ΥΛΟΠΟΙΗΣΗ ΜΕΤΑΤΡΟΠΕΑ GREEKLISH ΣΕ ΕΛΛΗΝΙΚΑ

ΜΑΣΤΡΑΝΤΩΝΗΣ ΧΑΡΑΛΑΜΠΟΣ

Επιβλέπων καθηγητής: Αριστομένης Θανόπουλος

Τ.Ε.Ι. ΠΕΛΟΠΟΝΝΗΣΟΥ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΣΧΟΛΗ ΤΕΧΝΙΚΩΝ ΕΦΑΡΜΟΓΩΝ (έδρα Σπάρτη)

ΙΑΝΟΥΑΡΙΟΣ 2017 ΑΘΗΝΑ ΑΤΤΙΚΗ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΗΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάση επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δε μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

Όνομα και Επώνυμο Συγγραφέα (Με Κεφαλαία):

.....

Υπογραφή (Ολογράφως, χωρίς μονογραφή):

.....

Ημερομηνία (Ημέρα - Μήνας - Έτος)

.....

ΠΕΡΙΛΗΨΗ

Η εργασία αυτή έχει σαν στόχο την υλοποίηση ενός προγράμματος σε μία γλώσσα που να μετατρέπει Greeklish λέξεις σε Ελληνικά, όπως επίσης και τρόπους αξιολόγησης , ανακάλυψη προβλημάτων και επίλυσης αυτών.

Στο πλαίσιο της εργασίας υλοποιήθηκε μία εφαρμογή που επιτρέπει σε κάποιον χρήστη την εισαγωγή της λέξης και έπειτα εμφανίζονται οι πιθανές μεταφράσεις της στα Ελληνικά. Όπου από εκεί επιλέγεται αυτόματα η ομοιότερη και επιστρέφεται στον χρήστη.

Σήμερα η χρήση των greeklish αυξάνεται καθώς αυξάνεται και ο όγκος δεδομένων του διαδικτύου. Τα greeklish προτιμώνται σαν τρόπος γραφής για την ταχύτερη μέθοδο τους αλλά και του μεγάλου εύρους πιθανών τρόπων γραφής που κατανοούνται από όλους τους χρήστες. Κανένας όμως δεν αναλογίζεται τι προβλήματα δημιουργούνται με την συχνή χρήση αυτού του τρόπου.

Αναλύουμε τις μεθόδους μετατροπής και προσπαθούμε να καλύψουμε, όσο το δυνατόν γίνεται, την κάθε διαφορετική ψυχολογία χρήσης των greeklish.

ABSTRACT

This project aims to develop a program in a language that converts Greeklish words to Greek, as well as evaluating methods, discovering problems and solving them.

As part of the work, an application was designed that allows a user to enter a word and then it displays the possible translation in Greek. From there the most identical to Greek word is selected and returned to the user.

Today the use of greeklish writing increases as the volume of internet digital data does. Greeklish is preferred as a way of writing for their fastest method but also the wide range of possible writing methods that every user can understand. But no one reflects on what problems are created by the frequent use of this way.

We analyze the conversion methods, and we try to cover as much as possible each different psychology of the use of Greeklish

ΕΥΧΑΡΙΣΤΙΕΣ

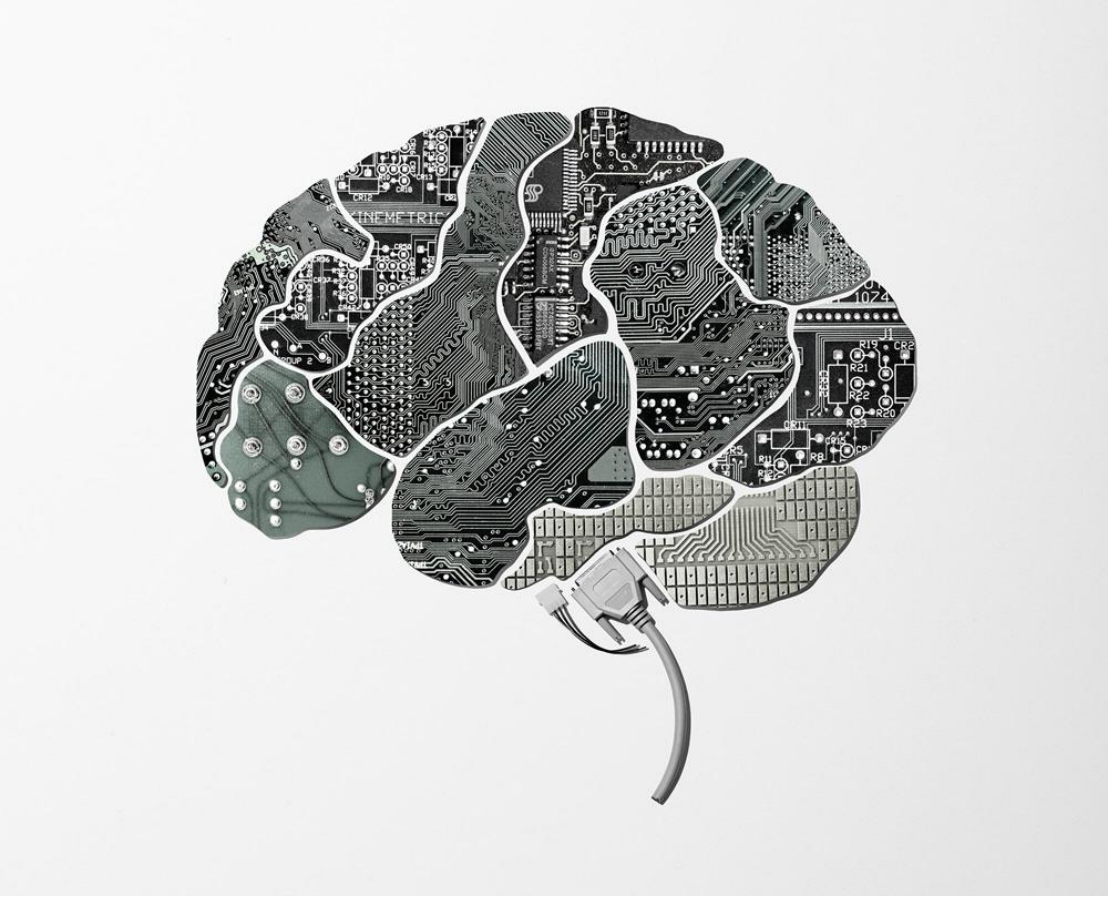
Ευχαριστώ από τα βάθη της καρδιάς μου τον κύριο Αριστομένη Θανόπουλο καθηγητή του τμήματος Μηχανικών Πληροφορικής Τ.Ε., για την ευκαιρία που μου έδωσε και με άφησε να ασχοληθώ με αυτό το θέμα, όπως και για την κατανόηση και την υπομονή που μου έδειξε στις δύσκολες στιγμές στα πλαίσια αυτής της εργασίας.

Εν κατακλείδι, αφιερώνω αυτή την εργασία στους γονείς μου, Παναγιώτη και Αρετή, και στην σύντροφο μου, Δέσποινα που στάθηκαν στο πλάι μου καθ' όλη την διάρκεια.

Πίνακας περιεχομένων

ΕΙΣΑΓΩΓΗ 7
1.1 Χρήση των greeklish: αιτία και μειονεκτήματα 9
1.2 Αντικείμενο και στόχος της πτυχιακής 11
1.2.1. Αντικείμενο 11
1.2.2. Στόχος 13
1.3 Ορισμοί 14
ΚΕΦΑΛΑΙΟ 2 16
2.1 Μεθοδολογία 18
2.1.1 Αλγόριθμος Μεθοδολογίας 20
2.1.2. Υπολογισμός ομοιότητας λέξεων 23
2.1.3. Εξαγωγή κανόνων μετατροπής 25
2.2. Εργαλεία δημιουργίας 28
2.3 Υλοποίηση Εφαρμογής 29
2.3.1. Σχεδίαση 29
2.3.2. Evaluation της εφαρμογής 33
2.3.3. Διαδικασία 35
ΕΠΙΛΟΓΟΣ ΚΕΦΑΛΑΙΟ 3 44
3.1 Συμπεράσματα 45
3.2 Χρήση Εφαρμογής 46
3.3 Μελλοντική αναβάθμιση Εφαρμογής 46
ΒΙΒΛΙΟΓΡΑΦΙΑ 48
ΠΑΡΑΡΤΗΜΑ Α: 50
Ο πλήρης κώδικας της εφαρμογής 50

ΕΙΣΑΓΩΓΗ



Περίληψη της ανάπτυξης της Τεχνολογίας

Με βάση τα τελευταία 10 χρόνια η τεχνολογία αποτελεί βασικό κομμάτι στην ζωή των ανθρώπων και ιδιαίτερα στην καθημερινότητα τους· σε θέματα όπως είναι η υγεία και η διαβίωση αλλά και σε διάφορες περιπτώσεις. Ορισμένα παραδείγματα αφορούν την εργασία, την παιδεία, την ψυχαγωγία και φυσικά την ενημέρωση. Με την ανάπτυξη της τεχνολογίας, αναπτύχθηκαν και δυνατότητες για την αναβάθμιση του συστήματος της οικονομίας, όπως επίσης και μέσα για ένα βιώσιμο περιβάλλον. Μιλώντας μεμονωμένα για την απασχόληση ενός ατόμου, υπάρχει η δυνατότητα μετακίνησης για έναν περίπατο, η προβολή μίας ταινίας στον κινηματογράφο, μία τηλεοπτική εκπομπή, μία περιήγηση στο διαδίκτυο όπως και στα μέσα κοινωνικής δικτύωσης.

Το διαδίκτυο πλέον καλύπτει μία μεγάλη γκάμα από δυνατότητες, χρησιμοποιείται για την ενημέρωση έχοντας πρόσβαση σε πολλά μέσα όπως είναι οι ηλεκτρονικές εφημερίδες, οι διαδικτυακές εκπομπές κ.α. Χρησιμοποιείται για την διασκέδαση με πάρα πολλές δραστηριότητες αναλόγως με τις προτιμήσεις του κάθε χρήστη. Χρησιμοποιείται επίσης και για την εργασία με παροχές όπως το ηλεκτρονικό ταχυδρομείο ή η άμεση σύνδεση με τον χώρο εργασίας του εργαζομένου.

Καθώς προχωράει ο χρόνος έτσι και η τεχνολογία αναπτύσσεται και μαζί του και το διαδίκτυο. Η χρήση του έχει γίνει πολύ πιο απλοποιημένη και ταχύτερη, παράλληλα αυξάνονται οι απαιτήσεις των συστημάτων, γίνεται όμως απαραίτητη και η εξειδίκευση των ανθρώπων που τα χρησιμοποιούν.

Εφευρέθηκε ένας τρόπος γραφής από τους χρήστες ώστε η διαδικτυακή επικοινωνία με την χρήση του αγγλικού τύπου (QWERTY) πληκτρολογίου να γίνει και πιο εύκολη και πιο γρήγορη για τους έλληνες, καθώς οι περισσότεροι είχαν μάθει να γράφουν στα αγγλικά. Ανακαλύφθηκε λοιπόν η χρήση του greeklish.

Παρατήρηση: Η ανάπτυξη χρήσης αυτού του τρόπου σε καμία περίπτωση δεν δηλώνει την υποστήριξη μιας χρήσης λατινικών χαρακτήρων αντί ελληνικών αν μη τι άλλο αποσκοπεί στην βελτίωση απόδοσης συστημάτων που επιτυγχάνουν την αντίστροφη διαδικασία (Λύρας Δημητριος,2009).

1.1 Χρήση των greeklish: αιτία και μειονεκτήματα

Τα greeklish (Γκρίκλις), προέρχονται από τις λέξεις Greek (ελληνικά) και English (αγγλικά), ονομάζονται αλλιώς και ως Grenglish, Λατινοελληνικά ή Φραγκολεβαντίνικα και είναι η ελληνική γλώσσα γραμμένη με το λατινικό αλφάβητο.

Η χρήση των greeklish έχει ξεκινήσει από το παρελθόν καθώς η αναγνώριση ελληνικών χαρακτήρων στους υπολογιστές δεν ήταν διαθέσιμη για όλα τα λειτουργικά συστήματα. Ο τρόπος αυτός επικράτησε διότι όπως αναφέραμε και παραπάνω, είναι γρηγορότερα στην πληκτρολόγηση και δεν υπάρχει λόγος να έχουν σωστή ορθογραφία.

Τα greeklish χρησιμοποιούνται στο διαδίκτυο από Έλληνες που παραδείγματος χάριν επικοινωνούν μέσω ηλεκτρονικού ταχυδρομείου, Facebook, Skype και διαδικτυακά forum που πολλοί χρήστες μιλάνε για την άποψη τους. Επίσης, άλλη μια χρήση τους είναι σε μηνύματα τύπου SMS για την κινητή τηλεφωνία και μεταξύ Ελλήνων που ζούν σε χώρες του εξωτερικού.

Το συγκεκριμένο φαινόμενο ορίζεται ως διγραφία και αποτελεί έναν τρόπο που έχει χρησιμοποιηθεί και από άλλες χώρες. Στην διγραφία εμφανίζονται δύο μορφές: *η συγχρονική και η διαχρονική*. Ο πρώτος τύπος φανερώνει τη συμβίωση δύο διαφορετικών συστημάτων γραφής σε μια γλώσσα, ενώ ο δεύτερος τύπος αναφέρεται στο γεγονός ότι στο πέρασμα του χρόνου το σύστημα γραφής μιας γλώσσας έχει υποστεί διάφορες αλλαγές και τελικά έφτασε να έχει αντικατασταθεί από άλλο καινούριο σύστημα.

Η πρώτη μορφή διγραφίας, η συγχρονική, είναι αυτή που εφαρμόζεται στις μέρες μας στην επικοινωνία των ανθρώπων μέσω του διαδικτύου, παρόλο που έχουν θεσπιστεί πρωτόκολλα κωδικοποίησης όλων των χαρακτήρων γραφής στα υπολογιστικά συστήματα. Υπάρχουν περιπτώσεις που η υποστήριξη για γλώσσες με μη λατινικούς χαρακτήρες είναι ελλιπής (όπως π.χ. Κινέζικα, Ιαπωνικά).

Ενδεικτικό παράδειγμα σύγχρονης διγραφίας είναι τα Σέρβικα καθώς παράλληλα με το Σερβικό-Λυριλλικό αλφάβητο έχει υιοθετηθεί και η χρήση ενός αντίστοιχου αλφαβήτου που αποτελείται από λατινικούς χαρακτήρες.

Στην περίπτωση της διαχρονικής διγραφίας το σύστημα γραφής έχει αντικατασταθεί πλήρως από ένα καινούριο που βασίζεται σε λατινικούς χαρακτήρες, για παράδειγμα

στην περίπτωση ρουμανικών (το αρχικό Κυριλλικό αλφάβητο αντικαταστάθηκε από το Λατινικό), τα τούρκικα (από αραβικό σε λατινικό αλφάβητο) και σε χώρες τις κεντρικής σοβιετικής Ασίας οι οποίες εγκατέλειψαν το Κυριλλικό αλφάβητο μετά την διάλυση της πρώην Σοβιετικής Ένωσης(el.wikipedia.org,2011).

Μειονεκτήματα

Τα greeklish στη γραφή τους, είναι αρκετά εύκολα, στην ανάγνωσή τους, όμως, είναι δυσανάγνωστα και κουραστικά ακόμα και για τους πιο φανατικούς υποστηρικτές τους.

Η μορφή της γλώσσας και τα χαρακτηριστικά της, αλλάζουν, πράγμα που φτωχαίνει και υπεραπλουστεύει το λεξιλόγιο. Αποτελεί κοινή παραδοχή ότι η μορφολογία της ψηφιακής γλώσσας είναι αρκετά χαλαρή και μεταβαλλόμενη. Παρατηρείται ότι με τη χρήση των greeklish επηρεάζεται και η εκμάθηση της ελληνικής γλώσσας, όπου το γεγονός αυτό, έχει επιπτώσεις στην ορθή γραφή των λέξεων της ελληνικής γλώσσας. Ακόμη, η αμέλεια της ορθής γραφής στα ελληνικά οδηγεί στα ορθογραφικά και συντακτικά λάθη, με αποτέλεσμα η εκτεταμένη χρήση τους, εκτός από ανορθογραφία μπορεί σε ορισμένες περιπτώσεις να προκαλέσει και δυσλεξία. (greeklishteam.wikispaces.com,2012).

Η εθνική μας ταυτότητα σιγά σιγά έστω και έμμεσα, εξασθενεί. Η ψηφιακή γλώσσα αποτελεί ένα είδος πολιτισμικού ιμπεριαλισμού των χρηστών της αγγλικής γλώσσας πίσω απ' την προμετωπίδα των τεχνολογικών συμβάσεων και περιορισμών. Απώτερος στόχος της είναι η αποδυνάμωση του στοιχείου εκείνου που οι γλωσσολόγοι ονομάζουν "εθνογλωσσική" συνείδηση και αποτελεί μία πτυχή του κυρίαρχου οικονομικού , πολιτικού και πολιτισμικού μοντέλου της εποχής μας , που είναι γνωστό ως παγκοσμιοποίηση.

Τα greeklish είναι γεγονός πως έχουν μπει για τα καλά στη ζωή μας και είναι χρέος μας να διαφυλάξουμε την ελληνική γλώσσα. Η εξολοκλήρου διαγραφή τους δεν αποτελεί λύση, αφού ορισμένες φορές μπορούν να μας φανούν χρήσιμα (greeklishproj4.blogspot.gr,2011).

1.2 Αντικείμενο και στόχος της πτυχιακής

1.2.1. Αντικείμενο

Απόπειρες για την ανάπτυξη τέτοιας εφαρμογής, ανά καιρούς έχουν γίνει αρκετές, άλλες με μεγάλο ποσοστό επιτυχίας και άλλες με λιγότερο. Κάτι αντίστροφο, δηλαδή ένας τρόπος μετατροπής από ελληνικά σε greeklish, δεν φαίνεται να είναι δύσκολος αφού μπορεί με ευκολία να γίνει, με την ανάθεση των ελληνικών χαρακτήρων σε αντίστοιχους αγγλικούς.

Παρόλα αυτά, ο τρόπος που οι άνθρωποι χρησιμοποιούν τα greeklish είναι απρόβλεπτος καθώς δεν υπάρχει κάποιος ειδικός κανόνας για την χρήση τους, ούτε έχει δημιουργηθεί από τους χρήστες κάποια τυποποιημένη μεταγραφή.

Έρευνες έδειξαν ότι έχει θεσπιστεί ένα επίσημο πρότυπο με την ονομασία ΕΛΟΤ 743:1982 (ΕΛΟΤ, 1982) που υπαγορεύει για την μεταγραφή της μίας γλώσσας στην άλλη, αλλά καταλήγουμε ότι κατά κύριο λόγο η μετατροπή αυτή γίνεται κατά την προτίμηση του χρήστη ακόμα και σήμερα.

Το πιο συχνό φαινόμενο στην γραφή των greeklish είναι αυτό του "ιωτακισμού" όπου για παράδειγμα στο στόχαστρο μπαίνουν τα φωνήεντα "ι", "η", "υ", "ει", "οι" της Νέας Ελληνικής, τα οποία αντικαθιστούνται από τον λατινικό χαρακτήρα "i".

Η ποικιλομορφία του τρόπου γραφής greeklish είναι μεγάλη. Σαν αντικείμενο το πρόγραμμα έχει την σωστή μετατροπή τους από την γραφή των greeklish σε αυτή των ελληνικών , όπως επίσης και την σύγκριση του ποσοστού επιτυχίας του με άλλα προγράμματα παρομοίου περιεχομένου και την επίδειξη διαφόρων προβλημάτων που προκαλούνται από το ίδιο η από αλλού.

Γενικότερα , αυτές οι πρακτικές μεταγραφής θα μπορούσαν να κατηγοριοποιηθούν σε Φωνητική, Οπτική, Δακτυλογραφική και μεικτός τρόπος μεταγραφής(διδακτορική διατριβή, Λύρας Δημητριος,2009). Πιο αναλυτικά:

a. *Φωνητική μεταγραφή:* Στην περίπτωση αυτή κάθε χαρακτήρας ή συνδυασμός γραφημάτων της Νεοελληνικής γλώσσας αντικαθίσταται από την αντίστοιχη αγγλική έκφραση που "ακούγεται" πιο ικανοποιητικά στην προφορά των γραφημάτων. Μεγάλο ρόλο παίζει και η καταγωγή του χρήστη· για παράδειγμα Έλληνας γεννημένος ή μεγαλωμένος στην Αμερική ή την Αυστραλία χρησιμοποιεί το "th" για την απόδοση του χαρακτήρα "δ" η και του "θ". Κάποιος άλλος στην Γερμάνια συνηθίζεται να χρησιμοποιεί το "w" αντί "β" και το "v" για την ακουστική απόδοση του "φ". Εν τέλη, οι Έλληνες γεννημένοι στην Γαλλία προτιμούν το "ου" για το αγγλικό "u".(Στα γαλλικά το "u" έχει δίφθογγη ακουστική απόδοση.)

b. *Οπτική μεταγραφή:* Εδώ, ο τύπος μεταγραφής βασίζεται στην οπτική ομοιότητα που έχουν τα ελληνικά με τα αντίστοιχα λατινικά σύμβολα. Συχνό φαινόμενο είναι η αντικατάσταση των ελληνικών χαρακτήρων όχι μόνο από λατινικούς αλλά παράλληλα και από αριθμούς, σύμβολα ή και συνδυασμό τους (κλασικό παράδειγμα είναι το γράμμα "θ" που αναπαριστάται ως "8" ή "9" ή το ελληνικό γράμμα "ξ" ως "3" και το "ω" ως "w").

c. *Δακτυλογραφική μεταγραφή (με βάση το πληκτρολόγιο):* Αυτός ο τύπος μεταγραφής αποτελείται από την θέση των χαρακτήρων του πληκτρολογίου του υπολογιστή. Οπότε αυτό έχει σαν αποτέλεσμα την πληκτρολόγηση της λατινοποιημένης ελληνικής λέξης με τον ίδιο τρόπο όπως θα χρησιμοποιούσε ο χρήστης με ελληνικούς χαρακτήρες (παρόλα αυτά συνηθίζεται να παραλείπονται οι τόνοι). Αυτό έχει σαν συνέπεια η μεταγραφή να μην έχει ούτε ακουστική αλλά ούτε και οπτική ομοιότητα. Χρήστες εξοικειωμένοι με το τυφλό σύστημα πληκτρολόγησης υιοθετούν αυτόν τον τρόπο.

d. *Μικτός τρόπος μεταγραφής:* Αυτή η κατηγορία αποτελεί και το μεγαλύτερο κομμάτι των Ελλήνων και γενικότερα όσων χρησιμοποιούν την ψηφιακή γλώσσα για αυτό είναι και πιο διαδεδομένη. Κατά βάση είναι στην ψυχολογική κρίση και την προσωπικότητα του κάθε χρήστη ο τρόπος μεταγραφής των ελληνικών γραφημάτων. Ο καθένας υιοθετεί οποιοδήποτε από τους προαναφερθέντες τρόπους (*φωνητική, οπτική ή δακτυλογραφική μεταγραφή*) ή και το συνδυασμό αυτών διαμορφώνοντας έτσι ένα δικό του σύστημα, χωρίς όμως αυτό να σημαίνει καταναγκαστικά εφαρμογή αυτού του τρόπου σε κάθε περίπτωση που τον χρησιμοποιεί.

1.2.2. Στόχος

Για να καλύψουμε την κάθε ανάγκη, αλλά και ψυχολογία και προσωπικότητα του κάθε ανθρώπου ο οποίος χρησιμοποίει τα greeklish, αναζητείται ο πιο στοχαστικός μεταγραφέας λατινοποιημένων ελληνικών σε ελληνικά.

Η υλοποίηση ενός μετατροπέα σε επίπεδο γραμμάτων και η ανάλυση της επιτυχίας αναλόγως και την τεχνική είναι ο σκοπός αυτής της έρευνας. Φυσικά, αν και με την πάροδο του χρόνου, επιστήμονες έχουν πλησιάσει το ποσοστό, ποτέ κανένας μετατροπέας δεν έχει αγγίξει το 100% της σωστής μετατροπής των προτάσεων. Θα αποκαταστήσουμε τυχών λάθη που εμφανίζονται και ανακαλύπτουμε τεχνικές που μπορούν να χρησιμοποιηθούν μελλοντικά και σε λειτουργικά συστήματα ή και προγράμματα όπου παλαιότερα δεν υποστήριζαν ελληνικούς χαρακτήρες.

Για αυτό τον λόγο και συγκρίνουμε και τις διάφορες τεχνικές για να καταδείξουμε και τα διάφορα προβλήματα που μπορούν να δημιουργηθούν από το θέμα αλλά και φυσικά και να εμφανιστούν και τρόπους επίλυσης τους, κάποιοι έχουν δοκιμαστεί με την τελευταία μέθοδο και άλλοι θα την συνεχίσουν σαν επέκταση σε άλλους τομείς.

Έχοντας, αναλύσει πλήρως την ιστορία και τις μεθόδους που θα ληφθούν υπόψη στο επόμενο κεφάλαιο θα ανακαλύψουμε την δημιουργία της εφαρμογής.

1.3 Ορισμοί

- **determined** [(ντετερμινιστικό) έχει σημασιολογική δομή και αποδεικνύει την σωστή αιτιολογία και σημασιολογία](http://accessibleculture.org,2011)

- **Αντικειμενοστραφής προγραμματισμός**[(object-oriented programming)μεθοδολογία όπου ο χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά γίνεται από κοινού](The Java Programming Language, Ken Arnold, James Gosling, David Holmes,2007)

- **map**[(χάρτης διασύνδεσης) αντιστοιχούν τιμές με κλειδιά](The Java Programming Language, Ken Arnold, James Gosling, David Holmes,2007)

- **mapping**[διασύνδεση χαρακτήρων και κλειδιών με άλλους]

- **class**[(κλάση) ορίζεται ως ταμπλέτα που περιγράφει της συνήθειες των αντικειμένων](oracle.com,1995-2016)

- **UTF-8**[είναι ένα μη-απωλεστικό σχήμα κωδικοποίησης χαρακτήρων μεταβλητού μήκους](el.wikipedia.org)

- **arraylist**[(πίνακας δεδομένων)αποθηκεύουν ακολουθίες αντικειμένων αλλα δεν μπορούν να αλλάξουν το μέγεθος τους](The Java Programming Language, Ken Arnold, James Gosling, David Holmes,2007)

- **string**[αντιπροσωπεύει χαρακτήρες στις προγραμματισμού](oracle.com,1995-2016)

- **constructor**[(κατασκευαστές)σαν τις μεθόδους για την δήλωση κλάσεων](oracle.com,1995-2016)
- **frame**[πλαίσιο κειμένου](oracle.com,1995-2016)

- **label**[ταμπέλα ονόματος πλαισίου](oracle.com,1995-2016)
- **textfield**[πλαίσιο γραφόμενου κειμένου](oracle.com,1995-2016)
- **input**[εντολή εισχώρησης δεδομένων]oracle.com,1995-2016 ()
- **action**[εντολή](oracle.com,1995-2016)
- **button**[ξεκίνημα εντολής](oracle.com,1995-2016)
- **symbolsets[**στοιχεία σύμβολα που δηλώνουν τον κάθε λατινικό χαρακτήρα**]**
- **user interface[**πίνακας διασύνδεσης με τον χρήστη**]**
- **evaluation[**κριτήριο αξιολόγησης**]**

ΚΕΦΑΛΑΙΟ 2


if(parameters.contains(
hql += " and p.name = :name";
}
if(parameters.contains("age")){
hql += " and p.age = :age";
}
TypedQuery<Person> query = em.
if(parameters.contains("name")){
query.setParameter("name",
}
if(parameters.contains("age")){
Integer.

Πρόλογος

Είναι γεγονός ότι , το μεγαλύτερο και το πιο σύνθετο πρόβλημα για την μετατροπή από greeklish σε ελληνικά με σωστή ορθογραφία, πέρα ότι πολλές λατινοποιημένες εκφράσεις μπορεί να αντιστοιχίζονται μονάχα σε μια ελληνική λέξη, είναι και ότι υπάρχει ακόμη πολλαπλή ανάγκη συλλογής μεγάλων σωμάτων επισημειωμένου κειμένου (annotated corpora) που θα περιλαμβάνει μεταγραφές ελληνικών ορθογραφημένων λέξεων με τις αντίστοιχες greeklish εκφράσεις τους.

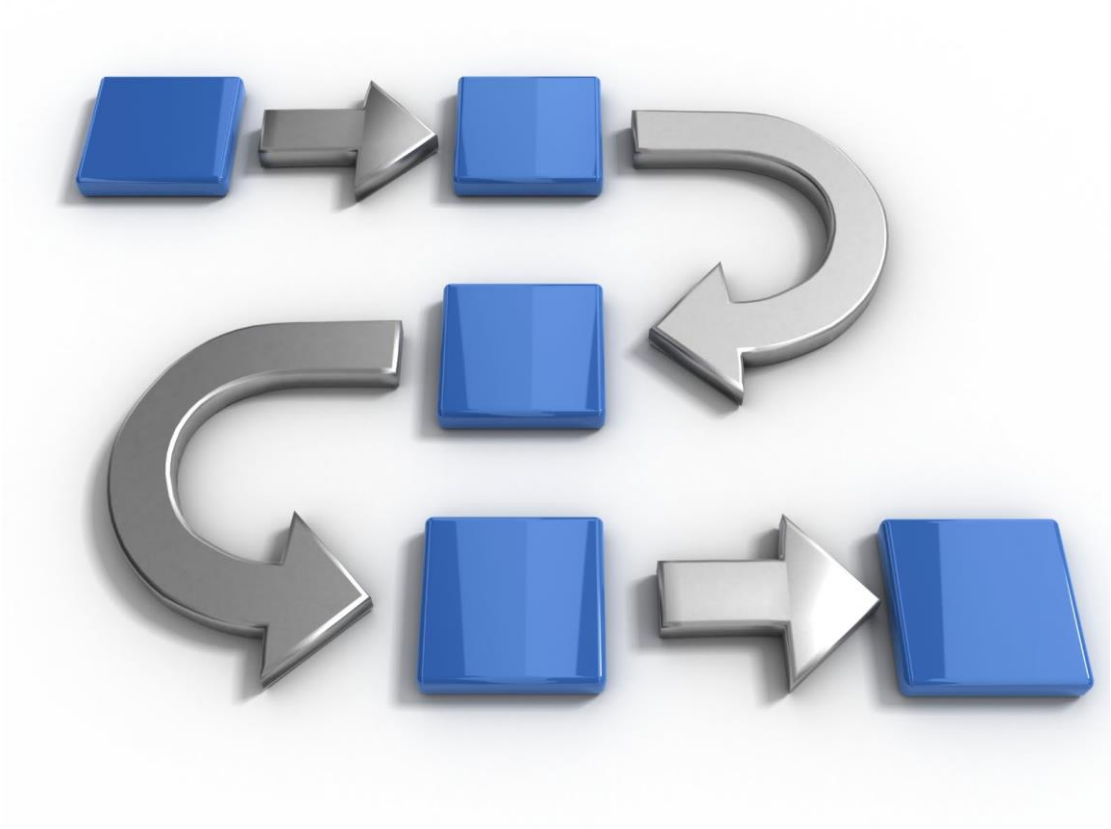
Η χρησιμότητα αυτών των σωμάτων κειμένου είναι:

- είτε ως βάσεις δεδομένων για την εκπαίδευση μοντέλων που θα επιτυγχάνουν τη μεταγραφή από greeklish σε ελληνικά
- είτε και ως δεδομένα εξέτασης για την αξιολόγηση των συστημάτων αυτών.

Ως εκ τούτου, για να καλύψουμε αυτά τα ζητήματα έγινε έρευνα και δημιουργία μιας εφαρμογής μετατροπής από την ψηφιακή γλώσσα greeklish σε ελληνικά για να μπορέσουμε να την συγκρίνουμε με άλλους εξίσου στοχαστικούς μεταγραφείς και να βρεθεί η καλύτερη λύση για την, όσο το δυνατόν καλύτερα, ορθοποιημένη μεταγραφή από λατινικούς χαρακτήρες και λέξεις σε ελληνικούς. Από άποψη μετατροπής χαρακτήρων φαίνεται να υπερισχύει καθώς οι αντιστοιχίες και η εναλαγή τους ήταν αρκετά απλός αλγόριθμος.

Μετά από την μετατροπή ακολουθεί η ομοιότητα με τις λέξεις στο λεξικό που δεν κατακτά πλήρως τον στόχο που αποζητάμε για την ορθά μεταφρασμένη λέξη.

2.1 Μεθοδολογία



Στην μεθοδολογία ανακαλύπτουμε το μεγαλύτερο στοιχείο που χρησιμοποιούμε για την εργασία.

Αντιλαμβανόμαστε από τα παραπάνω πως εξαιτίας της μεγάλης ποικιλομορφίας σχετικά με τους τρόπους μεταγραφής από ελληνικά σε greeklish, παρουσιάζεται η ανάγκη ανάπτυξης, εναλλαγής από το ένα σύστημα γραφής στο άλλο.

Μέχρι στιγμής υπάρχουν πολλές προσεγγίσεις για την μετατροπή των ελληνικών χαρακτήρων σε greeklish (για παράδειγμα ASDA(ASDA), Greeklish to Greek[αποτελεί και του δυο τρόπους (greeklish.net)]) που κατά κόρον στηρίζονται σε κανόνες στατικούς και κανόνες που βασίζονται σε λεξικά. Παρόλα αυτά κανένας από τους παραπάνω τρόπους δεν μπορεί να θεωρηθεί ικανοποιητικός καθώς δεν αντιπροσωπεύει τον κάθε χρήστη.

Οι ποικιλομορφίες της γραφής των greeklish σήμερα, έχουν πάρει πολύ διαφορετική τροπή από ότι παλαιότερα. Αρκετοί χρήστες τείνουν να παραλλάσσουν τις λέξεις (για παράδειγμα να τις κόβουν, να προσθέτουν σύμβολα, να απλοποιούν την ορθογραφία των λέξεων) πράγμα που καθιστά σχεδόν αδύνατο σε οποιοδήποτε πρόγραμμα να κάνει σωστή μετατροπή.

Λαμβάνοντας αυτό υπόψιν, καταλαβαίνουμε ότι περιοριζόμαστε από άποψη κανόνων. Με έναυσμα τα παραπάνω προβλήματα σε αυτό που θα επικεντρωθούμε είναι η δημιουργία ενός προγράμματος για την άμεση μεταγραφή από greeklish σε ελληνικά.

2.1.1 Αλγόριθμος Μεθοδολογίας

Για αρχή συλλέχτηκαν διάφορες λέξεις από γνωστές ιστοσελίδες στο διαδίκτυο με σκοπό την μετατροπή τους και τοποθετηθήκαν σε μια βάση δεδομένων, αφαιρώντας βέβαια συντομογραφίες και λέξεις που δεν υπάρχουν σε κάποιο λεξικό, τις οποίες χρησιμοποιούσαν άλλοι χρήστες. Έπειτα πραγματοποιήθηκε έλεγχος για το ποιός είναι ο πιο κοινός κανόνας των greeklish που χρησιμοποιούν οι χρήστες.

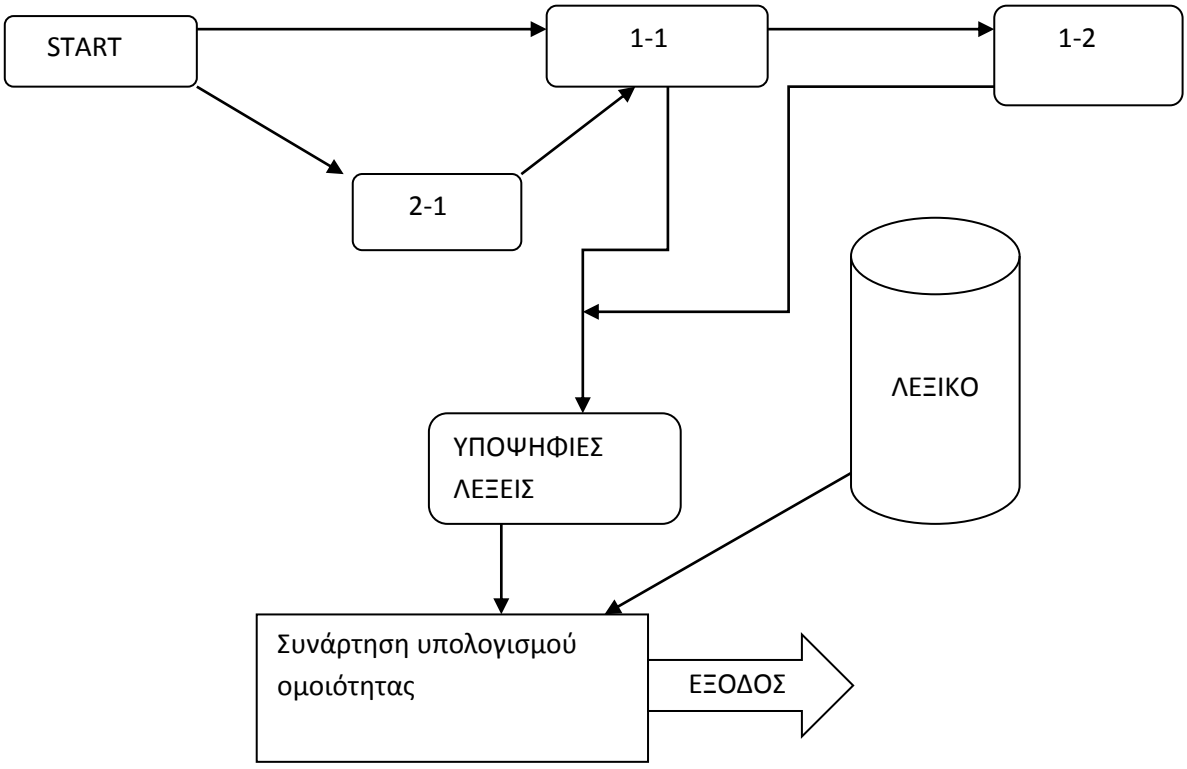
Κατά μέσο όρο το μεγαλύτερο ποσοστό χρήσης των greeklish στο διαδίκτυο είναι ο οπτικός κανόνας. Με βάση αυτόν τον κανόνα, στην παρούσα διατριβή, καταλήξαμε στο συμπέρασμα ότι η άμεση μετατροπή θα ήταν αποτελεσματικότερη σε αντιστοιχίες γραμμάτων και σε κανόνες που στοχεύουν σε μία μεμονωμένη λέξη.

Το πρόγραμμα αποτελείται από τρεις αλγορίθμους, ο καθένας εκ των οποίων εκτελεί μία αρχή παρόμοια αυτής των βάσεων δεδομένων.(1-1[ένα προς ένα],1-ν[ένα προς πολλά],ν-1[πολλά προς ένα])

Κάθε ένας από αυτούς συμπεριφέρεται διαφορετικά από τον άλλον, όλοι όμως έχουν σαν αρχή την αντικατάσταση ενός λατινικού γράμματος με έναν ελληνικό.

Στη συνέχεια χρησιμοποιούμαι έναν τύπο πολλαπλής σύγκρισης ή αλλιώς του cross-validation για να συγκρίνουμε την αποτελεσματικότητα του αλγορίθμου σε 3 ξεχωριστά μεταφρασμένα επίσημα κείμενα. Έτσι εξασφαλίζουμε την ακρίβεια των κειμένων ώστε το πρόγραμμα να επιτρέπει στην σύγκριση να γίνει όσο το δυνατόν καλύτερη.

Αναλυτικότερη επεξήγηση παρατίθεται στο παρακάτω σχεδιάγραμμα:



Έχοντας το σχεδιάγραμμα υπόψιν υλοποιούμε και αντίστοιχους αλγόριθμους.

Αναλυτικότερα:
[START]:Παίρνουμε την λέξη που εισάγει ο χρήστης. Η λέξη αυτή χωρίζεται σε γράμματα.

[1-1]:Εφαρμόζουμε απλή αντιστοιχία αγγλικών χαρακτήρων με ελληνικούς. Ο λόγος που ο αλγόριθμος ονομάστηκε ένα προς ένα είναι γιατί αντιστοιχήσαμε το κάθε χαρακτήρα με μόνο έναν χαρακτήρα.

[2-1]:Στην δεύτερη λειτουργία όπως στο σχεδιάγραμμα, ο αλγόριθμος δύο προς ένα βασίζεται πάρα πολύ στον πρώτο αλγόριθμο. Σε πολλές περιπτώσεις υπάρχουν ελληνικοί χαρακτήρες οι οποίοι είναι δίφθογγοι (πχ. ξ="κσ") αλλά αποτελούνται από έναν αγγλικό χαρακτήρα όπως για παράδειγμα “ξ” με “x” ή και δύο χαρακτήρες “ks”, “θ” με “th”. Ο αλγόριθμος αυτός καλύπτει τις περιπτώσεις οπού ο χρήστης χρησιμοποιεί τους διπλούς χαρακτήρες αντί για τους αριθμούς για να δηλώσει αυτά τα γράμματα. Ψάχνει τους διπλούς χαρακτήρες με έναν αλγόριθμο αφού εκτελεστεί ο πρώτος και αλλάζει τις λέξεις που δεν βγάζουν νόημα λόγο της απλής αντιστοιχίας.

[1-2]: Κατά βάση ο τρίτος αλγόριθμος καλύπτει τις αντιστοιχίες των φωνήεντων «ι» και «ο» αλλά και λίγες περιπτώσεις όπως το αγγλικό “x”. Αυτό συμβαίνει γιατί συνηθίζεται στα greeklish όλα τα «ι» να γράφονται με το «i», οπότε ο αλγόριθμος αλλάζει τα συγκεκριμένα φωνήεντα σε όλες τις υπόλοιπες αντιστοιχίες τους.

Τέλος, όλες οι περιπτώσεις εμφανίζονται σαν πιθανές λέξεις σε ένα πλαίσιο στο πρόγραμμα**[ΥΠΟΨΗΦΙΕΣ ΛΕΞΕΙΣ]**.

Έπειτα από έρευνα βρέθηκε ένα μεγάλο λεξικό λέξεων από το ανοιχτό εργαλείο γραφής κειμένου (openoffice.org,1999.2010). Το λεξικό χρησιμοποιείται από το openoffice writer(openoffice.org,1999.2010) σαν πρόγραμμα διόρθωσης για την ελληνική ορθογραφία και το διαθέτουν ανοιχτά στο κοινό για όποιων έχει να προσθέσει λέξεις που έχει ανακαλύψει ότι δεν υπάρχουν. Αν και η java και το σύστημα μπορούσαν να υποστηρίξουν τον πολύ μεγάλο όγκο του (περίπου 100000 λέξεων) όχι μόνο έπαιρνε πολύ χρόνο να αναζητήσει όλο το λεξικό υπήρχαν περιπτώσεις που δεν είχαμε το επιθυμητό αποτέλεσμα διότι γινόταν η παύση του προγράμματος, λόγο πάλι του μεγάλου όγκου. Έτσι αναζητήθηκαν οι 2000 πιο χρησιμοποιημένες ελληνικές λέξεις για την χρήση ως λεξικού για την διατριβή αυτής της εργασίας. Με το μέτρο ομοιότητας δυο λέξεων παίρνουμε την πιο "κοντινή" λέξη για την σωστή αντικατάσταση.

2.1.2. Υπολογισμός ομοιότητας λέξεων

Το μέτρο αυτό είναι ένας αλγόριθμος similarity χρησιμοποιώντας τις αρχές του **Levenshtein distance** (Vladimir Levenshtein,1965) όπου με βάση την απόσταση των λέξεων και την θέση γραμμάτων αναλύει τα πόσα βήματα μετατροπής θα χρειαζόταν μία λέξη να μετατραπεί σε μία άλλη. Ο μαθηματικός τύπος της απόστασης του Levenshtein, μεταξύ δύο στοιχείων **α και β** είναι:

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j)+1 \\ \operatorname{lev}_{a,b}(i,j-1)+1 \\ \operatorname{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} \end{cases}$$

Όπου $1_{(a_i \neq b_j)}$ είναι η συνάρτηση ένδειξης για το αν οι λέξεις διαφέρουν ή όχι. Οι τιμές που μπορεί να πάρει είναι ή 0 αν το $a_i = b_j$ ή 1 όπου βγαίνει η απόσταση από τον τύπο $\operatorname{lev}_{a,b}(i,j)$, όπου το **i** είναι ο πρώτος χαρακτήρας του **α** και το **j** του **β**(en.wikipedia.org,2011)**.**

Καταρχήν αυτού τύπου ένα παράδειγμα υπολογισμου απόσαστης του Levenshtein μεταξύ των λέξεων "παίρνω" και "περνάω" είναι το αποτέλεσμα τους να είναι 2. Διότι για να αλλάξει η λέξη αυτή, χρειάζονται τουλάχιστον 2 "κινήσεις".

1. **παί**ρνω → π**έ**ρνω
2. π**έ**ρνω → περν**ά**ω

Στην παρούσα περίπτωση, υπολογίζουμε την ομοιότητα σε ένα μοντέλο 0%-100%. Κατά βάση το μέτρο αυτό έχοντας λάβει την απόσταση των δύο λέξεων, που πρόκειται να συγκρίνει, κοιτάει ανά την θέση του κάθε γράμματος και τέλος εξάγει το ποσοστό των γραμμάτων που διαφέρουν.

Ξεκινώντας από την αρχή και το τέλος των δύο λέξεων βλέπουμε καταρχάς, αν ο αριθμός χαρακτήρων είναι ίδιος.

Στην λέξη "υγεία" το μέτρο απόστασης είναι 5. Έπειτα σαν μέσο όρο το άθροισμα των χαρακτήρων της λέξης, ο αλγόριθμος αναζητά εάν υπάρχει διαφορά στον κάθε χαρακτήρα ξεκινώντας από τον πρώτο και υπολογίζουμε το αποτέλεσμα. Στο συγκεκριμένο παράδειγμα έχουμε ποσοστό ομοιότητας 100%. Εάν ένα από τα γράμματα ήταν διαφορετικό , όπως το "ήγεια" τότε το ποσοστό θα έπεφτε στο 80% (εννοώντας 4/5 ομοιότητα οπότε στα 100 βγαίνει και το αποτέλεσμα).

Ένα από τα επίσης μεγαλύτερα προβλήματα που προκύπτουν είναι η ύπαρξη τόνων στις ελληνικές λέξεις αυτό μειώνει το ποσοστό σύγκρισης κατά τουλάχιστον κατά 20%, οπότε αναγκαστήκαμε να θέσουμε τα «στάνταρ» για τις πιο κοντινές λέξεις πολύ χαμηλά. Βέβαια αυτό δεν αποδείχθηκε πρόβλημα καθώς το λεξικό είναι αρκετά μικρό σε μεγαλύτερα, όπως με το προηγούμενο που αναφέραμε πριν το πρόβλημα θα ήταν πολύ μεγαλύτερο.

Έχοντας λάβει υπόψη τις τρεις μεθόδους στοχεύουμε έπειτα στην καταλληλότερη λύση.

2.1.3. Εξαγωγή κανόνων μετατροπής

Αρκετές έρευνες έχουν διεξαχθεί για την μετατροπή από greeklish σε ελληνικά, τα πιο γνωστά παραδείγματα είναι: e-chaos (e-Chaos), Innoetics (Innoetics), All Greek to me! (Chalamandaris, et al, 2006), Greeklish to Greek (greeklish.net), deGreeklish (DeGreeklish) και ακόμη περισσότερα τα οποία είναι πιο ιδιωτικά.

Για την εργασία αυτή επιλέχθηκαν οι έρευνες και οι αλγόριθμοι από δυο αξιοπρεπείς μετατροπείς και έγινε η σύγκριση μεταξύ όλων για την επίτευξη μιας καλύτερης μεθόδου.

Η εταιρία που έχει αφοσιωθεί πάνω στο κομμάτι αυτό, ονομάζεται Innoetics. Έχει αναπτύξει τεχνολογίες σε ότι αφόρα σε εφαρμογές κειμένου με ομιλία και την τεχνολογία για αυτόματη μετατροπή από greeklish σε ελληνικά.

Η συγκεκριμένη τεχνολογία αποτελεί προϊόν των εργαστηρίων του Ινστιτούτου Επεξεργασίας του Λόγου, συνδυάζοντας διαφορετικές τεχνολογίες από διαφορετικά γνωστικά πεδία. Ενσωματώνει δύο διαφορετικά στατιστικά μοντέλα, ένα για τα Greeklish και τους διαφορετικούς του τύπους και ένα δεύτερο για την Ελληνική γλώσσα, τα οποία συνδυάζονται μέσω μίας απεικονιστικής ψευδογλώσσας. Χρειάστηκε να συλλεχθούν δεδομένα εκατομμυρίων λέξεων για να παραχθούν τα αντίστοιχα μοντέλα έτσι ώστε να αντιμετωπιστεί η μεγάλη ποικιλότητα των Greeklish.

Παράλληλα, ένα υποσύστημα αναγνώρισης γλώσσας υποβοηθά την αποτελεσματικότερη αντιμετώπιση λέξεων αλλά και προτάσεων που γράφονται με λατινικούς χαρακτήρες, αλλά δεν είναι Greeklish.

Το αποτέλεσμα: Ένα πρόγραμμα που μετατρέπει οποιονδήποτε τύπο Greeklish με ακρίβεια που ξεπερνάει το 99%! (Innoetics,2010). Ο όγκος αυτού του προγράμματος

αλλά και τι τεχνολογία έχει ερευνηθεί για την λειτουργία του, είναι πολύ μεγάλος για να μπορούσαμε να φτιάξουμε κάτι αντίστοιχο. Βεβαία η Innoetics αποτελείται και από άτομα ειδικευμένα για προγράμματα ορθογραφίας η και text-to-speech, που έχουν να κάνουν δηλαδή με την μετατροπή κειμένων.

Η δεύτερη μέθοδος είναι μια έρευνα του Σπύρου Μπλάνα [A Greeklish-to-Greek converter,(gr2el,Σπύρος Μπλανας,2009)] η οποία αφορά μία μέθοδο μεταγραφής από λατινικούς χαρακτήρες σε ελληνικούς. Τον αλγόριθμο που έχει αναπτύξει και θα λάβουμε υπόψη είναι αυτός που στηρίζεται στον νόμο του Bayes.

Στην προκείμενη περίπτωση αν το "g" είναι greeklish και το "ε" είναι ελληνικά τότε ψάχνουμε το "ε" για:

$$argmax_{\epsilon} p(\epsilon|g) =$$
$$= argmax_{\epsilon} p(g|\epsilon)p(\epsilon) =$$
$$= argmax_{\epsilon} \prod_{i} p(g_i|\epsilon_i)p(\epsilon) =$$
$$= argmax_{\epsilon} \sum_{i} logp(g_i|\epsilon_i) + logp(\epsilon)$$

όπου το $p(g_i|e_i)$ θεωρείται ότι έχει ομοιόμορφη πιθανότητα μεταξύ των διάφορων τρόπων μετατροπής μιας ελληνικής λέξης σε greeklish(Σπύρος Μπλάνας,2009).

Υπήρχαν πολλές επιλογές σχετικά με τι μεθοδολογία που θα αναπτυχθεί το πρόγραμμα. Μία εκδοχή ήταν η δημιουργία μιας βάσεων δεδομένων με greeklish λέξεις και η αυτόματη μετάφραση τους από μία άλλη με ελληνικές. Ουσιαστικά ταύτη η τεχνική μας περιορίζει, καθώς ο κάθε χρήστης δεν θα ακολουθήσει κάποιο βασικό κανόνα για την γραφή των λέξεων, και άρα αυτή η εκδοχή δεν είναι χρήσιμη. Εκτός από αυτό ένα άλλο πρόβλημα είναι ότι θα έπρεπε να συνδέσουμε την java με μία άλλη γλώσσα προγραμματισμού, όπως την MySQL. Δεν είναι κάτι αδύνατο αλλά πράγμα πολύ δύσκολο και με πολλούς πόρους για να δημιουργηθεί. Σαν λύση που μπορούσε να βρεθεί, αντί για την σύνδεση ενός άλλου προγράμματος, είναι αυτές οι λέξεις να δημιουργούσαν ένα Arraylist που θα μπορούσε να καλέσει το πρόγραμμα, αλλά ήταν αδύνατη η διασύνδεση του με το υπόλοιπο πρόγραμμα.

Το επόμενο, ήταν με την μετάφραση γράμμα προς γράμμα να υπολογίζεται, με βάση την συχνότητα/πιθανότητα μετάφρασης του γράμματος, δηλαδή, σε ένα γράμμα όπως το αγγλικό "v" να προκύπτουν οι πιθανότητες το "v" να είναι "β" 60%, να είναι "ν" 30% και για να είναι "υ" 10%. Άρα με αυτές τις πιθανότητες να βγαίνουν κάποιες υποψήφιες λέξεις ως σωστές. Παρόλα αυτά όμως ούτε αυτή η λύση είναι προβλεπόμενη καθώς υπάρχει η μικρή πιθανότητα να εμφανιστεί το λάθος γράμμα σε λάθος λέξη. Για να μην αποκλείσουμε τελείως αυτήν την θεωρία, καθώς όπως αναφέραμε, υπάρχει η πιθανότητα ένα γράμμα να έχει δύο έννοιες αφήσαμε την επιλογή ώστε να εμφανίζεται κάθε πιθανό αποτέλεσμα μίας τέτοια περίπτωσης

Συλλέγοντας αυτά τα δεδομένα καταλήξαμε σε αυτήν την επιλογή μεθοδολογίας που αναφέραμε παραπάνω καθώς στο μόνο, ίσως που δεν μας επιφέρει 100% απόδοση είναι στα ορθογραφικά λάθη αν και τα περνάμε από ορθογραφικό έλεγχο.

Μετά από την επιλογή της μεθοδολογίας βρίσκουμε τα εργαλεία για την πραγματοποίηση αυτού του σχεδίου.

2.2. Εργαλεία δημιουργίας

Η επιλογή της γλώσσας προγραμματισμού ήταν εύκολη αφού αναζητούσαμε κάτι το οποίο να λειτούργει σε κάθε λειτουργικό σύστημα και τι καλύτερο από μία αντικειμενοστραφή γλώσσα και πιο συγκεκριμένα η Java(oracle.com,1995-2016).

Σαν πλατφόρμα για την γραφή του κώδικα χρησιμοποιήθηκε το NetBeans IDE 8.0.2(netbeans.org,2000) που ειδικεύεται στην καταγραφή κώδικα για πολλές γλώσσες προγραμματισμού και είναι πολύ πιο ιδανικό για τον χρήστη. Κατανέμει και υποδεικνύει τα λάθη με πολύ σαφήνεια και επιτρέπει σε μία πολύ ομαλότερη δημιουργία διεπαφής με τον χρήστη.

Εν τέλη όπως αναφέρθηκε και προηγούμενος χρησιμοποιήσαμε το λεξικό με τους ελληνικούς τύπους από το πρόγραμμα γραφής κειμένου OpenOffice(openoffice.org,1999.2010) και από εκεί χωρίσαμε τις 2000 πιο συχνότερες.

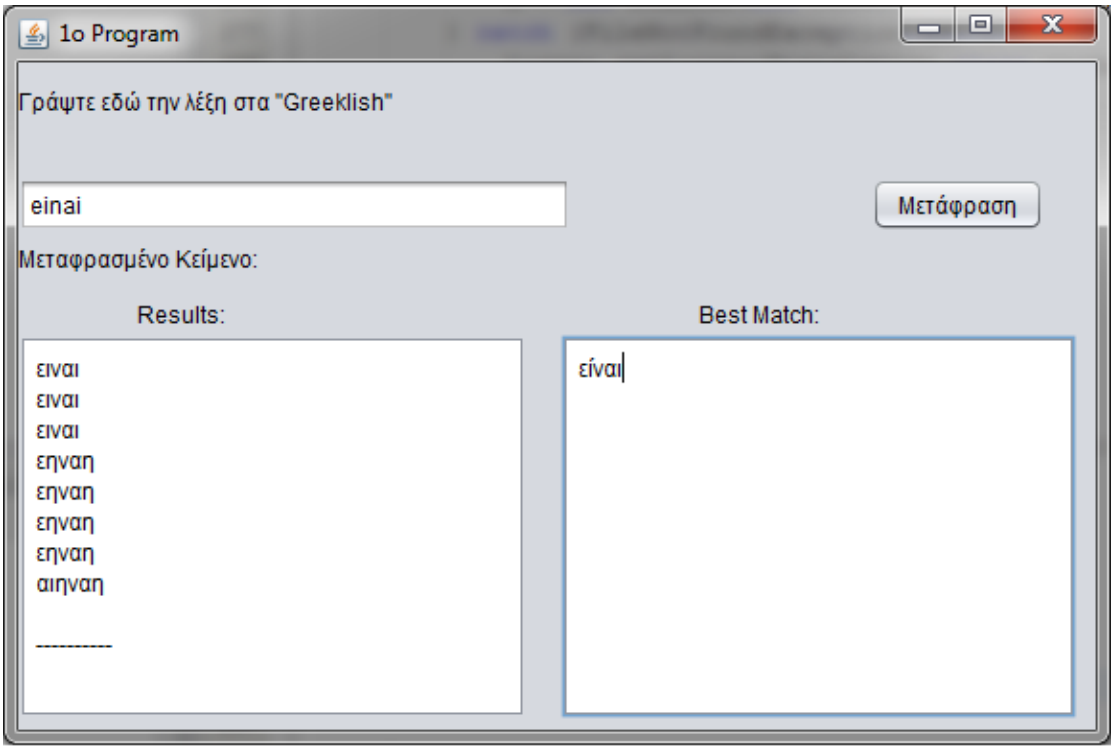
Συνεχίζοντας και έχοντας επιλέξει τα εργαλεία φτάνουμε στην υλοποίηση της εφαρμογής.

2.3 Υλοποίηση Εφαρμογής

2.3.1. Σχεδίαση

Για να καλύψουμε τις ανάγκες της διατριβής χρησιμοποιήσαμε τον κώδικα σε δύο προγράμματα παρόμοιας λειτουργίας για να μπορέσουμε να τραβήξουμε τα δεδομένα που θέλουμε.

Στο 1ο πρόγραμμα έχουμε επιλέξει το λεξικό για την αναζήτηση της καλύτερης λέξης. Στην εφαρμογή της έχει μια εικόνα σαν και αυτή:



Για να καλύψουμε την κάθε ανάγκη, αλλά και ψυχολογία και προσωπικότητα του κάθε ανθρώπου ο οποίος χρησιμοποίει τα greeklish, αναζητείται ο πιο στοχαστικός μεταγραφέας λατινοποιημένων ελληνικών σε ελληνικά.

Η υλοποίηση ενός μετατροπέα σε επίπεδο γραμμάτων και η ανάλυση της επιτυχίας αναλόγως και την τεχνική είναι ο σκοπός αυτής της έρευνας. Φυσικά, αν και με την πάροδο του χρόνου, επιστήμονες έχουν πλησιάσει το ποσοστό, ποτέ κανένας μετατροπέας δεν έχει αγγίξει το 100% της σωστής μετατροπής των προτάσεων.

Για αυτό τον λόγο και συγκρίνουμε και τις διάφορες τεχνικές για να καταδείξουμε και τα διάφορα προβλήματα που μπορούν να δημιουργηθούν από το θέμα αλλά και φυσικά και να εμφανιστούν και τρόπους επίλυσης τους, κάποιοι έχουν δοκιμαστεί με την τελευταία μέθοδο και άλλοι θα την συνεχίσουν σαν επέκταση σε άλλους τομείς.

Η μετατροπή γίνεται απευθείας με αντιστοίχηση, από greeklish προτάσεις και λέξεις σε αντίστοιχες ελληνικές. Στην περίπτωση της πρότασης για τους λατινικούς χαρακτήρες αναπτύσσει την πρόταση σε ενδείξεις για κάθε λέξη, δηλαδή χωρίζει την πρόταση και ορίζει τις λέξεις σε έναν πίνακα και έπειτα την κάθε λέξη σε σύμβολα δηλαδή για το κάθε γράμμα. Μέσω ενός χάρτη (hashmap) αντικαθιστά το κάθε χαρακτήρα - σύμβολο ακόμα και σε περίπτωση που δεν είναι μόνο λατινικό σύμβολο αλλά και κάποιος αριθμός (για παράδειγμα "3" σε "ξ").

Κατά βάση η εφαρμογή αποτελείται από πολλές παραλλαγές κώδικα java όπου κάθε μεθοδολογία ισούται και με μία εντολή. Εφαρμόζουμε της επιλογές κατά σειρά :

1. ένα προς πολλά (1:N) δηλαδή: η αντιστοιχεία ενός λατινικού γράμματος σε ένα ή και περισσότερων συνδυασμών γραμμάτων του ελληνικού αλφαβήτου

2. πολλά προς πολλά (N:M): δύο γράμματα σε ένα ή και περισσότερα

3. και ένα προς ένα (1:1): ένα σε ένα γράμμα (όχι αποτελεσματικός αλλά απαραίτητος για να εφαρμοστούν οι παραπάνω κανονές)

Μέσο των εφαρμογών και την συνάρτηση της ομοιότητας που υλοποιήσαμε μπορούμε να αντλήσουμε πληροφορίες και να συγκρίνουμε την επιτυχία μετάφρασης μεταξύ των διάφορων κανόνων και να ανακαλύψουμε τον πιο αποτελεσματικό.

Την εμφάνιση του ποσοστού επιτυχίας την έχουμε κρατήσει κρυφή για τον κάθε χρήστη αλλά εμφανίσιμη σε εμάς για να μπορέσει να μας χρησιμεύσει μόνο για την ένδειξη σύγκρισης της εργασίας αυτής.

Στην συνάρτηση αυτή έχουμε βάλει όριο της επιλογής των καταλληλότερων λέξεων το 60% σαν ποσοστό ομοιότητας λόγο της διαφοράς των τόνων των ελληνικών λέξεων που στο πρόγραμμα αυτό δεν μπορούν να υλοποιηθούν άμεσα στην μετάφραση.

Άρα εάν αυτή μία λέξη από τις προτεινόμενες έχει ποσοστό ομοιότητας μεγαλύτερο από 60% με μία από αυτές του λεξικού τότε επιλέγετε αυτόματα από το λεξικό η καταλληλότερη λέξη.

Όπως βλέπουμε στο παράδειγμα έχουμε χρησιμοποιήσει τις λέξεις «einai», «herete», «xairetai», «para8yro», «parathyro», «anthropos», «an8rwpos» και έχουμε βρει και τις υποψήφιες λέξεις αλλά και τον καλύτερο όρο στον οποίο θα αναφερθούμε σε λίγο.

ΕΙΣΑΓΩΓΗ	**ΥΠΟΨΗΦΙΕΣ ΛΕΞΕΙΣ**	**ΜΕ.Γ ΠΟΣΟΣΤΟ.**	**ΕΛΛΗΝΙΚΗ ΛΕΞΗ**	**ΠΟΣΟΣΤΟ ΟΜΟΙΟΤΗΤΑΣ**
Einai	Ειναι, εηναη, ευναυ, εειναει, εοιναοι, αιναι,	ειναι	είναι	83.3%
herete	ηερετε, χερετε, χαιραιται	χαιραιται	χαίρεται	70%
xairetai	Χαιρεται. ξαιρεται, ξερετε, ξαηρεταη, ξαιραιται	χαιρεται	χαίρεται	88.88%
Para8yro	παραθυρο, παραθειρο, παραθοιρο	παραθυρο	παράθυρο	88.88%
parathyro	Παρατηυρο, παραθυρο, παραθειρο, παραθοιρο	παραθυρο	παράθυρο	88.88%
Anthropos	Αντηροποσ, αντηροπος, αντυροπος, αντειροπος, αντοιροπος, αντοιρωπως,	αντηροπος	άνθρωπος	60.0%
An8rwpos	Ανθρωποσ, ανθρωπος, ανθρωπως,	ανθρωπος	άνθρωπος	88.88%

Αν παρατηρήσουμε και τον πίνακα βλέπουμε ότι μεγάλο πρόβλημα είναι η εμφάνιση της ίδιας πιθανής λέξης πολλές φορές. Για την επίλυση του μπορούμε να αποκλείσουμε λέξεις που εμφανίζονται επαναλαμβανόμενα αφού έχουν περάσει και από τους 3 αλγορίθμους.

Ένα ακόμα από τα φαινόμενα που εμφανίζονται είναι η έννοια ενός γράμματος να καλύπτεται με παραπάνω από μία τρόπους. Καλό παράδειγμα είναι το αγγλικό γράμμα “x” στην greeklish λέξη «xeri». Το φαινόμενο αυτό αποκαλείται overfitting καθώς ο ορισμός του είναι παραπάνω από ένα νοητό η λέξη μπορεί να μεταφράζεται ως “χέρι” ή και ως “ξέρει”.

Τέλος το μέγιστο θέμα που εμφανίζεται όπως παρατηρούμε και στον πίνακα είναι πάλι στην περίπτωση των greeklish γραμμάτων που γράφονται με διπλό χαρακτήρα (κατά βάση το"th"=θ) και έπειτα ακολουθεί σύμφωνο η μετατροπή παραμένει γράμμα προς γράμμα και το "θ" δεν μεταφράζεται σωστά με αποτέλεσμα η επιτυχία να φτάνει

μόνο στο 50%. Ο λόγος που ανακαλύφθηκε αυτό το λάθος είναι διότι οι υποψήφιες λέξεις εμφανίστηκαν κανονικά σαν "κοντινότερη" ελληνική λέξη δεν εμφανίστηκε, οπότε μετρήθηκε το ποσοστό επιτυχίας μόνο σε αυτή την λέξη και παρατηρήσαμε ότι δεν ξεπερνάει το όριο που του είχαμε αναθέσει.

Σαν λύση έχει βρεθεί μόνο για την περίπτωση που μετά από το "θ" ακολουθάει φωνήεν, στην περίπτωση ενός σύμφωνου θα μπορούσαμε να βάλουμε έναν παρόμοιο κανόνα. Όμως έτσι ακυρώνει τον κανόνα που ακολουθούν φωνήεν και για το εύρος των λέξεων που θα προσπαθήσουμε να καλύψουμε δεν θα μας προσφέρει μεγάλο συμφέρων. Προτιμήθηκε ο παρόν έλεγχος καθώς όπως αναφέραμε προηγουμένως η προτίμηση του κοινού για την χρήση των greeklish είναι αυτή των αριθμών και έτσι έχουμε καλύτερα αποτελέσματα.

Σαν επόμενο βήμα εφαρμόζουμε και με μεγάλη αποτελεσματικότητα, κατά την επιλογή του χρήστη (στην προκειμένη περίπτωση εμάς), τους κανόνες αυτούς σε κείμενα για να αναλύσουμε και αλλού την αποτελεσματικότητα τους.

2.3.2. Evaluation της εφαρμογής

Περνάμε των κώδικα από ένα evaluation, δηλαδή την χρήση του και την επίδειξη του σε διαφορετικά κείμενα.

Το evaluation υλοποιείται από τρεις προτάσεις μία από την εφαρμογή της ΕΛΟΤ, μία από την Innoetics και μία από το all-greek-to-me. Κάθε μετατροπέας έχει ξεχωριστό τρόπο υλοποίησης εναλλαγής των κειμένων οπότε έτσι καλύπτουμε διάφορες περιπτώσεις συσχετίσεων. Γενικά, πήραμε ελληνικές προτάσεις και τις εναποθέσαμε σαν μετάφραση στα διαδικτυακά πρόγραμμα αυτών των εταιριών, από ελληνικά σε greeklish την αντίστροφη διαδικασία δηλαδή. Έχουμε διαθέσιμο λοιπόν τον δικό τους τρόπο εφαρμογής και θα τον συγκρίνουμε με αυτών που αναπαράχθηκε για την εργασία αυτή. Το φαινόμενο αυτό αναλύεται και σε παρακάτω κεφάλαιο.

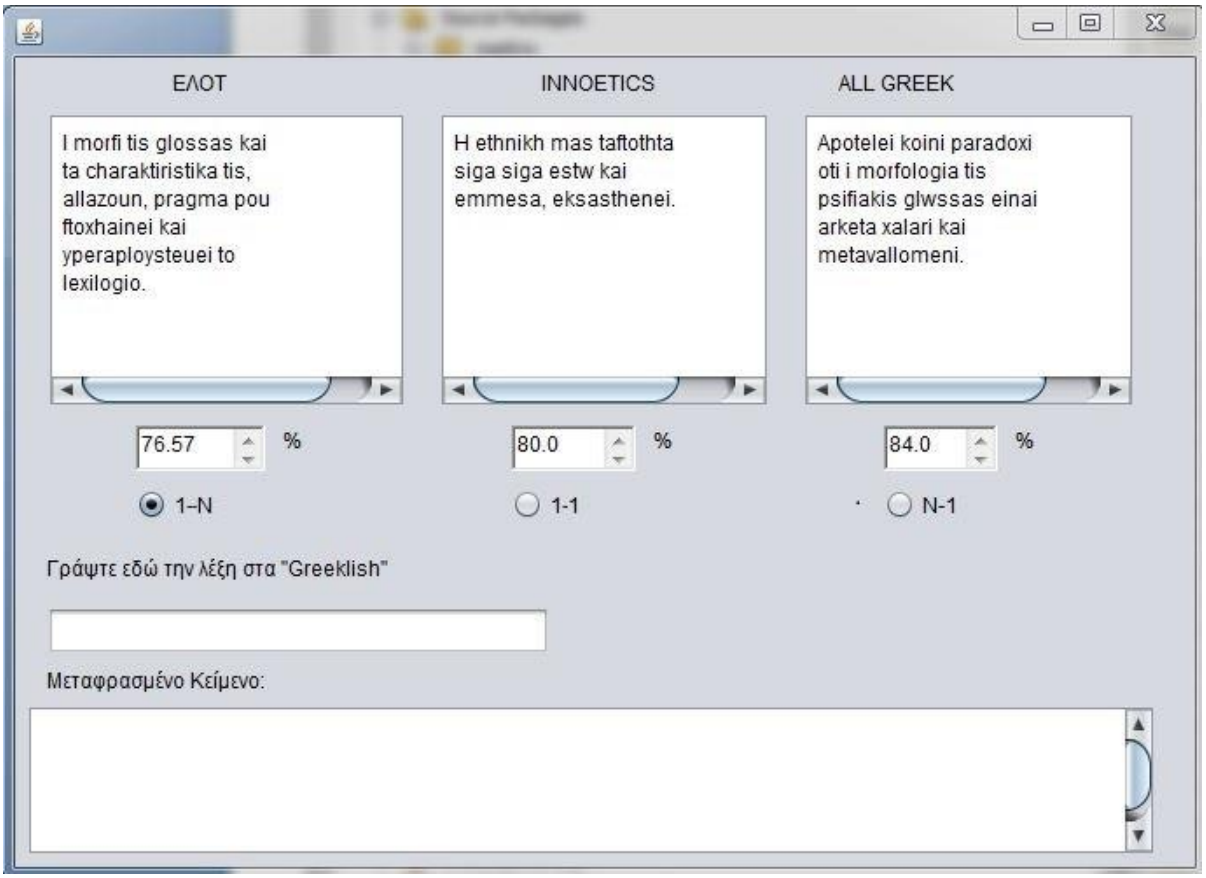
Έχοντας σαν κανόνα το μεταφρασμένο κείμενο με το σύστημα της ΕΛΟΤ 743, (καθώς όπως αναφέραμε είναι το μόνο επίσημο μεταφραστικό πρόγραμμα για την ελληνική γλώσσα), την μετάφραση της Innoetics μία από τις καλύτερες μεταφραστικές μηχανές greeklish σε ελληνικά και την all-greek-to me! την δεύτερη καλύτερη, και με βάση αυτές αναλύουμε τις μεθόδους μία-μία για να ανακαλύψουμε ποιά είναι η πιο αποτελεσματική.

Οι μέθοδοι μετατροπής 1-1(ένα προς ένα), 1-N(ένα προς δύο) και N-1(δύο προς ένα) διασπούνται σε ξεχωριστές εντολές για να δούμε τι αντίδραση έχουν στις 3 προτάσεις μετά από κάθε τους επεξεργασία.

Για να αντλήσουμε την αποτελεσματικότητα τους χρησιμοποιήθηκε η ίδια συνάρτηση ομοιότητας αλλά, αντί του λεξικού βάλαμε σαν μέτρο τις ίδιες, σωστά μεταφρασμένες, προτάσεις στα ελληνικά (φυσικά κρυφά από το **user interface).**

Το πλαίσιο εισαγωγής της λέξης παρέμεινε ίδιο απλά για καλαίσθητους λόγους. Αν κάποιος χρήστης εισάγει κάποια λέξη τότε θα εμφανιστεί μία αντίστοιχη μεταφρασμένη. Σε αυτή την περίπτωση μπορεί να πληκτρολογηθεί ολόκληρη πρόταση και όχι μόνο λέξη. Στην παρούσα περίπτωση το πλαίσιο αυτό έχει μείνει κενό.

Για την 2η εφαρμογή αναπτύσσουμε μία εικόνα σαν και αυτή:



Ο παρακάτω πίνακας μας επιδεικνύει όλες τις πληροφορίες που χρειαζόμαστε:

	ΕΛΟΤ	**INNOETICS**	**ALL GREEK**
1-N	76.57%	80.0%	84.0%
1-1	73.54%	79.25%	83.47%
N-1	64.3%	69.92%	75.61%

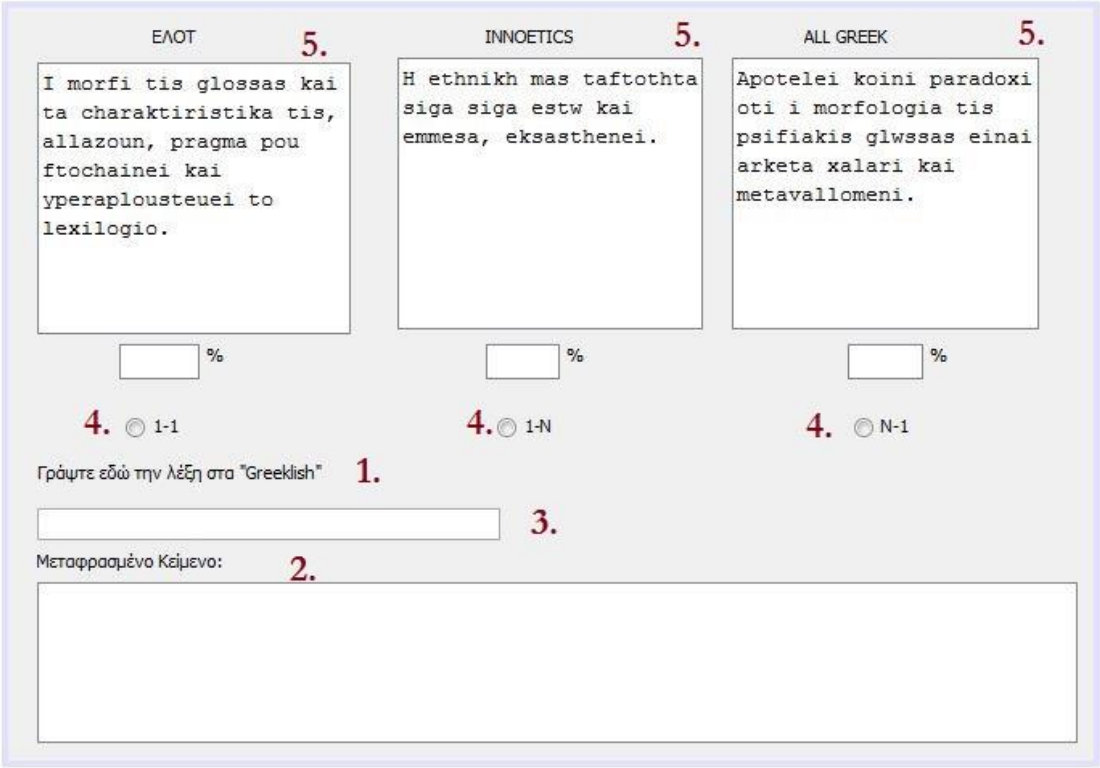
Παρατηρείτε πως η πρώτη μέθοδος είναι η αποτελεσματικότερη και ότι μεγάλο ρόλο παίζει το φαινόμενο που αναφέραμε προηγουμένως με τον διπλό χαρακτηρισμό ενός ελληνικού γράμματος. Για την συγκεκριμένη περίπτωση το πρόβλημα εμφανίζεται στο γράμμα "χ" παρόλα αυτά όμως, με την συνάρτηση της σύγκρισης βλέπουμε ότι το πρόγραμμα που δημιουργήθηκε δείχνει πολύ καλά αποτελέσματα.

Τέλος αν και όχι διαθέσιμο σαν συμπληρωματική διαδικασία, προσπαθήσαμε να εφαρμόσουμε την επιλογή της εκμάθησης κάποιων συνδυασμών λέξεων σε ένα αρχείο txt έτσι ώστε όταν εμφανίζεται παρόμοια λέξη να μεταφράζεται άμεσα και να αποφύγουμε το σοβαρό πρόβλημα που εντοπίστηκε και στις δύο εφαρμογές. Η τεχνική όμως θέλει καλύτερη εξειδίκευση και περισσότερους πόρους. Ακολουθεί ανάλυση τον πιο κύριων σημείων του κώδικα και έπειτα τα συμπεράσματα.

2.3.3. Διαδικασία

Μία πιο αναλυτική προσέγγιση στον κώδικα:

Για αρχή με την βοήθεια του Netbeans φτιάχτηκε πρώτα το παράθυρο για την μετατροπή.



1. ετικέτα για την περιγραφή του text field & text area

2. Textarea για την εμφάνιση της απάντησης

3. Textfield για το input της εφαρμογής

4.JRadiobutton για την εκτέλεση

5.Textarea για την δήλωση των σταθερών κειμένων.

Περιλιπτικά:

Στα πέντε πρώτα πλαίσια (5) αναφέρονται οι τρεις προτάσεις που χρησιμοποιήσαμε για το cross-validation και παρακάτω στα μικρά πλαίσια το ποσοστό επιτυχίας τους ανάλογα των αλγόριθμο που θα είναι επιλεγμένος.

Τα κουμπιά επιλογής (radio buttons)(4) επιλέγουν ποιος αλγόριθμος θα είναι σε ισχύ.

Κώδικας πλαισίου:

```
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jRadioButton1 = new javax.swing.JRadioButton();
    jRadioButton2 = new javax.swing.JRadioButton();
    jRadioButton3 = new javax.swing.JRadioButton();
    jLabel4 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    resultarea = new javax.swing.JTextArea();
    inputfield = new javax.swing.JTextField();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
    jScrollPane2 = new javax.swing.JScrollPane();
    jTextPane1 = new javax.swing.JTextPane();
    jScrollPane3 = new javax.swing.JScrollPane();
    jTextPane2 = new javax.swing.JTextPane();
    jScrollPane4 = new javax.swing.JScrollPane();
    jTextPane3 = new javax.swing.JTextPane();
    jScrollPane5 = new javax.swing.JScrollPane();
    jTextArea1 = new javax.swing.JTextArea();
    jScrollPane6 = new javax.swing.JScrollPane();
    jTextArea2 = new javax.swing.JTextArea();
    jScrollPane7 = new javax.swing.JScrollPane();
    jTextArea3 = new javax.swing.JTextArea();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jLabel1.setText("ΕΛΟΤ");

    jLabel2.setText("INNOETICS");

    jLabel3.setText("ALL GREEK");
```

```
jRadioButton1.setText("1-1");
jRadioButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton1ActionPerformed(evt);
    }
});

jRadioButton2.setText("1-N");

jRadioButton3.setText("N-1");

jLabel4.setText("Γράψτε εδώ την λέξη στα \"Greeklish\"");

resultarea.setColumns(20);
resultarea.setLineWrap(true);
resultarea.setRows(5);
jScrollPane1.setViewportView(resultarea);

inputfield.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        inputfieldActionPerformed(evt);
    }
});

jLabel5.setText("Μεταφρασμένο Κείμενο: ");

jLabel6.setText("%");

jLabel7.setText("%");

jLabel8.setText("%");

jScrollPane2.setViewportView(jTextPane1);
```

```
jScrollPane3.setViewportView(jTextPane2);

jScrollPane4.setViewportView(jTextPane3);

jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jTextArea1.setText("Apotelei koini paradochi oti i morfologia tis psifiakis
jScrollPane5.setViewportView(jTextArea1);

jTextArea2.setColumns(20);
jTextArea2.setRows(5);
jTextArea2.setText("H ethnikh mas taftothta\nsiga siga estw kai \nemmesa, ek
jScrollPane6.setViewportView(jTextArea2);

jTextArea3.setColumns(20);
jTextArea3.setRows(5);
jTextArea3.setText("Apotelei koini paradoxi\noti i morfologia tis \npsifiaki
jScrollPane7.setViewportView(jTextArea3);
```

Στην συνέχεια ενεργοποιούμε το textfile για να εισάγει την πρόταση που θα του δώσει ο χρήστης.

Ελέγχει αν ο χρήστης δεν τοποθέτησε σωστά η άφησε άδειο το σημείο με μία εξαίρεση και έπειτα δίνει την εντολή. Επίσης είναι ενεργοποιημένη και η εντολή να δοθεί από το πληκτρολόγιο με το κουμπί "Enter".

```
public Translation() {
    initComponents();
    KeyListener klb = new KeyListener(){
        @Override
        public void keyPressed(KeyEvent e) {
            int c = e.getKeyCode();
            if(c==10)
            {
                String input= inputfield.getText();
                if(input.equals(""))
                    JOptionPane.showMessageDialog(null,"Παρακαλώ εισάγετε μία φράση.","Προσοχή!",JOptionPane.ERROR_MESSAGE);
                else
                    translate(input);
            }
        }

        @Override
        public void keyTyped(KeyEvent ke) {
        }

        @Override
        public void keyReleased(KeyEvent ke) {
        }


    };
    inputfield.addKeyListener(klb);
}
```

Αυτό τοποθετεί την πρόταση σε μία μεταβλητή τύπου input.

```
String input= inputfield.getText();
if(input.equals(""))
    JOptionPane.showMessageDialog(null,"Παρακαλώ εισάγετε μία φράση.","Προσοχή!",JOptionPane.ERROR_MESSAGE);
else
    translate(input);   // TODO add your handling code here:
```

Η input πρόταση χωρίζεται λέξη με λέξη σε σημεία με μεταβλητή τύπου string tokens και μετά ο κάθε χαρακτήρας σε σύμβολα (symbolsets) με μεταβλητή Arraylist letters.

```
public void translate (String input){
    final Map<String,String> mymap;

    ArrayList<ArrayList<String>> letters = new ArrayList<>();

    String[] tokens = input.split("");

    ArrayList<String> temp = new ArrayList<>();
    for(String token:tokens){
        if(token.equals(" ")){
            letters.add(temp);
            temp = new ArrayList<>();
        }else{
            temp.add(token);
        }
    }
    letters.add(temp);
```

Αφού δημιουργηθεί το map τα σύμβολα εισάγονται μέσω του map και αντικαθιστούνται με τα αντίστοιχα ελληνικά τα οποία δηλώνονται σε έναν πίνακα lettersgr.

```
mymap= new HashMap<String,String>();
createmap(mymap);

ArrayList<ArrayList<String>> lettersgr = new ArrayList<>();
for(ArrayList<String> b:letters){
    ArrayList<String> temp2= new ArrayList<>();

    for(String p:b){
        if(mymap.containsKey(p))
        {
            temp2.add(mymap.get(p).toString());


        }
    }
    lettersgr.add(temp2);
}
```

Η κλάση του χάρτη και οι αντιστοιχίες στου χαρακτήρες.

```
private static void createmap(Map<String,String> map){

    map.put("a","α");
    map.put("b", "β");
    map.put("v", "β");
    map.put("g", "γ");    map.put("A","A");
    map.put("d", "δ");    map.put("B", "B");
    map.put("e", "ε");    map.put("V", "B");
    map.put("z", "ζ");    map.put("G", "Γ");
    map.put("h", "η");    map.put("D", "Δ");
    map.put("8", "θ");    map.put("E", "E");
    map.put("9", "θ");    map.put("Z", "Z");
    map.put("i", "ι");    map.put("H", "H");
    map.put("c", "κ");    map.put("I", "I");
    map.put("k", "κ");    map.put("C", "K");
    map.put("l", "λ");    map.put("K", "K");
    map.put("m", "μ");    map.put("L", "Λ");
    map.put("n", "ν");    map.put("M", "M");
    map.put("3", "ξ");    map.put("N", "N");
    map.put("o", "ο");    map.put("O", "O");
    map.put("p", "π");    map.put("P", "Π");
    map.put("r", "ρ");    map.put("R", "P");
    map.put("s", "σ");    map.put("S", "Σ");
    map.put("t", "τ");    map.put("T", "T");
    map.put("u", "υ");    map.put("U", "Y");
    map.put("y", "υ");    map.put("Y", "Y");
    map.put("f", "φ");    map.put("F", "Φ");
    map.put("x", "χ");    map.put("X", "X");
    map.put("w", "ω");    map.put("W", "Ω");
```

Οι καινούργιοι χαρακτήρες ενώνονται στην μεταγραφόμενη πρόταση με την χρήση της συνάρτησης του append το οποίο "κολλάει" κάθε string πίσω από το άλλο για να δημιουργήσει μια πρόταση. Έπειτα διάφορες συνθήκες ελέγχουν την ορθότητα της μετατροπής σε περίπτωση λόγου χάρη εμφάνισης ενός αριθμού και την σωστή αντιστοιχία. Δηλαδή οι κανόνες μετατροπής 2 ή παραπάνω γραμμάτων σε 1.

```
StringBuilder sb = new StringBuilder();
for(ArrayList<String> a:lettersgr){
    for(String t:a){
        sb.append(t);
    }
    sb.append(" ");
}
String phrasegr= sb.toString();
String[] vowel={"α","ε","υ","ι","ο","η","ω",};
for(String s:vowel){

    phrasegr=phrasegr.replaceAll("τη"+s, "θ"+s);
    phrasegr=phrasegr.replaceAll("Τη"+s, "Θ"+s);

}
phrasegr=phrasegr.replaceAll("κσ", "ξ");
phrasegr=phrasegr.replaceAll("πσ", "ψ");
phrasegr=phrasegr.replaceAll("Κσ", "Ξ");
phrasegr=phrasegr.replaceAll("Πσ", "Ψ");
phrasegr=phrasegr.replaceFirst("^θ","Θ");

phrasegr=phrasegr.replaceFirst("^ξ","Ξ");
phrasegr=phrasegr.replaceAll("σ ", "ς ");

phrasegr=phrasegr.replaceAll(" ετκ "," ετσι ");

phrasegr=phrasegr.replaceAll("ετκ "," ετσι ");


resultarea.append(phrasegr);
resultarea.append("\n----------\n");
```

Τέλος με την βοήθεια του scannerfile το πρόγραμμα διαβάζει το λεξικό με τους ελληνικούς τύπους και διορθώνει τυχών ορθογραφικά λάθη, αυτό κυρίως συμβαίνει με το φαινόμενο του "ιωτακισμού" που όλα τα φωνητικά "i"(SAMBA,1982) γράφονται και με το λατινικό γράμμα "i".

```
    String detemp= "";
     Scanner inFile1 = new Scanner(new File("c:/indic.txt")).useDelimiter(",\\s*");
     List<String> temps = new ArrayList<String>();

   while (inFile1.hasNext()) {

  detemp = inFile1.next();
  temps.add(detemp);
}
inFile1.close();
String[] dicArray = temps.toArray(new String[0]);
```

Κώδικας εκτέλεσης της κλάσης.

```
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new Translation().setVisible(true);
    }
});
```

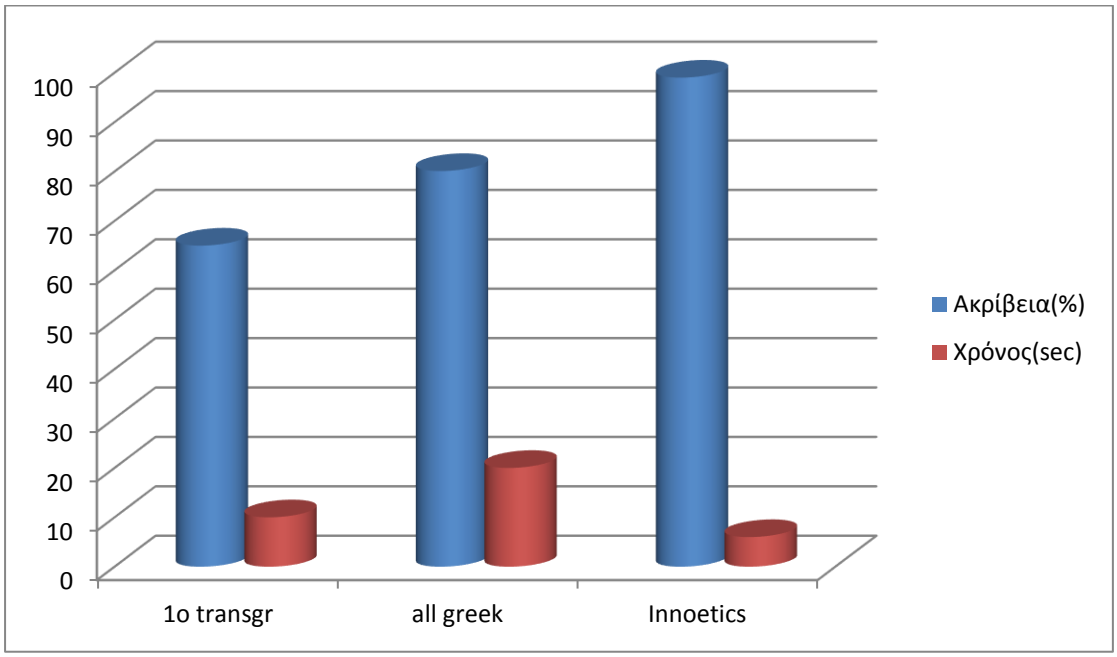
Αφού είδαμε για την εκτέλεση της εφαρμογής τώρα θα δούμε τι συμπεράσματα βγάζουμε.

ΕΠΙΛΟΓΟΣ ΚΕΦΑΛΑΙΟ 3

3.1 Συμπεράσματα

Έπειτα από πολλές έρευνες και πολλές δοκιμές με όλες τις τεχνικές και όλα τα προγράμματα καταλήγουμε ότι η Innoetics έχει σχεδόν τελειοποιήσει πολλά ζητήματα τα οποία προκύπτουν με άλλα προγράμματα.

Βλέποντας παρακάτω:



Ο μέσος όρος επιτυχίας του δικού μας προγράμματος (ονόματι 1ο transgr) βρέθηκε με βάση τα ποσοστά και στις 2 εφαρμογές(μικρότερο 55% μέγιστο 84%). Για τα υπόλοιπα προγράμματα αντλήθηκαν από το διαδίκτυο.

Ο αναπτυγμένος αλγόριθμος λειτούργησε πολύ καλά και ορθά σε σχέση με τους άλλους δύο, δεδομένου του τρόπου δημιουργίας του, καλύπτει και τις περιπτώσεις διπλού γράμματος, αλλά και αριθμού. Βασικό μειονέκτημα είναι ότι η μετάφραση για να γίνει πιο ακριβείς πρέπει να αποκλείει λέξεις που δεν μπορεί να μετατρέψει (μία περίπτωση η εμφάνιση πολλών "ι" σε μία λέξη) , πόσο μάλλον και για το πρόβλημα που συναντήσαμε παραπάνω. Όπως αναφέραμε όμως δεν καλύπτει την εφαρμογή μας και σε στην περίπτωση προτάσεων τα ποσοστά επιτυχίας μετάφρασης θα έπεφταν.

Η Innoetics έχει αναπτύξει αλγόριθμο ο οποίος "μαθαίνει" τυχών λέξεις που συναντάει πρώτη φορά , αυτό έχει σαν αποτέλεσμα, την επόμενη φορά που θα τύχει κάποιον συνδυασμό λέξης με λατινικούς χαρακτήρες πιο σπάνιο, να γνωρίζει την

σωστή μετατροπή του. Μία τέτοια εταιρία που έχει ασχοληθεί σε πολύ περισσότερο βάθος χρόνου είναι πολύ λογικό να κατάφερε να βελτιστοποιήσει μία τέτοια εφαρμογή.

Ασχέτως όμως την εφαρμογή ή τον αλγόριθμο καταλήγουμε στο πιο βασικό συμπέρασμα: ο ανθρώπινους νους δεν θα πάψει να σκαρφίζεται τρόπους για την γραφή αυτής της ψηφιακής γλώσσας, όποτε όσος χρόνος και να περάσει, το να κατορθώσουμε να αγγίξουμε το 100% της ακρίβειας στην μετατροπή από Greeklish σε Ελληνικά είναι ακατόρθωτο.

3.2 Χρήση Εφαρμογής

Αυτές οι εφαρμογές έχουν πολλαπλές χρήσεις και στην κοινωνία και γενικότερα για την διευκόλυνση καταστάσεων. Μπορεί ο οποιοσδήποτε και οπουδήποτε να χρησιμοποιηθούν για παράδειγμα σε κάποια διαδικτυακά φόρουμ, σε ιστοσελίδες που χρήστες επικοινωνούν μεταξύ τους, σε μέσα κοινωνικής δικτύωσης. Κατά κόρον για να φαίνεται και ο ισότοπος πιο ελκυστικός αλλά και λειτουργικός αφού πολλές φορές μια μηχανή αναζήτησης δεν ανταποκρίνεται αποτελεσματικά στην εμφάνιση των greeklish.

Ακόμα υπάρχει και η εκδοχή για μαθησιακούς σκοπούς όπου μαθητές μπορούν να ασχοληθούν με την ανάπτυξη και της γλωσσολογίας αλλά και της πληροφορικής. Όπου και να βοηθήσουν κάποιο άτομο με βοηθητικές μαθησιακές ανάγκες ή και σε εξοικείωση καλύτερης πληκτρολόγησης.

3.3 Μελλοντική αναβάθμιση Εφαρμογής

Μελλοντικά σχέδια για την περεταίρω ανάπτυξης της εφαρμογής δεν είναι τίποτε άλλο από την βελτιστοποίηση του μετατροπέα, στο να γίνεται πιο ορθογραφημένα και να καλύπτει ακόμα περισσότερες πιθανότητες γραφήματος μια λέξης σε greeeklish.

Επίσης θα μπορούσε να αναπτυχτεί στο θέμα με των threats δηλαδή των πολλαπλών λειτουργιών έτσι ώστε το πρόγραμμα να δουλεύει παράλληλα με μια άλλη

λειτουργία σε έναν υπολογιστή για την πιο άμεση εξυπηρέτηση και φυσικά την καλυτέρευση της ικανότητας του χρηστή.

Γλώσσες όπως τα Greeklish είναι ευρέως γνωστές για την λειτουργία τους και την διευκόλυνση που παρέχουν. Όλοι οι άνθρωποι έχουν την απαίτηση να εκτελούνται οι εντολές το δυνατόν γρηγορότερα ειδικά σε ότι έχει να κάνει με το διαδίκτυο. Δεν παύει όμως να αλλοιώνει την κανονική γλώσσα και να μειώνει την ιστορία της.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Accessible Culture, http://accessibleculture.org

2. Androutsopoulos, J. 1999. Lating-greek orthography in electronic mails: Use and stances. Technical report, Aristotle University of Thessaloniki.

3. ASDA: Greek-to-Greeklish converter, http://home.asda.gr/active/ GrLish2.asp

4. Blog greeklish, http://greeklish.blog.com/

5. Blog greeklishproj, greeklishproj4.blogspot.gr

6. Chalamandaris, A.; Protopapas, A.; Tsiakoulis, P.; andRaptis, S. 2006. All greek to me! an automatic greeklishto greek transliteration system. In *5th International Conferenceon Language Resources and Evaluation (LREC)*.

7. DeGreeklish, http://tools.wcl.ece.upatras.gr/degreeklish

8. ELOT (1982), Greek Organisation of Standardization

9. e-Chaos: freeware Greeklish converter, http://www.paraschis.gr/files.php

10. Galatas, G. 2008. Greeklish to greek converter based on lexicon structures. Master's thesis, University of Patras.

11. Greeklish to Greek converter, http://www.greeklish.net/

12. Greek to Greeklish by Innoetics, http://services.innoetics.com/greeklish/

13. Kotimas Elias,Lyras P. Dimitrios, Kyriakos Sgarbas, Fakotakis Nikos, 2010, A Stochastic Greek-to-Greeklish Transcriber Modeled By Real User Data

14. Ken Arnold, James Gosling, David Holmes,2007, The Java Programming Language

15. Lyras P. Dimitrios, Σεπτέμβριος 2010, Παραμετροποιήση Στοχαστικών Μεθόδων Εξόρυξης Γνώσης από Δεδομένα, Μετασχηματισμούς Συμβολοσειρών και Τεχνικών Συμπερασματικού Λογικού Προγραμματισμού

16. Netbeans,www.netbeans.org

17. Oracle, https://www.oracle.com/gr

18. OpenOffice, www. openoffice.org

19. Spyros Balnas,2009, gr2ελ: A Greeklish-to-Greek converter

20. Wikipedia: Greeklish, https://el.wikipedia.org/wiki/Greeklish

21. Wikispaces: greeklishteam, greeklishteam.wikispaces.com

ΠΑΡΑΡΤΗΜΑ Α:

Ο πλήρης κώδικας της εφαρμογής

Παρακάτω ο κώδικας στην γλώσσα προγραμματισμού Java.

-κώδικας-

```
package metrafrasths;

import java.io.File;
import java.io.FileNotFoundException;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import javax.swing.JOptionPane;

/**
 *
 * @author Kefka
 */
public class Translation extends javax.swing.JFrame {

    /**
     * Creates new form Transaltion
```

```
 */
public Translation() {
    initComponents();
    KeyListener klb = new KeyListener(){
        @Override
        public void keyPressed(KeyEvent e) {
            int c = e.getKeyCode();
            if(c==10)
            {
                String input= inputfield.getText();
                if(input.equals(""))
                    JOptionPane.showMessageDialog(null,"Παρακαλώ εισάγετε μία φράση.","Προσοχή!",JOptionPane.ERROR_MESSAGE);
                else
                    translate(input);
            }
        }

        @Override
        public void keyTyped(KeyEvent ke) {
        }

        @Override
        public void keyReleased(KeyEvent ke) {
        }
```

```
    };
    inputfield.addKeyListener(klb);
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jButton1 = new javax.swing.JButton();
    inputfield = new javax.swing.JTextField();
    jScrollPane1 = new javax.swing.JScrollPane();
    resultarea = new javax.swing.JTextArea();
    jLabel2 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("1o Program");
    setBackground(new java.awt.Color(204, 255, 204));

    jLabel1.setText("Γράψτε εδώ την λέξη στα \"Greeklish\"");

    jButton1.setText("Μετάφραση");
```

```
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        inputfield.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                inputfieldActionPerformed(evt);
            }
        });

        resultarea.setColumns(20);
        resultarea.setLineWrap(true);
        resultarea.setRows(5);
        jScrollPane1.setViewportView(resultarea);

        jLabel2.setText("Μεταφρασμένο Κείμενο: ");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
```

```
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel1)
                    .addGap(0, 0, Short.MAX_VALUE))
                .addGroup(layout.createSequentialGroup()
                    .addComponent(inputfield,
javax.swing.GroupLayout.PREFERRED_SIZE,                    288,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,      80,
Short.MAX_VALUE)
                    .addComponent(jButton1)))
            .addContainerGap())
        .addGroup(layout.createSequentialGroup()
            .addComponent(jLabel2,     javax.swing.GroupLayout.PREFERRED_SIZE,
178, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(0, 0, Short.MAX_VALUE))
        .addComponent(jScrollPane1)
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(12, 12, 12)
            .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                .addComponent(jButton1)
```

```
                .addComponent(inputfield,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

            .addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

            .addComponent(jScrollPane1,  javax.swing.GroupLayout.DEFAULT_SIZE,
235, Short.MAX_VALUE))

        );


        pack();

        setLocationRelativeTo(null);

    }// </editor-fold>


    private void inputfieldActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

    }


    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        String input= inputfield.getText();

        if(input.equals(""))

            JOptionPane.showMessageDialog(null,"Παρακαλώ εισάγετε μία
φράση.","Προσοχή!",JOptionPane.ERROR_MESSAGE);

        else

            translate(input);   // TODO add your handling code here:

    }

    public void translate (String input){
```

```
final Map<String,String> mymap;

ArrayList<ArrayList<String>> letters = new ArrayList<>();

String[] tokens = input.split("");

ArrayList<String> temp = new ArrayList<>();
for(String token:tokens){
   if(token.equals(" ")){
      letters.add(temp);
      temp = new ArrayList<>();
   }else{
      temp.add(token);
   }
}
letters.add(temp);
mymap= new HashMap<String,String>();
createmap(mymap);

ArrayList<ArrayList<String>> lettersgr = new ArrayList<>();
for(ArrayList<String> b:letters){
   ArrayList<String> temp2= new ArrayList<>();

   for(String p:b){
      if(mymap.containsKey(p))
      {
         temp2.add(mymap.get(p).toString());
```

```
        }
    }
    lettersgr.add(temp2);
}


StringBuilder sb = new StringBuilder();
for(ArrayList<String> a:lettersgr){
    for(String t:a){
        sb.append(t);
    }
    sb.append(" ");
}
String phrasegr= sb.toString();
String[] vowel={"α","ε","υ","ι","ο","η","ω",};
for(String s:vowel){

    phrasegr=phrasegr.replaceAll("τη"+s, "θ"+s);
    phrasegr=phrasegr.replaceAll("Τη"+s, "Θ"+s);

}
phrasegr=phrasegr.replaceAll("κσ", "ξ");
phrasegr=phrasegr.replaceAll("πσ", "ψ");
phrasegr=phrasegr.replaceAll("Κσ", "Ξ");
phrasegr=phrasegr.replaceAll("Πσ", "Ψ");
```

```
    phrasegr=phrasegr.replaceFirst("^θ","Θ");

    phrasegr=phrasegr.replaceFirst("^ξ","Ξ");
    phrasegr=phrasegr.replaceAll("σ ", "ς ");




    resultarea.append(phrasegr);
    resultarea.append("\n----------\n");

}
private static void createmap(Map<String,String> map){

    map.put("a","α");
    map.put("b", "β");
    map.put("v", "β");
    map.put("g", "γ");
    map.put("d", "δ");
    map.put("e", "ε");
    map.put("z", "ζ");
    map.put("h", "η");
    map.put("8", "θ");
    map.put("9", "θ");
    map.put("i", "ι");
    map.put("c", "κ");
```

```
map.put("k", "κ");
map.put("l", "λ");
map.put("m", "μ");
map.put("n", "ν");
map.put("3", "ξ");
map.put("o", "ο");
map.put("p", "π");
map.put("r", "ρ");
map.put("s", "σ");
map.put("t", "τ");
map.put("u", "υ");
map.put("y", "υ");
map.put("f", "φ");
map.put("x", "χ");
map.put("w", "ω");

map.put("A","Α");
map.put("B", "Β");
map.put("V", "Β");
map.put("G", "Γ");
map.put("D", "Δ");
map.put("E", "Ε");
map.put("Z", "Ζ");
map.put("H", "Η");
map.put("I", "Ι");
map.put("C", "Κ");
map.put("K", "Κ");
```

```
        map.put("L", "Λ");
        map.put("M", "M");
        map.put("N", "N");
        map.put("O", "O");
        map.put("P", "Π");
        map.put("R", "P");
        map.put("S", "Σ");
        map.put("T", "T");
        map.put("U", "Y");
        map.put("Y", "Y");
        map.put("F", "Φ");
        map.put("X", "X");
        map.put("W", "Ω");
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
        * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
        */
        try {
```

```
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(Translation.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(Translation.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(Translation.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Translation.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    }
    //</editor-fold>
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
```

```
            new Translation().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JTextField inputfield;
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextArea resultarea;
// End of variables declaration
}
```

-τέλος κώδικα-