο του ρήματος είναι ένα ή περισσότερα άτομα ή πράγματα. Ενώ το πρόσωπο δείχνει το ποιος κάνει την πράξη που δηλώνει το ρήμα.

Liczba pojedyńcza (Ενικός αριθμός)

- Πρώτο πρόσωπο: Ja (Εγώ)
- Δεύτερο πρόσωπο: Ty (Εσύ)
- Τρίτο πρόσωπο: On, Ona, Ono (Αυτός, Αυτή, Αυτό)

Liczba Mnoga (Πληθυντικός αριθμός)

- Πρώτο πρόσωπο: My (Εμείς)
- Δεύτερο πρόσωπο: Wy (Εσείς)
- Τρίτο πρόσωπο: Oni, One (Αυτοί, Αυτές + Αυτά)

Τα ρήματα επίσης κλίνονται ως προς τρεις χρόνους τον teraźniejszy (ενεστώτας), przeszły (παρατατικός), przyszły (μέλλοντας).

- Teraźniejszy (ενεστώτας) εκφράζει την πράξη που γίνεται τώρα (πχ. idę – πάω)
- Przeszły (παρατατικός) εκφράζει την πράξη που έγινε στο παρελθόν (πχ. poszedłem – πήγα αρσενικό γένος, poszłam – πήγα θηλυκό γένος)
- Przyszły (μέλλοντας) εκφράζει την πράξη που θα γίνει στο μέλλον (πχ. pójdę – θα πάω). Ο μέλλοντας χωρίζεται σε δυο όψεις. Όταν μια ενέργεια θα ολοκληρωθεί στο μέλλον και μπορούμε να προβλέψουμε το αποτέλεσμα της ή ξέρουμε ότι θα τελειώσει τότε έχουμε τον przyszły prosty (απλός μέλλοντας).

Αντιθέτως εάν δεν μπορούμε να προβλέψουμε το αποτέλεσμα μιας ενέργειας στο μέλλον ή δεν ξέρουμε αν θα τελειώσει τότε έχουμε przyszły złożony (σύνθετος μέλλοντας). Για παράδειγμα η λέξη “pójdę” (θα πάω) είναι στον απλό μέλλοντα διότι ξέρουμε η ενέργεια κάποτε θα τελειώσει και το υποκείμενο του ρήματος θα φτάσει κάπου. Ενώ η λέξη “grać” (παίζω) στον μέλλοντα γίνεται “będę grał” (θα παίζω), το οποίο δεν υποδηλώνει πότε θα τελειώσει, ούτε αν θα τελειώσει ποτέ η ενέργεια. Για αυτόν τον λόγο η λέξη “będę grał” βρίσκεται στον σύνθετο μέλλοντα.

Η φωνή ενός ρήματος δείχνει την σχέση μεταξύ του υποκειμένου και του ρήματος. Στα πολωνικά έχουμε τρεις φωνές και πιο συγκεκριμένα:

- Strona czynna (ενεργητική φωνή), όπου το υποκείμενο εκτελεί την ενέργεια του ρήματος. Στην πρόταση “Janek goli swojego tatę.” (Ο Γιάννης ξυρίζει τον πατέρα του.) βλέπουμε ότι υποκείμενο (Γιάννης) εκτελεί την ενέργεια του ρήματος (ξυρίζει).
- Strona bierna (παθητική φωνή), όπου το υποκείμενο δέχεται την ενέργεια του ρήματος αλλά δεν την εκτελεί. Στην πρόταση “Janek został ogolony przez swojego tatę.” (Ο Γιάννης ξυρίστηκε από τον πατέρα του.) βλέπουμε ότι υποκείμενο (Γιάννης) δέχεται την ενέργεια του ρήματος (ξυρίστηκε) από το αντικείμενο (πατέρα).
- Strona zwrotna (αντανακλαστική φωνή), όπου το υποκείμενο δέχεται την ενέργεια του ρήματος και ταυτόχρονα την εκτελεί. Στην πρόταση “Janek goli się.” (Ο Γιάννης ξυρίζεται.) βλέπουμε ότι υποκείμενο (Γιάννης) δέχεται την ενέργεια του ρήματος (ξυρίζεται) αλλά και ταυτόχρονα είναι αυτός κάνει την ενέργεια.

Ως έγκλιση χαρακτηρίζουμε την στάση του ομιλητή απέναντι στην ενέργεια του ρήματος. Με πιο απλά λόγια, με την έγκλιση καταλαβαίνουμε τι προσπαθεί να καταφέρει ο ομιλητής με την πρότασή του. Στα πολωνικά οι εγκλίσεις είναι τρεις:

- Tryb oznajmujący (ή orzekający) (Οριστική Έγκλιση). Στην έγκλιση αυτή ο ομιλητής είναι ουδέτερος και προσπαθεί απλά να μεταφέρει μια πληροφορία. (πχ. Gram w grę. – Παίζω ένα παιχνίδι.)

- Tryb przypuszczający (Υποτακτική Έγκλιση). Στην έγκλιση αυτή ο ομιλητής εκφράζει εικασία, αμφιβολία ή αναποφασιστικότητα. Τα ρήματα σε αυτή την έγκλιση συνήθως τελειώνουν σε “bym/byś/by” (πχ. Pograłbym w jakąś grę, ale muszę się uczyć. – Θα έπαιζα κάποιο παιχνίδι, αλλά πρέπει να διαβάσω.)
- Tryb roskazujący (Προστακτική Έγκλιση). Στην έγκλιση αυτή ο ομιλητής δίνει εντολές ή εκφράζει μια επιθυμία ή ένα αίτημα. (πχ. Wyciągnij ciasto z lodówki. – Βγάλε το κέικ από το ψυγείο.)

Τα ρήματα κλίνονται ως προς το γένος όπως και τα ουσιαστικά και τα επίθετα. Το γένος χωρίζεται σε αρσενικό, θηλυκό, ουδέτερο στον ενικό αριθμό και αρσενικό και μη αρσενικό στον πληθυντικό αριθμό. Η κλίση των ρημάτων ως προς το γένος είναι μόνο δυνατή στον αόριστο, σύνθετο μέλλοντα και στην υποτακτική έγκλιση.

Όπως και αναφέραμε στην περίπτωση του μέλλοντα τα ρήματα χωρίζονται σε όψεις. Οι όψεις δείχνουν αν μια ενέργεια είναι σε εξέλιξη, θα βρίσκεται σε εξέλιξη, έχει τελειώσει ή θα τελειώσει στο μέλλον. Οι κατηγορίες των ρημάτων ως προς όψεις είναι δύο:

- Czasowniki dokonane (Ολοκληρωμένα ρήματα). Είναι η κατηγορία ρημάτων που δηλώνουν ότι μια ενέργεια έχει ολοκληρωθεί ή θα ολοκληρωθεί στο μέλλον και μπορούμε να προβλέψουμε το αποτέλεσμα της ή ξέρουμε ότι θα τελειώσει. Για παράδειγμα το “ogolił się” (ξυρίστηκε) δηλώνει ότι η ενέργεια του ρήματος τέλειωσε.
- Czasowniki niedokonane (Μη ολοκληρωμένα ρήματα). Είναι η κατηγορία ρημάτων στα οποία δεν είναι φανερό αν η ενέργεια έχει ολοκληρωθεί ή θα ολοκληρωθεί στο μέλλον. Για παράδειγμα το “goli się” (ξυρίζεται) δηλώνει ότι η ενέργεια του ρήματος εκτελείται ακόμα αλλά δεν ξέρουμε αν θα τελειώσει (και τελικά θα ξυριστεί).

Τέλος όταν θέλουμε να εκφράσουμε άρνηση (πχ. nie idę - δεν πάω) τα ρήματα τα γράφουμε πάντα χωριστά από το “nie” (όχι). Αντιθέτως από τα επίθετα τα οποία συνήθως γράφονται μαζί με το “nie”.

	Ενεστώτας	Παρατατικός	Σύνθετος Μέλλ.
Ja	jestem	byłem	będę
Ja		byłam	będę
Ty	jesteś	byłeś	będziesz
Ty		byłaś	będziesz
On/Ona/Ono	jest	był	będzie
On/Ona/Ono		była	będzie
On/Ona/Ono		było	będzie
My	jesteśmy	byliśmy	będziemy
My		byłyśmy	będziemy
Wy	jesteście	byliście	będziecie
Wy		byłyście	będziecie
Oni/One	są	byli	będą
Oni/One		były	będą

Πίνακας 2.8: Παράδειγμα κλίσης λέξης być (είμαι) στα πολωνικά (Οριστική έγκλιση)

Ja	byłbym
Ja	byłabym
Ty	byłbyś
Ty	byłabyś
On/Ona/Ono	byłby
On/Ona/Ono	byłaby
On/Ona/Ono	byłoby
My	bylibyśmy
My	byłybyśmy
Wy	bylibyście
Wy	byłybyście
Oni/One	byliby
Oni/One	byłyby

Πίνακας 2.9: Παράδειγμα κλίσης λέξης być (είμαι) στα πολωνικά (Υποτακτική Έγκλιση)

Ty	bądź
On/Ona/Ono	niech będzie
My	bądźmy
Wy	bądźcie
Oni/One	niech będą

Πίνακας 2.10: Παράδειγμα κλίσης λέξης być (είμαι) στα πολωνικά (Προστακτική Έγκλιση)

Bezokolicznik (Απαρέμφατο)	Imiesłów Przymiotnikowy Teraźniejszy Czynny (Μετοχή επιθέτου ενεστώτας ενεργητική φωνή)	Imiesłów Przyłówkowy Współczesny (Μετοχή επιρρήματος παρατατικός)
być	będący	będąc

Πίνακας 2.11: Υπόλοιπες μορφές του ρήματος być (είμαι) στα πολωνικά

	Ενεστώτας	Παρατατικός	Μέλλοντας
Εγώ	είμαι	ήμουν	θα είμαι
Εσύ	είσαι	ήσουν	θα είσαι
Αυτός/Αυτή/Αυτό	είναι	ήταν	θα είναι
Εμείς	είμαστε	ήμασταν / ήμαστε	θα είμαστε
Εσείς	είσαστε / είστε	ήσασταν / ήσαστε	θα είσαστε / είστε
Αυτά	είναι	ήταν	θα είναι

Πίνακας 2.12: Παράδειγμα κλίσης λέξης być (είμαι) στα ελληνικά (Οριστική έγκλιση)

	Ενεστώτας	Παρατατικός
Εγώ	να είμαι	να ήμουν
Εσύ	να είσαι	να ήσουν
Αυτός/Αυτή/Αυτό	να είναι	να ήταν
Εμείς	να είμαστε	να ήμασταν / ήμαστε
Εσείς	να είσαστε / είστε	να ήσασταν / ήσαστε
Αυτά	να είναι	να ήταν

Πίνακας 2.13: Παράδειγμα κλίσης λέξης być (είμαι) στα ελληνικά (Υποτακτική έγκλιση)

Συνοψίζοντας, στο κεφάλαιο αυτό μελετήσαμε τα βασικότερα στοιχεία της πολωνικής γλώσσας. Είδαμε την αλφάβητο που χρησιμοποιείται και κάποιους βασικούς κανόνες προφοράς λέξεων. Τέλος μελετήσαμε τα κύρια κλιτά μέρη του λόγου (ουσιαστικά, ρήματα, επίθετα) για να καταλάβουμε πόσες διαφορετικές μορφές μπορεί να διαθέτει μια λέξη το οποίο τελικά οδηγεί στα πολωνικά να είναι μια δύσκολη γλώσσα για stemming.

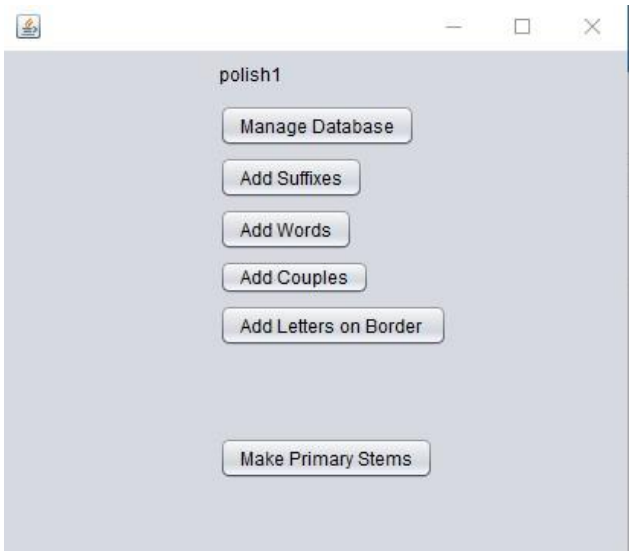
ΚΕΦΑΛΑΙΟ 3: Stemmer Σουίτα

ΕΙΣΑΓΩΓΗ

Στο πλαίσιο αυτής της εργασίας θα χρησιμοποιήσουμε την Stemmer Builder σουίτα του Δρ. Νικήτα Καρανικόλα. Πρόκειται για μια σουίτα προγραμμάτων λογισμικού μέσω της οποίας ειδικοί (experts) μπορούν να φτιάξουν stemmers χωρίς να γράψουν ούτε μια γραμμή κώδικα. Μέσω της σουίτας οι ειδικοί παράγουν έναν μηχανικό και επιβλεπόμενο stemmer τον οποίο ονομάζουμε primary stemmer. Με βάση τον primary stemmer και τα επιχειρήματα ειδικών (πάνω στην έξοδο του primary stemmer) μέσω μιας διεπαφής (wizard) οι ειδικοί μπορούν να δημιουργήσουν όσους μηχανικά παραγμένους stemmer επιθυμούν (trial stemmers). Οι ειδικοί δεν είναι απαραίτητο να είναι ομιλητές της γλώσσας για την οποία προσπαθούμε να δημιουργήσουμε stemmers. Σε αυτό το κεφάλαιο θα μελετήσουμε την Stemmer Builder σουίτα και όλη την λειτουργικότητα που μας προσφέρει.

3.1 Language Manager

Μέσω της διεπαφής του Language Manager ένας χρήστης μπορεί να ορίσει νέες γλώσσες, να ρυθμίζει τις παραμέτρους γλωσσών που ήδη υπάρχουν και να δημιουργεί νέους primary stemmers.

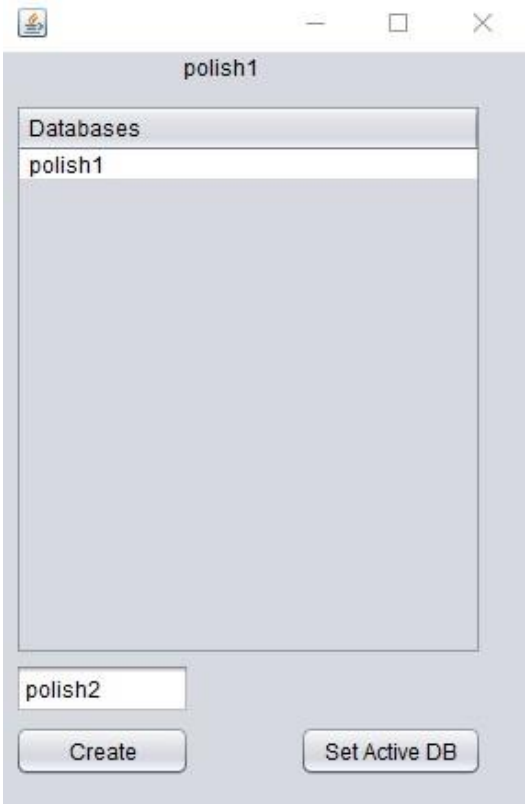


Εικόνα 3.1: Language Manager: Διεπαφή

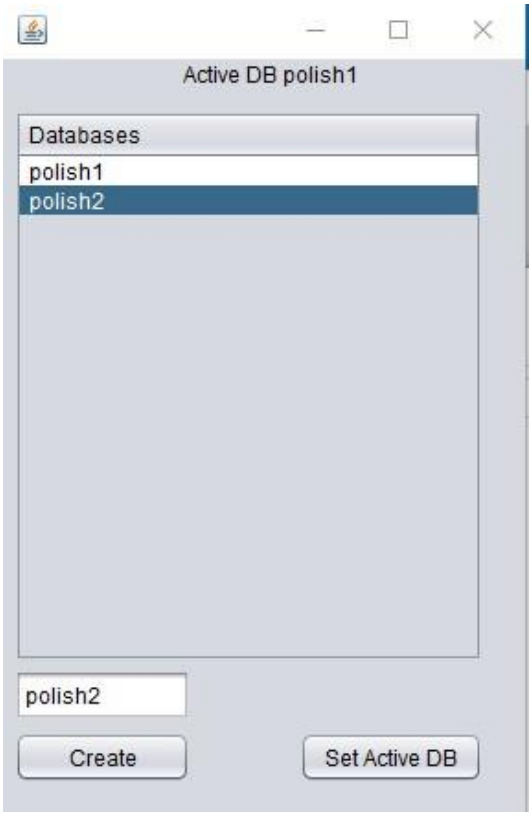
Το κουμπί “Manage Database” μας ανοίγει μια νέα διεπαφή μέσω της οποίας μπορούμε να εισάγουμε νέες γλώσσες στο σύστημα και να διαλέγουμε σε ποια γλώσσα θα εργαζόμαστε. Το Manage Database διαθέτει δυο κουμπιά:

- “Create” μέσω του οποίου δημιουργούμε νέες γλώσσες. Η νέα γλώσσα παίρνει για όνομα ό,τι βάλαμε στο πεδίο κειμένου πάνω από το κουμπί. Στην πραγματικότητα δημιουργείται μία νέα βάση και ένας νέος φάκελος (στο C:/stemSuite/<ONOMA>) με το όνομα που διαλέξαμε. Είναι σημαντικό να σημειωθεί ότι μπορούμε να δημιουργήσουμε δυο βάσεις για την ίδια γλώσσα, αρκεί να έχουν διαφορετικό όνομα.
- “Set Active DB” μέσω του οποίου επιλέγουμε την γλώσσα (βάση) πάνω στην οποία θέλουμε να εργαστούμε.

Εικόνα 3.2: Αρχική διεπαφή του Manage Database



Εικόνα 3.3: Manage Database: Εισαγωγή ονόματος νέας γλώσσας



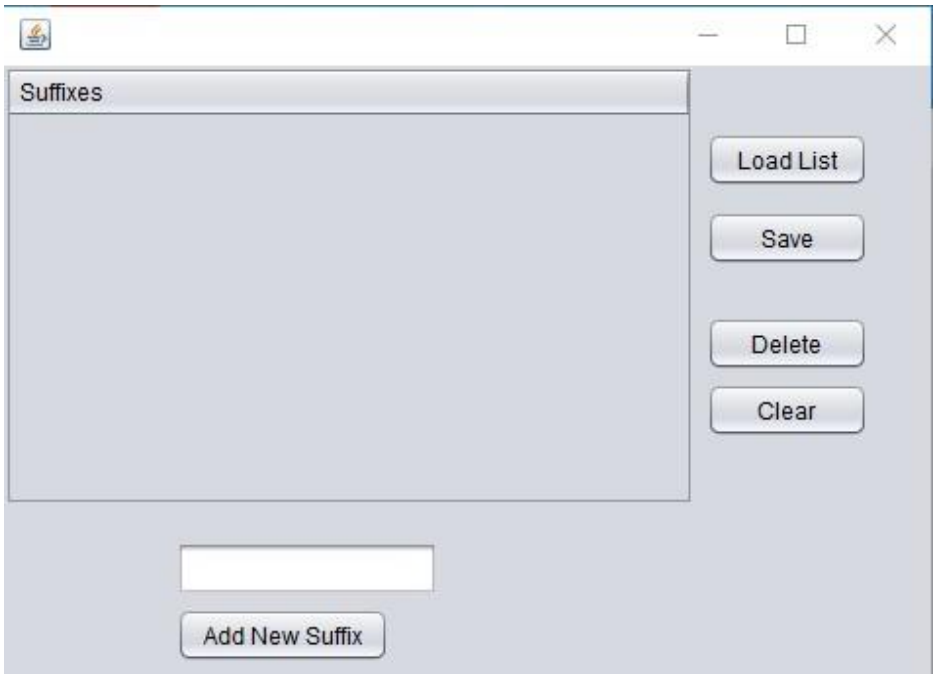
Εικόνα 3.4: Manage Database: Εισαγωγή νέας γλώσσας

Εικόνα 3.5: Manage Database: Αλλαγή ενεργής γλώσσας

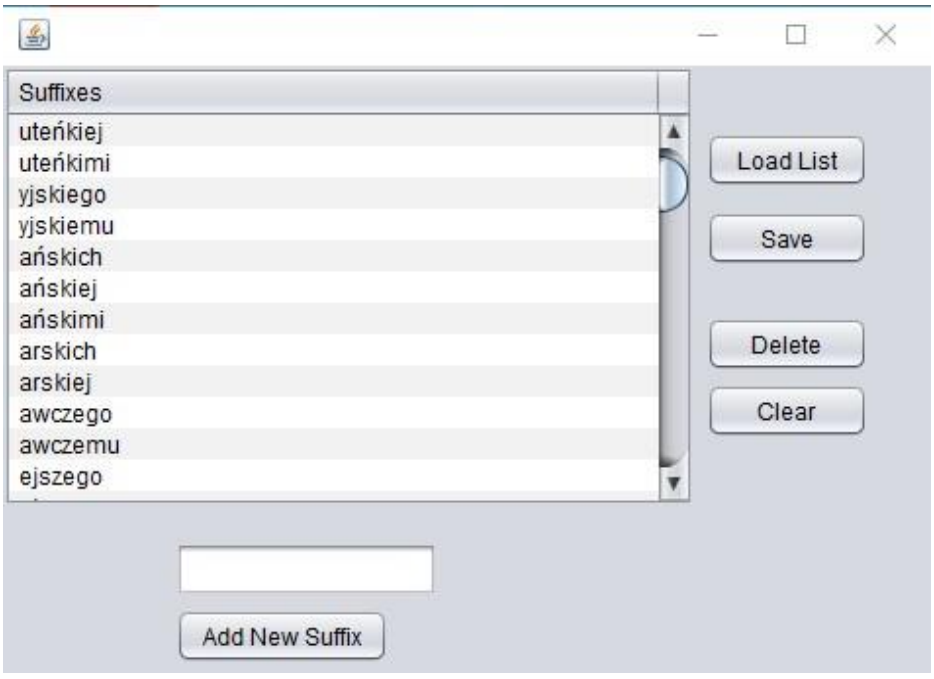
Με το κουμπί “Add Suffixes” του Language Manager ανοίγουμε νέα διεπαφή μέσω της οποίας διαχειριζόμαστε τις καταλήξεις οι οποίες θα χρησιμοποιηθούν στο stemming. Το Add Suffixes διαθέτει πέντε κουμπιά:

- “Add new Suffix” μέσω του οποίου προσθέτουμε νέες καταλήξεις. Για να προστεθεί η κατάληξη στην λίστα στην διεπαφή, ο χρήστης πρέπει να την γράψει στο πεδίο κειμένου και ύστερα να πατήσει το κουμπί. Ο χρήστης μπορεί να βάλει όσες καταλήξεις θέλει, με τον μόνο περιορισμό ότι μπορεί να τις βάζει μια μια.
- “Save” μέσω του οποίου αποθηκεύουμε σε αρχείο κειμένου τις καταλήξεις που έχουν προστεθεί ή αφαιρεθεί από την λίστα στην διεπαφή (C:/stemSuite/<ONOMA>/suffixelist.txt).
- “Load List” φορτώνει από το παραπάνω αρχείο όλες τις καταλήξεις και τις εμφανίζει πάνω στην διεπαφή.

- “Delete” μέσω του οποίου διαγράφουμε μια κατάληξη από την λίστα. Εάν επιλέξουμε πάνω από μια κατάληξη έχουμε την δυνατότητα να διαγράψουμε όσες επιλέξαμε.
- “Clear” αφαιρούμε όλες τις καταλήξεις από την διεπαφή.



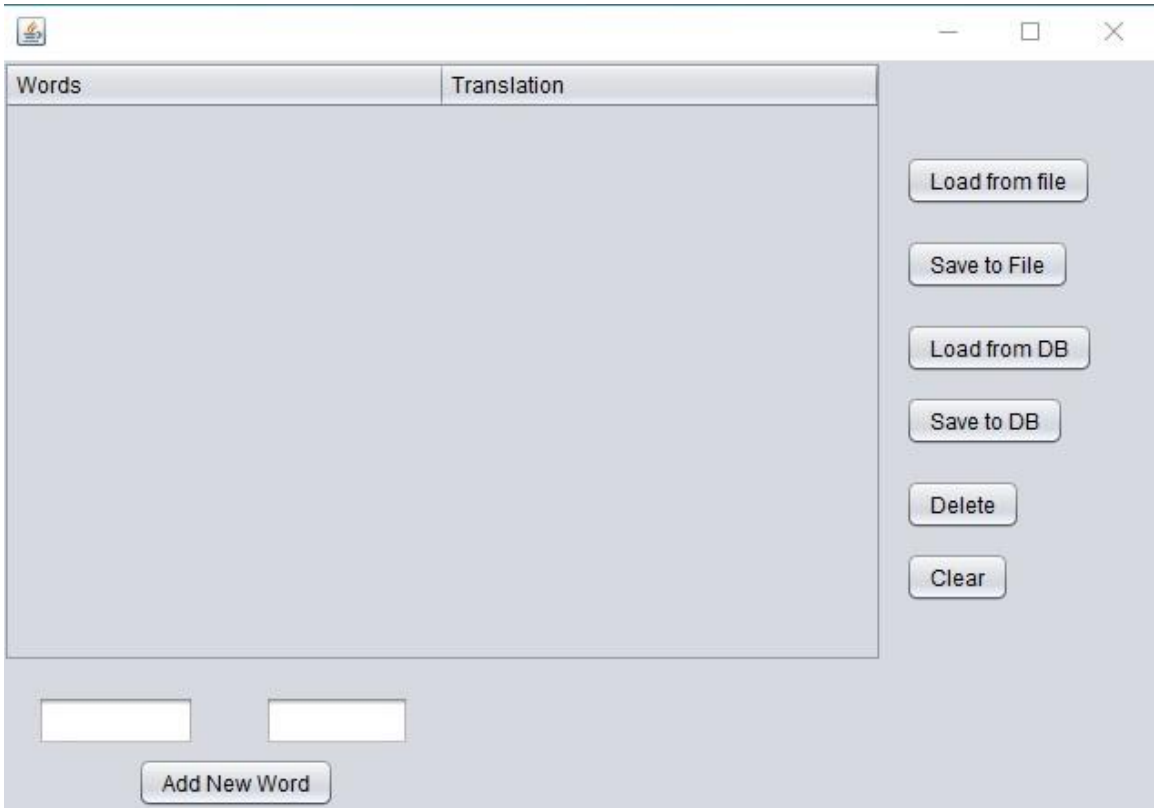
Εικόνα 3.6: Add suffixes: Διεπαφή χωρίς καταλήξεις



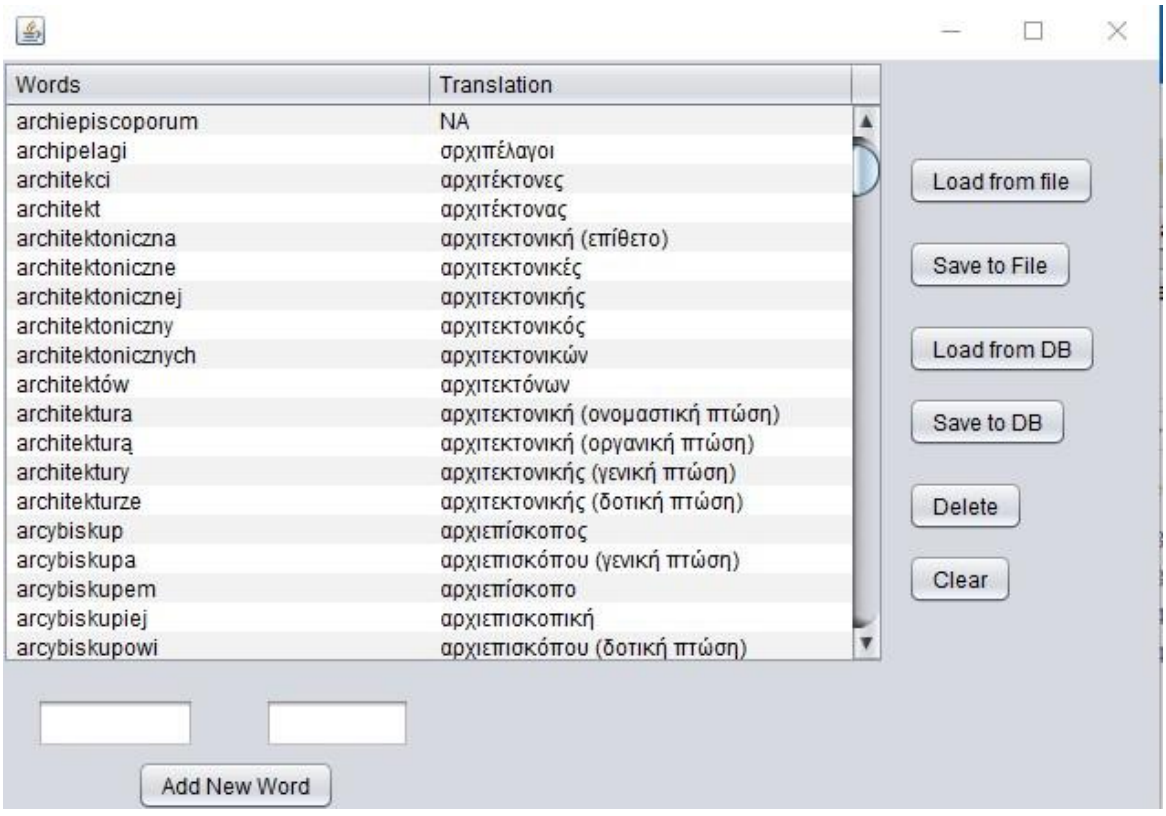
Εικόνα 3.7: Add suffixes: Διεπαφή με φορτωμένες καταλήξεις

Με το κουμπί “Add Words” του Language Manager ανοίγουμε νέα διεπαφή μέσω της οποίας διαχειριζόμαστε λέξεις και τις μεταφράσεις τους. Από τις λέξεις που θα εισάγει εδώ ο χρήστης θα δημιουργηθούν οι ρίζες του primary stemmer και οι ρίζες όλων των trial stemmers. Αυτές οι λέξεις συν οι μεταφράσεις μαζί με τις ρίζες τους από τον primary stemmer θα δοθούν ύστερα στους ειδικούς, ώστε να τα συγκρίνουν και να εκφράσουν εάν συμφωνούν ή όχι με τα αποτελέσματα. Το Add Words διαθέτει επτά κουμπιά:

- “Add New Word” μέσω του οποίου προσθέτουμε νέες λέξεις. Για να προστεθεί η λέξη στην λίστα στην διεπαφή, ο χρήστης πρέπει να την γράψει στο πεδίο κειμένου αριστερά και ύστερα να πατήσει το κουμπί. Είναι πολύ χρήσιμο να προσθέσουμε και την μετάφραση στο πεδίο δεξιά διότι δεν είναι απαραίτητο ο ειδικός να είναι ομιλητής της γλώσσας. Ο χρήστης μπορεί να βάλει όσες νέες λέξεις θέλει, με τον μόνο περιορισμό ότι μπορεί να τις βάζει μια μια.
- “Save to File” μέσω του οποίου αποθηκεύουμε σε αρχείο κειμένου τις λέξεις και τις μεταφράσεις που έχουν προστεθεί ή αφαιρεθεί από την λίστα στην διεπαφή (C:/stemSuite/<ONOMA>/words.txt).
- “Load from File” φορτώνει από το παραπάνω αρχείο όλες τις λέξεις και τις μεταφράσεις και τις εμφανίζει πάνω στην διεπαφή.
- “Save to DB” αποθηκεύει τις λέξεις και τις μεταφράσεις από την διεπαφή στην ενεργή βάση δεδομένων (αυτήν που διαλέξαμε στο Manage Database). Είναι πολύ σημαντικό να αποθηκεύσουμε τις λέξεις στην βάση διότι κάποια επόμενα κομμάτια (πχ. δημιουργία primary stemmer) δουλεύουν μόνο με τα δεδομένα της βάσης.
- “Load from DB” φορτώνει από την βάση δεδομένων όλες τις λέξεις και τις μεταφράσεις και τις εμφανίζει πάνω στην διεπαφή.
- “Delete” μέσω του οποίου διαγράφουμε μια λέξη και μετάφραση από την λίστα. Εάν επιλέξουμε πάνω από μια λέξη έχουμε την δυνατότητα να διαγράψουμε όσες επιλέξαμε.
- “Clear” αφαιρούμε όλες τις λέξεις και καταλήξεις από την διεπαφή.



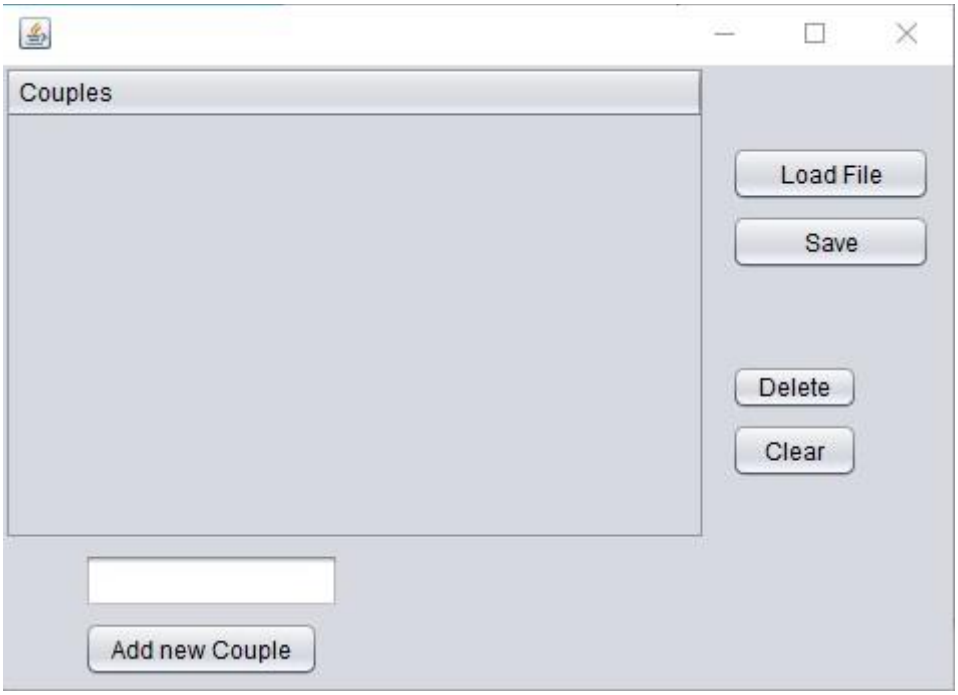
Εικόνα 3.8: Add Words: Διεπαφή χωρίς τις λέξεις



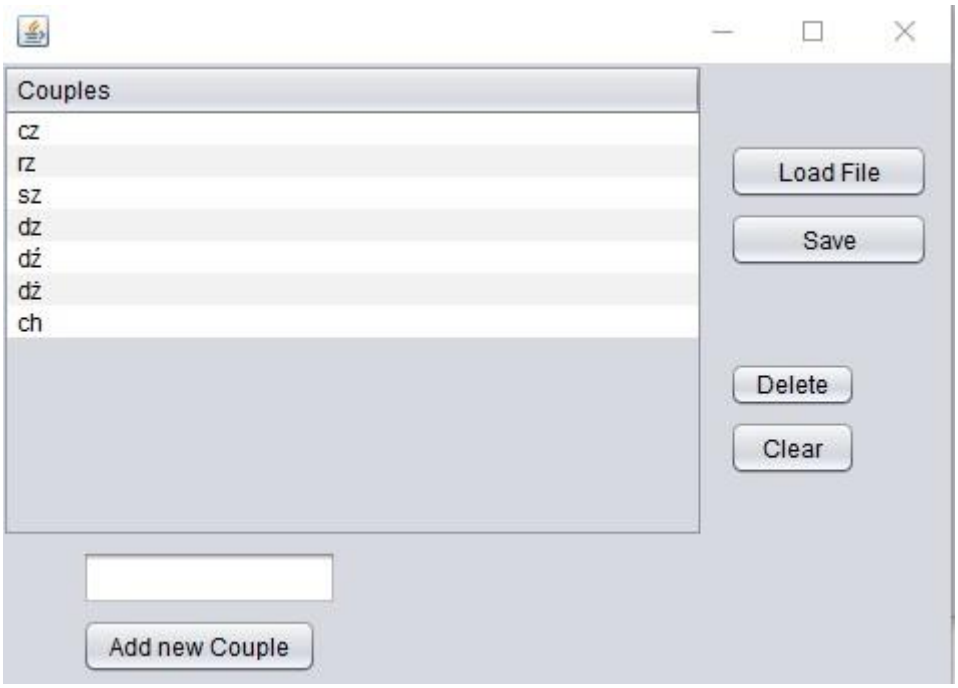
Εικόνα 3.9: Add Words: Διεπαφή με λέξεις

Με το κουμπί “Add Couples” του Language Manager ανοίγουμε νέα διεπαφή μέσω της οποίας διαχειριζόμαστε τα διπλά γράμματα. Το Add Couples διαθέτει πέντε κουμπιά:

- “Add new Couple” μέσω του οποίου προσθέτουμε νέα διπλά γράμματα. Για να προστεθεί το διπλό γράμμα στην λίστα στην διεπαφή, ο χρήστης πρέπει να το γράψει στο πεδίο κειμένου και ύστερα να πατήσει το κουμπί. Ο χρήστης μπορεί να βάλει όσα διπλά γράμματα θέλει, με τον μόνο περιορισμό ότι μπορεί να τα βάζει ένα ένα.
- “Save” μέσω του οποίου αποθηκεύουμε σε αρχείο κειμένου τα διπλά γράμματα που έχουν προστεθεί ή αφαιρεθεί από την λίστα στην διεπαφή (C:/stemSuite/<ONOMA>/couples.txt).
- “Load List” φορτώνει από το παραπάνω αρχείο όλα τα διπλά γράμματα και τα εμφανίζει πάνω στην διεπαφή.
- “Delete” μέσω του οποίου διαγράφουμε ένα διπλό γράμμα από την λίστα. Εάν επιλέξουμε πάνω από ένα διπλό γράμμα έχουμε την δυνατότητα να διαγράψουμε όσες επιλέξαμε.
- “Clear” αφαιρούμε όλα τα διπλά γράμματα από την διεπαφή.



Εικόνα 3.10: Add Couples: Διεπαφή χωρίς τα διπλά γράμματα



Εικόνα 3.11: Add Couples: Διεπαφή με τα διπλά γράμματα

Με το κουμπί “Add Letters on Borders” του Language Manager ανοίγουμε νέα διεπαφή μέσω της οποίας διαχειριζόμαστε γράμματα τα οποία χρειάζονται ειδικό χειρισμό. Η λογική αυτής της διεπαφής είναι ακριβώς ίδια με αυτές για τις καταλήξεις και τα διπλά γράμματα. Διαθέτει πέντε κουμπιά:

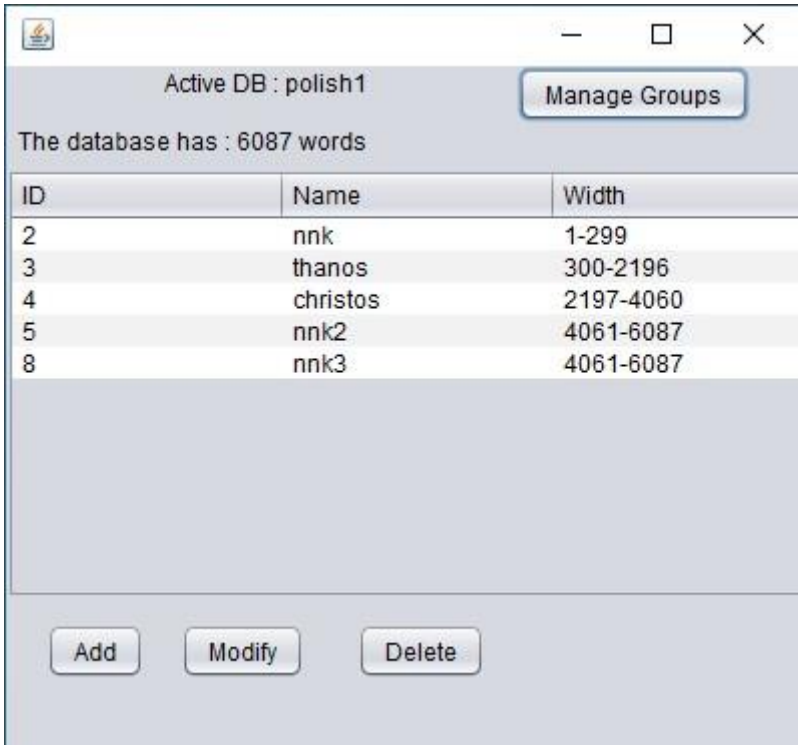
- “Add new Border Letter” μέσω του οποίου προσθέτουμε νέα γράμματα. Για να προστεθεί το διπλό γράμμα στην λίστα στην διεπαφή, ο χρήστης πρέπει να το γράψει στο πεδίο κειμένου και ύστερα να πατήσει το κουμπί.
- “Save” μέσω του οποίου αποθηκεύουμε σε αρχείο κειμένου τα γράμματα που έχουν προστεθεί ή αφαιρεθεί από την λίστα στην διεπαφή (C:/stemSuite/<ONOMA>/ lettersOnBorderlist.txt).
- “Load List” ” φορτώνει από το παραπάνω αρχείο όλα τα γράμματα και τα εμφανίζει πάνω στην διεπαφή.
- “Delete” μέσω του οποίου διαγράφουμε ένα ή περισσότερα γράμματα από την λίστα.
- “Clear” αφαιρούμε όλα τα γράμματα από την διεπαφή.

Εικόνα 3.12: Add Letters on Borders: Διεπαφή

Το τελευταίο κουμπί στην του Language Manager είναι το “Make Primary Stems”. Αυτό δεν ανοίγει νέα διεπαφή, αλλά τρέχει στο παρασκήνιο. Δημιουργεί ρίζες του primary stemmer παίρνοντας τις λέξεις που προσθέσαμε μέσω Add Words και αφαιρώντας την μεγαλύτερη κατάληξη που ταιριάζει από αυτές που βάλαμε στο Add Suffixes. Τέλος προσθέτει όλες τις ρίζες στην βάση δεδομένων.

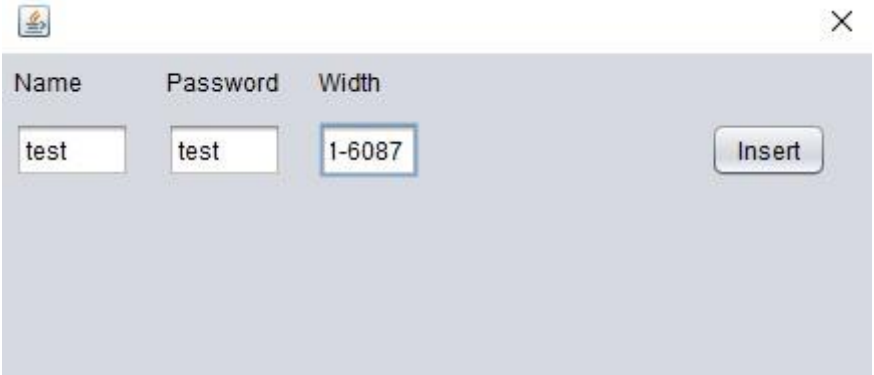
3.2 Experts Manager

Όπως αναφέραμε και προηγουμένως οι ειδικοί (experts) είναι τα άτομα τα οποία θα ελέγξουν την έξοδο του primary stemmer και θα μας δώσουν επιχειρήματα (arguments) για το πόσο σωστά ομαδοποιήθηκαν οι ρίζες. Μέσω του Experts Manager μπορούμε να ορίζουμε τα ονόματα των ειδικών, τους κωδικούς πρόσβασης και τις ομάδες λέξεων στις οποίες θα έχει πρόσβαση ο κάθε ειδικός.

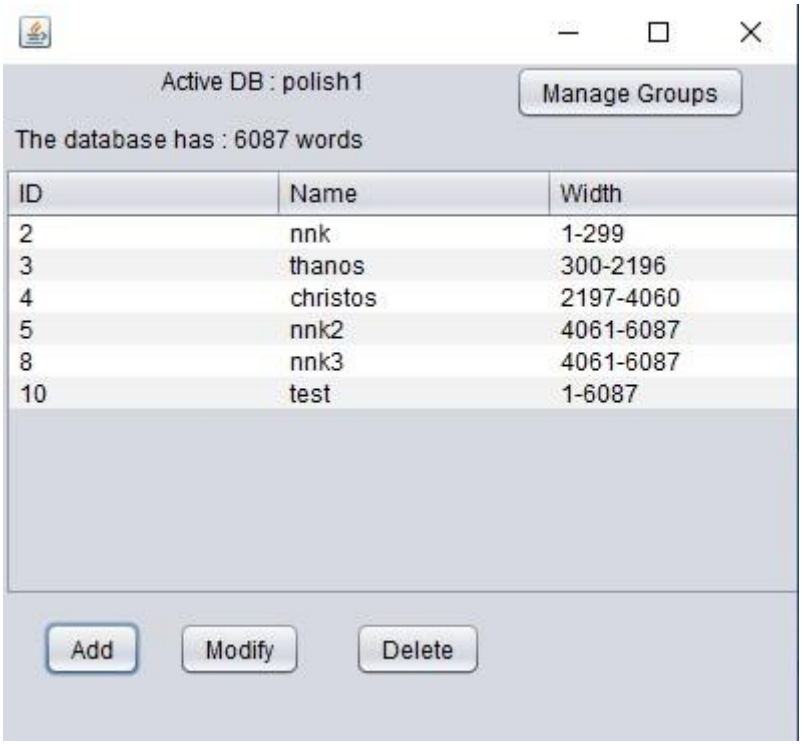


Εικόνα 3.13: Experts Manager: Αρχική διεπαφή

Το κουμπί “Add” μας ανοίγει μια νέα διεπαφή πάνω στην δίνουμε όνομα, κωδικό πρόσβασης και εύρος λέξεων στις οποίες θα έχει πρόσβαση ο νέος ειδικός. Μπορούμε να αφήσουμε το πεδίο κωδικού άδειο, το οποίο σημαίνει ότι ο συγκεκριμένος ειδικός δεν θα έχει κωδικό πρόσβασης. Στο παρακάτω παράδειγμα προσθέτουμε νέο ειδικό με όνομα “test”, κωδικό “test” και εύρος λέξεων “1-6087”.

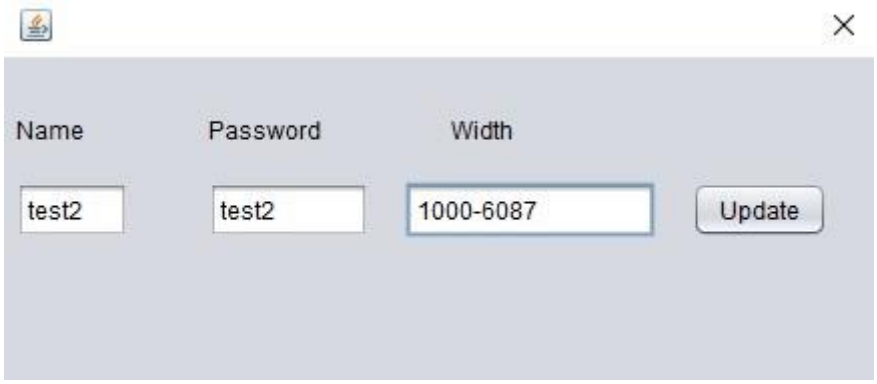


Εικόνα 3.14: Add: Προσθήκη νέου ειδικού

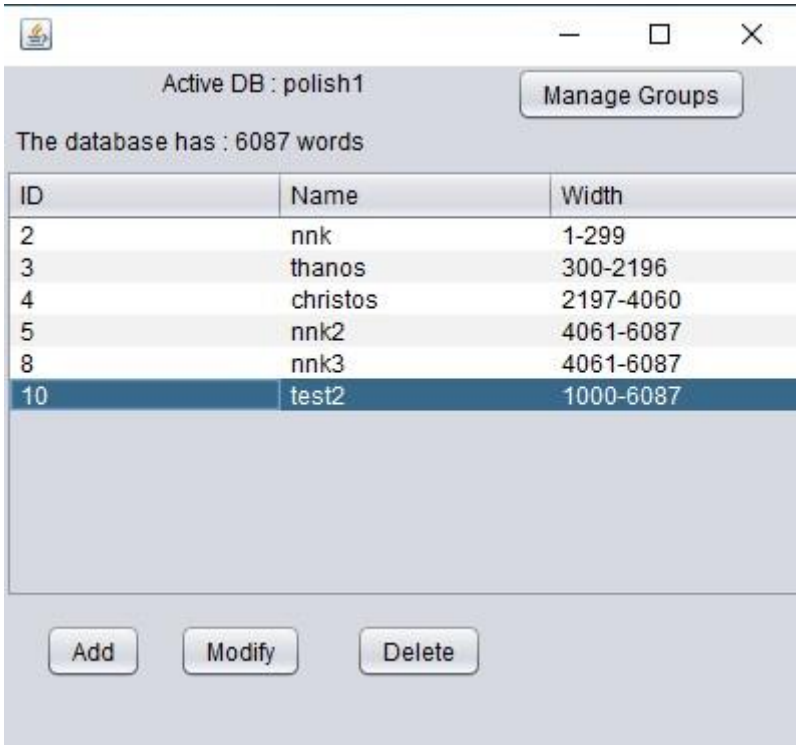


Εικόνα 3.15: Experts Manager: Διεπαφή μετά την προσθήκη ειδικού “test”

Με το “Modify” ανοίγουμε μια νέα διεπαφή πάνω στην οποία αλλάζουμε τα στοιχεία κάποιου συγκεκριμένου ειδικού που επιλέξαμε. Στο παρακάτω παράδειγμα αλλάξαμε στον ειδικό “test” το όνομα σε “test2”, τον κωδικό σε “test2’ και το εύρος λέξεων στις οποίες έχει πρόσβαση σε “1000-6087”.

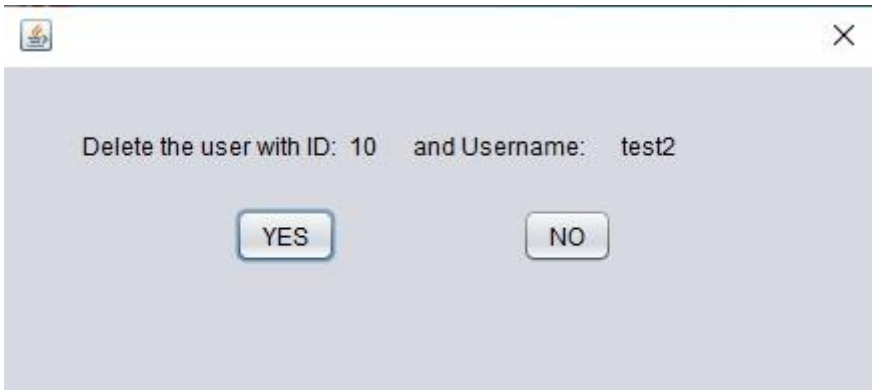


Εικόνα 3.16: Modify: Αλλαγή στοιχείων ειδικού που ήδη υπάρχει στο σύστημα



Εικόνα 3.17: Experts Manager: Διεπαφή μετά την αλλαγή στοιχείων του ειδικού “test”

Με το κουμπί “Delete” διαγράφουμε κάποιον συγκεκριμένο ειδικό. Πριν ο ειδικός διαγραφτεί όμως, ανοίγεται μια νέα διεπαφή “Ναι/Όχι” η οποία ρωτάει άμα είμαστε σίγουροι στο εάν θέλουμε να διαγράψουμε τον ειδικό μονίμως.



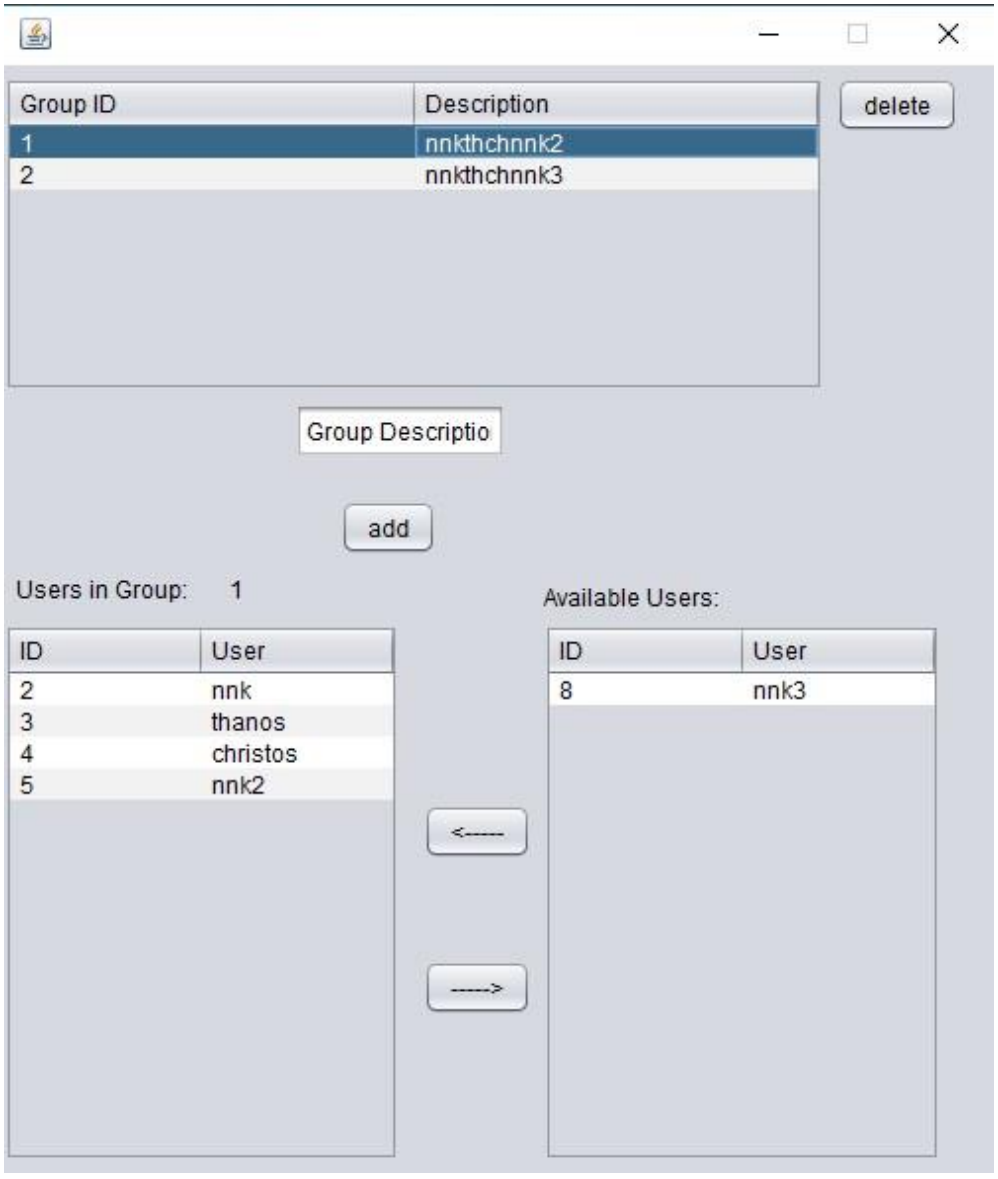
Εικόνα 3.18: Delete: Διαγραφή ειδικού “test2”

Τέλος το κουμπί “Manage Groups” μας ανοίγει μια νέα διεπαφή μέσω της οποίας μπορούμε να ορίζουμε ομάδες ειδικών. Αυτό θα φανεί χρήσιμο κατά την δημιουργία των trial stemmers, όπου θα μπορούμε να χρησιμοποιήσουμε επιχειρήματα ενός ειδικού ή ομάδων ειδικών στην συναρμολόγηση των trial stemmers. Ο wizard τότε θα

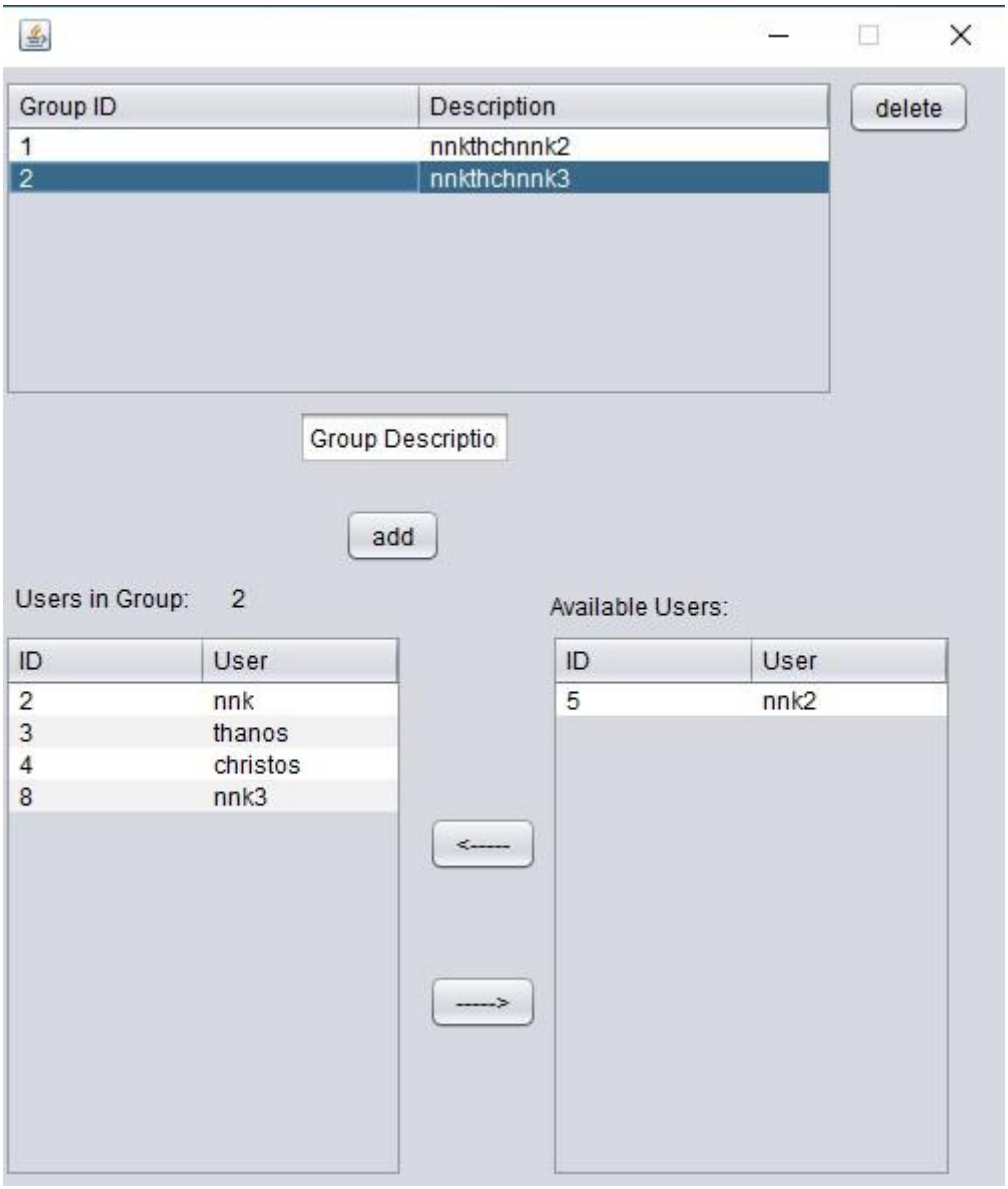
προσπαθήσει όσο το δυνατόν περισσότερο να συμμορφωθεί με τα επιχειρήματα ειδικών.

Μέσω της διεπαφής προσθέτουμε ομάδες με το κουμπί “Add” και το πεδίο κειμένου ακριβώς από πάνω. Με το “Delete” διαγράφουμε την επιλεγμένη ομάδα ή ομάδες. Τους ειδικούς τους προσθέτουμε στην επιλεγμένη ομάδα με τα βελάκια.

Στο παρακάτω παράδειγμα βλέπουμε δυο ομάδες ειδικών (nnkthchnnk2 και nnkthchnnk3) οι οποίες έχουν όλους τους ειδικούς εκτός του “nnk3” στην περίπτωση του “nnkthchnnk2” και του “nnk2” στην περίπτωση του “nnkthchnnk3”.



Εικόνα 3.19: Manage Groups: Ομάδα “nnkthchnnk2”



Εικόνα 3.20: Manage Groups: Ομάδα “nnkthchnnk3”

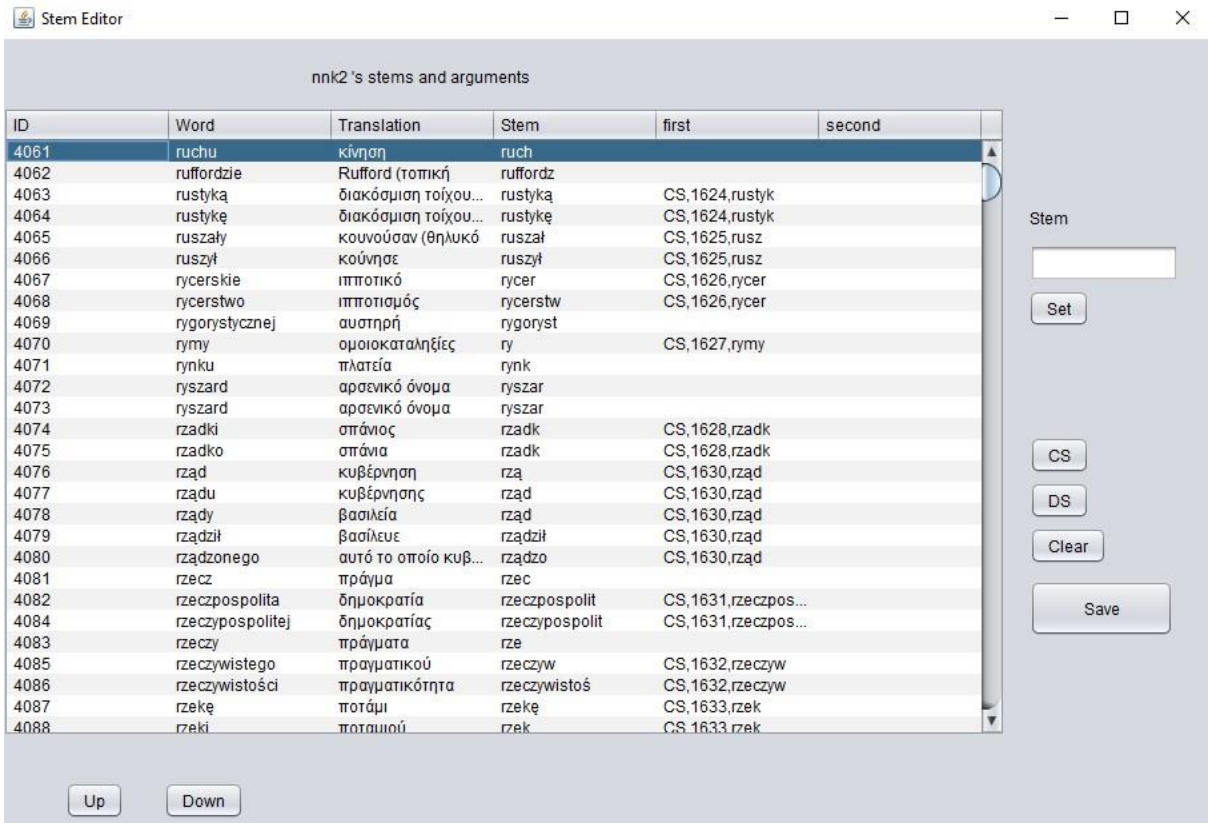
3.3 Stem Editor

Το Stem Editor είναι το εργαλείο με το οποίο οι ειδικοί μας δίνουν τα επιχειρήματά (arguments) τους. Μέσω του λογισμικού αυτού οι ειδικοί συγκρίνουν τις ρίζες του primary stemmer και δίνουν επιχειρήματα (ή αλλιώς σχολιάζουν) για το πόσο σωστή είναι η έξοδος (η ρίζα) κατά την γνώμη τους. Τα επιχειρήματα θα χρησιμοποιηθούν έπειτα στην συναρμολόγηση των trial stemmers όπου ο wizard θα βασιστεί πάνω σε αυτά για την ρύθμισή τους.

Όταν ο ειδικός ανοίγει το λογισμικό αρχικά πρέπει να εισάγει το όνομά του και τον κωδικό πρόσβασής του.

Εικόνα 3.21: Stem Editor: Log In

Μόλις ο ειδικός εισάγει τους κωδικούς, τότε ανοίγει μια διεπαφή πάνω στην οποία βλέπει όλες τις λέξεις στις οποίες του έχουν δοθεί πρόσβαση μέσω του Experts Manager μαζί με την μετάφρασή τους και τα αντίστοιχα stems του primary stemmer.



Εικόνα 3.22: Stem Editor: Διεπαφή

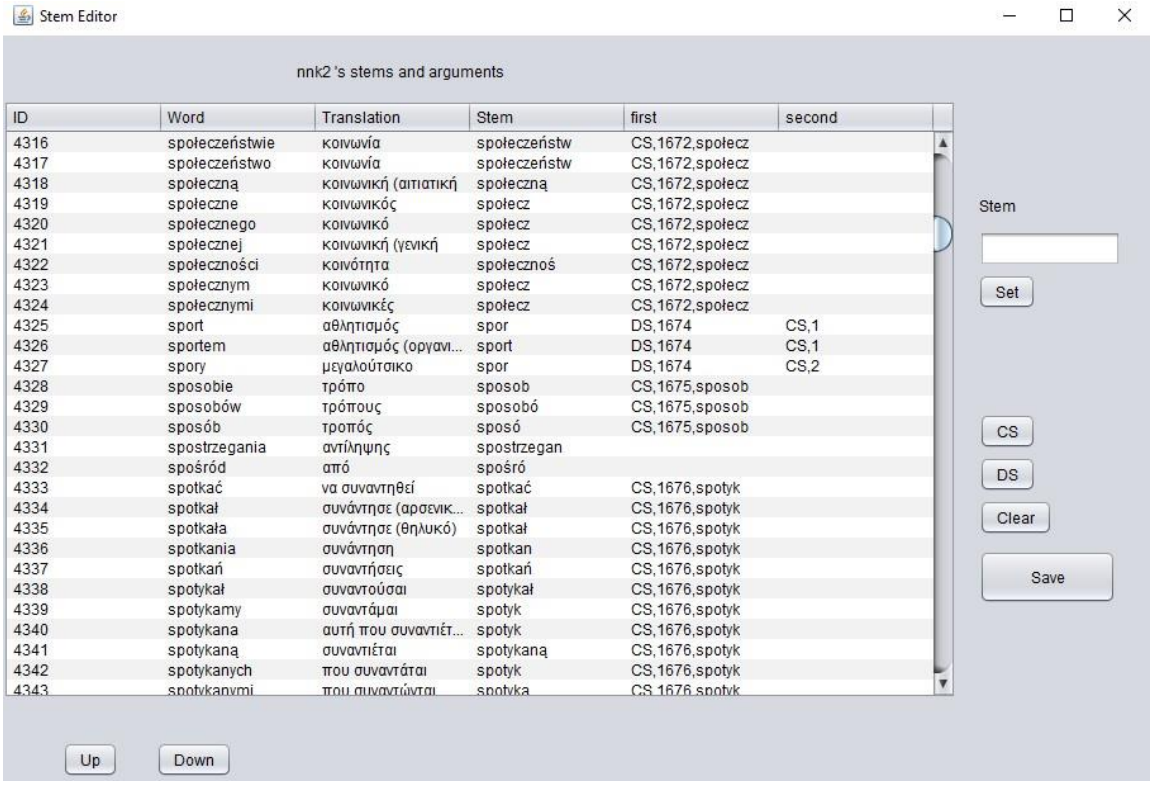
Τα επιχειρήματα (arguments) των ειδικών χωρίζονται σε τρεις κύριες κατηγορίες. Πιο συγκεκριμένα:

- CS είναι μια ομάδα λέξεων με κοινή ρίζα (Common Stem)
- DS είναι μια ομάδα λέξεων οι οποίες θα έπρεπε να έχουν διαφορετικές ρίζες (Different Stem)
- DS/CS έχουμε όταν σε μια ομάδα DS υπάρχουν πολλαπλά υποσύνολα από CS

Για παράδειγμα στην παραπάνω εικόνα οι λέξεις 4076-4080 είναι σημασιολογικά παρόμοιες (κυβερνάω / βασιλεύω). Ο ειδικός εντόπισε την ομοιότητα αυτή και θεώρησε ότι οι λέξεις αυτές πρέπει όλες να ανήκουν σε μια ομάδα. Για αυτόν τον λόγο τις έβαλε όλες σε μια υποομάδα CS και πρότεινε σαν κοινή ρίζα την μεγαλύτερη κοινή το “rząd”. Την ομάδα CS την όρισε μέσω του κουμπιού “CS”, ενώ την ρίζα την ρίζα την εισήγαγε στο πεδίο κειμένου και πάτησε το κουμπί “Set”.

Δεν είναι απαραίτητο σε μια ομάδα “CS” να υπάρχουν πολλές λέξεις. Μπορεί ο ειδικός να μην συμφωνεί στην ρίζα μιας λέξης και μπορεί να θέλει να προτείνει μια άλλη. Στην παραπάνω εικόνα για παράδειγμα ο ειδικός παρατήρησε την λέξη 4070 “rymy”. Ο primary stemmer για αυτήν την λέξη έβγαλε την ρίζα “ry”. Ο ειδικός θεώρησε ότι μια ρίζα με δυο χαρακτήρες μπορεί να προκαλέσει προβλήματα εξαιτίας του μικρού μήκους. Για τον λόγο αυτό έβαλε την λέξη μόνη της σε μια ομάδα CS και πρότεινε σαν νέα ρίζα την λέξη ολόκληρη αφού δεν είχε άλλες λέξεις με όμοια σημασία για να τις συγκρίνει.

Για να ορίσει μια ομάδα DS/CS ο ειδικός πρέπει αρχικά να ορίσει όλη την ομάδα λέξεων σαν DS και ύστερα πρέπει να κατηγοριοποιεί τα CS χωριστά. Στην παρακάτω εικόνα ο primary stemmer έβγαλε την ίδια ρίζα “spor” στην λέξη “sport” (αθλητισμός) και στην λέξη “spory” (μεγαλούτσικος). Αυτό είναι λάθος διότι αυτές οι δυο λέξεις δεν έχουν καμία σημασιολογική σχέση μεταξύ τους. Ο ειδικός το είδε και έφτιαξε μια ομάδα DS με τις τρεις λέξεις 4325, 4326, 4327. Τις λέξεις 4235 και 4326 τις έβαλε σε μια ομάδα CS διότι προέρχονται από την λέξη “αθλητισμός” ενώ την λέξη 4237 την έβαλε σε χωριστή ομάδα CS αφού σημαίνει “μεγαλούτσικος”.



Εικόνα 3.23: Stem Editor: Παράδειγμα DS/CS

3.4 StemmerEvaluatorV3

Μέσω της εφαρμογής StemmerEvaluatorV3 δημιουργούμε τους trial stemmers και συγκρίνουμε τα αποτελέσματα των διάφορων stemmers (primary ή trial) με τα επιχειρήματα των ειδικών ή και με άλλα stemmers. Το λογισμικό αυτό αποτελείται από μια κύρια κλάση (Matching) με την οποία μπορούμε να “φωνάξουμε” δυο διαφορετικές διεπαφές, την Stemmer2UI και EvaluatorUI. Συνεπώς δεν μπορούμε απλά να ανοίξουμε την εφαρμογή, αλλά πρέπει να “φωνάξουμε” την κατάλληλη κλάση μέσω της γραμμής εντολών (cmd).

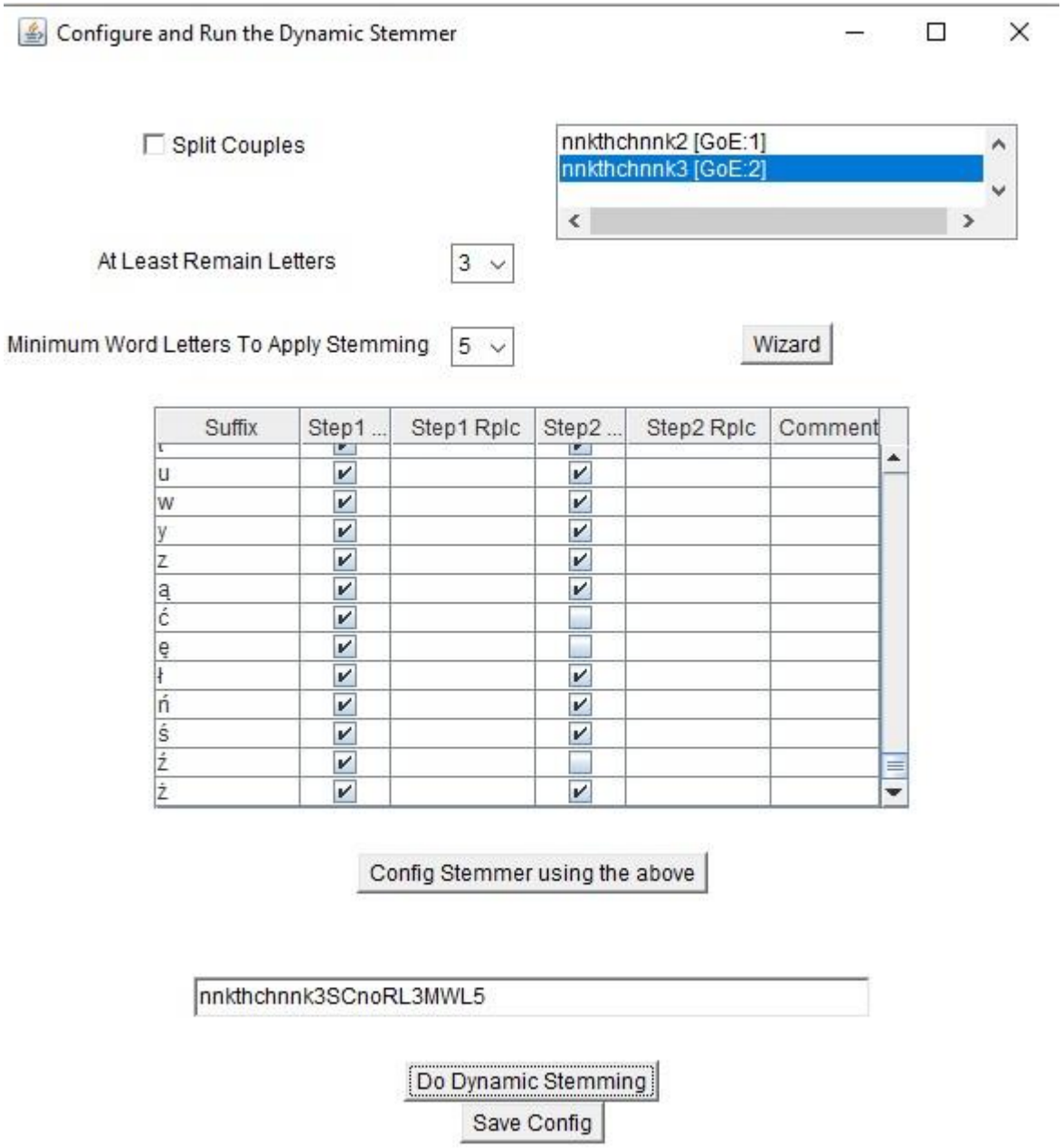
java -classpath StemmerEvaluatorV3.jar Matching.Stemmer2UI
java -classpath StemmerEvaluatorV3.jar Matching.EvaluatorUI

Η κλάση Stemmer2UI είναι υπεύθυνη για την ρύθμιση και δημιουργία των trial stemmers. Οι trial stemmers λειτουργούν με βάση αφαίρεσης των καταλήξεων σε δυο στάδια, η κατάληξη που κόβεται είναι η μεγαλύτερη που ταιριάζει από την λίστα. Όταν φωνάξουμε αυτήν την κλάση ανοίγεται μια διεπαφή και για να δημιουργήσουμε έναν trial stemmer πρέπει να ακολουθήσουμε τα παρακάτω βήματα:

- Διαλέγουμε τον ειδικό ή ομάδα ειδικών πάνω στην οποία θέλουμε ο trial stemmer μας να προσαρμοστεί.
- Ρυθμίζουμε την παράμετρο “Split Couples” με την οποία δηλώνουμε αν επιθυμούμε να χωρίσουμε τα διπλά γράμματα.
- Ρυθμίζουμε την παράμετρο “At Least Remaining Letters” με την οποία δηλώνουμε το ελάχιστο μέγεθος των ριζών.
- Ρυθμίζουμε την παράμετρο “Minimum Word Length to Apply Stemming” με την οποία δηλώνουμε το μέγεθος της μικρότερης λέξης που μπορεί να περάσει από stemming.
- Πατάμε το κουμπί “Wizard” το οποίο ρυθμίζει με βάση τον ειδικό ή ομάδα ειδικών που διαλέξαμε ποιες καταλήξεις και σε ποιο από τα δυο στάδια θα τις κόψει (δεν θα τις χρησιμοποιήσει) ο trial stemmer.
- Πατάμε το “Configue Stemmer using the above” το οποίο οριστικοποιεί τις ρυθμίζεις του trial stemmer μέσα στο πρόγραμμα.

- Προσθέτουμε στο πεδίο κειμένου το όνομα για το trial stemmer. Μπορεί να αποτελείται από γράμματα, αριθμούς και κενά. Είναι σημαντικό όμως το όνομα να αρχίζει από γράμμα διότι αυτό θα είναι και το όνομα της Java κλάσης που θα παράγουμε μετά. Η κλάση αυτή θα είναι ο trial stemmer σε μορφή ξεχωριστού προγράμματος.
- Πατάμε το “Do Dynamic Stemming” το οποίο κάνει stemming με τον trial stemmer που ρυθμίσαμε πάνω στις λέξεις στην βάση και στο τέλος αποθηκεύει τις ρίζες που παράχθηκαν στην βάση.
- Τέλος πατάμε το “Save Confing” το οποίο αποθηκεύει τις ρυθμίσεις του trial stemmer μέσα σε ένα αρχείο κειμένου στο C:/stemSuite/<ONOMA>/. Οι καταλήξεις στο αρχείο αυτό είναι κρυπτογραφημένες με τον SHA-256 αλγόριθμο.

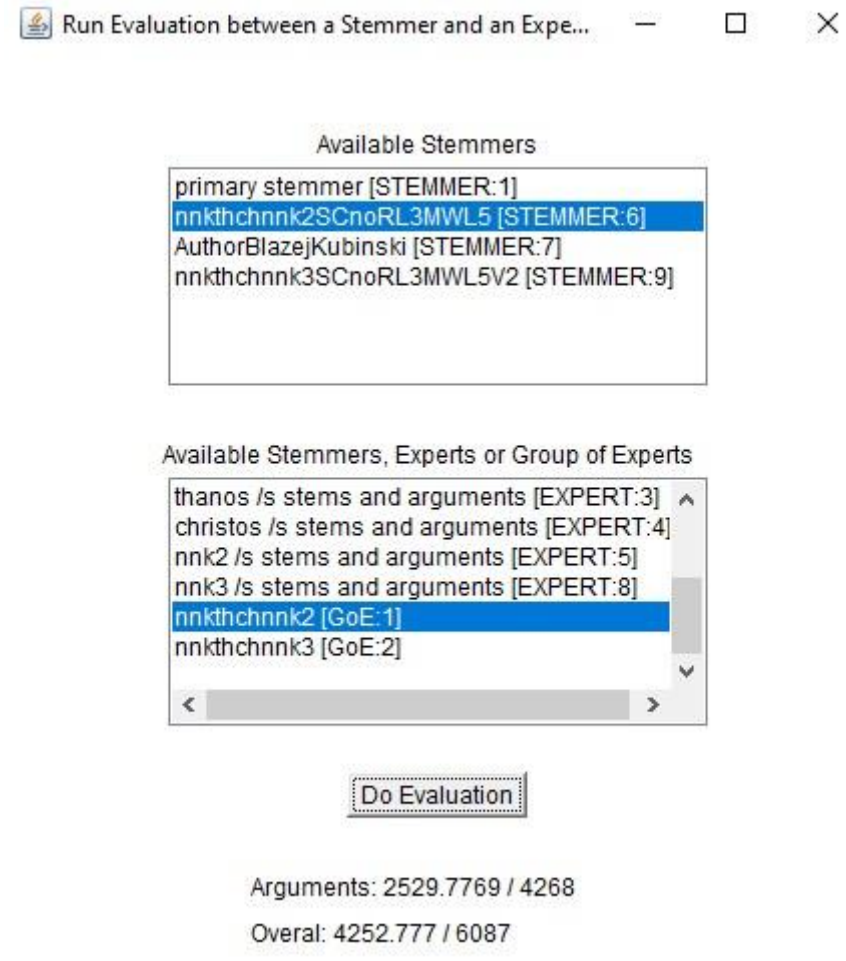
Την επιλογή καταλήξεων, σε ποιο στάδιο stemming θα γίνουν, μπορούμε να την κάνουμε και χειροκίνητα. Το λογισμικό μας δίνει την δυνατότητα να ενεργοποιήσουμε και απενεργοποιήσουμε όποια κατάληξη θέλουμε και σε όποιο στάδιο επιθυμούμε.



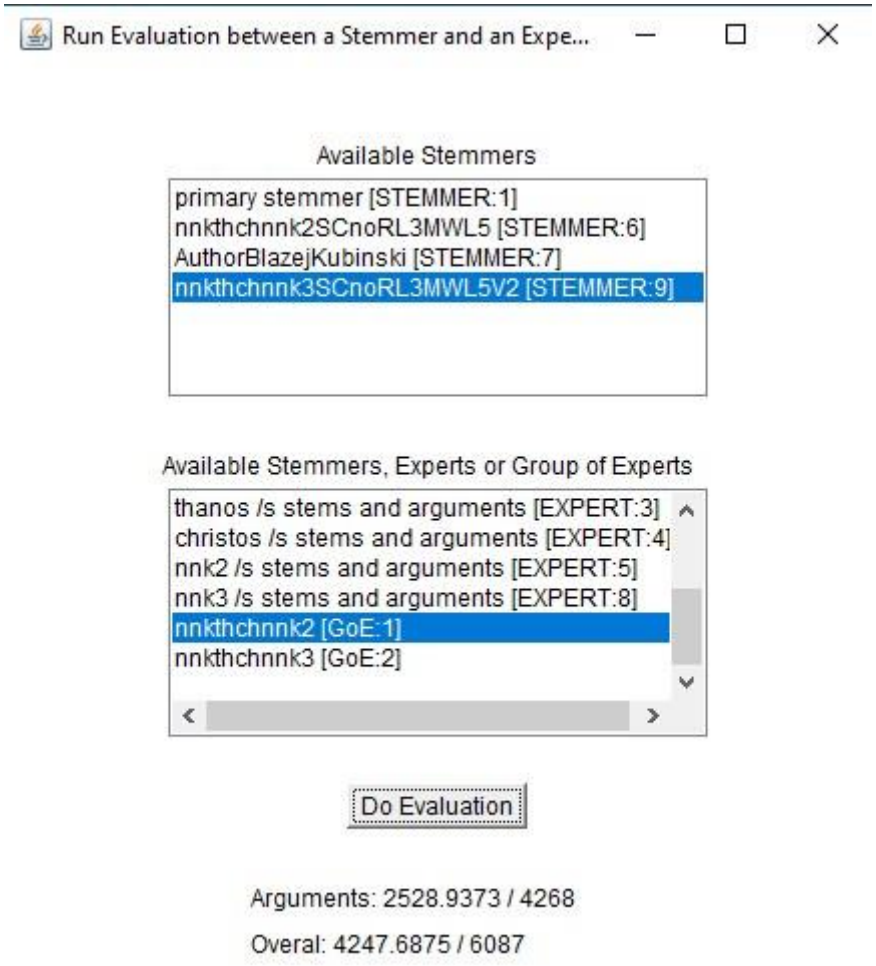
Εικόνα 3.24: Stemmer2UI: Ρύθμιση trial stemmer

Με την κλάση EvaluatorUI αξιολογούμε τους διαφορετικούς stemmers που δημιουργήσαμε ώστε να βρούμε τον καλύτερο. Η κύρια λογική του ελέγχου είναι να συγκρίνουμε τα αποτελέσματα των stemmers με τα επιχειρήματα των ειδικών ώστε να βρούμε τον βαθμό ομοιότητας μεταξύ τους. Δεν είναι απαραίτητο όμως να συγκρίνουμε stemmer με ομάδες ειδικών, μπορούμε να συγκρίνουμε και πόσο όμοιοι είναι δυο stemmers μεταξύ τους

Στις παρακάτω εικόνες συγκρίνουμε δυο stemmers (nnkthchnnk2SCnoRL3MWL5 και nnkthchnnk3SCnoRL3MWL5V2) με τα επιχειρήματα μιας ομάδας ειδικών (nnkthchnnk2). Βλέπουμε ότι ο βαθμός ομοιότητας στην περίπτωση του nnkthchnnk2SCnoRL3MWL5 είναι 4252.777/6087, ενώ ο βαθμός ομοιότητας του nnkthchnnk3SCnoRL3MWL5V2 είναι 4247.6875/6087. Βάση των αποτελεσμάτων συμπεραίνουμε ότι ο nnkthchnnk2SCnoRL3MWL5 είναι ελάχιστα καλύτερος αφού τα αποτελέσματά του είναι περισσότερο όμοια με την ομάδα ειδικών από ότι του δεύτερου trial stemmer με την ομάδα ειδικών.



Εικόνα 3.25: EvaluatorUI: nnkthchnnk2SCnoRL3MWL5 vs nnkthchnnk2



Εικόνα 3.26: EvaluatorUI: nnkthchnnk3SCnoRL3MWL5V2 vs nnkthchnnk2

3.5 Code Builder

Τελευταίο εργαλείο το οποίο μας παρέχει η σουίτα είναι το Code Builder. Με το Code Builder μπορούμε να μετατρέψουμε το αρχείο κειμένου με τις ρυθμίσεις ενός trial stemmer σε κώδικα Java. Δηλαδή δημιουργεί τον trial stemmer σαν χωριστό πρόγραμμα, το οποίο μπορούμε να χρησιμοποιήσουμε έξω από το περιβάλλον της σουίτας.

Το Code Builder δεν διαθέτει γραφική διεπαφή. Πρέπει να τρέξουμε το πρόγραμμα μέσω της γραμμής εντολών (cmd) και να δώσουμε ως παράμετρο το όνομα του αρχείου με τις ρυθμίσεις του trial stemmer για τον οποίο θέλουμε να φτιάξουμε κώδικα. Είναι σημαντικό το όνομα της παραμέτρου να μην έχει την επέκταση “.txt” και το αρχείο

κειμένου με τις ρυθμίσεις πρέπει να βρίσκεται μέσα στον υποφάκελο της επιλεγμένης γλώσσας C:/stemSuite/<ONOMA>/.
Με την παρακάτω εντολή καλούμε το πρόγραμμα.

```
java -jar CodeBuilder.jar ONOMATRIALSTEMMER
```

Το αποτέλεσμα της παραπάνω εντολής είναι .java αρχείο το οποίο δημιουργείται στον υποφάκελο της επιλεγμένης γλώσσας. Για να δημιουργήσουμε το εκτελέσιμο αρχείο εκτελούμε την παρακάτω εντολή.

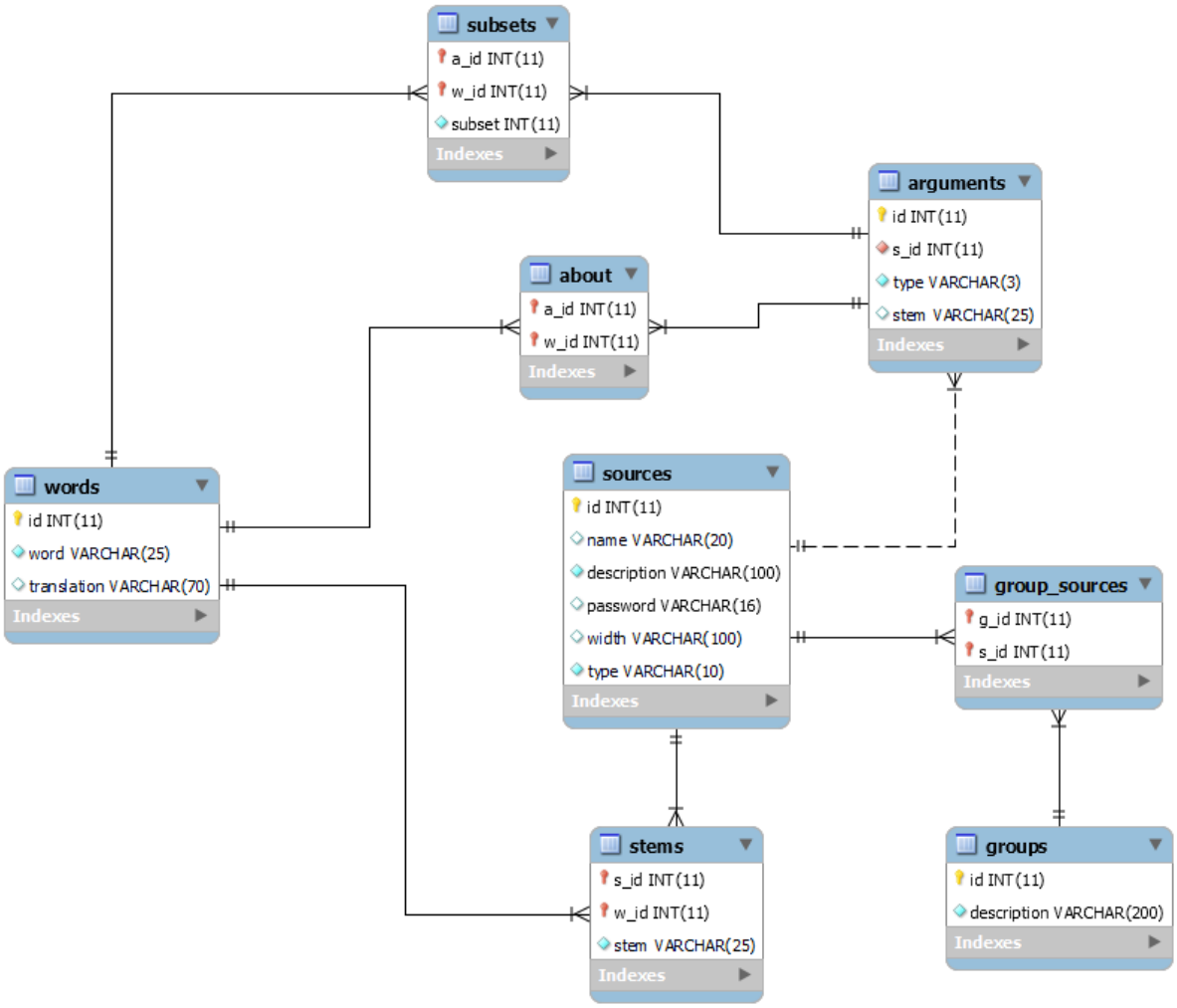
```
javac –encoding UTF-8 ONOMATRIALSTEMMER.java
```

Το εκτελέσιμο που δημιουργείται έχει το όνομα ONOMATRIALSTEMMER.class. Το τρέχουμε με την παρακάτω εντολή στην οποία θα πρέπει να δώσουμε δυο αρχεία κειμένου σαν παραμέτρους. Η πρώτη παράμετρος είναι αρχείο κειμένου με τις λέξεις πάνω στις οποίες στις οποίες θέλουμε να εφαρμόσουμε το stemming. Οι λέξεις πρέπει να είναι χωρισμένες με αλλαγή γραμμή. Η δεύτερη παράμετρος του προγράμματος είναι αρχείο μέσα στο οποίο θα αποθηκεύσουμε τις ρίζες των λέξεων της πρώτης παραμέτρου.

```
java ONOMATRIALSTEMMER  input.txt stemmed.txt
```

3.6 Βάση Δεδομένων

Η παρακάτω εικόνα αναπαριστά το σχήμα της βάσης δεδομένων της σουίτας με όλους τους πίνακες και όλες τις εξαρτήσεις.



Εικόνα 3.27: Σχήμα βάσης δεδομένων

Στο κεφάλαιο αυτό μελετήσαμε το πακέτο λογισμικού Stemmer Builder Suite του κύριου Νικήτα Καρανικόλα. Είδαμε πως μπορούμε να ρυθμίσουμε την σουίτα ώστε να μπορεί να υποστηρίζει νέες γλώσσες. Επίσης είδαμε πως μπορούμε να προσθέσουμε ειδικούς και πως οι ειδικοί μπορούν να επιχειρηματολογούν πάνω στις εξόδους του primary stemmer. Τέλος μάθαμε πως μπορούμε να δημιουργήσουμε μηχανικά stemmer βάση των επιχειρημάτων των ειδικών και πως μπορούμε να τον αποθηκεύσουμε ώστε να μπορεί να λειτουργεί έξω από το περιβάλλον της σουίτας.

ΚΕΦΑΛΑΙΟ 4: Błażej Kubiński Stemmer

ΕΙΣΑΓΩΓΗ

Η Stemmer σουίτα δημιουργεί μηχανικά παραγόμενους stemmers. Για να έχουμε έναν βαθμό σύγκρισης πρέπει να συγκρίνουμε τις εξόδους (τις ρίζες) τους με έναν άλλον, διαφορετικό stemmer. Ψάξαμε στο διαδίκτυο για έναν έτοιμο stemmer με χειροποίητα φτιαγμένους κανόνες. Δηλαδή έναν stemmer του οποίου οι κανόνες φτιάχτηκαν από έναν ειδικό της πολωνικής γλώσσας και όχι από μηχανή. Ένα άλλο προαπαιτούμενο για τον stemmer που ζητάμε είναι να βρίσκεται σε πηγαία γλώσσα. Αυτά τα κριτήρια τα ικανοποιεί ο stemmer του Błażej Kubiński ο οποίος βρίσκεται στην πλατφόρμα ανοιχτού κώδικα GitHub. Στο κεφάλαιο αυτό θα μελετήσουμε πλήρως τον stemmer μαζί με τους κανόνες του.

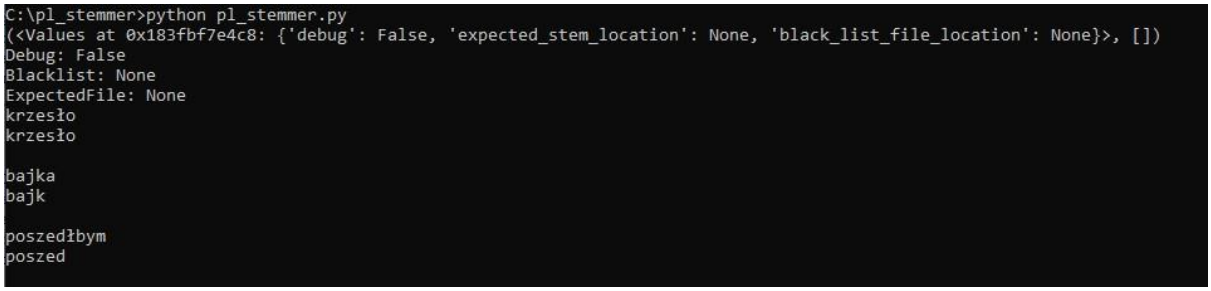
4.1 Błażej Kubiński Stemmer

Ο stemmer του Błażej Kubiński βασίζεται στον αλγόριθμο του Porter. Είναι ένας σχετικά απλός stemmer βασισμένος σε κανόνες, βάση τους οποίους αφαιρούνται καταλήξεις και σε κάποιες περιπτώσεις και τα προθήματα. Το πρόγραμμα είναι διαθέσιμο σε μορφή πηγαίου κώδικα στην πλατφόρμα GitHub από το 2014 και είναι γραμμένο στην γλώσσα προγραμματισμού python.

Το πρόγραμμα αυτό δεν διαθέτει γραφική διεπαφή (GUI), η αλληλεπίδραση με τον χρήστη γίνεται αποκλειστικά μέσω της γραμμής εντολών (cmd). Το πρόγραμμα ζητάει από τον χρήστη να εισάγει μια λέξη. Πάνω στην λέξη αυτή εφαρμόζει το stemming και εμφανίζει το αποτέλεσμα (την ρίζα) πάνω στην οθόνη.

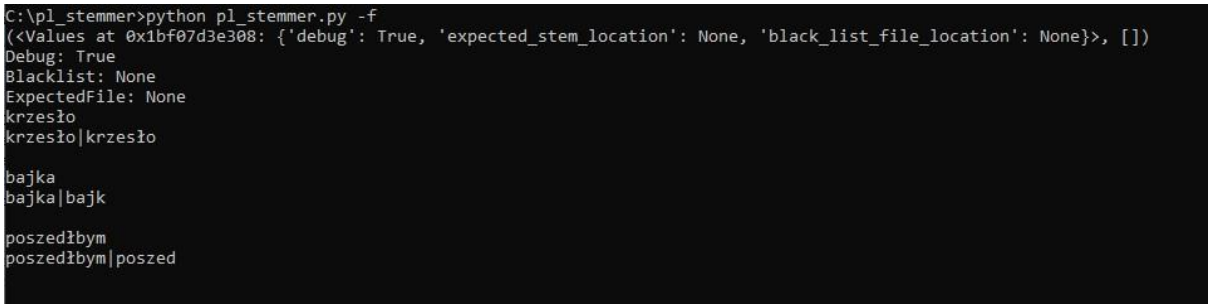
Το πρόγραμμα το καλούμε με την παρακάτω εντολή.

```
python pl_stemmer.py [-f] [-b blacklist] [-e expected_stems]
```



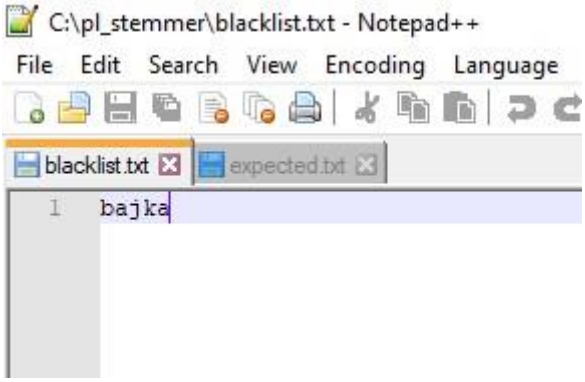
Εικόνα 4.1: Błażej Kubiński Stemmer Λειτουργία

Όπως βλέπουμε το πρόγραμμα διαθέτει επιπλέον λειτουργίες τις οποίες μπορούμε να τις ενεργοποιήσουμε προσθέτοντας παραμέτρους όταν το καλούμε. Η πρώτη παράμετρος είναι η “-f”. Εμφανίζει την ρίζα μαζί με την αρχική λέξη. Στην περίπτωση που δώσαμε την παράμετρο “-e” αυτό που εμφανίζεται είναι αρχική λέξη | ρίζα | αναμενόμενη ρίζα.



Εικόνα 4.2: Błażej Kubiński Stemmer Λειτουργία -f

Η παράμερος “-b” παίρνει την διαδρομή ως προς αρχείο με κείμενο το οποίο περιέχει τις λέξεις οι οποίες δεν πρέπει να περάσουν από τον αλγόριθμο stemming. Στο παρακάτω παράδειγμα δημιουργήσαμε ένα αρχείο “blacklist.txt” με την λέξη “bajka”. Ο αλγόριθμος (κανονικά) εφαρμόζει stemming στην λέξη αυτή και μας γυρνάει την ρίζα “bajk”. Επειδή όμως βάλαμε την λέξη στην λίστα λέξεων που δεν πρέπει να περάσουν από stemming, το πρόγραμμα θα μας την γυρίσει όπως είναι.



Εικόνα 4.3: blacklist.txt

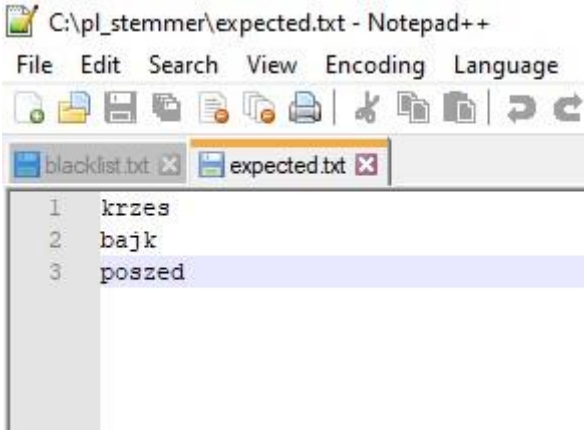
```
C:\pl_stemmer>python pl_stemmer.py -f -b blacklist.txt
(<Values at 0x1b56e02e5c8: {'debug': True, 'expected_stem_location': None, 'black_list_file_location': 'blacklist.txt'}>
, [])
Debug: True
Blacklist: blacklist.txt
ExpectedFile: None
The file with list of words not to stem was named: blacklist.txt
krzesło
krzesło|krzesło

bajka
bajka|bajka

poszedłbym
poszedłbym|poszed
```

Εικόνα 4.4: Błażej Kubiński Stemmer Λειτουργία -f -b

Η παράμετρος “-e” παίρνει την διαδρομή ως προς αρχείο με κείμενο (που περιέχει μέσα ρίζες) το οποίο ο χρήστης αναμένει ως αποτέλεσμα. Η επιλογή αυτή επίσης αξιολογεί τον αλγόριθμο. Δηλαδή το πρόγραμμα συγκρίνει την ρίζα που παράγει ο αλγόριθμος με αυτήν που περιμένουμε (αυτήν που βρίσκεται στο αρχείο κειμένου που δώσαμε). Ύστερα αυξάνει κάποιες από τις παρακάτω μεταβλητές (counters):

- Good – Ρίζα που έβγαλε ο αλγόριθμος είναι ίδια με την αναμενόμενη
- Bad – Ρίζα που έβγαλε ο αλγόριθμος είναι διαφορετική με την αναμενόμενη
- True positive – Ρίζα που έβγαλε ο αλγόριθμος είναι ίδια με την αναμενόμενη
- True negative – Ρίζα που έβγαλε ο αλγόριθμος είναι διαφορετική με την αναμενόμενη
- False positive – Stemming έγινε σε λέξη που δεν έπρεπε
- False negative – Stemming δεν έγινε σε λέξη στην θα έπρεπε να γίνει

Στο παρακάτω παράδειγμα δημιουργήσαμε αρχείο “expected.txt” με τις ρίζες τις οποίες περιμένουμε να βγάλει ο αλγόριθμος.



Εικόνα 4.5: expected.txt

```
C:\pl_stemmer>python pl_stemmer.py -f -b blacklist.txt -e expected.txt
(<Values at 0x26e1eebf648: {'debug': True, 'expected_stem_location': 'expected.txt', 'black_list_file_location': 'blacklist.t
xt'}>, [])
Debug: True
Blacklist: blacklist.txt
ExpectedFile: expected.txt
The file with list of words not to stem was named: blacklist.txt
The file with the prepared stems is located in:  expected.txt
krzesło
krzesło|krzesło|krzes

bajka
bajka|bajka|bajk

poszedłbym
poszedłbym|poszed|poszed

^Z

Good: 1 , bad: 2
Good / Bad: 0.3333333333333333

True positives: 1
True negatives: 0
False positives: 0
False negatives: 2
Precision: 1 / ( 1 + 0 ) = 1.0
Recall: 1 / ( 1 + 2 ) = 0.3333333333333333
Accuracy: ( 1 + 0 ) / ( 1 + 2 + 0 + 0 ) = 0.3333333333333333
```

Εικόνα 4.6: Błażej Kubiński Stemmer Λειτουργία -f -b -e

4.2 Κανόνες Błażej Kubiński Stemmer

Όπως αναφέραμε και προηγούμενος ο αλγόριθμος του Błażej Kubiński βασίζεται σε κανόνες γραμμένους από ειδικούς. Ο αλγόριθμος αυτός ελέγχει το μήκος της λέξης και τις καταλήξεις και τις κόβει σε επτά στάδια. Παρακάτω παρουσιάζουμε τους κανόνες που εφαρμόζονται. Οι κανόνες δεν υπάρχουν σε κάποιο είδος τεκμηρίωσης. Εξήχθησαν από μελέτη του πηγαίου κώδικα.

1ο στάδιο: Κόβει τις πιο συχνές καταλήξεις ουσιαστικών. Παίρνει την αρχική λέξη και

- Αν το μήκος της λέξης είναι μεγαλύτερο από 7 και τελειώνει σε "zacja", "zacją", "zacji" τότε κόβει τα τέσσερα τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 6 και τελειώνει σε "acja", "acji", "acją", "tach", "anie", "enie", "eniu", "aniu" τότε κόβει τα τέσσερα τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 6 και τελειώνει σε "tyka" τότε κόβει τα δυο τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 5 και τελειώνει σε "ach", "ami", "nia", "niu", "cia", "ciu" τότε κόβει τα τρία τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 5 και τελειώνει σε "cji", "cja", "cją" τότε κόβει τα δύο τελευταία γράμματα.
- Αλλιώς κόβει το μήκος της λέξης είναι μεγαλύτερο από 5 και τελειώνει σε "ce", "ta" τότε κόβει τα δύο τελευταία γράμματα.
- Αλλιώς δεν κάνει τίποτα και επιστρέφει την λέξη όπως είναι.

2ο στάδιο: Κόβει τις πιο συχνές καταλήξεις υποκοριστικών. Παίρνει ό,τι έβγαλε το 1ο στάδιο και

- Αν το μήκος της λέξης είναι μεγαλύτερο από 6 και τελειώνει σε "eczek", "iczek", "iszek", "aszek", "uszek" τότε κόβει τα πέντε τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 6 και τελειώνει σε "enek", "ejek", "erek" τότε κόβει τα δυο τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 4 και τελειώνει σε "ek", "ak" τότε κόβει τα δυο τελευταία γράμματα.
- Αλλιώς δεν κάνει τίποτα και επιστρέφει την λέξη όπως είναι.

3ο στάδιο: Κόβει τις πιο συχνές καταλήξεις επιθέτων. Παίρνει ό,τι έβγαλε το 2ο στάδιο και

- Αν το μήκος της λέξης είναι μεγαλύτερο από 7 και αρχίζει με "naj" και ταυτόχρονα τελειώνει σε "sze" ή "szy" τότε κόβει τα τρία πρώτα και τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 7 και αρχίζει με "naj" και ταυτόχρονα τελειώνει σε "szych" τότε κόβει τα τρία πρώτα και πέντε τελευταία γράμματα.

- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 6 και τελειώνει σε "czny" τότε κόβει τα τέσσερα τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 5 και τελειώνει σε "owy", "owa", "owe", "ych", "ego" τότε κόβει τα τρία τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 5 και τελειώνει σε "ej" τότε κόβει τα δύο τελευταία γράμματα.
- Αλλιώς δεν κάνει τίποτα και επιστρέφει την λέξη όπως είναι.

4ο στάδιο: Κόβει τις πιο συχνές καταλήξεις ρημάτων. Παίρνει ό,τι έβγαλε το 3ο στάδιο και

- Αν το μήκος της λέξης είναι μεγαλύτερο από 5 και τελειώνει σε "bym" τότε κόβει τα τρία τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 5 και τελειώνει σε "esz", "asz", "cie", "eść", "ać", "łem", "amy", "emy" τότε κόβει τα τρία τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 3 και τελειώνει σε "esz", "asz", "eść", "ać", "eć", "ać" τότε κόβει τα δυο τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 3 και τελειώνει σε "aj" τότε κόβει το τελευταίο γράμμα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 3 και τελειώνει σε "ać", "em", "am", "ał", "ił", "ić", "ąc" τότε κόβει τα δύο τελευταία γράμματα.
- Αλλιώς δεν κάνει τίποτα και επιστρέφει την λέξη όπως είναι.

5ο στάδιο: Κόβει τις πιο συχνές καταλήξεις επιρρημάτων. Παίρνει ό,τι έβγαλε το 4ο στάδιο και

- Αν το μήκος της λέξης είναι μεγαλύτερο από 4 και τελειώνει σε "nie" ή "wie" τότε κόβει τα δυο τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 4 και τελειώνει σε "rze" τότε κόβει τα δυο τελευταία γράμματα.
- Αλλιώς δεν κάνει τίποτα και επιστρέφει την λέξη όπως είναι.

6ο στάδιο: Κόβει τις πιο συχνές καταλήξεις πληθυντικού. Παίρνει ό,τι έβγαλε το 5ο στάδιο και

- Αν το μήκος της λέξης είναι μεγαλύτερο από 4 και τελειώνει σε "ów" ή "om" τότε κόβει τα δυο τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 4 και τελειώνει σε "ami" τότε κόβει τα τρία τελευταία γράμματα.
- Αλλιώς δεν κάνει τίποτα και επιστρέφει την λέξη όπως είναι.

7ο στάδιο: Κόβει κάποιες άλλες, γενικές καταλήξεις. Παίρνει ό,τι έβγαλε το 6ο στάδιο και

- Αν το μήκος της λέξης είναι μεγαλύτερο από 4 και τελειώνει σε "ia" ή "ie" τότε κόβει τα δυο τελευταία γράμματα.
- Αλλιώς αν το μήκος της λέξης είναι μεγαλύτερο από 4 και τελειώνει σε "u", "ą", "i", "a", "ę", "y", "ę", "ł" τότε κόβει το τελευταίο γράμμα.
- Αλλιώς δεν κάνει τίποτα και επιστρέφει την λέξη όπως είναι.

Συνοψίζοντας, στο κεφάλαιο αυτό μελετήσαμε τον Błażej Kubiński Stemmer. Είδαμε πως καλούμε το πρόγραμμα και τις διάφορες επιλογές που μας δίνει, καθώς και μελετήσαμε τους κανόνες που χρησιμοποιεί για την παραγωγή των ριζών.

ΚΕΦΑΛΑΙΟ 5: Υλοποίηση

Στα δυο προηγούμενα κεφάλαια μελετήσαμε τα εργαλεία τα οποία χρησιμοποιήσαμε στο τεχνικό κομμάτι. Στο κεφάλαιο αυτό θα δούμε το πως προσαρμόσαμε τα εργαλεία αυτά, έτσι ώστε να μπορούμε να πάρουμε τα αποτελέσματα των δυο stemmers και να τα συγκρίνουμε μεταξύ τους για να δούμε τελικά πόσο καλά αποτελέσματα παράγουν οι μηχανικά δημιουργημένοι stemmers.

5.1 Προσαρμογή stemming σουίτας

Αρχικά πρέπει να δημιουργήσουμε στο Language Manager της σουίτας την βάση δεδομένων για την γλώσσα μας. Αφού θέλουμε να δημιουργήσουμε stemmers για πολωνικά δημιουργήσαμε την “polish1”.

Εικόνα 5.1: polish1

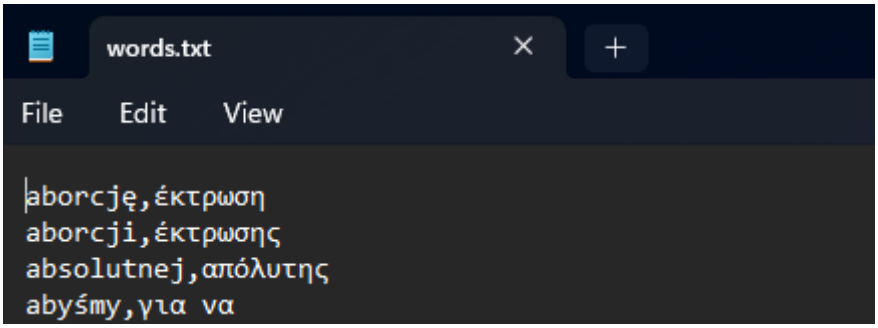
Ύστερα πρέπει να προσθέσουμε λέξεις με μεταφράσεις, καταλήξεις και διπλά γράμματα των πολωνικών. Για τον σκοπό αυτό βασιστήκαμε σε μια παλιά εργασία της

Άννας Μαρίας Βάλτσακ με θέμα “Αλγόριθμοι Εύρεσης Ριζών Λέξεων στην Πολωνική Γλώσσα”. Στο πλαίσιο εκείνης της εργασίας η κυρία Βάλτσακ συγκέντρωσε 669 καταλήξεις της πολωνικής γλώσσας, καθώς και 6087 πολωνικές λέξεις από διάφορες πηγές όπως για παράδειγμα βιογραφίες ή ποιήματα.

Οι λέξεις αρχικά βρισκόντουσαν σε αρχείο τύπου excel σε μορφή “λέξη | ρίζα | κατάληξη | χώρα προέλευσης | μετάφραση | αριθμός επαναλήψεων της λέξης”. Για να μην προσθέτουμε στην σουίτα τις λέξεις μια μια φτιάξαμε ένα απλό script στην γλώσσα python (createWT.py) το οποίο παίρνει από το excel όλες τις γραμμές με τις λέξεις και μεταφράσεις και τις προσθέτει σε αρχείο κειμένου “words.txt” σε μορφή “λέξη, μετάφραση”. Αυτό γιατί έτσι αποθηκεύονται τα ζεύγη όταν τα προσθέτουμε στην σουίτα και επιλέγουμε να τα αποθηκεύσουμε σε αρχείο κειμένου. Με μια αντιγραφή-επικόλληση του αρχείο κειμένου στο κατάλληλο υποφάκελο (C:/stemSuite/polish1/) η σουίτα μπορεί να φορτώσει όλες τις λέξεις. Αφού τις φορτώσουμε στο Add Words του Language Manager μέσω του "Load from File” είναι υποχρεωτικό να αποθηκεύσουμε τα ζεύγη στην βάση μέσω του “Save to DB”.

word	stem	ending	other language	translation	existencens
aborcję	aborc	ję		έκτρωση	1
aborcji	aborcj	i		έκτρωσης	1
absolutnej	absolut	nej		απόλυτης	1
abyśmy	abyś	my		για να	1

Εικόνα 5.2: Μορφή δεδομένων από την Άννα Μαρία Βάλτσακ



Εικόνα 5.3: words.txt

Επειδή από το σύνολο των λέξεων περίπου οι 2600 ήταν χωρίς μετάφραση έπρεπε να τις μεταφράσουμε. Αυτό επειδή οι ειδικοί μας δεν είναι ομιλητές της πολωνικής γλώσσας και χωρίς τις μεταφράσεις τα επιχειρήματά τους δεν θα ήταν αρκετά ακριβή.

Σημειώνουμε ότι στις λέξεις που προέρχονται από ξένες γλώσσες δεν έχουμε προσθέσει μετάφραση.

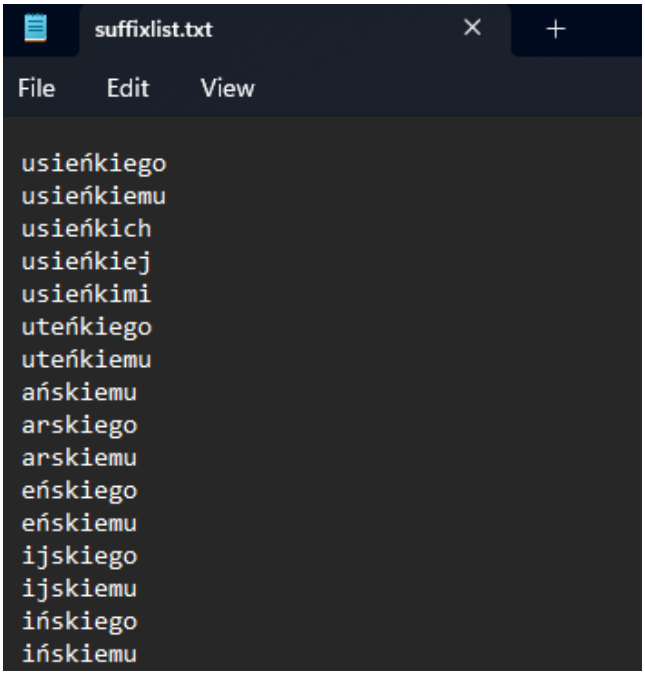
5849	zbawienia	zbawi	enia			3
5850	zbawienie	zbawi	enie			1
5851	zbawieniu	zbawie	niu			1

Εικόνα 5.4: Λέξεις χωρίς μετάφραση

5849	zbawienia	zbawi	enia		σωτηρίας	3
5850	zbawienie	zbawi	enie		σωτηρία	1
5851	zbawieniu	zbawie	niu		σωτηρία (δοτική πτώση)	1

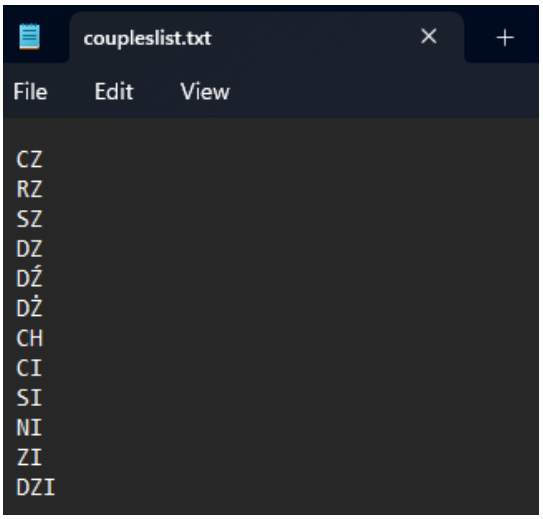
Εικόνα 5.5: Λέξεις με μετάφραση

Στις καταλήξεις δεν έχουμε κάνει καμία αλλαγή, χρησιμοποιήσαμε όλες τις 669 όπως ήταν. Τις αποθηκεύσαμε σε αρχείο κειμένου “suffixlist.txt” και προσθέσαμε στον υποφάκελο C:/stemSuite/polish1/.



Εικόνα 5.6: suffixlist.txt

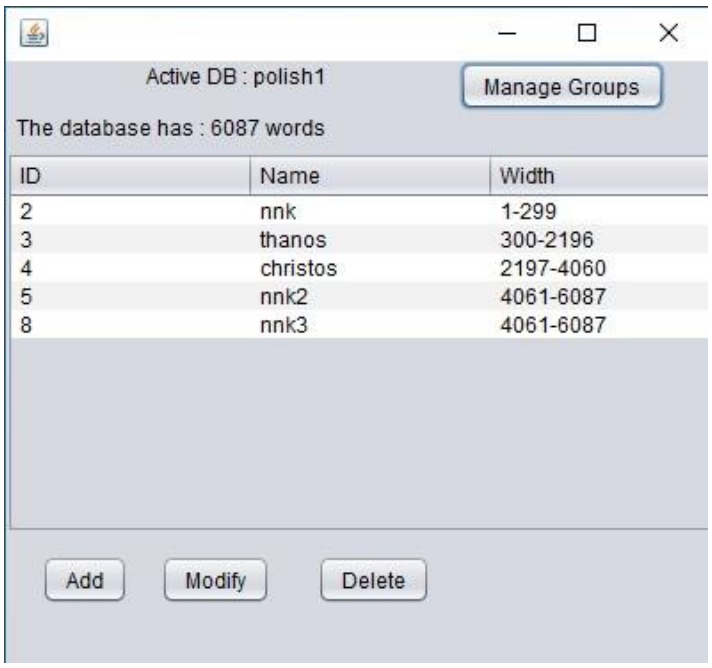
Ως διπλά γράμματα ορίσαμε τα CZ, RZ, SZ, DZ. DŹ, DŻ, CH καθώς και τα CI, SI, NI, ZI. Αυτό γιατί το “I” δίπλα από τα συγκεκριμένα σύμφωνα μαλακώνει την προφορά τους και συνεπώς παράγουν έναν φθόγγο. Για αυτόν τον λόγο επίσης ορίσαμε και το τριπλό γράμμα DZI.



Εικόνα 5.7: coupleslist.txt

Στα πολωνικά δεν έχουμε γράμματα τα οποία χρειάζονται ειδική προσοχή. Για αυτό το Letters on Borders το αφήνουμε άδειο.

Αφού προσθέσουμε όλα τα παραπάνω δεδομένα στην σουίτα είμαστε έτοιμοι να δημιουργήσουμε τα primary stems πάνω στα οποία οι ειδικοί μας θα δώσουν επιχειρήματα. Έχουμε τους παρακάτω ειδικούς:

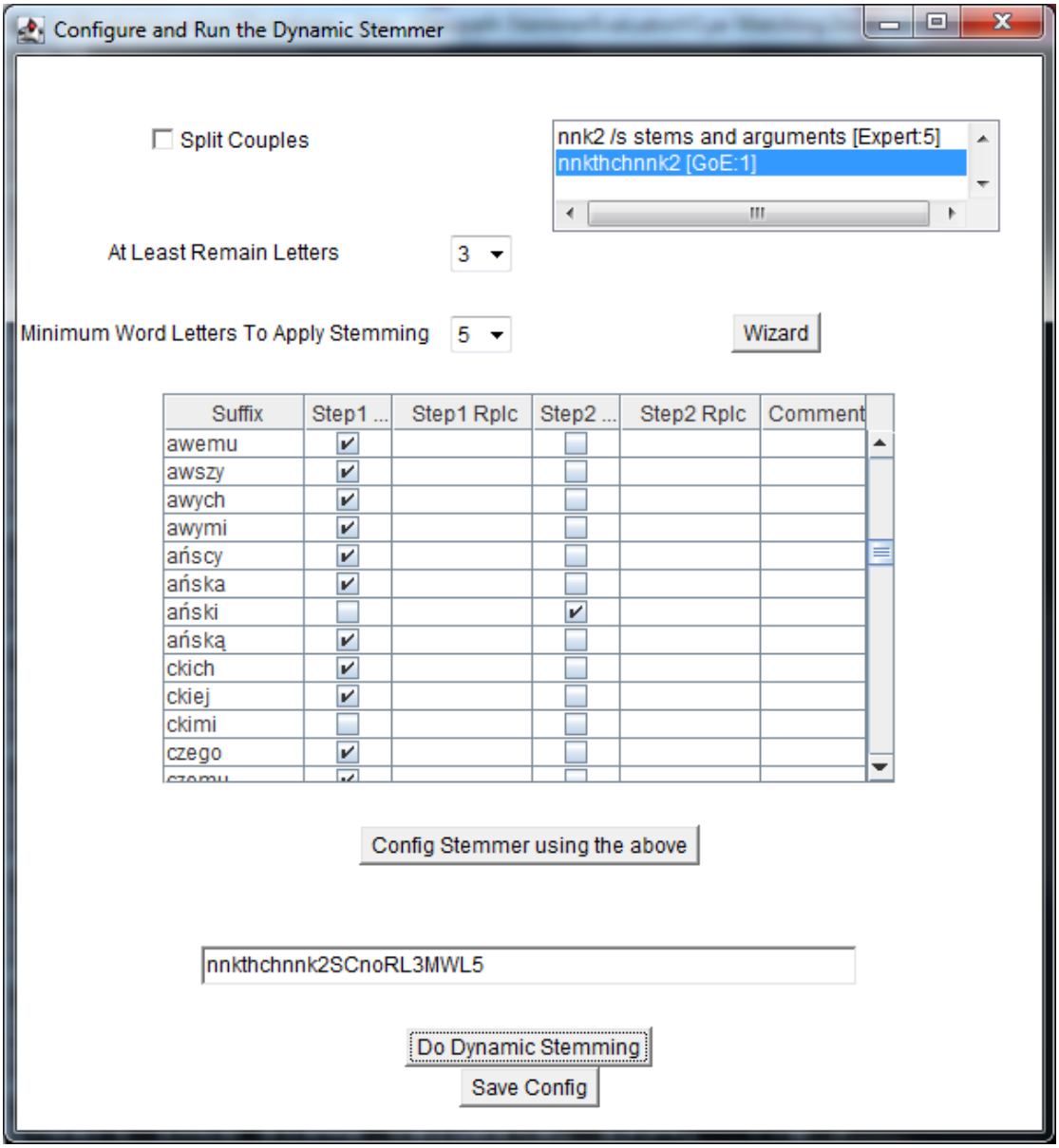


Εικόνα 5.8: Λίστα με τους ειδικούς

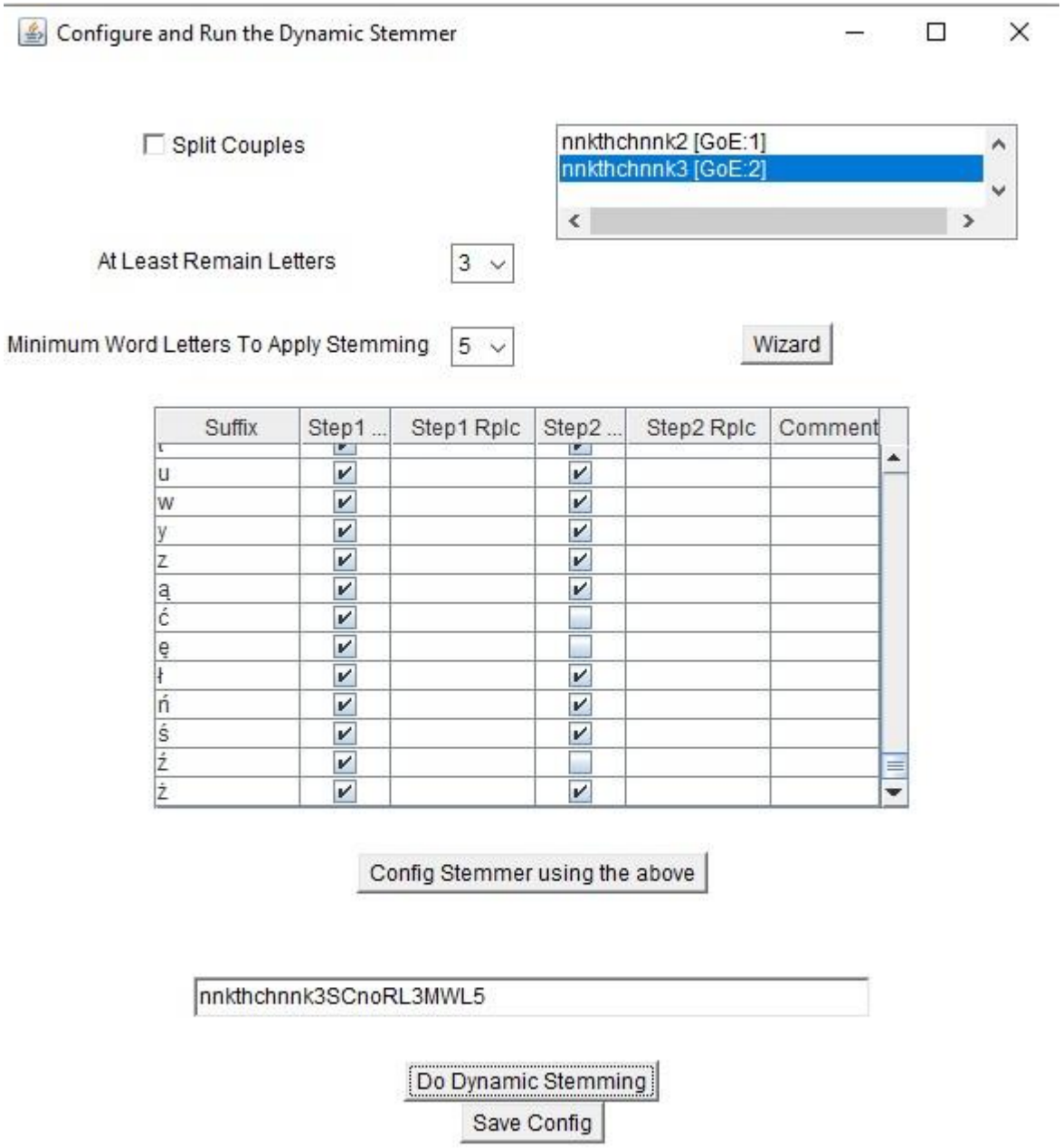
Οι ειδικοί nnk, thanks, christos και nnk3 έδιναν τα επιχειρήματα κανονικά. Δηλαδή όριζαν ομάδες CS, DS και DS/CS και προτείναν είτε δικές τους ρίζες είτε ρίζες που πρότεινε ο primary stemmer. Ο ειδικός nnk2 θεώρησε ότι το σύστημα υποστηρίζει αντικαταστάσεις (replacements). Στα πολωνικά είναι αρκετά συχνό όταν κλίνονται οι λέξεις κάποια γράμματα να μετατρέπονται σε άλλα. Για παράδειγμα το “Ł” μετατρέπεται συχνά στο “L” (SZKOŁA -> SZKOLE) ή το “IĄ” σε “IĘ” (PIENIĄDZE -> PIENIĘDZY). Η λογική της αντικατάστασης είναι να αλλάξουμε ένα κομμάτι κάποιας λέξης αντί να κόβουμε όλο και περισσότερο από αυτήν. Ο λόγος που θα θέλαμε να κάνουμε αυτό είναι ότι όσο μικρότερες είναι οι ρίζες τόσο μεγαλύτερη πιθανότητα να έχουμε over conflation. Δηλαδή σε δυο ομάδες με διαφορετικές κοινές ρίζες, ο stemmer βγάζει την ίδια ρίζα. Στις πάνω λέξεις SZKOŁA και SZKOLE (σημαίνει σχολείο, βρίσκεται σε δυο κλήσεις) χωρίς αντικατάσταση κοινή ρίζα θα ήταν SZKO, ενώ με αντικατάσταση η κοινή ρίζα θα μπορούσε να είναι SZKOŁ ή SZKOL.

Το σύστημα της σουίτας δεν υποστηρίζει προς το παρόν τις αντικαταστάσεις. Όμως ο nnk2 θεώρησε ότι τις υποστηρίζει ώστε να μελετήσουμε τις επιπτώσεις που θα έχει στα αποτελέσματα.

Με τα επιχειρήματα των ειδικών δημιουργήσαμε δυο trial stemmers. Ο ένας από τους trial stemmers δημιουργήθηκε με βάση τα επιχειρήματα των nnk, thanos, christos και nnk2, ενώ ο δεύτερος από τα επιχειρήματα των nnk, thanos, christos και nnk3. Με το “Do Dynamic Stemming” τρέξαμε τους stemmers στις 6087 λέξεις που προσθέσαμε προηγούμενος και οι ρίζες εισάχθηκαν αυτομάτως στην βάση δεδομένων. Στις παρακάτω εικόνες βλέπουμε τις ρυθμίσεις των δυο trial stemmers.



Εικόνα 5.9: Trial stemmer της ομάδας nnkthchnnk2

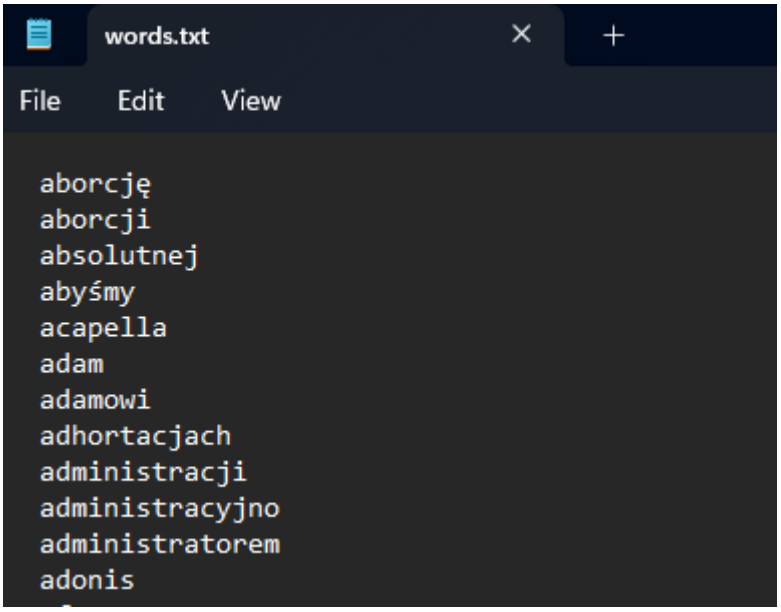


Εικόνα 5.10: Trial stemmer της ομάδας nnkthchnnk3

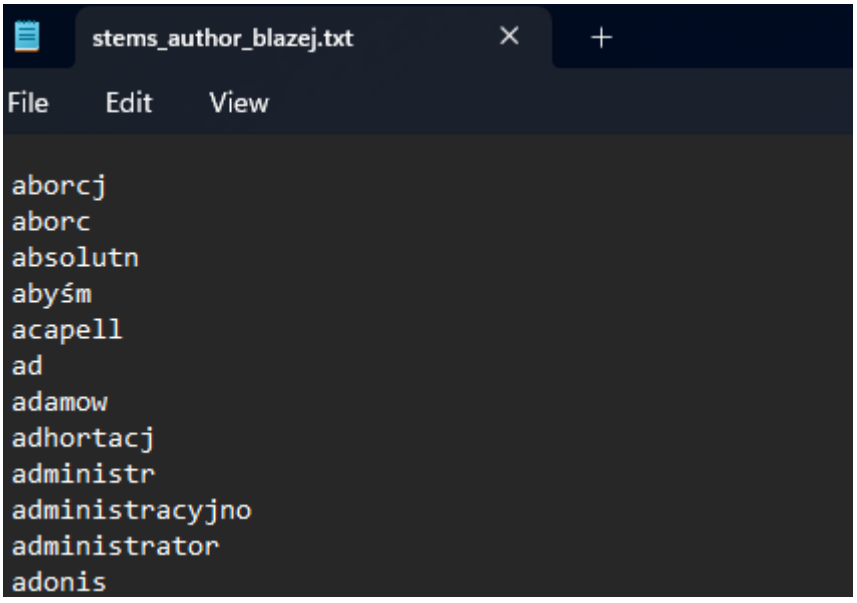
5.2 Προσαρμογή Błażej Kubiński Stemmer

Ο Błażej Kubiński Stemmer ζητάει από τον χρήστη να του γράψει τις λέξεις μέσω της γραμμής εντολών και εμφανίζει την ρίζα από κάτω στην οθόνη. Αυτό δεν είναι αποδοτικό για σύνολο 6087 λέξεων. Για τον λόγο αυτό μετατρέψαμε λίγο την λειτουργικότητα του προγράμματος. Προσαρμόσαμε το πρόγραμμα έτσι ώστε να μπορεί να διαβάζει τις λέξεις από ένα αρχείο κειμένου (words.txt) και τα αποτελέσματα

να τα αποθηκεύει επίσης σε αρχείο κειμένου (stems_author_blazej.txt). Το αρχείο “words.txt” πρέπει να διαθέτει μια λέξη ανά γραμμή ώστε να δουλεύει σωστά το πρόγραμμα. Μπορούμε να το φτιάξουμε απλά για σύνολο λέξεων μας κάνοντας μικρές αλλαγές στο createWT.py, το οποίο αντί να αποθηκεύει σε αρχείο κειμένου “λέξη, μετάφραση”, να αποθηκεύει μόνο “λέξη”. Το αρχείο εξόδου “stems_author_blazej.txt” χωρίζει την κάθε ρίζα με νέα γραμμή.

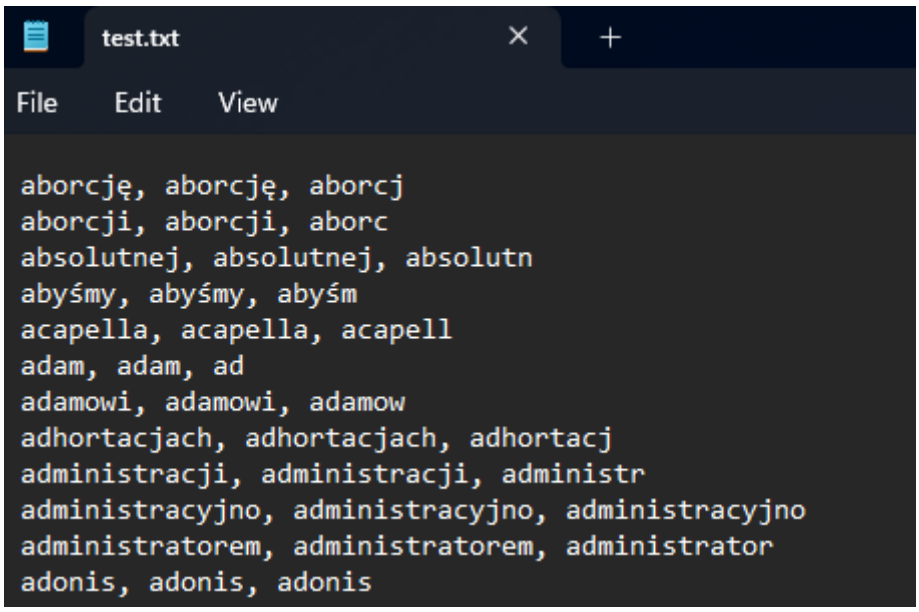


Εικόνα 5.11: words.txt

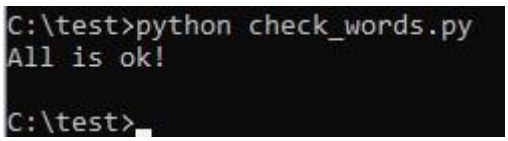


Εικόνα 5.12: stems_author_blazej.txt

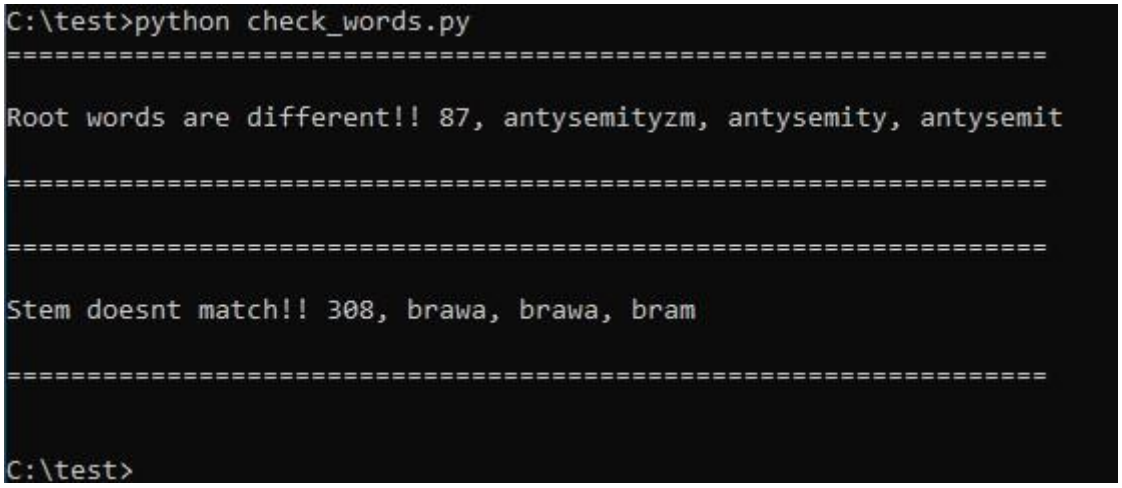
Για να είμαστε σίγουροι ότι έγιναν όλα σωστά και δεν μας ξέφυγε κάτι δημιουργήσαμε ένα python script (check_words.py) με το οποίο κάνουμε έλεγχο. Ελέγχουμε αν οι λέξεις είναι αποθηκευμένες με την ίδια σειρά στην βάση δεδομένων και στο αρχείο words.txt, καθώς και εάν η ρίζα του Błażej Kubiński Stemmer περιέχεται στην αρχική λέξη. Αποθηκεύουμε επίσης τον συνδυασμό “λέξη από βάση, λέξη από αρχείο, ρίζα” μέσα σε αρχείο κειμένου “text.txt” για την περίπτωση που θα θέλαμε να συγκρίνουμε τα αποτελέσματα με τα μάτια μας. Αν κάποια από τις δυο συνθήκες δεν είναι αληθής εμφανίζεται κατάλληλο μήνυμα και στην οθόνη και στο αρχείο κειμένου. Ο έλεγχος αυτός είναι πολύ σημαντικός διότι όταν συγκρίνουμε τα αποτελέσματα μέσω της σουίτας, οι λέξεις με τις ρίζες πρέπει να βρίσκονται στις ίδιες θέσεις. Στην περίπτωση που δεν είναι τα αποτελέσματα που θα παράγει η σουίτα θα είναι λανθασμένα.



Εικόνα 5.13: test.txt



Εικόνα 5.14: check_words.py: Όλα σωστά

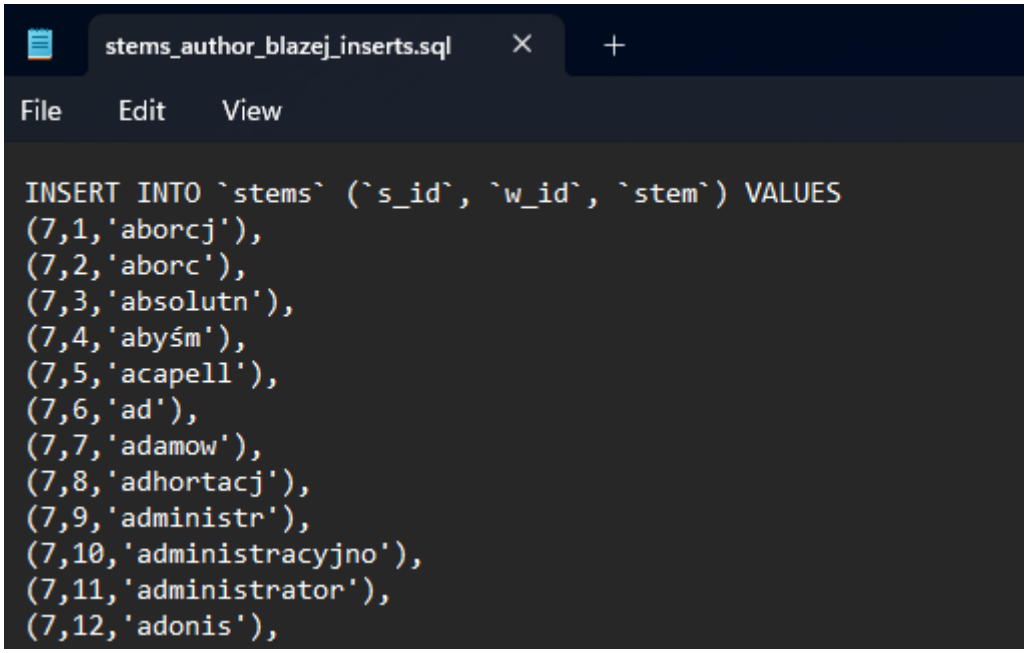


Εικόνα 5.15: check_words.py: Εντοπισμένα λάθη

Αφού φτιάξαμε τις ρίζες με τον Błażej Kubiński Stemmer και ελέγξαμε ότι αντιστοιχούν με τις λέξεις της βάσης δεδομένων με την ίδια ακριβώς σειρά, πρέπει να προσθέσουμε και αυτές (τις ρίζες) μέσα στην βάση. Επειδή δεν έχουμε την δυνατότητα να προσθέσουμε stemmer μέσω της σουίτας (μπορούμε να προσθέτουμε μόνο ειδικούς μέσω διεπαφής), πρέπει να εκτελέσουμε εντολή MySQL μέσα σε κάποιο περιβάλλον (είτε MySQL, είτε phpMyAdmin). Η εντολή που πρέπει να εκτελέσουμε είναι η εξής:

```
INSERT INTO `sources` (`id`, `name`, `description`, `password`, `width`, `type`)
VALUES
(7, NULL, 'AuthorBlazejKubinski', NULL, NULL, 'STEMMER');
```

Με την καταχώρηση στην βάση της σουίτας του Błażej Kubiński Stemmer ως 'STEMMER', μπορούμε να προσθέσουμε και τις ρίζες που παρήγαγε μέσα στην βάση. Για τον σκοπό αυτό δημιουργήσαμε python script (create_inserts.py) το οποίο φτιάχνει αρχείο .sql με την εντολή INSERT και όλες τις ρίζες από το “stems_author_blazej.txt”. Είναι σημαντικό το “s_id” να είναι το ίδιο με το “id” που δώσαμε στον Błażej Kubiński Stemmer μέσα στην βάση.



Εικόνα 5.16: stems_author_blazej_inserts.sql

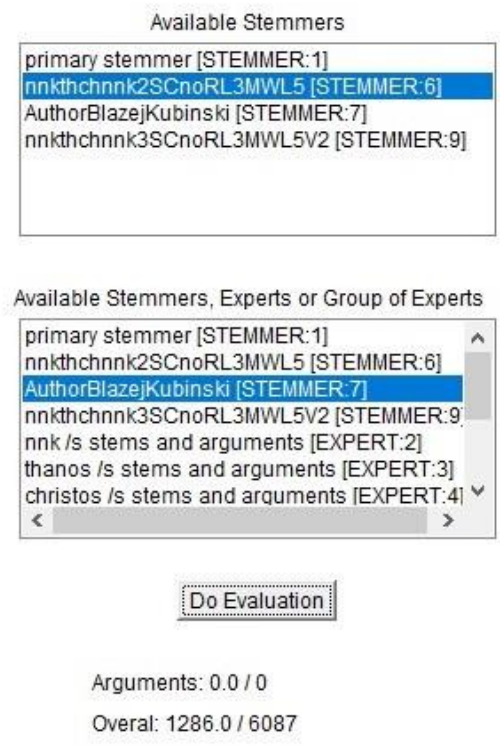
Με τις ρίζες των trial stemmers και του Błażej Kubiński Stemmer μέσα στην βάση της σουίτας μπορούμε να προχωρήσουμε στους ελέγχους και τις συγκρίσεις.

5.3 Αποτελέσματα EvaluatorUI

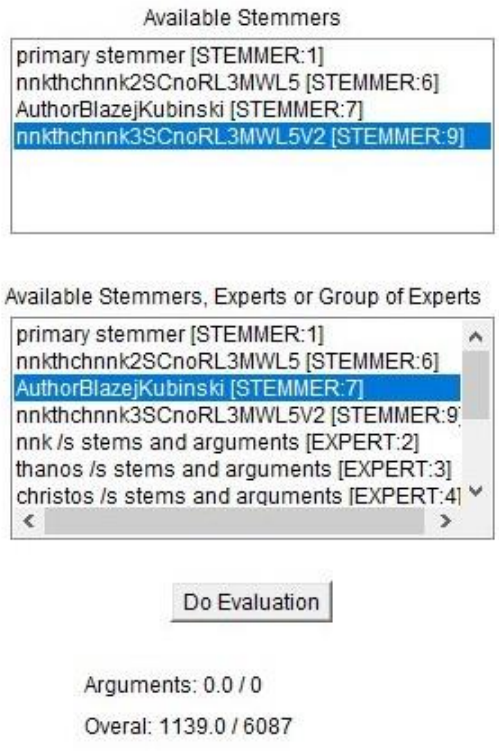
Αρχικά για τους ελέγχους χρησιμοποιήσαμε το EvaluatorUI της σουίτας. Μέσω του λογισμικού αυτού συγκρίναμε τον βαθμό ομοιότητας μεταξύ των τριών stemmers και των επιχειρημάτων των ειδικών. Ελέγξαμε επίσης το πόσο όμοιες είναι οι ρίζες των μηχανικά δημιουργημένων stemmers με τον stemmer που βασίζεται σε κανόνες. Τα αποτελέσματα τα παρουσιάζουμε στον παρακάτω πίνακα. Σημειώνουμε πως όπου γράφουμε “GOE1” εννοούμε την ομάδα ειδικών nnk, thanos, christos, nnk2 ενώ σαν “GOE2” αναφερόμαστε στην ομάδα nnk, thanos, christos, nnk3.

	Επιχειρήματα	Σύνολο	Ποσοστό (%)
NNK STEMMER ME AUTHOR BŁAŻEJ STEMMER	0.0 / 0	1286.0 / 6087	21.13
NNK STEMMER ME ΜΕΙΩΜΕΝΕΣ ΑΝΤΙΚΑΤΑΣΤΑΣΕΙΣ (MA) ΜΕ AUTHOR BŁAŻEJ STEMMER	0.0 / 0	1139.0 / 6087	18.71
NNK STEMMER ME GOE1	2529.7769 / 4268	4252.777 / 6087	69.87
NNK STEMMER ME GOE2	2551.7769 / 4278	4268.027 / 6087	70.12
NNK STEMMER ME ΜΕΙΩΜΕΝΕΣ ΑΝΤΙΚΑΤΑΣΤΑΣΕΙΣ ΜΕ GOE1	2528.9373 / 4268	4247.6875 / 6087	69.78
NNK STEMMER ME ΜΕΙΩΜΕΝΕΣ ΑΝΤΙΚΑΤΑΣΤΑΣΕΙΣ ΜΕ GOE2	2548.9373 / 4278	4260.9375 / 6087	70.00
AUTHOR BŁAŻEJ STEMMER ME GOE1	2068.16 / 4268	3849.41 / 6087	63.24
AUTHOR BŁAŻEJ STEMMER ME GOE2	2081.1602 / 4278	3853.4102 / 6087	63.31

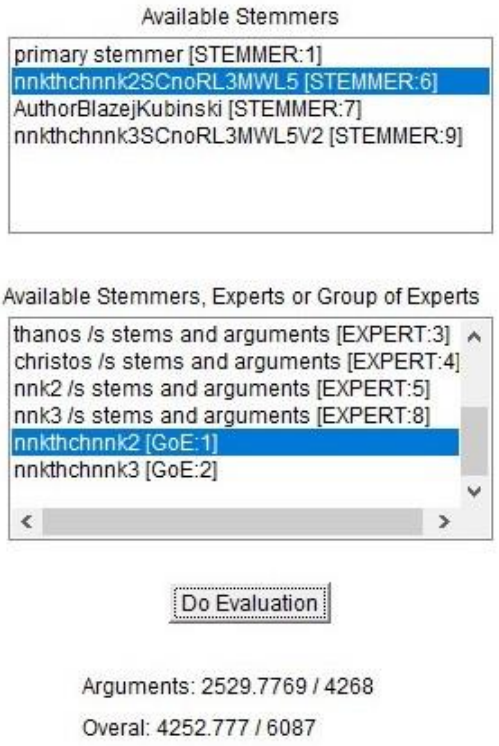
Πίνακας 5.1: Αποτελέσματα συγκρίσεων από το EvaluatorUI



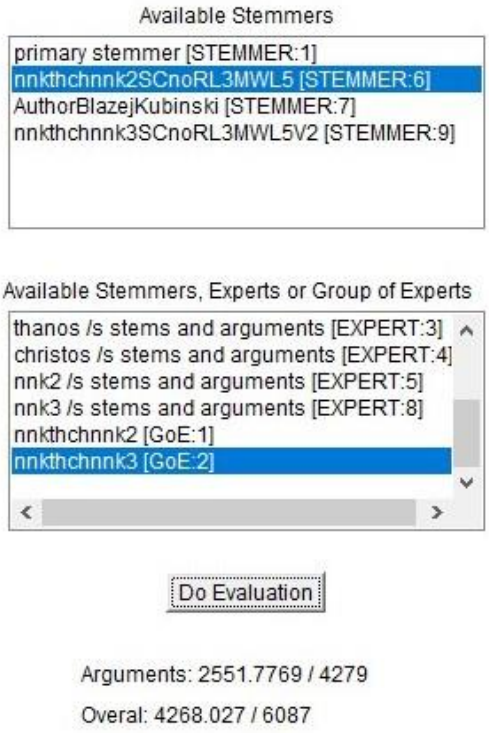
Εικόνα 5.17: NNK ΜΕ BŁAŻEJ



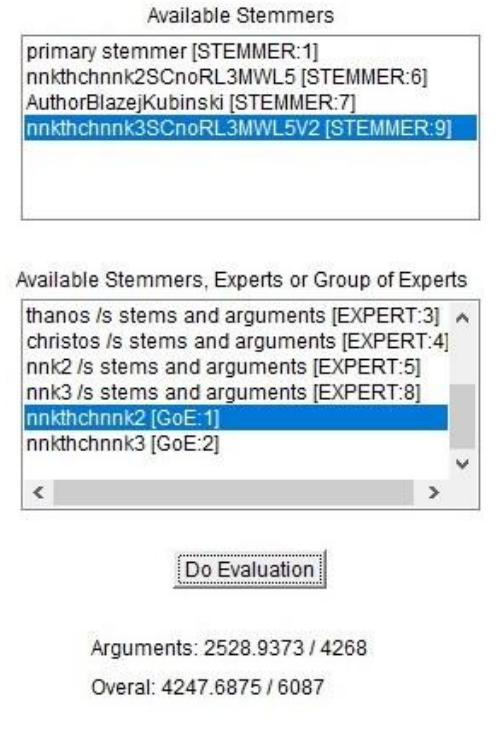
Εικόνα 5.18: NNKMA ΜΕ BŁAŻEJ



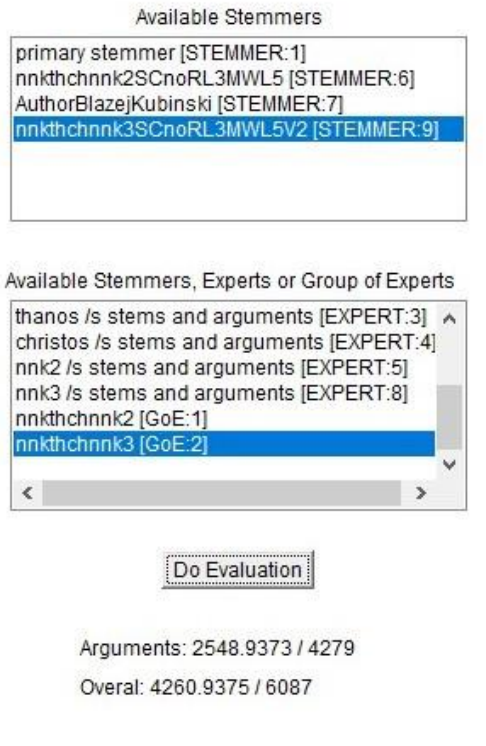
Εικόνα 5.19: NNK ME GOE1



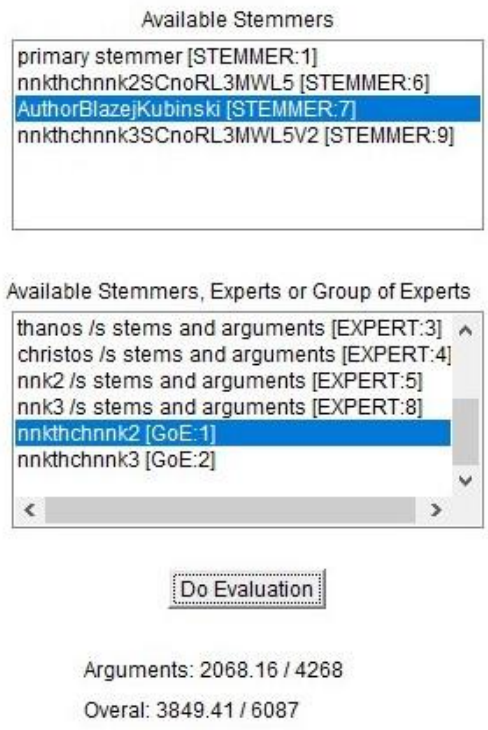
Εικόνα 5.20: NNK ME GOE2



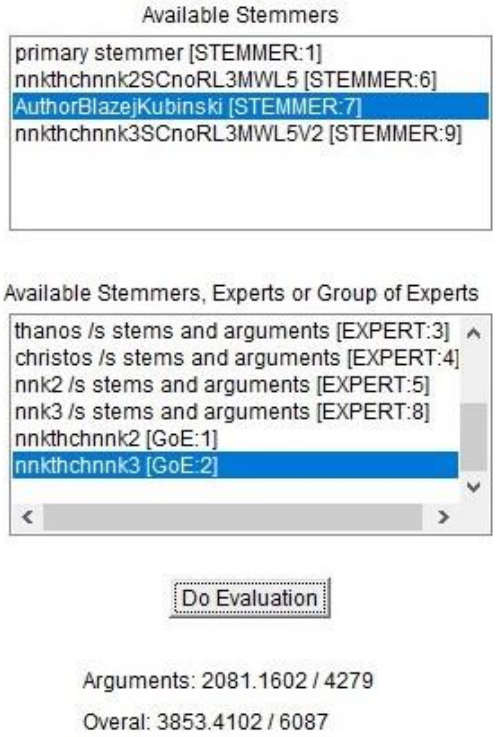
Εικόνα 5.21: NNKMA ME GOE1



Εικόνα 5.22: NNKMA ME GOE2



Εικόνα 5.23: BŁAŻEJ ME GOE1



Εικόνα 5.24: BŁAŻEJ ME GOE2

Από τα παραπάνω αποτελέσματα παρατηρούμε ότι οι μηχανικά δημιουργημένοι stemmers έχουν μεγαλύτερο βαθμό ομοιότητας με τα επιχειρήματα των ειδικών σε σχέση με τον stemmer που βασίζεται σε κανόνες. Αυτό όμως είναι ένα αναμενόμενο φαινόμενο αφού οι μηχανικά δημιουργημένοι stemmers βασίζονται πάνω στα επιχειρήματα των ειδικών και οι κανόνες που δημιουργούνται προσπαθούν να προσαρμοστούν όσο το πιο δυνατόν κοντά σε αυτά. Βάση αυτό περιμέναμε ο Błażej Kubiński Stemmer να έχει χειρότερα αποτελέσματα διότι οι κανόνες του είναι διαφορετικοί και συνεπώς και οι ρίζες στις οποίες καταλήγει μπορεί να είναι διαφορετικές από αυτές που πρότειναν οι ειδικοί.

Ένα ενδιαφέρον φαινόμενο που παρατηρούμε στα αποτελέσματα είναι ότι ο μηχανικός stemmer που δημιουργήθηκε από τα επιχειρήματα της ομάδας στην οποία ο ένας ειδικός θεώρησε ότι υπάρχουν αντικαταστάσεις προσαρμόζεται περισσότερο στα επιχειρήματα των ειδικών σε σύγκριση με τον άλλον μηχανικά δημιουργημένο stemmer. Περιμέναμε ότι ο stemmer ο οποίος βασίζεται στα επιχειρήματα της GOE2 θα βγάλει ελάχιστα καλύτερα αποτελέσματα αφού οι αντικαταστάσεις στην παρούσα

υλοποίηση της σουίτας δεν υποστηρίζονται. Και συνεπώς τα επιχειρήματα του nnk2 θα έπρεπε να έχουν μικρότερη ανταπόκριση στο σύστημα της σουίτας. Τα αποτελέσματα αυτά προέκυψαν καθαρά από σύμπτωση λόγο της φύσης της μηχανικής μάθησης.

Τέλος οι ρίζες των μηχανικά δημιουργημένων stemmers έχουν βαθμό ομοιότητας περίπου 20% με τις ρίζες του Błażej Kubiński Stemmer. Το αποτέλεσμα αυτό ήταν αναμενόμενο αφού όπως και αναφέραμε προηγουμένος οι κανόνες μεταξύ των stemmers είναι διαφορετικοί και συνεπώς οι ρίζες που παράγονται είναι διαφορετικές.

5.4 Έλεγχος από ομιλητή της πολωνικής γλώσσας

Το εργαλείο EvaluatorUI είναι αρκετά καλό εργαλείο για τον σκοπό της αξιολόγησης των stemmers. Όμως επειδή οι ρίζες των stemmers συγκρίνονται με τα επιχειρήματα των ειδικών τα τελικά αποτελέσματα είναι πιθανός διαστρεβλωμένα υπέρ των μηχανικά δημιουργημένων stemmers. Αφού αυτοί προσπαθούν να προσαρμοστούν όσο το περισσότερο δυνατόν στα επιχειρήματα των ειδικών. Για τον λόγο αυτόν έπρεπε να γίνει και σύγκριση των ριζών από ομιλητή της πολωνικής γλώσσας.

Αρχικά για την ευκολία μας προσθέσαμε όλες τις αρχικές λέξεις, τις ρίζες ενός μηχανικά δημιουργημένου stemmer (επιλέξαμε αυτόν που βασιζόταν σε επιχειρήματα της GOE1) και τις ρίζες του Błażej Kubiński Stemmer σε ένα αρχείο excel.

Orginal Words	NNK Stemmer	Błażej Stemmer
aborcję	abor	aborcj
aborcji	aborc	aborc
absolutnej	absolu	absolutn
abyśmy	abyś	abyśm
acapella	acapel	acapell
adam	adam	ad
adamowi	ada	adamow
adhortacjach	adhortac	adhortacj
administracji	administrac	administr
administracyjno	administracyj	administracyjno
administratorem	administrator	administrator
adonis	ado	adonis

Εικόνα 5.25: Παράλληλη παράθεση αποτελεσμάτων

Ύστερα υπολογίσαμε τις περιπτώσεις των over conflation και under conflation που έκανε ο καθένας από τους δυο stemmer. Τις μετρήσεις τις κάναμε και στο επίπεδο των ομάδων αλλά και στο επίπεδο των λέξεων. Δηλαδή μετρήσαμε τον αριθμό των ομάδων που παρουσίασαν λάθη καθώς και τον αριθμό των λέξεων που κατηγοριοποιήθηκαν λάθος. Στην παρακάτω εικόνα παρουσιάζουμε παράδειγμα σωστής ομαδοποίησης και από τους δυο stemmers.

Orginal Words	NNK Stemmer	Błażej Stemmer
doktryn	doktr	doktryn
doktryna	doktr	doktryn
doktryną	doktr	doktryn
doktryny	doktr	doktryn

Εικόνα 5.26: Σωστή ομαδοποίηση από τους δυο stemmers

Under conflation έχουμε όταν σε μια ομάδα λέξεων με κοινή ρίζα, μια ή περισσότερες ρίζες που παράγει ο stemmer είναι διαφορετικές. Για παράδειγμα στην παρακάτω εικόνα όλες οι λέξεις προέρχονται από την λέξη “filozofia” (φιλοσοφία) και συνεπώς θα έπρεπε να έχουν μια κοινή ρίζα. Όμως ο ένας stemmer σε μια λέξη έβγαλε μια διαφορετική ρίζα, ενώ ο δεύτερος έβγαλε διαφορετική ρίζα σε τέσσερεις λέξεις. Συνεπώς παρουσιάζεται φαινόμενο under conflation σε αυτές τις λέξεις.

Orginal Words	NNK Stemmer	Błażej Stemmer
filozofią	filozof	filozofi
filozoficzna	filozof	filozoficzn
filozoficzny	filozoficz	filozof
filozoficznych	filozof	filozoficzn
filozoficznym	filozof	filozoficznym
filozofii	filozof	filozofi

Εικόνα 5.27: Παράδειγμα Under Conflation

Over Conflation έχουμε όταν σε δυο ομάδες με διαφορετικές κοινές ρίζες, ο stemmer παράγει μια κοινή ρίζα. Για παράδειγμα στην παρακάτω εικόνα παρουσιάζουμε τα αποτελέσματα για τις λέξεις “głośny” και “głową”. Ο μηχανικά δημιουργημένος stemmer προτείνει για τις δυο λέξεις μια κοινή ρίζα το “gło”. Αυτό είναι λάθος επειδή η λέξη “głośny” σημαίνει “δυνατός ήχος” ενώ η λέξη “głową” σημαίνει “κεφάλι”. Οι λέξεις

αυτές δεν έχουν καμία σημασιολογική σχέση μεταξύ τους και συνεπώς θα έπρεπε να έχουν δυο διαφορετικές ρίζες.

Orginal Words	NNK Stemmer	Błażej Stemmer
głośny	gło	głośn
głową	gło	głow

Εικόνα 5.28: Παράδειγμα Over Conflation

Στις περιπτώσεις διαφωνίας των δυο stemmers για την ομαδοποίηση, μετρήσαμε επίσης σε πόσες περιπτώσεις είναι σωστότερη η έξοδος του μηχανικά δημιουργημένου stemmer και σε πόσες περιπτώσεις είναι σωστότερη η έξοδος του Błażej Kubiński Stemmer. Επειδή στις περισσότερες περιπτώσεις είναι δύσκολο να ορίσουμε την σωστότερη ρίζα, μετρήσαμε τις περιπτώσεις όπου θα είχαμε under conflation ή over conflation εάν στο σύνολο δεδομένων μας πιθανά εμφανιζόταν λέξεις με ίδια ρίζα.

Στο παρακάτω παράδειγμα βλέπουμε την έξοδο των δυο stemmers για την λέξη “gwizdnął”. Το σύνολο δεδομένων μας δεν έχει άλλες λέξεις με ίδια κοινή ρίζα. Για αυτόν τον λόγο στο πλαίσιο των μετρήσεών μας και οι δυο ρίζες είναι σωστές. Όμως η λέξη “gwizdnął” προέρχεται από την λέξη “gwizdać” και εάν υπήρχε η λέξη αυτή στο σύνολο δεδομένων μας οτιδήποτε και να έβγαζε ο Błażej Kubiński Stemmer θα είχαμε περίπτωση under conflation. Επομένως ο μηχανικά δημιουργημένος stemmer σε αυτήν την περίπτωση είναι ο σωστότερος.

Orginal Words	NNK Stemmer	Błażej Stemmer
gwizdnął	gwizd	gwizdną

Εικόνα 5.29: Παράδειγμα καλύτερης ρίζας

Παρακάτω παρουσιάζουμε τα αποτελέσματα των μετρήσεών μας.

	NNK ομάδες	NNK λέξεις	Błażej ομάδες	Błażej λέξεις
Under Conflate	747	1329	898	1869
Over Conflate	113	194	12	14
Καλύτερη ρίζα	178		62	

Πίνακας 5.2: Αποτελέσματα συγκρίσεων από ομιλητή πολωνικής γλώσσας

Όπως βλέπουμε ο μηχανικά δημιουργημένος stemmer έβγαλε καλύτερα αποτελέσματα ως προς το under conflation και την καλύτερη ομαδοποίηση. Ένας από τους λόγους για αυτό είναι ότι προτιμάει να κόβει μεγαλύτερες καταλήξεις και συνεπώς να δημιουργεί μικρότερες ρίζες. Εξαιτίας αυτού όμως, είναι χειρότερος στις περιπτώσεις του over conflation αφού όσο μικρότερη η ρίζα τόσο περισσότερη πιθανότητα να ομαδοποιηθούν λάθος οι λέξεις.

Παρατηρούμε ότι στο under conflation ο μηχανικά δημιουργημένος stemmer ομαδοποιεί λάθος κατά μέσο όρο περίπου 1.78 λέξεις ανά ομάδα και στο over conflation περίπου 1.72 λέξεις ανά ομάδα. Σε σύγκριση με τον Błażej Kubiński Stemmer ο οποίος στο under conflation ομαδοποιεί λάθος περίπου 2.08 λέξεις ανά ομάδα ενώ στο over conflation περίπου 1.17 λάθος λέξεις ανά ομάδα.

Ένα συχνό φαινόμενο στα αποτελέσματα του μηχανικά δημιουργημένου stemmer είναι ότι σε πολλές περιπτώσεις του under conflation η λέξη που δεν κατηγοριοποιούταν σωστά διέφερε για ένα φωνήεν από την υπόλοιπη ομάδα και τα κυρίως φωνήεντα ήταν “a”, “i”, “ó”. Συνεπώς με λίγο διαφορετική παραμετροποίηση ίσως ο stemmer αυτός είχε καλύτερα αποτελέσματα.

Orginal Words	NNK Stemmer	Błażej Stemmer
gnębiony	gnęb	gnębion
gnębionych	gnębi	gnębion
Orginal Words	**NNK Stemmer**	**Błażej Stemmer**
krokiem	krok	krok
kroków	krokó	krok

Εικόνα 5.29: Παραδείγματα under conflation με διαφορά ένα φωνήεν

Σε αυτό το κεφάλαιο παρουσιάσαμε το τεχνικό κομμάτι της εργασίας μας. Είδαμε το πως προσαρμόσαμε τους δυο stemmers στις ανάγκες μας. Ύστερα αφού λάβαμε τις εξόδους των δυο stemmers συγκρίναμε τα αποτελέσματα μεταξύ τους με δυο τρόπους. Ο πρώτος ήταν με το έτοιμο εργαλείο της σουίτας το οποίο συγκρίνει τις εξόδους με τα επιχειρήματα των ειδικών, ενώ ο δεύτερος τρόπος σύγκρισης ήταν από ομιλητή της γλώσσας. Στο παρακάτω κεφάλαιο παρουσιάζουμε τα συμπεράσματά μας.

ΚΕΦΑΛΑΙΟ 6: ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα πολωνικά είναι μια δύσκολη γλώσσα για την δημιουργία stemmers. Ένας από τους λόγους για αυτό είναι η πολυπλοκότητα στις κλίσεις, όπως είδαμε ένα ρήμα μπορεί να έχει πενήντα κλίσεις. Συνεπώς είναι πολύ δύσκολο να δημιουργηθεί ένας stemmer για την γλώσσα αυτή είτε μηχανικά, είτε με κανόνες γραμμένους από ειδικό.

Τα αποτελέσματα των μηχανικά δημιουργημένων stemmers ήταν αρκετά κοντά σε αυτά του stemmer με κανόνες γραμμένους με το χέρι. Σε κάποιες περιπτώσεις ήταν καλύτερα (under conflation, καλύτερες ρίζες, πιο κοντά στα επιχειρήματα ειδικών), ενώ σε κάποιες άλλες χειροτέρα (over conflation).

Στα δεδομένα μας υπήρχαν περίπου 1200 σετ επιχειρημάτων των ειδικών. Έχοντας υπόψη ότι στα αποτελέσματα διαπιστώσαμε 747 περιπτώσεις under conflation και 114 over conflation, παρατηρούμε πως οι μηχανικά δημιουργημένοι stemmers κατηγοριοποίησαν σωστά περίπου το 38% όλων των ομάδων σωστά ως προς το under conflation και περίπου το 90% ως προς το over conflation.

Βάση των παραπάνω νούμερων μπορούμε να πούμε ότι οι μηχανικά δημιουργημένοι stemmers έχουν αρκετό περιθώριο βελτίωσης. Την βελτίωση αποτελεσμάτων πιθανά θα μπορούσαμε να πετύχουμε είτε με διαφορετική παραμετροποίηση, είτε με υλοποίηση αντικαταστάσεων.

Στο προηγούμενο κεφάλαιο αναφέραμε ότι σε αρκετές περιπτώσεις του under conflation ο μηχανικά δημιουργημένος stemmer “χάνει” ομάδες λόγω κάποιων συγκεκριμένων φωνήεντων. Θα μπορούσαμε χειροποίητα να ενεργοποιήσουμε (σε οποιοδήποτε στάδιο) στην διεπαφή ρύθμισης stemmers, κάποιες καταλήξεις που διαθέτουν αυτά τα φωνήεντα σαν πρώτο γράμμα. Επίσης θα μπορούσαμε να ενεργοποιήσουμε τα ίδια τα φωνήεντα να αφαιρούνται στο δεύτερο στάδιο του stemming.

Σημειώσαμε επίσης ότι ο μεγάλος αριθμός του over conflation μπορεί να οφείλεται στο μικρό μέγεθος των ριζών που παράγει. Η σουίτα μας δίνει την επιλογή να ορίσουμε το

ελάχιστο μέγεθος της ρίζας που παράγει ο stemmer. Πιθανά μια λύση για αυτό το πρόβλημα θα μπορούσε να είναι η αύξηση του ελάχιστου μήκους της ρίζας από τρείς χαρακτήρες που είχαμε ορίσει σε τέσσερεις.

Μια άλλη πρόταση για την βελτίωση των αποτελεσμάτων είναι η υλοποίηση στο σύστημα της σουίτας των αντικαταστάσεων. Στα πολωνικά είναι συχνό φαινόμενο όταν κλίνονται οι λέξεις, κάποια γράμματα να μετατρέπονται σε κάποια άλλα, για παράδειγμα το “l” σε “ł” (szkoła -> szkole, gardło -> gardle). Στο σύστημα όπως είναι τώρα, κατά την εύρεση της κοινής ρίζας πρέπει να κόβουμε αυτά τα γράμματα αφού αλλιώς θα είχαμε φαινόμενο under conflation. Αυτό όμως κάνει τις ρίζες που παράγονται να είναι μικρότερες σε μέγεθος, το οποίο τις προκαλεί να είναι πιο ευάλωτες στο φαινόμενο του over conflation. Με την αντικατάσταση των καταλήξεων είτε των αρχικών λέξεων, είτε των ριζών που προκύπτουν θα είχαμε σαν αποτέλεσμα μεγαλύτερες τελικές ρίζες.

Μια πιθανή ιδέα για την υλοποίηση των αντικαταστάσεων είναι η υλοποίηση ενός δεύτερου wizard, ο οποίος ελέγχει τα επιχειρήματα των ειδικών και τις ρίζες που προτείνουν. Αφού κάνει τον έλεγχο, εμφανίζει στον χρήστη όλες τις αντικαταστάσεις που πρότειναν οι ειδικοί και δίνει στον χρήστη την επιλογή να ενεργοποιήσει ή απενεργοποιήσει όποιες επιθυμεί. Ύστερα ο wizard με τις αντικαταστείς θα πρέπει να επικοινωνήσει με τον wizard στον οποίο ρυθμίζονται οι stemmers και να περάσει όλες τις επιλογές του χρήστη.

Είναι ενδεχόμενο οι αντικαταστάσεις να πρέπει να εφαρμόζονται στο δεύτερο στάδιο του stemming, διότι οι ειδικοί επιχειρηματολογούν πάνω στις εξόδους του primary stemmer. Αλλιώς θα πρέπει να γίνουν στο πρώτο στάδιο, πάνω στην αρχική λέξη, με την αφαίρεση της ρίζας να γίνει στο δεύτερο στάδιο.

Συμπεραίνοντας η δημιουργία μηχανικών stemmer μέσω της σουίτας είναι μια αρκετά καλή δυνατότητα για το χώρο του Information Retrieval. Αυτό το συμπέρασμα επαληθεύτηκε στην πολωνική γλώσσα. Από τα επιχειρήματα μη ομιλητών της πολωνικής γλώσσας δημιουργήσαμε stemmer οι οποίοι παράγουν συγκρίσιμα αποτελέσματα με stemmer φτιαγμένο από γνώστη της γλώσσας. Ωστόσο τα τελικά

αποτελέσματα δεν είναι τα καλύτερα και οι stemmer και η σουίτα έχουν ένα αρκετά μεγάλο περιθώριο βελτίωσης.

ΚΕΦΑΛΑΙΟ 7: ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Nikitas N. Karanikolas, A methodology for building simple but robust stemmers without language knowledge: Overview, data model and ranking algorithm. CompSysTech'2013: 14th International Conference on Computer Systems and Technologies, June 2013, Ruse, Bulgaria. ACM ICPS, doi:10.1145/2516775.2516783.

[2] Nikitas N. Karanikolas. Supervised learning for building stemmers. Journal of Information Science, Vol. 41 (3), pp. 315-328, 2015, doi:10.1177/0165551515572528.

[3] Nikitas N. Karanikolas. A methodology for building simple but robust stemmers without language knowledge: Stemmer configuration. Procedia, Social and Behavioral Sciences, vol. 147, pp. 370-375, doi:10.1016/j.sbspro.2014.07.113.

[4] Nikitas N. Karanikolas. Stemmer Builder suite Manual, Belgrade, January 2020 http://users.uniwa.gr/nnk/more/StemSuite/Stemmer_Builder_suite_Manual_Belgrade_January_2020.pdf

[5] Anna Maria Walczak. Αλγόριθμοι εύρεσης ριζών λέξεων στην πολωνική γλώσσα, Διπλωματική εργασία ΤΕΙ Αθήνας, 2016

[6] Błażej Kubiński, Polish stemmer, https://github.com/Tutanchamon/pl_stemmer

[7] Anna Kubica, Fonetyka, https://aniakubica.com/o-slowach/fonetyka/

[8] Maciołek Marcin, Tęczowa gramatyka języka polskiego w tabelach. Część I: Deklinacja. Część II: Czasownik, 2016,
(Γραμματική της πολωνικής γλώσσας σε πίνακες. Μέρος Ι: Κλίση. Μέρος ΙΙ: Ρήμα.)
https://www.sjikp.us.edu.pl/pl/publikacje/publikacje-szkoly-jezyka-i-kultury-polskiej-uniwersytetu-slaskiego/tabele-i-plansze/teczowa-gramatyka-jezyka-polskiego-w-tabelach-czesc-i-deklinacja-czesc-ii-czasownik/

[9] Koniugacja (Κλίση), https://pl.bab.la/koniugacja/