ΕΠΕΞΕΡΓΑΣΙΑ ΦΩΝΗΣ ΚΑΙ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

ΠΡΟΠΑΡΑΣΚΕΥΗ 1ΗΣ ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ:

ΜΕΤΑΤΡΟΠΕΑΣ ΓΙΑ GREEKLISH

Μελίστας Θωμάς *03114149*
Κουμούτσου Δήμητρα *03114022*

Για την υλοποίηση της άσκησης μας δίνεται ένας φάκελος με data τα οποία επεξεργαζόμαστε στα επόμενα βήματα. Για κάθε βήμα που ακολουθεί έχουμε δημιουργήσει ξεχωριστά αρχεία με κώδικα σε python, έχουμε εντολές που δίνονται στο shell με τη χρήση του εργαλείου openfst και τελικά προκύπτουν τα επιθυμητά αρχεία με τα αποτελέσματα. Η διαδικασία που ακολουθούμε σε κάθε βήμα περιγράφεται αναλυτικά στη συνέχεια.
Για να εκτελεστούν όλα τα βήματα μαζί, έχοντας μόνο τα αρχεία data, έχουμε δημιουργήσει ένα shell script (*run.sh*), που υλοποιεί το σύνολο των python αρχείων και fstopen εντολών.

ΒΗΜΑ 1

(Κώδικας train.py)

Θέλουμε να αντιστοιχίσουμε τους ελληνικούς με τους λατινικούς χαρακτήρες από τα αρχεία train_gr.txt και train_greng.txt που δίνονται.
Αφού ανοίξουμε τα αρχεία, χωρίζουμε σε λέξεις.
Δημιουργούμε μια λίστα με λίστες όπου καταχωρούνται όλες οι αντιστοιχίσεις.
Δημιουργούμε επίσης ένα αγγλικό λεξικό το οποίο «γεμίζουμε» καθώς διασχίζουμε το αρχείο, με τα γράμματα και τους διφθόγγους των λατινικών. Για να γίνει η καταχώρηση στη λίστα αντιστοιχίζουμε κάθε γράμμα από το αγγλικό λεξικό στη λίστα με το engnum του.

Αρχικά ελέγχουμε αν η λέξη από το ελληνικό αρχείο και το λατινικό ταυτίζονται, οπότε καταλαβαίνουμε ότι έχουμε μια αγγλική λέξη στο κείμενο greenglish, και δεν υπάρχει κάποια αντιστοιχία να γίνει σε αυτή την περίπτωση, αφού οι αγγλικές λέξεις διατηρούνται.
Έπειτα, ελέγχουμε αν το μήκος της ελληνικής λέξης είναι ίδιο με της λέξης σε greenglish.
Πρώτα εξετάζουμε την περίπτωση που οι δύο λέξεις έχουν ίδιο μήκος, οπότε θεωρούμε ότι υπάρχει 1-1 αντιστοίχιση μεταξύ των γραμμάτων στην ελληνική και τη greenglish λέξη.
Κινούμαστε γράμμα-γράμμα και αποθηκεύουμε στη λίστα, με την ακόλουθη διαδικασία.
Ελέγχουμε αν το αγγλικό λεξικό περιέχει το γράμμα που διαβάστηκε. Αν το περιέχει, προσθέτω το ελληνικό γράμμα στον οποίο έγινε μετατροπή στη θέση της λίστας που αντιστοιχεί στο συγκεκριμένο λατινικό χαρακτήρα. Αλλιώς προσθέτουμε πρώτα στο λεξικό το λατινικό χαρακτήρα και μετά προσθέτουμε στη λίστα τον ελληνικό χαρακτήρα στον οποίο μετατράπηκε.

Στη συνέχεια εξετάζουμε την περίπτωση που η ελληνική λέξη και η greenglish λέξη δεν έχουν ίδιο πλήθος χαρακτήρων. Αυτό σημαίνει ότι οι χαρακτήρες δεν μπορούν να αντιστοιχιστούν ακριβώς, αλλά οι αντιστοιχίσεις θα γίνονται 1-2 ή 2-1 (αναλογία ελληνικών/αγγλικών χαρακτήρων).
Πρώτα, στην περίπτωση που η ελληνική λέξη έχει λιγότερα γράμματα, σημαίνει ότι ένα ελληνικό γράμμα αντιστοιχίζεται σε 2 λατινικά. Ψάχνουμε τη λέξη γράμμα-γράμμα για να βρούμε το δίφθογγο. Αν το γράμμα που συναντήσουμε ανήκει σε ένα πλήθος γραμμάτων (που εξηγούμε παρακάτω) και προχωράμε στο επόμενο γράμμα. Όταν βρεθεί άλλο γράμμα, κρατάμε αυτό και το επόμενο του. Αποθηκεύουμε τις αντιστοιχίσεις με ανάλογο τρόπο με προηγουμένως, δηλαδή αν ο «λατινικός» δίφθογγος υπάρχει στο λεξικό

προσθέτουμε στη λίστα το αντίστοιχο ελληνικό γράμμα, αλλιώς πρώτα ανανεώνουμε το λεξικό κι έπειτα επεκτείνουμε τη λίστα.
Εντελώς ανάλογα κινούμαστε για τις 1-2 αναλογίες, δηλαδή όταν ένας λατινικός χαρακτήρας αντιστοιχίζεται σε 2 ελληνικούς. Ελέγχουμε αν το γράμμα που συναντάμε στην greenglish λέξη ανήκει σε ένα πλήθος γραμμάτων, και όταν βρεθεί κάποιο άλλο γράμμα το αποθηκεύουμε μαζί με το επόμενο του στη λίστα, στο σημείο που βρίσκεται ο αντίστοιχος λατινικός χαρακτήρας.

Κριτήριο επιλογής των γραμμάτων που θα αποθηκευτούν είναι αν ανήκουν σε ένα σύνολο που δεν αντιστοιχίζεται σε δίφθογγο ('ΑΒΓΖΚΛΜΝΟΠΡΣΤΧΩ' για τα ελληνικά και 'ABCFKLMNOPQRSTVWXZ' για τους λατινικούς), ή αν η εμφάνισή του στις 1-1 αντιστοιχίσεις ήταν μεγαλύτερη από 3 φορές. Με αυτό τον τρόπο προσπαθούμε να ελαχιστοποιήσουμε τους λανθασμένους κανόνες που προκύπτουν π.χ. από ορισμένες λέξεις που τυχαίνει να είναι ισομήκεις αλλά περιέχουν διφθόγγους. Φυσικά, θα υπάρχουν και πάλι ορισμένοι λανθασμένοι κανόνες αλλά στη συνέχεια του εργαστηρίου κάποιες από αυτές τις αστοχίες θα διορθώσει ο ορθογράφος.

Τελικά, γράφουμε σε ένα αρχείο (*stats.txt*) τα στατιστικά των μετατροπών, δηλαδή ποιο γράμμα αντιστοιχίστηκε σε ποιο/ποια και πόσες φορές.

Αρχεία εξόδου *stats.txt*

ΒΗΜΑ 2

(Κώδικας probs_and_symbols.py)

Εδώ δημιουργούμε τα αρχεία *isyms.txt* και *osyms.txt*, με πρώτη σειρά το <epsilon> και ένα αρχείο *probs.txt* με την πιθανότητα εμφάνισης κάθε μετατροπής. Διαβάζουμε το αρχείο με τα στατιστικά που δημιουργήσαμε στο προηγούμενο βήμα και με κατάλληλη μορφοποίηση της εισόδου κρατάμε την πιθανότητα εμφάνισης κάθε μετατροπής. Συγκεκριμένα, χωρίζουμε στις γραμμές τους χαρακτήρες πριν και μετά την ":" απομονώνοντας έτσι τον λατινικό χαρακτήρα και αποθηκεύουμε στη λίστα I[] του isyms.txt. Γράφουμε στο αρχείο το χαρακτήρα και δίπλα έναν αύξον αριθμό. Έπειτα χωρίζουμε κάθε ελληνικό με το κενό, και μετράμε πόσες ήταν οι συνολικές εμφανίσεις, αθροίζοντας τις εμφανίσεις κάθε κανόνα. Αποθηκεύουμε τους ελληνικούς χαρακτήρες στην O[] για το αρχείο osyms.txt. Τέλος, αποθηκεύουμε στο αρχείο (*probs.txt*) με κάθε μετατροπή κανονικοποιημένη ως προς τις συνολικές εμφανίσεις του λατινικού γράμματος.
Τέλος, ελέγχουμε αν κάποιος λατινικός χαρακτήρας δεν εμφανίστηκε καμία φορά, ψάχνοντας στη λίστα I[] και τον προσθέτουμε στο αρχείο isyms μαζί με τον αύξοντα αριθμό.

Αρχεία εξόδου *isyms.txt* *osyms.txt* *probs.txt*

ΒΗΜΑ 3

(Κώδικας G.py)

Διαβάζουμε το αρχείο με τις πιθανότητες μετατροπής των χαρακτήρων και δημιουργούμε ένα αρχείο (*G.txt*) το οποίο έχει γραμμές στη μορφή (0 0 'latin' 'greek' 'cost') όπου το κόστος υπολογίζεται ως ο αρνητικός λογάριθμος. Το αρχείο έχει αυτή τη μορφή ώστε να τηρεί τη μορφή που χρειάζεται στη συνέχεια για να γίνει compile και να δημιουργηθεί το fst.
Χρησιμοποιούμε τα αρχεία *osyms.txt* και *isyms.txt* που περιέχουν τους ελληνικούς και λατινικούς χαρακτήρες αντίστοιχα, με την πρώτη στήλη κάθε αρχείου να περιέχει τον χαρακτήρα και τη δεύτερη την αρίθμηση σε αύξουσα σειρά. Στην πρώτη σειρά και των δύο αρχείων έχουμε συμπεριλάβει το <epsilon>, με αρίθμηση 0.

Το compile γίνεται στο shell με την εντολή

```
$ fstcompile --isymbols=isyms.txt --osymbols=osyms.txt G.txt G.fst
```

Αν θέλουμε να απεικονίσουμε το fst, γίνεται με τις εντολές

```
$ fstdraw --isymbols=isyms.txt --osymbols=osyms.txt G.fst G.dot
$ dot -Tpdf G.dot >G.pdf
```

ΒΗΜΑ 4

(Κώδικας I.py)

Θέλουμε όπου υπάρχουν αγγλικές λέξεις στο κείμενο των greenglish αυτές να διατηρούνται κατά τη μετατροπή του κειμένου. Επομένως πρέπει να υπάρχει μια κατάσταση όπου ο κάθε λατινικός χαρακτήρας σε μια λέξη θα παραμένει ίδιος. Φυσικά, αυτή η μετατροπή δεν είναι η συχνότερη, οπότε αποφεύγεται με το πολύ μεγάλο κόστος που θέτουμε.
Η διαδικασία δημιουργίας της λίστας με τα δεδομένα για το fst είναι ανάλογη με το προηγούμενο βήμα, έχουμε δηλαδή την ίδια μορφή αλλά εδώ το κόστος είναι σταθερό.
Έχουμε τις εντολές:

```
$ fstcompile --isymbols=isyms.txt --osymbols=isyms.txt I.txt I.fst
```

Αν θέλουμε, απεικονίζουμε ως εξής:

```
$ fstdraw --isymbols=isyms.txt --osymbols=isyms.txt I.fst I.dot
$ dot -Tpdf I.dot >I.pdf
```

ΒΗΜΑ 5

(Κώδικας A1.py)

Γνωρίζουμε ότι ένα FSA είναι ένα FST με την είσοδο να ταυτίζεται με την έξοδο.
Δημιουργούμε ένα αρχείο (*A1.txt*) που περιέχει μορφοποιημένα τα δεδομένα για τη δημιουργία του fsa.
Διαβάζουμε κάθε σειρά του αρχείου που δίνεται ως λεξικό (el_caps_noaccent).
Θέλουμε να δημιουργήσουμε ένα FSA με αρχική κατάσταση το πρώτο γράμμα της λέξης/γραμμής, το οποίο θα εκτείνεται μέχρι το τέλος της λέξης. Για την επόμενη λέξη θέλουμε να γυρίσουμε στην αρχική κατάσταση (μηδενίζοντας τον μετρητή i) και από εκεί να μεταβούμε σε μια νέα (μοναδική) κατάσταση που θα αντιστοιχεί στο δεύτερο γράμμα, έπειτα στο τρίτο κλπ (αυξάνοντας τον μετρητή i). Δηλαδή, για κάθε νέα λέξη επιστρέφουμε στην αρχική κατάσταση και από εκεί αρχίζουμε ένα νέο “μονοπάτι” για κάθε λέξη.
Για να υλοποιήσουμε την επιστροφή στην αρχική κατάσταση, όταν είμαστε στην αρχή κάθε λέξης, γράφουμε μια γραμμή στο αρχείο με 0 στην αντίστοιχη θέση , δηλαδή μορφή
(0, i, ‘letter’, ‘letter’, ‘cost’)
και για τα υπόλοιπα γράμματα έχουμε τη γενική μορφή
(i-1,i, ‘letter’, ‘letter’, ‘cost’).

Το αρχείο txt, με ανάλογη διαδικασία όπως αυτή που ακολουθήσαμε στα προηγούμενα βήματα, γίνεται compile στη συνέχεια.

```
$ fstcompile ––isymbols=osyms.txt ––osymbols=osyms.txt A1.txt A1.fst
```

Το FSA που έχουμε δημιουργήσει σε αυτό το σημείο γνωρίζουμε ότι είναι πάρα πολύ μεγάλο οπότε η απεικόνιση του καθίσταται αδύνατη.

ΒΗΜΑ 6

Για μετατροπή σε ντετερμινιστικό χρησιμοποιούμε την εντολή

```
$ fstdeterminize A1.fst A1det.fst
```

Για ελαχιστοποίηση χρησιμοποιούμε την εντολή

```
$ fstminimize A1det.fst A1min.fst
```

Δε σχεδιάζουμε κάποιο λόγω μεγέθους του αρχικού FSA.

ΒΗΜΑ 7

(Κώδικας A2.py)

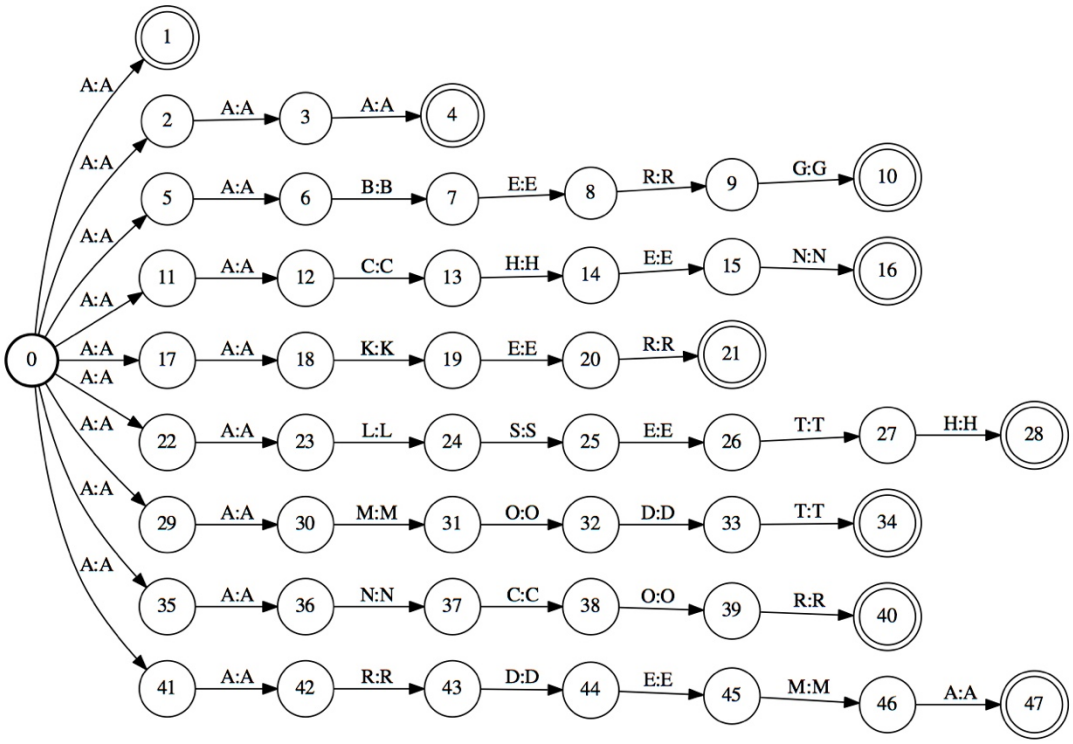
Επαναλαμβάνουμε ακριβώς τα ανάλογα βήματα 5,6 για το αγγλικό λεξικό.
Οι εντολές για τη δημιουργία fsa, τη μετατροπή σε ντετερμινιστικό και την ελαχιστοποίση είναι για το A2 οι ακόλουθες

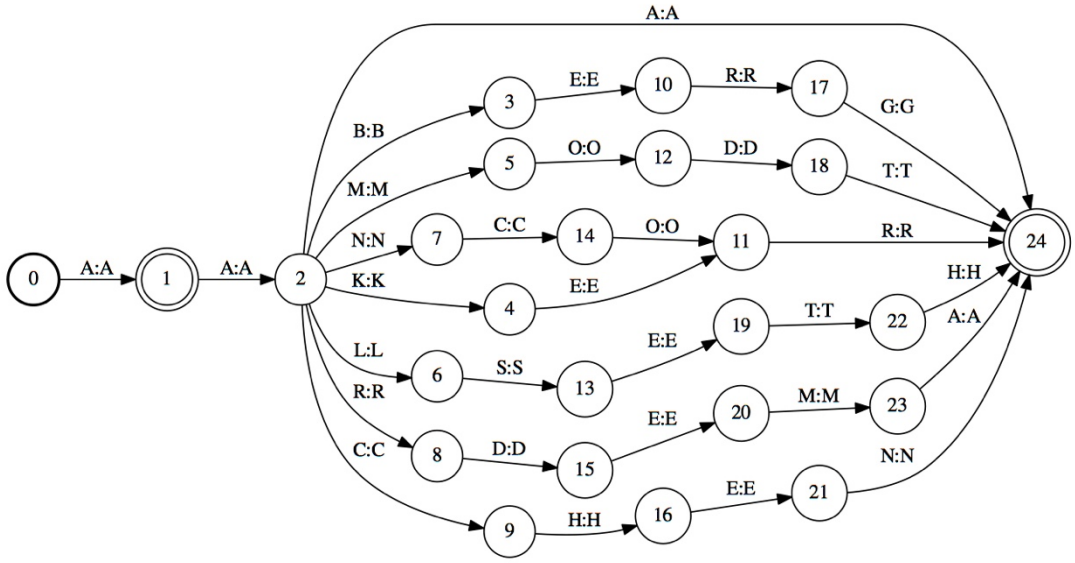
```
$ fstcompile --isymbols=isyms.txt --osymbols=isyms.txt A2.txt A2.fst

$ fstdeterminize A2.fst A2det.fst

$ fstminimize A2det.fst A2min.fst
```

Όπως και προηγουμένως, το FSA που θα προκύψει από την παραπάνω διαδικασία είναι πολύ μεγάλο σε μέγεθος οπότε η σχεδίαση του δεν είναι χρήσιμη. Για να έχουμε όμως έστω μια μερική απεικόνιση του αποτελέσματός μας, έχουμε επιλέξει ένα μικρό κομμάτι από το FSA A2 με ένα μικρό αριθμό λέξεων και σχεδιάσαμε το FSA του και το optimised FSA. Τα αρχεία βρίσκονται στον φάκελο ”for check”.





Αρχεία εξόδου *A2.pdf* *A2_opt.pdf*