ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«ΑΝΑΠΤΥΞΗ ΣΤΑΤΙΣΤΙΚΩΝ ΓΛΩΣΣΙΚΩΝ ΜΟΝΤΕΛΩΝ ΓΙΑ ΤΑ ΕΛΛΗΝΙΚΑ ΜΕ ΧΡΗΣΗ ΡΙΖΑΣ ΚΑΙ ΜΕΡΟΥΣ ΤΟΥ ΛΟΓΟΥ»

ΔΙΠΛΑΡΗΣ ΣΩΤΗΡΗΣ
ΠΡΑΤΣΟΛΗΣ ΔΗΜΗΤΡΗΣ

Επιβλέπων:
Καθηγητής Διγαλάκης Βασίλης

Εξεταστική Επιτροπή:
Καθηγητής Πατεράκης Μιχάλης
Καθηγητής Κουμπαράκης Μανώλης

Χανιά, Ιούνιος 2001

ΕΥΧΑΡΙΣΤΙΕΣ

Θα θέλαμε να εκφράσουμε τις ειλικρινείς μας ευχαριστίες στον καθηγητή κ. Βασίλη Διγαλάκη, που μας εμπιστεύτηκε το θέμα, και κυρίως για την καθοδήγησή του και τις πολύτιμες συμβουλές του, που βοήθησαν σημαντικά στην ολοκλήρωση της διπλωματικής αυτής εργασίας. Επιπλέον, για την ευκαιρία που μας έδωσε να αποκτήσουμε σημαντικές εμπειρίες πάνω σε έναν συνεχώς αναπτυσσόμενο τομέα, όπως αυτός της αναγνώρισης φωνής.

Επίσης, θα θέλαμε να ευχαριστήσουμε τους αναγνώστες της εργασίας μας, καθηγητές κ. Μιχάλη Πατεράκη και κ. Μανώλη Κουμπαράκη για τον χρόνο που διέθεσαν για να κάνουν τις παρατηρήσεις τους.

Ευχαριστούμε τους συναδέλφους μας στο Εργαστήριο Τηλεπικοινωνιών για την κατανόησή τους κάθε φορά που βρεθήκαμε αντιμέτωποι με κάποιο πρόβλημα. Ιδιαίτερα ευχαριστούμε τον Δημήτρη Οικονομίδη για τις πολύτιμες υποδείξεις του πάνω σε γλωσσικά μοντέλα.

Τέλος, χρωστάμε ευγνωμοσύνη στις οικογένειές μας, χωρίς την υλική και ηθική συμπαράσταση των οποίων,δε θα ήταν δυνατή η ολοκλήρωση των σπουδών μας, καθώς και σε όλους τους φίλους μας, εδώ στα Χανιά, από τους οποίους αντλούσαμε δύναμη για να συνεχίσουμε την προσπάθειά μας.

Σωτήρης Διπλάρης
Δημήτρης Πρατσόλης

Πολυτεχνείο Κρήτης
Ιούνιος 2001

«Αφιερώνεται στους γονείς μου.
Στους δύο ανθρώπους που ,
με τις θυσίες τους,
μου έχουν δώσει το δικαίωμα
να κάνω όνειρα για τη ζωή μου...»

Δημήτρης Πρατσόλης

«Αφιερώνεται στην οικογένειά μου»

Σωτήρης Διπλάρης

ΠΕΡΙΕΧΟΜΕΝΑ

Λίστα Σχημάτων

Λίστα Πινάκων

Λίστα Γραφικών Παραστάσεων

Κεφάλαιο 1

Εισαγωγή

Η έρευνα που γίνεται πάνω στα γλωσσικά μοντέλα έχει ως σκοπό την ανάπτυξη υπολογιστικών τεχνικών και δομών που περιγράφουν ακολουθίες λέξεων όπως αυτές παράγονται από τους ανθρώπους. Τέτοια μοντέλα μπορούν να δημιουργηθούν για να παρέχουν εκτιμήσεις της ορθότητας και της αληθοφάνειας δεδομένων δειγμάτων κειμένου και έχουν γίνει απαραίτητα εργαλεία για διάφορα ερευνητικά πεδία. Η διπλωματική αυτή εργασία αναφέρεται στην εφαρμογή των μοντέλων αυτών στην επεξεργασία φωνής, η οποία έχει εξελιχθεί σε ένα κύριο ερευνητικό πεδίο τα 30 τελευταία χρόνια. Ο βασικός στόχος αυτή τη στιγμή είναι είναι η ανάπτυξη αναγνωριστών με μεγάλο μέγεθος λεξικού (large-vocabulary recognisers) για φυσικό, αυθόρμητο και συνεχή λόγο. Παρά τη σταθερή πρόοδο, όμως, η βελτίωση κάποιων παραγόντων είναι ακόμα απαραίτητη πριν γίνει εφικτή η εκτεταμένη βιομηχανική χρήση των αναγνωριστών. Αλλά το πολύ ευρύ φάσμα των δυνατών εφαρμογών μπορεί να εγγυηθεί ότι η τεχνολογία αυτή θα κερδίσει πολύ έδαφος, όταν ωριμάσει.

Δύο βασικές προσεγγίσεις της μοντελοποίησης της ανθρώπινης γλώσσας μπορούν να προσδιοριστούν. Η πρώτη βασίζεται στην συντακτική και σημασιολογική ανάλυση του κειμένου για να καθορίσει την ιεραρχική δομή των προτάσεων. Μια τέτοια ανάλυση χρησιμοποιεί μια σειρά κανόνων για να διαπιστωθεί το αν μια πρόταση είναι επιτρεπτή ή όχι. Αν και η προσέγγιση αυτή είναι δυνατόν να περιγράψει ένα σημαντικό ποσοστό μιας γλώσσας, η πλήρης κάλυψη της είναι σχεδόν αδύνατη, κυρίως λόγω των συνεχών αλλαγών που συμβαίνουν σε μια γλώσσα. Επιπλέον, κάποιες *φραστικές διατυπώσεις (utterances)* που παρατηρούνται σε μια γλώσσα και δεν είναι σωστές γραμματικά, δεν μπορούν να περιγραφούν από μια τέτοια ανάλυση.

Από την άλλη μεριά, μια δεύτερη προσέγγιση βασιζόμενη σε στατιστικές τεχνικές έχει μεγαλύτερη ευρωστία σε γραμματικές ανωμαλίες . Τέτοια *στατιστικά γλωσσικά μοντέλα (statistical language models)* αντιστοιχούν σε κάθε λέξη μιας φραστικής διατύπωσης μια τιμή πιθανότητας σύμφωνα με την εκτιμώμενη πιθανοφάνεια της ανάμεσα στα συμφραζόμενα της ακολουθίας λέξεων που περιβάλλει την συγκεκριμένη λέξη. Αυτές οι πιθανότητες συνάγονται από ένα κείμενο μεγάλου μεγέθους, στο οποίο αναφερόμαστε ως *κείμενο εκπαίδευσης (training text)*. Με τον τρόπο αυτό το γλωσσικό μοντέλο μπορεί να αντικατοπτρίζει τη χρήση της γλώσσας όπως αυτή χρησιμοποιείται στην πράξη, και όχι μόνο την γραμματική της εξειδανίκευση. Ένα μεγάλο πλεονέκτημα είναι το ότι το μέγεθος των διαθέσιμων κειμένων εκπαίδευσης στις μέρες μας κυμαίνεται πια σε εκατοντάδες εκατομμύρια λέξεις.

Ένα σύστημα αναγνώρισης φωνής πρέπει να βρίσκει την πιο πιθανή *υπόθεση πρότασης (sentence hypothesis)* για κάθε φραστική διατύπωση. Στο συνεχή λόγο, όχι μόνο κάποιες λέξεις θα είναι άγνωστες, αλλά επίσης και τα όριά τους στο χρόνο. Για μεγάλα λεξικά αυτό συνεπάγεται ένα εξαιρετικά υψηλό αριθμό πιθανών εναλλακτικών κατατμήσεων του ακουστικού σήματος σε λέξεις. Σε αυτό το σημείο, το γλωσσικό μοντέλο υπολογίζει την γλωσσική ορθότητα και αληθοφάνεια μέρους ή ολόκληρων των υποθέσεων. Σε συνδυασμό με το υπόλοιπο σύστημα αναγνώρισης, η εκτίμηση αυτή βοηθά στο να βρεθούν εκείνες οι υποθέσεις που είναι πιο πιθανό να οδηγήσουν στο σωστό αποτέλεσμα, όπως επίσης και εκείνες οι υποθέσεις που μπορούν να απορριφθούν με στόχο τον περιορισμό του αριθμού των υποθέσεων σε πρακτικά επίπεδα.

Αν και η συγκεκριμένη διπλωματική εργασία εστιάζει στην εφαρμογή των γλωσσικών μοντέλων σε συστήματα αναγνώρισης φωνής, τα γλωσσικά μοντέλα είναι σημαντική παράμετρος και σε άλλες εφαρμογές όπως η αναγνώριση γραφικού χαρακτήρα, ορθογραφική διόρθωση, part-of-speech tagging.

Οι επόμενες παράγραφοι περιγράφουν τα κύρια μέρη στα οποία μπορεί να διαιρεθεί ένα σύστημα αναγνώρισης φωνής, αναδεικνύουν το ρόλο των γλωσσικών μοντέλων και σκιαγραφούν το πεδίο της διπλωματικής αυτής εργασίας.

1.1 Το πρόβλημα της αναγνώρισης φωνής

Στην αναγνώριση φωνής, ο σκοπός είναι να βρεθεί η περισσότερο πιθανή ακολουθία λέξεων *w*, δεδομένων των ακουστικών δεδομένων *x* που έχουν παρατηρηθεί. Αυτό επιτυγχάνεται βρίσκοντας την κατάλληλη ακολουθία λέξεων *w* που μεγιστοποιεί την υπό συνθήκη πιθανότητα *P(w|x)*. Από τον *κανόνα του Bayes* έχουμε,

$$P(w \mid x) = \frac{P(x \mid w) \cdot P(w)}{P(x)} \qquad (1.1)$$

αλλά καθώς το *P(x)* είναι σταθερό για ένα δεδομένο ακουστικό σήμα, μία ισοδύναμη στρατηγική είναι να υπολογίσουμε την ποσότητα :

$$\arg\max_{\forall w} \{P(x \mid w) \cdot P(w)\}, \qquad (1.2)$$

όπου *argmax* συμβολίζει το όρισμα που μεγιστοποιεί την αντίστοιχη ποσότητα. Η ποσότητα που πρέπει να μεγιστοποιηθεί είναι το γινόμενο των πιθανοτήτων *P(w)* και *P(x|w)*. Το *P(x|w)* αναπαριστά την πιθανότητα εμφάνισης μιας ακολουθίας ακουστικών διανυσμάτων *x*, δεδομένης μιας ακολουθίας λέξεων *w*. Η πιθανότητα αυτή υπολογίζεται από το ακουστικό μέρος του αναγνωριστή και είναι γνωστή ως ***ακουστικό μοντέλο*** (*acoustic model*). Ο όρος *P(w)* αναπαριστά την εκτίμηση της εκ των προτέρων (*a-priori*) πιθανότητας μιας δεδομένης ακολουθίας λέξεων *w*, ανεξάρτητα από το ακουστικό σήμα που παρατηρήθηκε. Η εκτίμηση γίνεται με βάση κάποιο στατιστικό μοντέλο και η πιθανότητα είναι γνωστή ως ***γλωσσικό μοντέλο*** (*language model*).

Ας θεωρήσουμε την ακολουθία λέξεων *w*= «*το καλοκαίρι κάνει πολλή ζέστη*». Τότε ένα γλωσσικό μοντέλο θα υπολόγιζε την *P(w)* ως εξής:

P(w)= P(«το καλοκαίρι κάνει πολλή ζέστη»)=

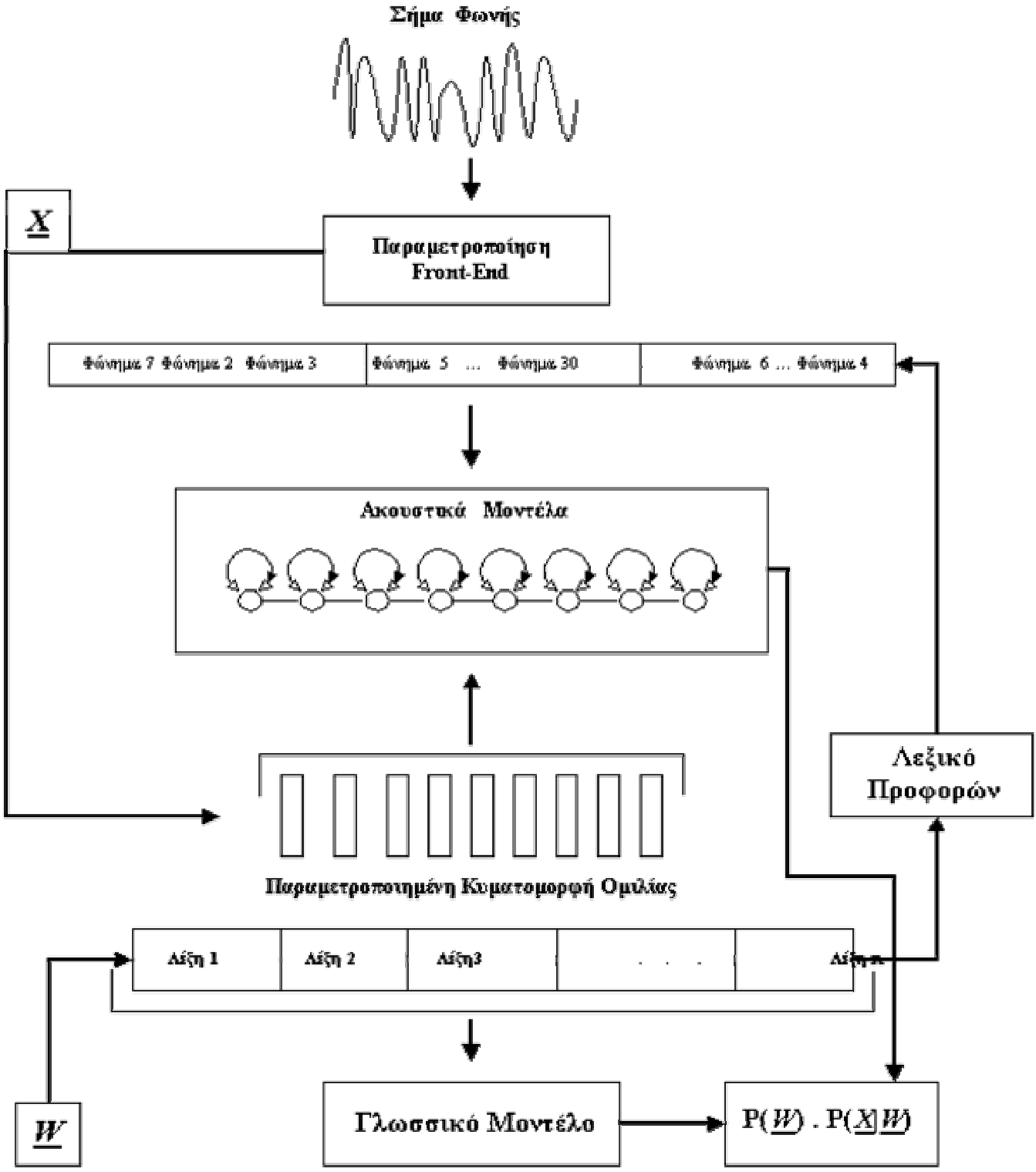
= P(«το») · P(«καλοκαίρι» | «το») · P(«κάνει» | «το καλοκαίρι»)

· P(«πολλή» | «το καλοκαίρι κάνει») · P(«ζέστη» | «το καλοκαίρι κάνει πολλή»)

Η γλωσσική μονάδα που μοντελοποιείται είναι συνήθως η λέξη. Για να υπάρχει η δυνατότητα γενίκευσης και να μοντελοποιούνται λέξεις που δεν παρατηρήθηκαν στα δεδομένα εκπαίδευσης, χρησιμοποιούνται μικρότερες γλωσσικές μονάδες όπως το ***φώνημα*** *(phoneme)* ή η *συλλαβή*. Η όλη διαδικασία ονομάζεται σχεδίαση του ακουστικού μοντέλου. Στην παρούσα εφαρμογή κάθε λέξη μετατρέπεται σε μία ακολουθία βασικών ήχων, τα φωνήματα, χρησιμοποιώντας ένα ***λεξικό προφορών*** *(dictionary*). Το λεξικό προφορών είναι ένα αρχείο που περιέχει τις ηχητικές αποδόσεις όλων των λέξεων που περιέχονται στη γραμματική και πρέπει να συνταχθεί ώστε να περιγράφει ακριβώς τις προφορές των λέξεων ακόμη και με περισσότερους του ενός τρόπους. Η δομή ενός τέτοιου λεξικού φαίνεται με το εξής παράδειγμα:

άγνωστες	a+ j n o s t e s
άνθρωπος	a+ n th r o p o s
δυναμικών	dh i n a m i k o n
κωνσταντίνος	k o n s t a d i+ n o s
κωνσταντίνος	k o s t a d i+ n o s
νοικιασμένο	n i k2 y a s m e+ n o
	.
	.
	.

Όλα τα παραπάνω φαίνονται στο *Σχήμα 1.1,* όπου περιγράφεται η διαδικασία υπολογισμού της πιθανότητας $P(x|w)$ μιας ακολουθίας λέξεων w δεδομένου ενός παραμετροποιημένου ακουστικού σήματος x.


Σήμα Φωνής
X
Παραμετροποίηση
Front-End
Φώνημα 7 Φώνημα 2 Φώνημα 3
Φώνημα 5 ... Φώνημα 30
Φώνημα 6 ... Φώνημα 4
Ακουστικά Μοντέλα
Λεξικό
Προφορών
Παραμετροποιημένη Κυματομορφή Ομιλίας
Λέξη 1
Λέξη 2
Λέξη3
. . .
Λέξη n
W
Γλωσσικό Μοντέλο
P(W) . P(X|W)

Σχήμα 1.1

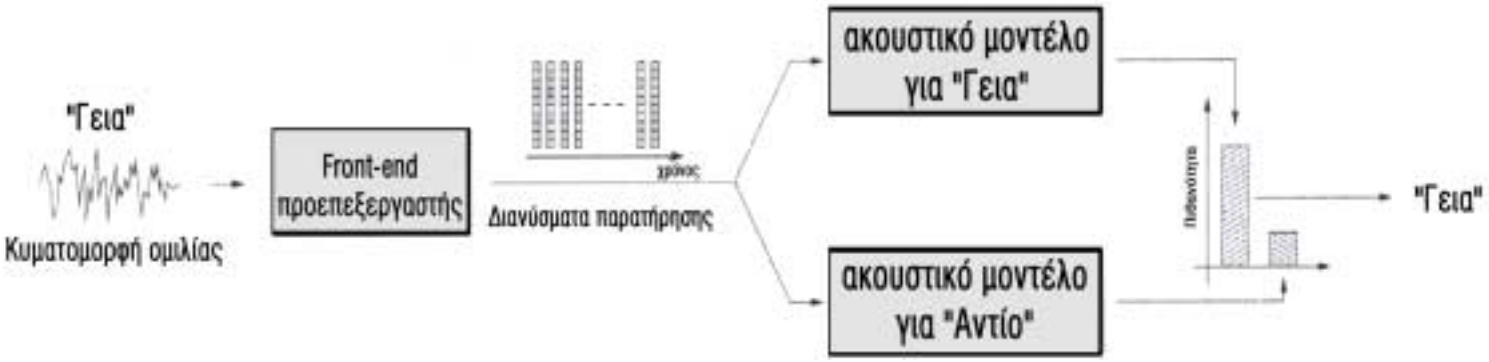
1.1.1 Προεπεξεργασία του σήματος φωνής

Το σήμα φωνής που παράγεται από ένα μικρόφωνο ή μια συσκευή εγγραφής είναι στη μορφή ενός αναλογικού ηλεκτρικού σήματος. Το σήμα αυτό επεξεργάζεται σε μια μορφή κατάλληλη για χρήση από το σύστημα αναγνώρισης φωνής, από μια σειρά λειτουργιών στην οποία αναφερόμαστε *ως προεπεξεργασία σήματος (front-end).* Συνήθως αυτά τα βήματα περιλαμβάνουν *περιορισμό του εύρους του σήματος (band limiting), δειγματοληψία του* και στη συνέχεια εφαρμογή κάποιων *φασματικών μετασχηματισμών* για να κωδικοποιηθούν τα χαρακτηριστικά της συχνότητας του σήματος. Το τελευταίο αυτό βήμα περιλαμβάνει τον *διακριτό μετασχηματισμό Fourier βραχέως χρόνου (short-time discrete Fourier transform)*, και μια συγκεκριμένη δημοφιλής επιλογή είναι να κωδικοποιηθεί το σήμα σαν *συντελεστές Cepstrum [1]*. Το αποτέλεσμα είναι μια διακριτού χρόνου σειρά από διανύσματα παρατηρήσεως (observation vectors), η οποία είναι και η έξοδος του front-end. Η έξοδος του front-end περνάει στη συνέχεια στον αλγόριθμο αναγνώρισης.

1.1.2 Το ακουστικό μοντέλο

Το ακουστικό μοντέλο είναι κεντρική έννοια σε κάθε σύστημα αναγνώρισης φωνής και είναι ένας τρόπος κωδικοποίησης των ήχων που εμπεριέχονται στην ανθρώπινη φωνή. Η πιο επιτυχής μέθοδος κωδικοποίησης των ήχων της φωνής είναι η χρήση των *κρυφών μαρκοβιανών μοντέλων (Hidden Markov Models).* Περιγράφοντας τα διανύσματα παρατήρησης σαν μια πιθανοτική χρονική σειρά, τα HMMs λαμβάνουν υπόψη την έμφυτη φυσική μεταβλητότητα των χαρακτηριστικών της ανθρώπινης φωνής. Δεδομένου ενός αριθμού παραδειγμάτων ενός συγκεκριμένου ήχου στη μορφή των αντίστοιχων σειρών διανυσμάτων παρατήρησης X, οι παράμετροι του μοντέλου M μπορούν να ρυθμιστούν έτσι ώστε να αναπαριστούν με τον καλύτερο τρόπο αυτά τα δεδομένα με μια πιθανοτική σημασία, η οποία συχνά συμβολίζεται ως $P(X \mid M)$. Το μοντέλο μπορεί επομένως να χρησιμοποιηθεί για τον υπολογισμό της πιθανότητας μιας νέας σειράς διανυσμάτων παρατήρησης αναφορικά με τις παραμέτρους του, κι επομένως να δώσει μια ένδειξη του πόσο όμοια είναι η

νέα μέτρηση με αυτές που αρχικά χρησιμοποιήθηκαν για να καθορίσουν τις παραμέτρους του μοντέλου. Η πιθανότητα αυτή χρησιμοποιείται για την εξαγωγή μιας στατιστικής απόφασης σχετικά με την φραστική διατύπωση που πρόκειται να αναγνωριστεί όπως φαίνεται στο σχήμα που ακολουθεί. *(Σχήμα 1.2)*.



Σχήμα 1.2

Τα HMMs μπορούν να εκπαιδευτούν είτε για να μοντελοποιήσουν απ'ευθείας ολόκληρες λέξεις, είτε για να μοντελοποιήσουν μικρότερες μονάδες λέξεων (φωνήματα), οι οποίες συνδέονται για να σχηματίσουν λέξεις. Αυτή η τελευταία προσέγγιση είναι αυτή που συνήθως επιλέγεται, καθώς ακόμα και για μικρού μεγέθους λεξικά, είναι πιθανό να μην υπάρχει επαρκές υλικό εκπαίδευσης για να καθοριστούν αξιόπιστα μοντέλα ολόκληρων λέξεων.

1.1.3. Το γλωσσικό μοντέλο

Ενώ το ακουστικό μοντέλο δείχνει πόσο πιθανό είναι μία ακολουθία λέξεων να ταιριάζει με τα ακουστικά δεδομένα, το γλωσσικό μοντέλο εκτιμά την *a-priori* πιθανότητα της ακολουθίας λέξεων αυτής καθεαυτής, π.χ. :

$$\hat{P}\big(w(0,K-1)\big) \qquad (1.3)$$

,όπου $w(0,K-1)= w(0),w(1),\ldots,w(K-1)$ είναι η ακολουθία K ζητούμενων λέξεων, και η περισπωμένη δηλώνει εκτιμήτρια συνάρτηση. Η πιθανότητα αυτή μπορεί να βοηθήσει το σύστημα αναγνώρισης φωνής να αποφασίσει σχετικά με τους διάφορους

πιθανούς, όμοια ακουστικούς, ανταγωνιζόμενους τρόπους κατάτμησης των διανυσμάτων παρατηρήσης σε λέξεις σύμφωνα με την ακουστική τους πιθανότητα. Από τον ορισμό των υπό συνθήκη πιθανοτήτων, μπορούμε να αναλύσουμε την από κοινού πιθανότητα της ποσότητας *(1.3)* σε ένα γινόμενο από υπό συνθήκη πιθανότητες.

$$\hat{P}\big(w(0,K-1)\big)=\prod_{i=0}^{K-1}\hat{P}\big(w(i)\,|w(0,i-1)\big) \qquad (1.4)$$

,όπου πρακτικά το $w(0,-1)$ υποδεικνύει το σύμβολο της αρχής της πρότασης (*start of sentence symbol*). Εφόσον η χρήση ενός γλωσσικού μοντέλου στη διαδικασία της αναγνώρισης φωνής συνήθως απαιτεί υπολογισμό των υπό συνθήκη πιθανοτήτων που εμφανίζονται στο δεξιό μέλος της εξίσωσης *(1.4)*, αυτές κανονικά υπολογίζονται κατευθείαν. Αναπτύσσοντας ένα γλωσσικό μοντέλο, ο κύριος σκοπός είναι να βρεθούν κατάλληλες δομές που θα μοντελοποιήσουν τις πιθανοτικές εξαρτήσεις μεταξύ των λέξεων στη φυσική γλώσσα, και στη συνέχεια να χρησιμοποιηθούν αυτές οι δομές για τον υπολογισμό είτε των υπό συνθήκη είτε των από κοινού πιθανοτήτων των ακολουθιών λέξεων.

1.2 Σκοπός της διπλωματικής εργασίας

Στη διπλωματική αυτή εργασία σκοπός μας είναι να κατασκευάσουμε γλωσσικά μοντέλα μεγάλου λεξιλογίου για τα ελληνικά. Τα γλωσσικά αυτά μοντέλα χρησιμοποιούνται για αναγνώριση ομιλίας και πιο συγκεκριμένα για υπαγόρευση ελληνικού κειμένου, στα πλαίσια του έργου *«Λογοτυπογραφία – Βελτίωση της ροής στο δημοσιογραφικό χώρο με σύστημα αυτόματης υπαγόρευσης ελληνικού κειμένου»*. Τα βασισμένα σε λέξεις γλωσσικά μοντέλα (*word-based language models*) μοντελοποιούν τη γλώσσα με βάση τη σχέση μίας λέξης με τις προηγούμενές της. Το μειονέκτημά τους είναι ότι με τον τρόπο αυτό δε λαμβάνονται υπόψη συντακτικά και γραμματικά πρότυπα της γλώσσας. Ταυτόχρονα, σε σχετικά μικρά λεξικά όπου υπάρχει σποραδικότητα δεδομένων, τα *word-based* γλωσσικά μοντέλα αποτυγχάνουν

να αναπαραστήσουν με ευρωστία (*robustness*) τη γλώσσα. Για το λόγο αυτό, για μια γλώσσα με πλούσια μορφολογία όπως τα ελληνικά, χρησιμοποιούμε ομαδοποίηση λέξεων σε κλάσεις με σκοπό την καλύτερη αναπαράστασή της, τουλάχιστον στις περιπτώσεις όπου υπάρχει σποραδικότητα δεδομένων, δηλαδή όταν υπάρχει ανάγκη για μικρό σε μέγεθος λεξικό. Το αποτέλεσμα είναι η δημιουργία πιο γενικευμένων γλωσσικών μοντέλων, που μπορούν να καλύψουν και *bigrams* λέξεων που δεν έχουν παρατηρηθεί στα κείμενα εκπαίδευσης. Υλοποιούμε και μελετούμε την απόδοση *word-bigram backoff* γλωσσικών μοντέλων, μοντέλων βασισμένων σε κλάσεις λέξεων (*class-based language models*), υβριδικών μοντέλων που περιέχουν συνδυασμό λέξεων και κλάσεων, καθώς και *word-based* μοντέλων μεγαλύτερης τάξης. Για την ομαδοποίηση των λέξεων σε κλάσεις χρησιμοποιούμε δύο προσεγγίσεις, καθώς και το συνδυασμό τους. Οι δύο αυτοί τρόποι ομαδοποίησης λέξεων είναι το *stemming*, δηλαδή η ομαδοποίηση λέξεων με βάση την ομοιότητα της ρίζας ή της κατάληξής τους, και το *part-of-speech tagging*, δηλαδή η ομαδοποίηση λέξεων με βάση το μέρος του λόγου ή τη γραμματική ιδιότητά τους. Για την υλοποίηση του *stemming* αναπτύσσουμε έναν απλό αλγόριθμο *stemming* για τα ελληνικά που χωρίζει κάθε λέξη σε ρίζα και κατάληξη, ενώ για την υλοποίηση του *part-of-speech tagging* βασιζόμαστε σε *part-of-speech* αγγλοελληνικό λεξικό καθώς και σε κανόνες της ελληνικής γραμματικής για τον εμπλουτισμό των κλάσεων.

1.3 Δομή της διπλωματικής εργασίας

Στο δεύτερο κεφάλαιο περιγράφονται οι βασικές τεχνικές κατασκευής *n-gram* γλωσσικών μοντέλων, οι περιπτώσεις *word-based* και *class-based* γλωσσικών μοντέλων, οι τρόποι μέτρησης της απόδοσης των γλωσσικών μοντέλων, καθώς και θέματα υλοποίησής τους. Στο τρίτο κεφάλαιο παρουσιάζεται ο αλγόριθμος *stemming* για τα ελληνικά που αναπτύξαμε και τα αποτελέσματα που έδωσε για διάφορες περιπτώσεις λεξικών. Η θεωρία στην οποία βασίζεται το *part-of-speech tagging*, καθώς και ο τρόπος που υλοποιήθηκε περιγράφεται στο τέταρτο κεφάλαιο. Στο πέμπτο κεφάλαιο παρουσιάζονται όλα τα γλωσσικά μοντέλα που αναπτύχθηκαν στα πλαίσια αυτής της διπλωματικής, ο τρόπος που υλοποιήθηκαν, και τα χαρακτηριστικά

τους. Τα αποτελέσματα των πειραμάτων αυτών, καθώς και η ανάλυσή τους δίνονται στο έκτο κεφάλαιο, ενώ στο έβδομο κεφάλαιο παρουσιάζονται τα συμπεράσματα της διπλωματικής εργασίας και οι πιθανές μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Τεχνικές γλωσσικών μοντέλων

2.1 Εισαγωγή στα γλωσσικά μοντέλα

Τα γλωσσικά μοντέλα είναι ένας πρωτεύων παράγοντας σε πολλά πεδία, τα οποία περιλαμβάνουν την αναγνώριση φωνής, την αναγνώριση γραφικού χαρακτήρα, την ορθογραφική διόρθωση και άλλα. Η κυρίαρχη τεχνολογία στα γλωσσικά μοντέλα είναι τα *n-gram* μοντέλα. Ένα γλωσσικό μοντέλο, συνήθως, διατυπώνεται σαν μια κατανομή πιθανότητας $P(s)$ πάνω σε συμβολοσειρές s που προσπαθεί να αναπαραστήσει πόσο συχνά μια συμβολοσειρά s συμβαίνει σε μια πρόταση. Για παράδειγμα για ένα γλωσσικό μοντέλο που περιγράφει την ελληνική γλώσσα, μπορεί να έχουμε $P(«γεια») \approx 0,01$ αφού, ίσως, μία από κάθε εκατό προτάσεις που ένας άνθρωπος λέει είναι «γεια». Από την άλλη μεριά, μπορεί να είχαμε $P(«εγώ φασόλια τον φύλακα») \approx 0$ και $P(«ο αετός κολυμπάει θετικά») \approx 0$ αφού είναι εξαιρετικά απίθανο κάποιος να πει κάποια από αυτές τις δύο συμβολοσειρές. Βλέπουμε ότι σε αντίθεση με τη γλωσσολογία, η γραμματική ορθότητα δεν έχει σχέση με τη μοντελοποίηση της γλώσσας. Ακόμα κι αν η συμβολοσειρά *«ο αετός κολυμπάει θετικά»* είναι γραμματικά σωστή, το μοντέλο αντιστοιχεί σ' αυτή μία πιθανότητα κοντά στο μηδέν.

Τα πιο ευρέως χρησιμοποιούμενα γλωσσικά μοντέλα είναι, μακράν τα *n-gram* γλωσσικά μοντέλα. Ας πάρουμε την περίπτωση των συγκεκριμένων μοντέλων όταν $n=2$. Τα μοντέλα αυτά ονομάζονται *bigram* μοντέλα. Αρχικά για μια πρόταση s η οποία αποτελείται από τις λέξεις $w_1,..,w_l$, χωρίς απώλεια της γενικότητας μπορούμε να εκφράσουμε την πιθανότητα $P(s)$ ως:

$$P(s) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1 w_2)...P(w_l \mid w_{1...}w_{l-1}) = \prod_{i=1}^{l} P(w_i \mid w_1...w_{i-1}) \tag{2.1}$$

Στα *bigram* μοντέλα κάνουμε την προσέγγιση ότι η πιθανότητα μιας λέξης εξαρτάται μόνο από την ταυτότητα της αμέσως προηγούμενης λέξης και έτσι μπορύμε να εκφράσουμε την σχέση *(2.1)* ως εξής :

$$P(s) = \prod_{i=1}^{l} P(w_i \mid w_1...w_{i-1}) \cong \prod_{i=1}^{l} P(w_i \mid w_{i-1}) \tag{2.2}$$

Για να έχει νόημα η πιθανότητα $P(w_i|w_{i-1})$ για $i=1$ μπορούμε να εισάγουμε στην αρχή της πρότασης το σύμβολο <*s*> . Με αυτό τον τρόπο υποθέτουμε ότι το w_0 είναι το <*s*>. Επιπρόσθετα, για να ισούται το άθροισμα των πιθανοτήτων όλων των συμβολοσειρών $\sum_{s} P(s)$ με 1 είναι απαραίτητο να τοποθετήσουμε και ένα σύμβολο </*s*> στο τέλος των προτάσεων και να το συμπεριλάβουμε στο γινόμενο της εξίσωσης *(2.2)*. Για παράδειγμα, για να υπολογίσουμε την πιθανότητα *P(«αυτός διαβάζει ένα βιβλίο»)* θα έχουμε

P(«αυτός διαβάζει ένα βιβλίο») = P(«αυτός»|<s>) · P(«διαβάζει»|«αυτός») · P(«ένα»|«διαβάζει») · P(«βιβλίο»|«ένα») · P(</s>|«βιβλίο»)

Για να εκτιμήσουμε την πιθανότητα $P(w_i|w_{i-1})$, δηλαδή την συχνότητα με την οποία η λέξη w_i εμφανίζεται με δεδομένο ότι η τελευταία λέξη είναι η w_{i-1}, μπορούμε απλά να μετρήσουμε πόσο συχνά το bigram $w_{i-1}w_i$ εμφανίζεται σε κάποιο κείμενο και να κανονικοποιήσουμε. Έστω $c(w_{i-1}w_i)$ ο αριθμός των φορών που το bigram $w_{i-1}w_i$ εμφανίζεται στο δοσμένο κείμενο. Έτσι έχουμε:

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{\sum_{w_j} c(w_{i-1}w_j)} \tag{2.3}$$

Το διαθέσιμο κείμενο για τη δημιουργία ενός μοντέλου ονομάζεται *κείμενο εκπαίδευσης (training text)*. Για n-gram μοντέλα, το μέγεθος των δεδομένων εκπαίδευσης είναι τυπικά πολλά εκατομμύρια λέξεις. Η εκτίμηση της πιθανότητας $P(w_i|w_{i-1})$ που δίνεται από τον τύπο *(2.3)* ονομάζεται *εκτίμηση μεγίστης πιθανοφάνειας (maximum likelihood estimation ML)* της πιθανότητας $P(w_i|w_{i-1})$, επειδή η ισότητα αυτή παράγει το bigram μοντέλο που προσδίδει την μέγιστη πιθανότητα στο κείμενο εκπαίδευσης, από όλα τα πιθανά bigram μοντέλα.

Για n-gram μοντέλα όπου $n > 2$ αντί να συνθηκοποιούμε την πιθανότητα μιας λέξης με την ταυτότητα της αμέσως προηγούμενης λέξης, συνθηκοποιούμε αυτή την πιθανότητα με την ταυτότητα των $n - 1$ προηγούμενων λέξεων. Γενικεύοντας την εξίσωση *(2.3)* για $n > 2$ έχουμε:

$$P(s) = \prod_{i=1}^{l+1} P(w_i \mid w_{i-n+1}^{i-1}) \qquad (2.4)$$

όπου το w_i^j συμβολίζει τις λέξεις $w_i,...,w_j$. Επίσης θεωρούμε ότι το w_{-n+2}, μέσω του w_0, είναι το <*s*> και το w_{l+1} το </*s*>. Για να εκτιμήσουμε τις πιθανότητες $P(w_i|\ w_{i-n+1}^{i-1})$, η ανάλογη ισότητα στην *(2.4)* είναι:

$$P(w_i \mid w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})} \qquad (2.5)$$

Στην πράξη, το μεγαλύτερο n που χρησιμοποιείται είναι το $n=3$. Ένα τέτοιο μοντέλο ονομάζεται *trigram μοντέλο*. Οι λέξεις w_{i-n+1}^{i-1} πριν την τωρινή λέξη w_i, ονομάζονται συχνά *ιστορία (history)*. Επίσης, συχνά αναφερόμαστε στην τιμή του n ενός n-gram μοντέλου σαν *τάξη (order)* του μοντέλου.

Ας θεωρήσουμε ένα μικρό παράδειγμα. Έστω ότι τα δεδομένα εκπαίδευσης S αποτελούνται από τις εξής τρεις προτάσεις:

«αυτός διαβάζει ουμπέρτο έκο»,

«αυτή διαβαζει ένα διαφορετικό βιβλίο»,
«εκείνη διαβάζει ένα βιβλίο που της έδωσε ο νίκος»

και ας υπολογίσουμε την πιθανότητα *P(«αυτός διαβάζει ένα βιβλίο»)* για το bigram μοντέλο μεγίστης πιθανοφάνειας. Έχουμε λοιπόν:

$$P(\alpha\upsilon\tau\acute{o}\varsigma \mid < s >) = \frac{c(< s > \alpha\upsilon\tau\acute{o}\varsigma)}{\sum_w c(< s > w)} = \frac{1}{3}$$

$$P(\delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota \mid \alpha\upsilon\tau\acute{o}\varsigma) = \frac{c(\alpha\upsilon\tau\acute{o}\varsigma\, \delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota)}{\sum_w c(\alpha\upsilon\tau\acute{o}\varsigma\, w)} = \frac{1}{1}$$

$$P(\acute{\varepsilon}\nu\alpha \mid \delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota) = \frac{c(\,\delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota\ \acute{\varepsilon}\nu\alpha)}{\sum_w c(\delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota\ w)} = \frac{2}{3}$$

$$P(\beta\iota\beta\lambda\acute{\iota}o \mid \acute{\varepsilon}\nu\alpha) = \frac{c(\acute{\varepsilon}\nu\alpha\ \beta\iota\beta\lambda\acute{\iota}o)}{\sum_w c(\acute{\varepsilon}\nu\alpha\, w)} = \frac{1}{2}$$

$$P(< \backslash s > \mid \beta\iota\beta\lambda\acute{\iota}o) = \frac{c(\beta\iota\beta\lambda\acute{\iota}o < \backslash s >)}{\sum_w c(\beta\iota\beta\lambda\acute{\iota}o\ w)} = \frac{1}{2}$$

και τελικά:

P(«αυτός διαβάζει ένα βιβλίο»)=

$$P(\alpha\upsilon\tau\acute{o}\varsigma \mid < s >) \cdot P(\delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota \mid \alpha\upsilon\tau\acute{o}\varsigma) \cdot P(\acute{\varepsilon}\nu\alpha \mid \delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota) \cdot P(\beta\iota\beta\lambda\acute{\iota}o \mid \acute{\varepsilon}\nu\alpha) \cdot$$
$$P(< \backslash s > \mid \beta\iota\beta\lambda\acute{\iota}o) = \frac{1}{3} \cdot 1 \cdot \frac{2}{3} \cdot \frac{1}{2} \cdot \frac{1}{2} \approx 0.06$$

2.2. Smoothing

Ας θεωρήσουμε τώρα την πρόταση «ο νίκος διαβάζει ένα βιβλίο». Έχουμε:

$$P(\delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota \mid \nu\acute{\iota}\kappa o\varsigma) = \frac{c(\nu\acute{\iota}\kappa o\varsigma\ \delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota)}{\sum_w c(\nu\acute{\iota}\kappa o\varsigma\ w)} = \frac{0}{1}$$

που μας δίνει *P(*«ο νίκος διαβάζει ένα βιβλίο») = 0. Προφανώς αυτό είναι μια υποεκτίμηση της πιθανότητας της πρότασης *«ο νίκος διαβάζει ένα βιβλίο»*, αφού υπάρχει κάποια πιθανότητα να εμφανιστεί η συγκεκριμένη πρόταση. Για να δείξουμε γιατί είναι σημαντικό να δωθεί στην πιθανότητα αυτή μια μη μηδενική τιμή, ας αναλογιστούμε την πρωταρχική εφαρμογή των γλωσσικών μοντέλων, την αναγνώριση φωνής. Στην αναγνώριση φωνής προσπαθούμε να βρούμε μια πρόταση *s* που μεγιστοποιεί την ποσότητα:

$$P(s \mid A) = \frac{P(A \mid s) \cdot P(s)}{P(A)} \qquad (2.6)$$

για ένα δεδομένο ακουστικό σήμα *A*. Εάν η πιθανότητα *P(s)* είναι ίση με το μηδέν, τότε και η πιθανότητα *P(s|A)* θα είναι μηδέν και η συμβολοσειρά *s* δεν θα θεωρηθεί ποτέ σαν πιθανή πρόταση. Έτσι, οποτεδήποτε μια συμβολοσειρά *s,* τέτοια ώστε *P(s)=0,* εμφανίζεται κατά την διάρκεια μιας εφαρμογής αναγνώρισης φωνής, θα συμβαίνει λάθος. Αντιστοιχίζοντας όλες τις συμβολοσειρές με μια μη μηδενική πιθανότητα, αποφεύγουμε λάθη στην αναγνώριση φωνής.

Η τεχνική που χρησιμοποιούμε για το σκοπό αυτό ονομάζεται *smoothing*. Ο όρος smoothing περιγράφει τεχνικές παραγωγής περισσότερο ακριβών πιθανοτήτων από την εκτίμηση μέγιστης πιθανοφάνειας των πιθανοτήτων (όπως στους τύπους *2.3 ,2.5*). Η λέξη smoothing προέρχεται από το γεγονός ότι αυτές οι τεχνικές χρησιμοποιούνται για να κάνουν τις κατανομές πιθανοτήτων πιο ομοιόμορφες, αυξάνοντας τις χαμηλές πιθανότητες, όπως οι μηδενικές, και μειώνοντας τις υψηλές πιθανότητες. Οι μέθοδοι smoothing όχι μόνο αποτρέπουν τις μηδενικές πιθανότητες αλλά προσπαθούν επίσης να βελτιώσουν την ακρίβεια του γλωσσικού μοντέλου στο

σύνολο του. Όταν μια πιθανότητα έχει εκτιμηθεί από μη επαρκή δεδομένα εκπαίδευσης, οι τεχνικές smoothing μπορουν να βελτιώσουν σημαντικά την εκτίμηση.

Για να δώσουμε ένα παράδειγμα, μια απλή τεχνική smoothing είναι να θεωρήσουμε ότι κάθε bigram εμφανίζεται μια φορά περισσότερο απ'ότι στην πραγματικότητα. Έτσι θα έχουμε:

$$P(w_i \mid w_{i-1}) = \frac{1 + c(w_{i-1}w_i)}{\sum_{w_i}[1 + c(w_{i-1}w_i)]} = \frac{1 + c(w_{i-1}w_i)}{|V| + \sum_{w_i} c(w_{i-1}w_i)}, \qquad (2.7)$$

όπου V είναι το λεξικό, δηλαδή το σύνολο όλων των επιτρεπτών λέξεων. Ας θεωρήσουμε τώρα το παράδειγμα της προηγούμενης παραγράφου χρησιμοποιώντας τη νέα αυτή κατανομή πιθανοτήτων και θεωρώντας ότι το λεξικό V αποτελείται από όλες τις λέξεις που εμφανίζονται στα δεδομένα εκπαίδευσης S. Επομένως θα είναι |V| = 18.

Για την πρόταση «αυτός διαβάζει ένα βιβλίο», έχουμε τώρα:

$$P(\text{«αυτός διαβάζει ένα βιβλίο»}) =$$
$$P(\alpha\upsilon\tau\acute{o}\varsigma \mid < s >) \cdot P(\delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota \mid \alpha\upsilon\tau\acute{o}\varsigma) \cdot P(\acute{\varepsilon}\nu\alpha \mid \delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota) \cdot P(\beta\iota\beta\lambda\acute{\iota}o \mid \acute{\varepsilon}\nu\alpha) \cdot$$
$$P(< \backslash s > \mid \beta\iota\beta\lambda\acute{\iota}o) = \frac{2}{14} \cdot \frac{2}{12} \cdot \frac{3}{14} \cdot \frac{2}{13} \cdot \frac{2}{13} \approx 0.0001$$

Με άλλα λόγια , εκτιμήσαμε ότι η πρόταση «αύτος διαβάζει ένα βιβλίο» εμφανίζεται περίπου μια φορά σε κάθε δέκα χιλιάδες προτάσεις. Η εκτίμηση αυτή είναι περισσότερο λογική από την εκτίμηση μεγίστης πιθανοφάνειας (≈0.06). Για την πρόταση «ο νίκος διαβάζει ένα βιβλίο» έχουμε:

$$P(\text{«ο νίκος διαβάζει ένα βιβλίο»}) =$$
$$P(o \mid < s >) \cdot P(\nu\acute{\iota}\kappa o\varsigma \mid o) \cdot P(\delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota \mid \nu\acute{\iota}\kappa o\varsigma) \cdot P(\acute{\varepsilon}\nu\alpha \mid \delta\iota\alpha\beta\acute{\alpha}\zeta\varepsilon\iota) \cdot P(\beta\iota\beta\lambda\acute{\iota}o \mid \acute{\varepsilon}\nu\alpha) \cdot$$
$$P(< \backslash s > \mid \beta\iota\beta\lambda\acute{\iota}o) = \frac{1}{14} \cdot \frac{2}{13} \cdot \frac{1}{12} \cdot \frac{3}{14} \cdot \frac{2}{13} \cdot \frac{2}{13} \approx 0.00004$$

Πάλι βλέπουμε ότι η εκτίμηση αυτή είναι πιο λογική από την μηδενική πιθανότητα του μοντέλου μεγίστης πιθανοφάνειας.

Όπως είδαμε το smoothing είναι μια τεχνική αντιμετώπισης προβλημάτων που προκύπτουν από την έλλειψη επαρκών δεδομένων εκπαίδευσης. Υπάρχουν και άλλες μέθοδοι αντιμετώπισης τέτοιων προβλημάτων όπως είναι η *ομαδοποίηση των λέξεων (word classing).*

2.2.1 ***Additive smoothing***

Ένας από τους πιο απλούς τρόπους για *smoothing* που χρησιμοποιούνται στην πράξη είναι το *additive smoothing [2],[3],[4]*, που είναι απλώς μια γενικευμένη έκφραση της μεθόδου που περιγράφηκε στη σχέση *(2.7)*. Αντί να υποθέσουμε ότι κάθε *n-gram* εμφανίζεται μία παραπάνω φορά από όσες πραγματικά εμφανίζεται, υποθέτουμε ότι εμφανίζεται δ φορές παραπάνω, όπου τυπικά ισχύει $0 < \delta \leq 1$:

$$P_{add}(w_i \mid w_{i-n+1}^{i-1}) = \frac{\delta + c(w_{i-n+1}^{i})}{\delta |V| + \sum_{w_i} c(w_{i-n+1}^{i})} \qquad (2.8)$$

Οι *Lidstone* και *Jeffreys* χρησιμοποιούν για το δ την τιμή $\delta=1$. Οι *Gale* και *Church [5],[6]* υποστήριξαν ότι η μέθοδος αυτή έχει φτωχή απόδοση.

2.2.2 **Εκτιμήτρια *Good-Turing***

Η εκτιμήτρια *Good-Turing [7]* αποτελεί τη βάση για πολλές τεχνικές *smoothing*. Η εκτιμήτρια *Good-Turing* δηλώνει ότι για οποιοδήποτε *n-gram* που εμφανίζεται *r* φορές, πρέπει να υποθέτουμε ότι εμφανίζεται r^* φορές όπου :

$$r^* = (r+1)\frac{n_{r+1}}{n_r}, \qquad (2.9)$$

και όπου n_r είναι ο αριθμός των *n-grams* που εμφανίζονται ακριβώς *r* φορές στα δεδομένα εκπαίδευσης. Για να μετατρέψουμε τον αριθμό εμφανίσεων σε πιθανότητα, απλά κανονικοποιούμε . Για ένα *n-gram* α με *r* εμφανίσεις, έχουμε :

$$P_{GT}(\alpha) = \frac{r^*}{N} \qquad (2.10)$$

όπου $N = \sum_{r=0}^{\infty} n_r r^*$. Παρατηρούμε ότι :

$$N = \sum_{r=0}^{\infty} n_r r^* = \sum_{r=0}^{\infty} (r+1) n_{r+1} = \sum_{r=1}^{\infty} r n_r \ , \qquad (2.11)$$

δηλαδή το *N* είναι ίσο με το πραγματικό σύνολο των αριθμών εμφάνισης των *ngrams* στην κατανομή.

Για να παράγουμε αυτή την εκτιμήτρια, ας υποθέσουμε ότι υπάρχει ένα σύνολο από *s* διαφορετικά *n-grams* $\alpha_1,...,\alpha_s$ και ότι οι πραγματικές τους πιθανότητες ή συχνότητες εμφάνισης είναι $P_1,...,P_s$, αντίστοιχα. Ας εκτιμήσουμε τώρα την αληθινή πιθανότητα ενός *n-gram* α_i που εμφανίζεται *r* φορές σε κάποια δεδομένα, δεδομένου ότι δεν ξέρουμε την ταυτότητα του *n-gram* α_i ,αλλά ξέρουμε τις υποψήφιες πιθανότητες $P_1,...,P_s$. Ισοδύναμα μπορούμε να υπολογίσουμε την τιμή $E(P_i|c(\alpha_i)=r)$, όπου *E* είναι η αναμενόμενη τιμή και όπου $c(\alpha_i)$ είναι ο αριθμός των φορών που το *n-gram* α_i εμφανίζεται στα δεδομένα. Η ποσότητα αυτή αναλύεται ως εξής :

$$E(P_i \mid c(\alpha_i) = r) = \sum_{j=1}^{s} P(i=j \mid c(\alpha_i) = r) P_j \qquad (2.12)$$

Η πιθανότητα $P(i=j|c(\alpha_i)=r)$ είναι η πιθανότητα ενός άγνωστου *n-gram* α_i με r εμφανίσεις να είναι πραγματικά το j-οστό *n-gram* α_i (με αντίστοιχη συχνότητα εμφάνισης p_j). Αυτή η πιθανότητα μπορεί να γραφεί και ως εξής :

$$P(i=j \mid c(\alpha_i)=r) = \frac{P(c(\alpha_j)=r)}{\sum_{j=1}^{s} P(c(\alpha_j)=r)} = \frac{\binom{N}{r} P_j^r (1-P_j)^{N-r}}{\sum_{j=1}^{s} \binom{N}{r} P_j^r (1-P_j)^{N-r}} = \frac{P_j^r (1-P_j)^{N-r}}{\sum_{j=1}^{s} P_j^r (1-P_j)^{N-r}} \tag{2.13}$$

όπου $N = \sum_{j=1}^{s} c(\alpha_j)$ είναι το σύνολο των αριθμών εμφάνισης των *ngrams*. Αντικαθιστώντας αυτό στην εξίσωση *(2.12)*, παίρνουμε

$$E(P_i \mid c(\alpha_i)=r) = \frac{\sum_{j=1}^{s} P_j^{r+1} (1-P_j)^{N-r}}{\sum_{j=1}^{s} P_j^r (1-P_j)^{N-r}} \tag{2.14}$$

Ας θεωρήσουμε τώρα την ποσότητα $E_N(n_r)$ ως τον αναμενόμενο αριθμό *ngrams* που έχουν ακριβώς r αριθμούς εμφάνισης, δεδομένου ότι το σύνολο των αριθμών εμφάνισης των *n-grams* είναι N. Η ποσότητα αυτή είναι ίση με το άθροισμα της πιθανότητας για κάθε *n-gram* να έχει ακριβώς r αριθμούς εμφάνισης :

$$E_N(n_r) = \sum_{j=1}^{s} P(c(\alpha_j)=r) = \sum_{j=1}^{s} \binom{N}{r} P_j^r (1-P_j)^{N-r} \tag{2.15}$$

Αντικαθιστώντας την παράσταση αυτή στην εξίσωση *(2.14)* παίρνουμε :

$$E(P_i \mid c(\alpha_i)=r) = \frac{r+1}{N+1} \cdot \frac{E_{N+1}(n_{r+1})}{E_N(n_r)} \tag{2.16}$$

Αυτή είναι μια εκτιμήτρια για την αναμενόμενη πιθανότητα ενός *n-gram* a_i με αριθμό εμφάνισης r . Για να την εκφράσουμε με τους όρους του διορθωμένου αριθμού εμφάνισης r^* , χρησιμοποιώντας τη σχέση *(2.9)* εξάγουμε

$$r^* = N \cdot E(P_i \mid c(\alpha_i) = r) = N \frac{r+1}{N+1} \cdot \frac{E_{N+1}(n_{r+1})}{E_N(n_r)} \approx (r+1)\frac{n_{r+1}}{n_r}$$

(2.17)

Σημειώνουμε ότι στην παραπάνω εξίσωση χρησιμοποιούνται οι προσεγγίσεις $E_N(n_r) \approx n_r$ και $E_{N+1}(n_{r+1}) \approx n_{r+1}$. Με άλλα λόγια χρησιμοποιούμε τις εμπειρικές τιμές του n_r για να εκτιμήσουμε τις αναμενόμενες τιμές του.

Η εκτιμήτρια *Good-Turing* δεν μπορεί να χρησιμοποιηθεί όταν $n_r=0$. Γενικά είναι απαραίτητο να «εξομαλύνουμε» τα n_r, για παράδειγμα να προσαρμόσουμε τα n_r ώστε να είναι όλα μεγαλύτερα του μηδέν. Σχετικά πρόσφατα οι *Gale* και *Sampson [8]* πρότειναν έναν απλό και αποδοτικό αλγόριθμο για την εξομάλυνση αυτών των τιμών.

Στην πράξη, η εκτιμήτρια *Good-Turing* δε χρησιμοποιείται αυτούσια για *ngram smoothing*, επειδή δεν υποστηρίζει το συνδυασμό μοντέλων μεγαλύτερης τάξης με μοντέλα μικρότερης τάξης, συνδυασμό απαραίτητο για καλή απόδοση, όπως επεξηγείται στις επόμενες παραγράφους. Ωστόσο, χρησιμοποιείται σαν εργαλείο σε διάφορες άλλες τεχνικές *smoothing*.

2.2.3 ***Jelineck-Mercer smoothing***

Ας θεωρήσουμε την περίπτωση κατασκευής ενός *bigram* μοντέλου πάνω σε δεδομένα εκπαίδευσης από τα οποία παίρνουμε :

c(«παρατηρώ το») = 0
c(«παρατηρώ καταβεβλημένος») = 0

Τότε, σύμφωνα με το *additive smoothing*, αλλά και σύμφωνα με την εκτιμήτρια *Good-Turing*, θα έχουμε :

P(«το»|«παρατηρώ») = P(«καταβεβλημένος»|«παρατηρώ»)

Ωστόσο, διαισθητικά θα έπρεπε να ισχύει :

$$P(\text{«το»}|\text{«παρατηρώ»}) > P(\text{«καταβεβλημένος»}|\text{«παρατηρώ»})$$

καθώς η λέξη *«το»* είναι πολύ πιο κοινή από όσο η λέξη *«καταβεβλημένος»*. Για να εντάξουμε στο μοντέλο αυτή τη συμπεριφορά, μπορούμε να παρεμβάλουμε (*interpolate)* το *bigram* μοντέλο με ένα *unigram* μοντέλο. Το *unigram* μοντέλο δεν εξαρτά την πιθανότητα μιας λέξης από καμία άλλη λέξη, και έτσι απλώς αντανακλά τη συχνότητα των λέξεων στο κείμενο. Για παράδειγμα, το *unigram* μοντέλο μέγιστης πιθανοφάνειας *(maximum-likelihood unigram model)* είναι :

$$P_{ML}(w_i) = \frac{c(w_i)}{\sum_{w_i} c(w_i)} \qquad (2.18)$$

Μπορούμε να παρεμβάλουμε γραμμικά ένα *bigram* μοντέλο με ένα *unigram* μοντέλο ως εξής :

$$P_{interp}(w_i|w_{i-1}) = \lambda P_{ML}(w_i|w_{i-1}) + (1-\lambda)P_{ML}(w_i) \qquad (2.19)$$

Όπου $0 \le \lambda \le 1$. Επειδή :

$$P(\text{«το»}|\text{«παρατηρώ»}) = P(\text{«καταβεβλημένος»}|\text{«παρατηρώ»}) = 0$$

ενώ $P_{ML}(\text{«το»}) >> P_{ML}(\text{«καταβεβλημένος»})$, θα έχουμε ότι :

$$P_{interp}(\text{«το»}|\text{«παρατηρώ»}) > P_{interp}(\text{«καταβεβλημένος»}|\text{«παρατηρώ»})$$

όπως ακριβώς το επιθυμούμε.

Γενικά, είναι χρήσιμο να παρεμβάλουμε μεγαλύτερης τάξης *n-gram* μοντέλα με μικρότερης τάξης μοντέλα επειδή όταν υπάρχουν ανεπαρκή δεδομένα για να εκτιμηθεί μια πιθανότητα στο μεγαλύτερης τάξης μοντέλο, το μικρότερης τάξης μοντέλο μπορεί συχνά να προσφέρει χρήσιμη πληροφορία. Μια γενική κλάση παρεμβαλλόμενων μοντέλων περιγράφεται από τους *Jelinek* και *Mercer [9]*. Ένας

κομψός τρόπος υλοποίησης αυτής της παρεμβολής δίνεται από τους *Brown κ.ά. [10]* ως εξής :

$$P_{\text{int}erp}(w_i \mid w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} P_{ML}(w_i \mid w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) P_{\text{int}erp}(w_i \mid w_{i-n+2}^{i-1}) \quad (2.20)$$

Αυτό σημαίνει ότι το *n*-τάξης *smoothed* μοντέλο ορίζεται αναδρομικά σαν μία γραμμική παρεμβολή ανάμεσα στο *n*-τάξης μέγιστης πιθανοφάνειας μοντέλο και το (*n*-1)-τάξης *smoothed* μοντέλο. Για να τελειώσει η αναδρομή, μπορούμε να θεωρήσουμε το *smoothed* πρώτης τάξης μοντέλο ως την μεγίστης πιθανοφάνειας κατανομή, ή να θεωρήσουμε το *smoothed* 0-τάξης μοντέλο ως την ομοιόμορφη κατανομή

$$P_{unif}(w_i) = \frac{1}{|V|} \quad (2.21)$$

Για ένα σταθερό P_{ML} είναι δυνατό να ψάξουμε αποτελεσματικά για το $\lambda_{w_{i-n+1}^{i-1}}$ που μεγιστοποιεί την πιθανότητα μερικών δεδομένων, χρησιμοποιώντας τον αλγόριθμο *Baum-Welch [11]*. Για να εξάγουμε αποτελέσματα που έχουν νόημα, τα δεδομένα που χρησιμοποιούμε για εκτιμήσουμε το $\lambda_{w_{i-n+1}^{i-1}}$ πρέπει να είναι διαφορετικά από τα δεδομένα που χρησιμοποιούμε για να υπολογίσουμε το P_{ML} . Στη *held-out* παρεμβολή, κρατάμε ένα τμήμα των δεδομένων εκπαίδευσης για το λόγο αυτό, όπου αυτά τα «κρατημένα απ' έξω» δεδομένα δεν χρησιμοποιούνται στον υπολογισμό του P_{ML} . Εναλλακτικά, οι *Jelinek* και *Mercer* περιγράφουν μία τεχνική, γνωστή ως *deleted interpolation* ή *deleted estimation* όπου τα διαφορετικά μέρη των δεδομένων εκπαίδευσης εναλλάσσονται στην εκπαίδευση είτε του P_{ML} είτε του $\lambda_{w_{i-n+1}^{i-1}}$. Το τελικό αποτέλεσμα είναι ο μέσος όρος των επί μέρους αποτελεσμάτων.

Παρατηρούμε ότι το βέλτιστο $\lambda_{w_{i-n+1}^{i-1}}$ είναι διαφορετικό για διαφορετικά ιστορικά w_{i-n+1}^{i-1}. Για παράδειγμα, για κάποια συμφραζόμενα που έχουν παρατηρηθεί χιλιάδες φορές, θα είναι κατάλληλο ένα υψηλό λ καθώς η κατανομή μεγαλύτερης τάξης θα είναι πολύ αξιόπιστη. Αντίθετα, για ένα ιστορικό που έχει παρατηρηθεί μόνο μία φορά, θα είναι κατάλληλο ένα μικρό λ. Το να εκπαιδεύσουμε κάθε

παράμετρο $\lambda_{w_{i-n+1}^{i-1}}$ ανεξάρτητα , γενικά δεν είναι εύστοχο. Θα χρειαζόμαστε ένα τεράστιο ποσό δεδομένων για να εκπαιδεύσουμε με ακρίβεια τόσες πολλές ανεξάρτητες παραμέτρους. Έτσι, αντί γι' αυτό, οι *Jelinek* και *Mercer* προτείνουν τη διαίρεση των $\lambda_{w_{i-n+1}^{i-1}}$ σε έναν λογικό αριθμό τμημάτων *(buckets)*, και τον καθορισμό ίδιας τιμής για όλα τα $\lambda_{w_{i-n+1}^{i-1}}$ που ανήκουν στο ίδιο τμήμα, μειώνοντας με τον τρόπο αυτό τον αριθμό των ανεξάρτητων παραμέτρων που πρέπει να εκτιμηθούν. Στην ιδανική περίπτωση, θα έπρεπε να τοποθετήσουμε μαζί στο ίδιο τμήμα τα $\lambda_{w_{i-n+1}^{i-1}}$ αυτά για τα οποία έχουμε έναν εκ των προτέρων λόγο να πιστεύουμε ότι θα έχουν όμοιες τιμές. Οι *Bahl, Jelinek* και *Mercer [12]* προτείνουν την επιλογή αυτών των ομάδων $\lambda_{w_{i-n+1}^{i-1}}$ σύμφωνα με το $\sum_{w_i} c(w_{i-n+1}^{i})$, δηλαδή το συνολικό άθροισμα των αριθμών εμφάνισης που παρεμβάλλονται στην μεγαλύτερης τάξης κατανομή (το οποίο ισούται με το αριθμό φορών εμφάνισης στο αντίστοιχο ιστορικό). Όπως αναφέρθηκε πιο πάνω, αυτός ο συνολικός αριθμός φορών εμφάνισης θα πρέπει να συσχετίζεται με το πόσο πολύ θα πρέπει να βαρύνει η κατανομή μεγαλύτερης τάξης. Όσο μεγαλύτερος αυτός ο αριθμός φορών, τόσο υψηλότερη τιμή θα πρέπει να έχει το $\lambda_{w_{i-n+1}^{i-1}}$. Πιο συγκεκριμένα, οι *Bahl κ.ά.* προτείνουν το διαμερισμό *(partitioning)* της κλίμακας των δυνατών τιμών του συνολικού αριθμού φορών εμφάνισης και την τοποθέτηση στο ίδιο τμήμα *(bucket)* όλων των $\lambda_{w_{i-n+1}^{i-1}}$ που σχετίζονται με το ίδιο κομμάτι *(partition)*. Ο *Chen [13]* έδειξε ότι η διαίρεση *(bucketing)* σύμφωνα με το μέσο αριθμό φορών εμφάνισης ανά μη μηδενικό στοιχείο σε μια κατανομή $\frac{\sum_{w_i} c(w_{i-n+1}^{i})}{|\, w_i : c(w_{i-n+1}^{i}) > 0\,|}$ εξάγει καλύτερη απόδοση από τη χρησιμοποίηση της τιμής $\sum_{w_i} c(w_{i-n+1}^{i})$.

2.2.4 ***Katz Smoothing***

Η μέθοδος *Katz smoothing [14]* είναι μια προέκταση της μεθόδου *Good-Turing,* εισάγοντας το συνδυασμό *μοντέλων μεγαλύτερης τάξης (higher-order models)* με μοντέλα *μικρότερης τάξης (lower-order models)*. Αρχικά θα περιγράψουμε το Katz

smoothing για bigram μοντέλα. Για ένα bigram w_{i-1}^{i} με αριθμό εμφάνισης $r = c(w_{i-1}^{i})$ υπολογίζουμε τον διορθωμένο αριθμό εμφάνισης του χρησιμοποιώντας την εξίσωση:

$$c_{katz}(w_{i-1}^{i}) = \begin{cases} d_r \cdot r & if\ r > 0 \\ \alpha(w_{i-1}) \cdot P_{ML}(w_i) & if\ r = 0 \end{cases} \qquad (2.22)$$

Αυτό σημαίνει ότι σε όλα τα bigrams με ένα μη μηδενικό αριθμό εμφάνισης r έχουμε *μείωση (discount)* αυτού του αριθμού *σύμφωνα* με ένα συντελεστή d_r *(discounting ratio)*. Ο συντελεστής d_r είναι περίπου ίσος με $\frac{r^*}{r}$, δηλαδή ίσος με την μείωση που προβλέπεται από την εκτίμηση *Good-Turing,* και θα καθοριστεί αμέσως μετά. Οι αριθμοί εμφάνισης, που αφαιρούνται από bigrams με μη μηδενικό r, κατανέμονται στη συνέχεια στα bigrams με μηδενικό αριθμό εμφάνισης σύμφωνα με την επόμενη χαμηλότερης τάξης κατανομή, δηλαδή το unigram μοντέλο. Η τιμή $\alpha(w_{i-1})$ επιλέγεται έτσι ώστε το σύνολο των αριθμών εμφάνισης στην κατανομή $\sum_{w_i} c_{katz}(w_{i-1}^{i})$ να μένει αναλλοίωτο, δηλαδή να ισχύει:

$$\sum_{w_i} c_{katz}(w_{i-1}^{i}) = \sum_{w_i} c(w_{i-1}^{i}). \qquad (2.23)$$

Η κατάλληλη τιμή για το $\alpha(w_{i-1})$ είναι:

$$\alpha(w_{i-1}) = \frac{1 - \sum_{w_i : c(w_{i-1}^{i})>0} P_{katz}(w_i|w_{i-1})}{\sum_{w_i : c(w_{i-1}^{i})=0} P_{ML}(w_i)} = \frac{1 - \sum_{w_i : c(w_{i-1}^{i})>0} P_{katz}(w_i|w_{i-1})}{1 - \sum_{w_i : c(w_{i-1}^{i})>0} P_{ML}(w_i)} \qquad (2.24)$$

Για να υπολογίσουμε την πιθανότητα $P_{katz}(w_i|w_{i-1})$ από τον διορθωμένο αριθμό εμφάνισης, κανονικοποιούμε:

$$P_{katz}(w_i \mid w_{i-1}) = \frac{c_{katz}(w_{i-1}^{i})}{\sum_{w_i} c_{katz}(w_{i-1}^{i})} \qquad (2.25)$$

Η ποσότητα d_r υπολογίζεται με τον ακόλουθο τρόπο: οι μεγάλοι αριθμοί εμφάνισης θεωρούνται σωστοί, και επομένως δεν μειώνονται. Ειδικότερα, η μέθοδος Katz παίρνει $d_r=1$ για όλα τα r για τα οποία ισχύει $r>k$ για κάποιο k. Συνήθως στη μέθοδο

Katz είναι *k=5*. Οι συντελεστές μείωσης για τους χαμηλότερους αριθμούς εμφάνισης *r<=k* υπολογίζονται από την εφαρμογή του αλγορίθμου εκτίμησης Good-Turing στο σύνολο της κατανομής των bigrams. Δηλαδή το n_r στην εξίσωση *(2.9)* υποδηλώνει τους συνολικούς αριθμούς των bigrams που εμφανίζονται ακριβώς *r* φορές στα δεδομένα εκπαίδευσης. Αυτά τα d_r επιλέγονται έτσι ώστε οι τελικές μειώσεις στους αριθμούς εμφάνισης να είναι ανάλογες με τις μειώσεις που προβλέπονται από την εκτίμηση Good-Turing, και τέτοια ώστε το σύνολο των αριθμών εμφάνισης στο σύνολο της κατανομής των bigrams να είναι ίσο με τον σύνολο των αριθμών εμφάνισης που πρέπει να αποδωθούν στα bigrams με μηδενικούς αριθμούς εμφάνισης σύμφωνα με τον αλγόριθμο Good-Turing. Αυτός ο περιορισμός εκφράζεται από τις εξισώσεις:

$$1 - d_r = \mu(1 - \frac{r^*}{r}) \qquad (2.26)$$

για $r \in \{1, \ldots, k\}$ για κάποια σταθερά μ.

Ο αλγόριθμος Good-Turing προβλέπει ότι το σύνολο των αριθμών εμφάνισης που πρέπει να αποδωθεί στα bigrams με μηδενικό αριθμό εμφάνισης είναι

$$n_0 \cdot 0^* = n_0 \cdot \frac{n_1}{n_0} = n_1 \qquad (2.27)$$

Έτσι ο δεύτερος περιρισμός αντιστοιχεί στην εξίσωση:

$$\sum_{r=1}^{k} n_r \cdot (1 - d_r) \cdot r = n_1 \qquad (2.28)$$

Η μοναδική λύση των εξισώσεων είναι η:

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1) \cdot n_{k+1}}{n_1}}{1 - \frac{(k+1) \cdot n_{k+1}}{n_1}} \qquad (2.29)$$

Η μέθοδος Katz smoothing για μοντέλα υψηλότερης τάξης καθορίζεται με ανάλογο τρόπο. Όπως μπορούμε να δούμε στην εξίσωση *(2.22)*, το bigram μοντέλο καθορίζεται σε σχέση με το unigram μοντέλο. Γενικά το *Katz n-gram μοντέλο*

καθορίζεται σε σχέση με το *Katz (n-1)-gram μοντέλο,* με παρόμοιο τρόπο όπως και στο *Jelinek-Mercer smoothing.* Για να τελειώσει η αναδρομή, το Katz unigram μοντέλο θεωρείται ότι είναι το unigram μοντέλο *μεγίστης πιθανοφάνειας (maximum likelihood).*

Όπως αναφέρθηκε και στην παράγραφο *(§2.2.2)*, είναι συχνά απαραίτητο να γίνει smoothing στην ποσότητα n_r όταν χρησιμοποιούμε την εκτίμηση Good-Turing, για εκείνα τα n_r που είναι πολύ χαμηλά. Όμως, στο Katz smoothing αυτό δεν είναι απαραίτητο επειδή η εκτίμηση Good-Turing χρησιμοποιείται μόνο για μικρούς αριθμούς εμφάνισης $r<=k$ και το n_r είναι γενικά πολύ υψηλό για αυτές τις τιμές του r.

2.2.5 ***Absolute Discounting***

Η τεχνική *absolute discounting [15],* όπως και η τεχνική *Jelinek-Mercer,* εμπλέκει την *παρεμβολή (interpolation)* υψηλότερης και χαμηλότερης τάξης μοντέλων. Όμως, αντί να πολλαπλασιάζει την κατανομή μεγίστης πιθανοφάνειας του μοντέλου υψηλότερης τάξης με ένα συντελεστή $\lambda_{w_{i-n+1}^{i-1}}$, η κατανομή υψηλότερης τάξης δημιουργείται από την αφαίρεση ενός συντελεστή $D<=1$ από κάθε μη μηδενικό αριθμό εμφάνισης. Έτσι από την εξίσωση *(2.20)*:

$$P_{\text{int erp}}(w_i \mid w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} P(w_i \mid w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) P_{\text{int erp}}(w_i \mid w_{i-n+2}^{i-1}) \qquad (2.30)$$

έχουμε:

$$P_{abs}(w_i \mid w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^{i}) - D, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i-1}}) P_{abs}(w_i \mid w_{i-n+2}^{i-1}) \qquad (2.31)$$

Για να αθροίζει αυτή η κατανομή στο 1 παίρνουμε:

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1} \bullet) \qquad (2.32)$$

όπου το $N_{1+}(w_{i-n+1}^{i-1} \bullet)$ είναι:

$$N_{1+}(w_{i-n+1}^{i-1} \bullet) = |\{w_i : c(w_{i-n+1}^{i-1} w_i) > 0\}|$$

και θεωρώντας ότι *0<=D<=1*. Έχει βρεθεί ότι η καταλληλότερη τιμή για το *D* είναι η:

$$D = \frac{n_1}{n_1 + 2n_2} \tag{2.33}$$

όπου τα n_1, n_2 είναι ο συνολικός αριθμός από n-grams που έχουν ακριβώς μία ή δύο εμφανίσεις στα δεδομένα εκπαίδευσης.

2.2.6 Περίληψη αλγορίθμων

Όπως σημείωσαν οι *Kneser* και *Ney [16]*, οι περισσότεροι αλγόριθμοι *smoothing* που υπάρχουν μπορούν να περιγραφούν με την ακόλουθη εξίσωση:

$$P_{smooth}(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} \alpha(w_i \mid w_{i-n+1}^{i-1}) & \alpha\nu\ c(w_{i-n+1}^{i-1}) > 0 \\ \gamma(w_{i-n+1}^{i-1}) p_{smooth}(w_i \mid w_{i-n+2}^{i-1}) & \alpha\nu\ c(w_{i-n+1}^{i-1}) = 0 \end{cases} \tag{2.34}$$

Αυτό σημαίνει ότι, αν ένα *n-gram* έχει μη μηδενικό αριθμό εμφάνισης τότε χρησιμοποιούμε την κατανομή $\alpha(w_i \mid w_{i-n+1}^{i-1})$. Αλλιώς, «οπισθοχωρούμε» *(backoff)* στη μικρότερης τάξης κατανομή $p_{smooth}(w_i \mid w_{i-n+2}^{i-1})$, όπου ο συντελεστής $\gamma(w_{i-n+1}^{i-1})$ επιλέγεται έτσι ώστε η υπό συνθήκη κατανομή να αθροίζει στη μονάδα. Αναφερόμαστε στους αλγορίθμους που εμπίπτουν σε αυτή την κατηγορία ως *backoff* μοντέλα. Η μέθοδος *Katz smoothing* είναι το τυπικό παράδειγμα *backoff smoothing*.

Διάφοροι άλλοι αλγόριθμοι *smoothing* εκφράζονται ως γραμμική παρεμβολή μεγαλύτερης με μικρότερης τάξης *n-gram* μοντέλων, όπως είδαμε για παράδειγμα στην εξίσωση *(2.20)* που την υπενθυμίζουμε κι εδώ :

$$P_{smooth}(w_i \mid w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} P_{ML}(w_i \mid w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) P_{smooth}(w_i \mid w_{i-n+2}^{i-1})$$

Η σχέση αυτή μπορεί να ξαναγραφεί ως εξής :

$$P_{smooth}(w_i \mid w_{i-n+1}^{i-1}) = \alpha'(\ w_i \mid w_{i-n+1}^{i-1}) + \gamma(w_{i-n+1}^{i-1})\ P_{smooth}(w_i \mid w_{i-n+2}^{i-1}) \qquad (2.35)$$

όπου :

$$\alpha'(w_i \mid w_{i-n+1}^{i-1}) = \lambda_{\ w_{i-n+1}^{i-1}} P_{ML}(w_i \mid w_{i-n+1}^{i-1}) \qquad (2.36)$$

και $$\gamma(w_{i-n+1}^{i-1}) = (1 - \lambda_{\ w_{i-n+1}^{i-1}})\ . \qquad (2.37)$$

Στη συνέχεια, ορίζοντας :

$$\alpha(w_i \mid w_{i-n+1}^{i-1}) = \alpha'(w_i \mid w_{i-n+1}^{i-1}) + \gamma(w_{i-n+1}^{i-1}) P_{smooth}(w_{i-n+2}^{i-1}) \qquad (2.38)$$

παρατηρούμε ότι αυτά τα μοντέλα μπορούν να γραφούν στη μορφή της εξίσωσης *(2.35)*. Αναφερόμαστε στα μοντέλα αυτού του τύπου ως *παρεμβαλλόμενα (interpolated) μοντέλα*.

Η ειδοποιός διαφορά ανάμεσα σε *backoff* και *interpolated* μοντέλα είναι ότι στον καθορισμό της πιθανότητας των *n-grams* που έχουν μη μηδενικό αριθμό εμφάνισης, τα *interpolated* μοντέλα χρησιμοποιούν πληροφορία από μικρότερης τάξης κατανομές, ενώ τα *backoff* μοντέλα όχι. Και στις δύο περιπτώσεις χρησιμοποιούνται μικρότερης τάξης κατανομές για τον καθορισμό της πιθανότητας των *n-grams* με μηδενικό αριθμό εμφάνισης.

Σημειώνουμε ότι είναι εύκολο να δημιουργήσουμε μία *backoff* έκδοση ενός αλγορίθμου παρεμβολής. Αντί να χρησιμοποιήσουμε την εξίσωση *(2.38)*, μπορούμε απλώς να θέσουμε :

$$\alpha(w_i \mid w_{i-n+1}^{i-1}) = \alpha'(w_i \mid w_{i-n+1}^{i-1}) \qquad (2.39)$$

και στη συνέχεια να ρυθμίσουμε κατάλληλα την τιμή του $\gamma(w_{i-n+1}^{i-1})$, ώστε οι πιθανότητες να αθροίζουν στη μονάδα.

2.3 Γλωσσικά μοντέλα βασισμένα σε κλάσεις

Μέχρι στιγμής είδαμε ότι ένα γλωσσικό μοντέλο μπορεί να βρίσκει σχέσεις μεταξύ ξεχωριστών λέξεων. Είναι ξεκάθαρο, όμως, ότι μερικές λέξεις μοιάζουν πολύ με κάποιες άλλες, τόσο από σημασιολογικής, όσο και από συντακτικής άποψης. Για παράδειγμα, θα ήταν αναμενόμενο να παρατηρήσουμε παρόμοια πιθανότητα κατανομής λέξεων γύρω από τη λέξη *«καθαρή»*, με αυτή που θα παρατηρούσαμε γύρω από τη λέξη *«καθαρότερη»*. Φυσικά οι κατανομές αυτές δε θα είναι πανομοιότυπες, καθώς είναι σπάνιο να ακούσουμε κάποιον να λέει *«η καθαρότερη Δευτέρα»*. Αν λοιπόν μπορέσουμε, με επιτυχημένο τρόπο, να χωρίσουμε τις λέξεις σε *κατηγορίες (κλάσεις),* είναι δυνατό να σχεδιαστεί ένα γλωσσικό μοντέλο που να ανακαλύπτει σχέσεις ανάμεσα σε κλάσεις. Η προσέγγιση αυτή έχει τα ακόλουθα πλεονεκτήματα :

- Στα *βασισμένα σε κλάσεις γλωσσικά μοντέλα (class-based language models)* οι λέξεις που ανήκουν σε μια κλάση έχουν από κοινού κάποια στατιστικά στοιχεία και κατά συνέπεια γίνεται δυνατό να γενικευθεί το γλωσσικό μοντέλο, αναθέτοντας υψηλότερης τάξης πιθανότητες συνδυασμούς λέξεων που δεν εμφανίστηκαν ποτέ στο κείμενο εκπαίδευσης. Η ικανότητα αυτή του μοντέλου να επεξεργάζεται με κάποια λογική γεγονότα που δεν έχουν παρατηρηθεί ονομάζεται *ευρωστία (robustness)* του γλωσσικού μοντέλου.

- Η κατηγοριοποίηση των λέξεων μπορεί να μειώσει την ποικιλία των συμφραζομένων σε ένα μοντέλο, αντιμετωπίζοντας έτσι το πρόβλημα της *σποραδικότητας (sparseness)* των δεδομένων στο κείμενο εκπαίδευσης.

- Η μείωση της ποικιλίας των συμφραζομένων οδηγεί σε ένα πιο συμπαγές *(compact)* γλωσσικό μοντέλο με λιγότερες παραμέτρους. Σαν αποτέλεσμα, οι απαιτήσεις σε χώρο για την αποθήκευσή του μειώνονται, πράγμα που μπορεί να είναι σημαντικό από πρακτική σκοπιά.

Στη συνέχεια, θα εισάγουμε τους μαθηματικούς συμβολισμούς για το χειρισμό των κλάσεων λέξεων και θα περιγράψουμε πως αυτές εφαρμόζονται στην ανάπτυξη γλωσσικών μοντέλων.

2.3.1 Κλάσεις λέξεων

Ως *κλάση* θα θεωρούμε οποιαδήποτε ομαδοποίηση λέξεων. Έστω ότι υπάρχουν N_c τέτοιες κλάσεις που δηλώνονται ως εξής :

$$\mathbf{C} = \{c_0, c_1, \ldots, c_{Nc}\} \tag{2.40}$$

Ας ορίσουμε μια συνάρτηση $C(\cdot)$ που αντιστοιχίζει κάθε λέξη $w_i : i \in \{0,1,\ldots,N_w\}$ σε μία ή περισσότερες κλάσεις $c_j : j \in \{0,1,\ldots,N_c\}$, π.χ. :

$$c_j = C(w_i) \qquad j \in \{0,1,\ldots,N_c\text{-}1\} \qquad \text{και} \qquad i \in \{0,1,\ldots,N_w\text{-}1\} \tag{2.41}$$

όπου c_j είναι η κλάση στην οποία η συνάρτηση $C(\cdot)$ αντιστοιχίζει το w_i. Αν αυτή η αντιστοίχιση είναι *«πολλά προς ένα» (many-to-one)*, τότε μιλάμε για ντετερμινιστική σχέση μεταξύ μελών και κλάσης *(deterministic membership)* , ενώ αν είναι *«πολλά προς πολλά» (many-to-many)* αναφερόμαστε σε στοχαστική σχέση *(stochastic membership)*.

Υποθέτουμε τώρα ότι η πιθανότητα να παρατηρήσουμε μια λέξη *w(i)* ορίζεται απολύτως από τη γνώση της κλάσης στην οποία ανήκει, ώστε να μπορούμε να γράψουμε:

$$P(w_i \mid w_0^{i-1}) = P(w_i \mid c_i) \cdot P(c_i \mid w_0^{i-1}) \tag{2.42}$$

όπου w_0^{i-1} είναι το *ιστορικό (history)* της λέξης w_i .

Στην περίπτωση του *stochastic membership*, μας επιτρέπεται να αναλύσουμε τις εκτιμήτριες της υπό συνθήκη πιθανότητας ως εξής :

$$P(w_i \mid w_0^{i-1}) = \sum_{\forall c: c \in C(w(i))} P(w_i \mid c) \cdot P(c \mid w_0^{i-1}) \qquad (2.43)$$

Στη συνέχεια, ομαδοποιώντας το ιστορικό σε ισοδύναμες κλάσεις προκύπτει :

$$P\left(c_j \mid w_0^{i-1}\right) = \sum_{\forall h: h \in H(w(i))} P\left(c_j \mid h\right) \cdot P\left(h \mid w_0^{i-1}\right) \qquad (2.44)$$

Σαν αντιστοίχιση του ιστορικού μιας λέξης με ισοδύναμες κλάσεις μπορούμε να επιλέξουμε τις *n-1* πιο πρόσφατες κλάσεις που παρατηρήθηκαν :

$$\mathbf{H}(w_i) = \{ c_{i-n+1}, c_{i-n+2}, \ldots, c_{i-1} \} \qquad (2.45)$$

Από την παραπάνω αντιστοίχιση παίρνουμε ένα *class-based* γλωσσικό μοντέλο. Παρατηρούμε ότι όταν η συνάρτηση *C(·)* είναι *one-to-many,* η εξίσωση *(2.45)* αναπαριστά μια *many-to-many* αντιστοίχιση.

Όταν η αντιστοίχιση του ιστορικού με ισοδύναμες κλάσεις είναι *many-to-one*, η εξίσωση *(2.44)* απλοποιείται ως εξής :

$$P\left(c(i) \mid w_0^{i-1}\right) \approx P\left(c_j \mid H(w_i)\right) \qquad (2.46)$$

Έτσι, μπορούμε να ξαναγράψουμε την σχέση *(2.43)* ως εξής :

$$P\left(w(i) \mid w_0^{i-1}\right) \approx \sum_{\forall c: c \in C(w(i))} P\left(w_i \mid C(w_i)\right) \cdot P\left(C(w_i) \mid H(w_i)\right) \qquad (2.47)$$

Η παραπάνω εξίσωση χρησιμοποιήθηκε για την κατασκευή *class-based bigram* γλωσσικών μοντέλων, όπου οι κλάσεις ήταν είτε τα *μέρη του λόγου (part-of-speech based model),* είτε οι *ρίζες των λέξεων (stem based model)*, είτε ο συνδυασμός τους. Στα *part-of-speech* μοντέλα εκμεταλλευόμαστε τη γραμματική λειτουργία κάθε λέξης, ενώ τα *stemmed* μοντέλα αντιμετωπίζουν τον πολυμορφισμό της ελληνικής

γλώσσας. Τα μοντέλα αυτά έχουν ανταγωνιστική απόδοση σε σχέση με τα αντίστοιχα *word-based* μοντέλα όταν τα δεδομένα εκπαίδευσης χαρακτηρίζονται από σποραδικότητα. Αντίθετα, αποδίδουν λιγότερο καλά όταν το ποσό του κειμένου εκπαίδευσης αυξάνει. Προκειμένου να καλυφθεί αυτό το χάσμα, χρησιμοποιήσαμε το συνδυασμό των δύο αυτών τεχνικών, καθώς και *υβριδικά μοντέλα (hybrid word- and class-based models)* με μίξη κλάσεων και λέξεων. Τα *stemmed* και *part-of-speech* μοντέλα περιγράφονται αναλυτικότερα στα κεφάλαια 3 και 4.

2.4 Υπολογισμός της απόδοσης ενός γλωσσικού μοντέλου (*Perplexity*)

Με δεδομένο ότι ο σκοπός ενός γλωσσικού μοντέλου είναι να «μαντέυει» λέξεις, ακούγεται φυσικό να υπολογίσουμε την απόδοση του μοντέλου μέσω των πιθανοτήτων, που το συγκεκριμένο μοντέλο δίνει στις λέξεις ενός δείγματος κειμένου (sample text). Το κείμενο αυτό ονομάζεται *testing text.* Ο γεωμετρικός μέσος όρος αυτών των πιθανοτήτων δείχνει να είναι ένα καλό μέτρο της ικανότητας του μοντέλου να «μαντεύει» λέξεις. Η *αβεβαιότητα (perplexity),* το πρότυπο μέγεθος για συγκρίσεις αποδόσεων μεταξύ γλωσσικών μοντέλων, είναι απλά ο αντίστροφος αριθμός του γεωμετρικού μέσου όρου που αναφέραμε παραπάνω.

Ας δούμε, λοιπόν, το μέτρο της αβεβαιότητας πιο αναλυτικά. Έστω ότι $W=w_1,...,w_i,...,w_n$, μια ακολουθία λέξεων στο testing text και $c_{k(i)}$ το context που το μοντέλο «μαντεύει» για την λέξη w_i. Επιπλέον, έστω $P(w_i=w_{l(i)}|c_{k(i)})$ η πιθανότητα που αντιστοιχίζεται στη λέξη w_i από το μοντέλο. Η συνολική πιθανότητα *TP* για την ακολουθία των λέξεων θα είναι:

$$TP = P(W) = P(w[1:n]) = \prod_{i=1}^{i=n} P(w_i \mid c_i) \qquad (2.48)$$

Η *αβεβαιότητα (perplexity) PP* που περιγράψαμε διαισθητικά ως το γεωμετρικό μέσο όρο των πιθανοτήτων θα είναι:

$$PP = (TP)^{-\frac{1}{n}} \tag{2.49}$$

Για ένα μεγάλο δείγμα κειμένου η συνολική πιθανότητα TP μπορεί να είναι εξαιρετικά μικρή. Επομένως, είναι πρακτικά καλύτερη η χρησιμοποίηση του *λογαρίθμου της συνολικής πιθανότητας LTP*:

$$LTP = \log_2(TP) = \sum_{i=1}^{i=n} \log_2(P(w_i \mid c_i)) \tag{2.50}$$

και του *λογαρίθμου της αβεβαιότητας LPP:*

$$LPP = \log_2(PP) = -\frac{1}{n}\log_2(TP) = -\frac{1}{n}LTP \tag{2.51}$$

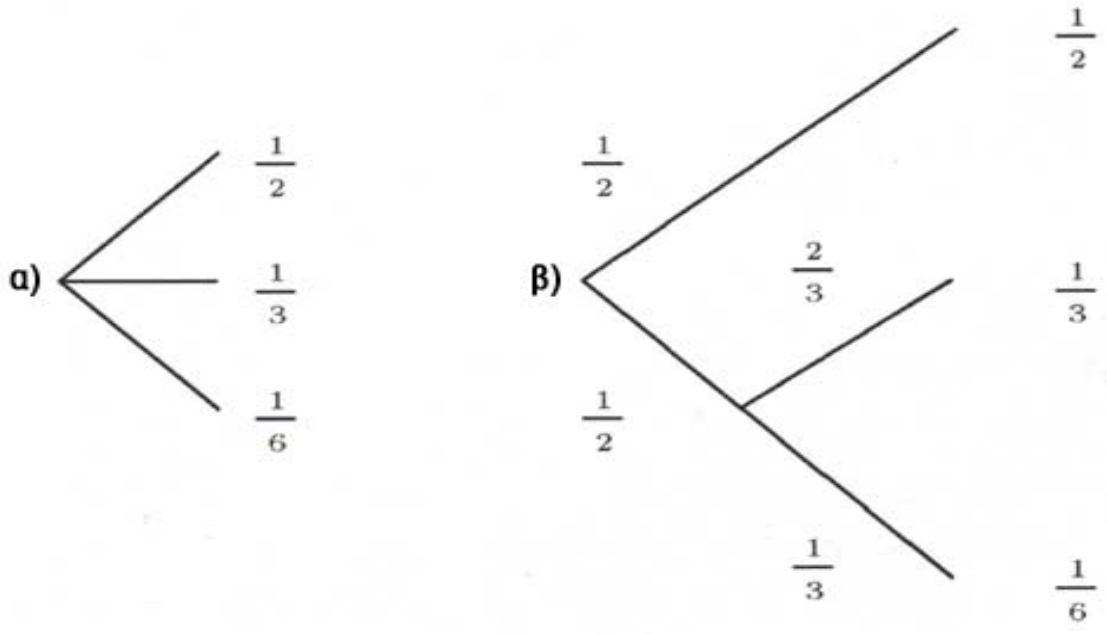
2.4.1 Πληροφορία, εντροπία και αβεβαιότητα από την οπτική γωνία της θεωρίας πληροφοριών

Στο κεφάλαιο αυτό θα εξάγουμε το λογάριθμο της πιθανότητας LP και της αβεβαιότητας PP, για τη μέτρηση της ποιότητας ενός μοντέλου, μέσω της θεωρίας πληροφοριών. Επίσης, θα εισάγουμε και την έννοια της εντροπίας, η οποία έχει άμεση σχέση με τις δύο προηγούμενες έννοιες.

Η θεωρία πληροφοριών ασχολείται με πηγές πληροφορίας. Με απλά λόγια, μπορούμε να φανταστούμε μια πηγή πληροφορίας σαν μια συσκευή που παράγει στην έξοδο της σύμβολα, τα οποία επιλέγονται μέσα από ένα πεπερασμένο σύνολο $V=\{x_1, \ldots, x_l\}$, γνωστό στον παρατηρητή.Τα σύμβολα επιλέγονται με βάση έναν στατιστικό κανόνα, που χαρακτηρίζει τη συσκευή. Συμβολίζουμε την πιθανότητα του συμβόλου x_i που παρατηρείται με $P(x_i)$. Όταν μια πηγή πληροφορίας παράγει στην

έξοδο της ένα σύμβολο, παρέχει πληροφορία με το να εξαλείφει την αβεβαιότητα για την ταυτότητα του συγκεκριμένου συμβόλου. Δηλαδή, μια πηγή παράγει περισσότερη πληροφορία, εάν η αβεβαιότητα για το επόμενο σύμβολο είναι μεγαλύτερη. Πώς μπορούμε, όμως, να μετρήσουμε την αβεβαιότητα για το επόμενο σύμβολο; Αν υπάρχει κάποιο μέτρο, έστω $H(P(x_1, \ldots, x_l))$, θα πρέπει να έχει τις παρακάτω ιδιότητες:

1) Το $H(P(x_1, \ldots, x_l))$ πρέπει να είναι συνεχές για τις πιθανότητες $P(x_i)$,
2) Αν όλα τα $P(x_i)$ είναι ίσα ($P(x_i) = \frac{1}{l}$), τότε το $H(P(x_1, \ldots, x_l))$ πρέπει να είναι μονοτονική αύξουσα συνάρτηση για τα l. Με άλλα λόγια, αν όλα τα σύμβολα είναι ισοπίθανα, υπάρχει μεγαλύτερη αβεβαιότητα όσο αυξάνει ο αριθμός των συμβόλων.
3) Εάν η επιλογή του επόμενου συμβόλου αναλυθεί σε δύο επιτυχημένες επιλογές, τότε το αρχικό $H(P(x_1, \ldots, x_l))$ πρέπει να είναι το *σταθμικό άθροισμα (weighted sum)* των H τιμών από κάθε επιλογή. Αυτό παριστάνεται στο *Σχήμα (2.1)*. Στην πρώτη περίπτωση έχουμε τρεις δυνατότητες με πιθανότητες



$P(x_i) = \frac{1}{2}, P(x_2) = \frac{1}{3}, P(x_3) = \frac{1}{6}$. Στην δεύτερη περίπτωση αρχικά

Σχήμα 2.1

επιλέγουμε ανάμεσα σε δύο δυνατότητες καθεμιά από τις οποίες έχει πιθανότητα $\frac{1}{2}$, και αν εμείς επιλέξουμε την δεύτερη δυνατότητα θα κάνουμε ακόμα μία επιλογή μεταξύ δύο δυνατοτήτων με πιθανότητες $\frac{2}{3}$ και $\frac{1}{3}$. Απαιτούμε, λοιπόν, να ισχύει:

$$H(\frac{1}{2},\frac{1}{3},\frac{1}{6}) = H(\frac{1}{2},\frac{1}{2}) + \frac{1}{2}\cdot H(\frac{2}{3},\frac{1}{3}).$$

Έχει αποδειχθεί *[17]* ότι το μόνο $H(P(x_1, \ldots, x_l))$ που ικανοποιεί τους τρεις αυτούς περιορισμούς είναι της μορφής:

$$H = -k\cdot\sum P(x_i)\cdot\log(P(x_i)) \qquad (2.52)$$

Η σταθερά *k* καθορίζει μερικώς την επιλογή μιας μονάδας μέτρησης. Ποσότητες της μορφής

$$H = -\sum P(x_i)\cdot\log(P(x_i)) \qquad (2.53)$$

παίζουν κεντρικό ρόλο στην θεωρία πληροφορίας σαν μονάδα μέτρησης της πληροφορίας, της επιλογής και της αβεβαιότητας. *[18]*

Ένας τρόπος για να καταλάβουμε διαισθητικά γιατί χρησιμοποιείται ο λογάριθμος είναι να κοιτάξουμε στην πληροφορία που παρέχεται από μια πηγή με *l* ισοπίθανα σύμβολα. Σύμφωνα με τον τύπο *(2.53)* το περιεχόμενο πληροφορίας μιας τέτοιας πηγής είναι:

$$H(X) = \sum_{i=1}^{i=l}\frac{1}{l}\cdot\log(l) = \frac{1}{l}\cdot l\cdot\log(l) = \log(l) \qquad (2.54)$$

Αν η πηγή βγάλει στην έξοδο της δυο σύμβολα στη σειρά, θα πάρουμε δύο φορές μια τέτοια πληροφορία. Όμως, η εμφάνιση δύο συμβόλων στη έξοδο της πηγής είναι ισοδύναμη με την εμφάνιση στην έξοδο δύο ανεξάρτητων πηγών πληροφορίας ενός από τα l^2 σύμβολα, με την ίδια πιθανότητα. Η ποσότητα πληροφορίας της δεύτερης

πηγής θα είναι,επομένως , διπλάσια από την ποσότητα πληροφορίας της πρώτης. Πράγματι,όμως , επειδή χρησιμοποιούμε λογάριθμο έχουμε:

$$log(l^2)=2 \cdot log(l) \qquad (2.55)$$

Επομένως, ο λογάριθμος συμφωνεί με την διαίσθησή μας για τις ποσότητες πληροφορίας.

Ένας άλλος τρόπος για να καταλάβουμε την ισότητα *(2.53)* είναι το να ξαναγράψουμε την ισότητα ως εξής:

$$H = -\sum P(x_i) \cdot \log(P(x_i)) = H = \sum P(x_i) \cdot \log(\frac{1}{P(x_i)}) \qquad (2.56)$$

Αν X είναι μια τυχαία μεταβλητή (η πηγή) για ένα σύνολο $V=\{x_i,...,x_l\}$, τότε το H είναι η *αναμενόμενη τιμή (expected value)* του $\log(\frac{1}{P(x_i)})$, όπου $\frac{1}{P(x_i)}$ είναι η αβεβαιότητα που σχετίζεται με το σύμβολο x_i. Αν το x_i δεν είναι πολύ πιθανό τότε το $\frac{1}{P(x_i)}$ είναι πολύ μεγάλο, κάτι που συμφωνεί και με τη διαίσθηση ότι συμβάντα μικρής πιθανότητας μεταφέρουν ένα μεγάλο ποσό αβεβαιότητας.

Σαν ένα παράδειγμα αυτού του ορισμού, ας θεωρήσουμε την εντροπία μιας μεταβλητής που μπορεί να πάρει μόνο μια τιμή, φυσικά με πιθανότητα 1:

$$H(X) = 1 \cdot \log\frac{1}{1} = 0$$

Οι λογάριθμοι στην εξίσωση *(2.53)* συνήθως χρησιμοποιούνται με βάση το 2 και σε αυτήν την περίπτωση, η μονάδα μέτρησης της πληροφορίας είναι το *bit.* Για παράδειγμα, η πληροφορία που παρέχεται από μια ομοιόμορφη πηγή με δύο σύμβολα είναι ένα bit:

$$I=log_2 2=1 \ bit$$

Σύμφωνα με τη θεωρία πληροφορίας, κατά μέσο όρο, χρειάζονται H *bits* για να αναπαρασταθεί ένα σύμβολο από μια πηγή με εντροπία H.Επιπλέον, μια πηγή με

εντροπία H παρέχει τόση πληροφορία, όση μια πηγή που επιλέγει τα σύμβολά της ανεξάρτητα και με ίσες πιθανότητες, από ένα λεξικό μεγέθους

$$l = 2^H \qquad (2.57)$$

Αυτό συμβαίνει γιατί, σύμφωνα με την εξίσωση *(2.54)* η εντροπία της δεύτερης πηγής είναι:

$$log2^H=H \qquad (2.58)$$

Τί γίνεται, όμως, με τις πηγές πληροφορίας που δεν επιλέγουν τα σύμβολά τους ανεξάρτητα από προηγούμενα σύμβολα; Έστω ότι x_i είναι το i^{th} σύμβολο στην έξοδο της πηγής και x_i^j, $i<=j$ μια συντομογραφία της ακολουθίας $x_i,x_{i+1},\ldots,x_j$. Γι'αυτήν την πιο γενική περίπτωση η εντροπία ορίζεται ως εξής:

$$H = -\lim_{n\to\infty}\frac{1}{n}\cdot \sum_{x[1:n]\in V^n} P(x_i^j\cdot \log(P(x_i^j)) \qquad (2.59)$$

Υπάρχει μια κατηγορία πηγών πληροφορίας για την οποία μπορούμε να απλοποιήσουμε τη σχέση *(2.59)*. Αυτή η κατηγορία πηγών ονομάζονται *εργοδικές πηγές (ergodic sources)*. Αν και ένας ακριβής ορισμός της εργοδικότητας είναι πολύπλοκος, η γενική ιδέα είναι απλή.

«Σε μια εργοδική διαδικασία κάθε ακολουθία που παράγεται από τη διαδικασία είναι η ίδια από πλευράς στατιστικής. Δηλαδή η συχνότητα γραμμάτων, η συχνότητα ζευγών γραμμάτων κλπ. που παράγεται από συγκεκριμένες ακολουθίες, όσο τα μήκη των ακολουθιών αυξάνουν, θα προσεγγίζει το άπειρο ανεξάρτητα από την ακολουθία. Στην πραγματικότητα αυτό δεν είναι αληθινό για όλες τις ακολουθίες, αλλά το σύνολο των ακολουθιών για τις οποίες δεν ισχύει έχει σχεδόν μηδενική πιθανότητα. Με άλλα λόγια η ιδιότητα της εργοδικότητας σημαίνει στατιστική ομοιογένεια.» [17, σελ.45,46]

Στη περίπτωση των εργοδικών πηγών πληροφορίας ο τύπος *(2.59)* γίνεται:

$$H = -\lim_{n\to\infty}\frac{1}{n}\log(P(x_i^j)) \qquad (2.60)$$

Δηλαδή, μπορούμε να υπολογίσουμε την εντροπία *H* από μεγάλου μήκους ακολουθίες συμβόλων που παράγονται από την πηγή.

Πώς μπορούμε να εφαρμόσουμε την θεωρία πληροφοριών στα γλωσσικά μοντέλα; Η γλώσσα μπορεί να θεωρηθεί σαν μια πηγή πληροφορίας της οποίας τα σύμβολα εξόδου είναι λέξεις από το λεξικό $V=\{w_1, \ldots, w_m\}$. Μπορούμε να χρησιμοποιήσουμε τον τύπο *(2.60)* για να υπολογίσουμε την πληροφορία που περιέχεται ανά λέξη σε ένα κείμενο πολύ μεγάλου μεγέθους.

$$H = \frac{1}{n} \cdot \log(P(w_i^j)) \qquad (2.61)$$

Αλλά πώς μπορούμε να υπολογίσουμε τις πιθανότητες των ακολουθιών των λέξεων w_i^j της γλώσσας , που χρειάζονται στη σχέση *(2.61)*; Μπορούμε να τις προσεγγίσουμε με τις πιθανότητες $\hat{P}(w_i^j)$ που δίνονται από το γλωσσικό μοντέλο. Αν αντικαταστήσουμε τις πραγματικές πιθανότητες της σχέσης *(2.61)* με τις προσεγγίσεις τους $\hat{P}(w_i^j)$ που δίνονται από το γλωσσικό μοντέλο, έχουμε το λογάριθμο της αβεβαιότητας *LPP* που είδαμε στο κεφάλαιο *(§2.4)*.

$$H = -\frac{1}{n} \log(\hat{P}(w_i^j)) \qquad (2.62)$$

Διαισθητικά το *LPP* είναι ένα μέτρο της εντροπίας του μοντέλου της γλώσσας. Μπορούμε να δείξουμε *[19 σελ.474]* ότι *LPP>=H,* αν θεωρήσουμε ότι η πηγή που παράγει το κείμενο έχει εργοδική συμπεριφορά. Αυτό είναι καθαρά διαισθητικό, επειδή το μοντέλο της γλώσσας μπορεί να είναι, στην καλύτερη περίπτωση, τόσο καλό όσο και η ίδια η γλώσσα. Από την πλευρά του αναγνωριστή φωνής η ποσότητα *LPP* μετρά την δυσκολία στην αναγνώριση λόγου, που προέρχεται από την ίδια πηγή που παρήγαγε και το κείμενο. Άρα, η ποσότητα *LPP* είναι ένα πολύ κατάλληλο μέτρο της για την ποιότητα ενός γλωσσικού μοντέλου.

Όμοια με την εξίσωση *(2.59)*, η δυσκολία σε μια εφαρμογή αναγνώρισης φωνής δίνεται, επίσης, από την αβεβαιότητα (perplexity):

$$PP = 2^{LPP} = \hat{P}(w_i^j) \qquad (2.63)$$

Συμπερασματικά, μια εφαρμογή αναγνώρισης φωνής με ένα γλωσσικό μοντέλο, που έχει πιθανότητα λογάριθμου *LPP,* μπορεί να θεωρηθεί ότι είναι τόσο δύσκολη όσο η αναγνώριση μιας γλώσσας με *PP* ισοπίθανες λέξεις.

2.4.2 Αξιολόγηση του μέτρου της αβεβαιότητας (*perplexity*)

Η θεμελιώδης μετρική για την απόδοση ενός συστήματος αναγνώρισης φωνής είναι η ακρίβεια της αναγνώρισης. Γιατί τότε μας ενδιαφέρει η ξεχωριστή μέτρηση της ποιότητας του γλωσσικού μοντέλου; Πρώτον, επειδή μας επιτρέπει να μετρήσουμε την ποιότητα ενός συστατικού του αναγνωριστή, του γλωσσικού μοντέλου, ανεξάρτητα από τα χαρακτηριστικά των άλλων συστατικών του υπό εξεταζόμενου συστήματος αναγνώρισης. Με τον τρόπο αυτό είναι δυνατό να συγκριθούν γλωσσικά μοντέλα που χρησιμοποιούνται σε διαφορετικά συστήματα αναγνώρισης, και ταυτόχρονα δίνεται στους ερευνητές η δυνατότητα να δουλέψουν ξεχωριστά πάνω στα δύο υποσυστήματα, ακολουθώντας έτσι την τακτική του «διαίρει και βασίλευε». Κατά δεύτερον, μπορούμε να μετρήσουμε την ποιότητα γλωσσικών μοντέλων που έχουν κατασκευαστεί για διαφορετικούς σκοπούς, π.χ. για *αποσαφήνιση (disambiguation) λέξεων* και *διόρθωση ορθογραφίας* ή *κωδικοποίηση κειμένου*.

Τι περιμένουμε από μια μετρική της ποιότητας ενός γλωσσικού μοντέλου, ειδικά σε συνάρτηση με την αναγνώριση φωνής; Ας υποθέσουμε ότι έχουμε δύο γλωσσικά μοντέλα *LM1* και *LM2*, και σύμφωνα με το μέτρο απόδοσης που χρησιμοποιούμε, το *LM1* είναι καλύτερο από το *LM2*. Περιμένουμε ότι, στη γενική περίπτωση, η ακρίβεια της αναγνώρισης ενός αναγνωριστή φωνής που χρησιμοποιεί το *LM1* θα μειωθεί αν στη θέση του *LM1* χρησιμοποιήσουμε το *LM2*. Με άλλα λόγια, η μετρική της ποιότητας του γλωσσικού μοντέλου θα έπρεπε να συσχετίζεται σε μεγάλο βαθμό με την ακρίβεια οποιουδήποτε συστήματος αναγνώρισης φωνής.

Η μετρική του *perplexity*, όπως φάνηκε στις προηγούμενες παραγράφους, έδειξε αρκετές φορές να συσχετίζεται ικανοποιητικά με την ακρίβεια της αναγνώρισης. Επιπλέον, είναι μία θεωρητικά έγκυρη μετρική σε ό,τι αφορά τα περιθώρια της επιλογής σε ένα κείμενο που έχει παραχθεί από γλωσσικό μοντέλο. Είναι έτσι μια πολύ κατάλληλη μετρική για την αξιολόγηση γλωσσικών μοντέλων. Παρ' όλα αυτά, έχει επίσης και μερικά μειονεκτήματα :

- Η μετρική του perplexity δεν παίρνει υπόψη της την ακουστική ομοιότητα που πιθανόν να έχουν δύο λέξεις. Έτσι, δεν υπάρχει τέλεια συσχέτιση μεταξύ του *perplexity* και της ακρίβειας της αναγνώρισης. Υπάρχουν παραδείγματα στη βιβλιογραφία *[20]*, όπου ένα γλωσσικό μοντέλο *LM1* με μεγαλύτερο *perplexity* από ένα άλλο μοντέλο *LM2* οδηγεί σε μεγαλύτερη ακρίβεια αναγνώρισης.

- Ηεπίδοση του γλωσσικού μοντέλου εξαρτάται από το κείμενο δοκιμής *(test set)*. Αν επιλέξουμε ένα *test set* που είναι πολύ διαφορετικό από το κείμενο που χρησιμοποιήθηκε για την εκπαίδευση *(training set)*, το μοντέλο θα αποδίδει πολύ φτωχά. Ωστόσο, αυτό δε σημαίνει ότι στην πραγματικότητα το μοντέλο είναι κακό, αλλά ότι το *test set* είναι πολύ διαφορετικό από το *training set*. Στην πραγματικότητα, το γλωσσικό μοντέλο μπορεί να έχει μάθει πολύ καλά τις στατιστικές ιδιότητες του *training set*.

- Το γλωσσικό μοντέλο, στην τελική περίπτωση, θα χρησιμοποιηθεί για τη διάκριση των λέξεων ανάμεσα σε πιθανές και μη πιθανές. Φαίνεται ότι γι' αυτό το σκοπό είναι σημαντικότερη η *διαφορά* ανάμεσα στις πιθανότητες πιθανών και μη πιθανών λέξεων, παρά η απόλυτη *τιμή* αυτών των πιθανοτήτων. Γενικά, η «αρνητική» πληροφορία είναι χρήσιμη σε ζητήματα μάθησης γλωσσών *[21][22]*. Κατά συνέπεια, φαίνεται κατάλληλο να χρησιμοποιηθεί ένα «ψεύτικο» κείμενο σαν *test set* σε ένα γλωσσικό μοντέλο (π.χ. μια τυχαία ακολουθία λέξεων επιλεγμένων από το λεξικό ή η αντιμετάθεση των λέξεων ενός κειμένου). Στη συνέχεια, θα μπορούσαμε για παράδειγμα να

μετρήσουμε τη διαφορά στην τιμή του *perplexity* ανάμεσα στα δύο κείμενα.

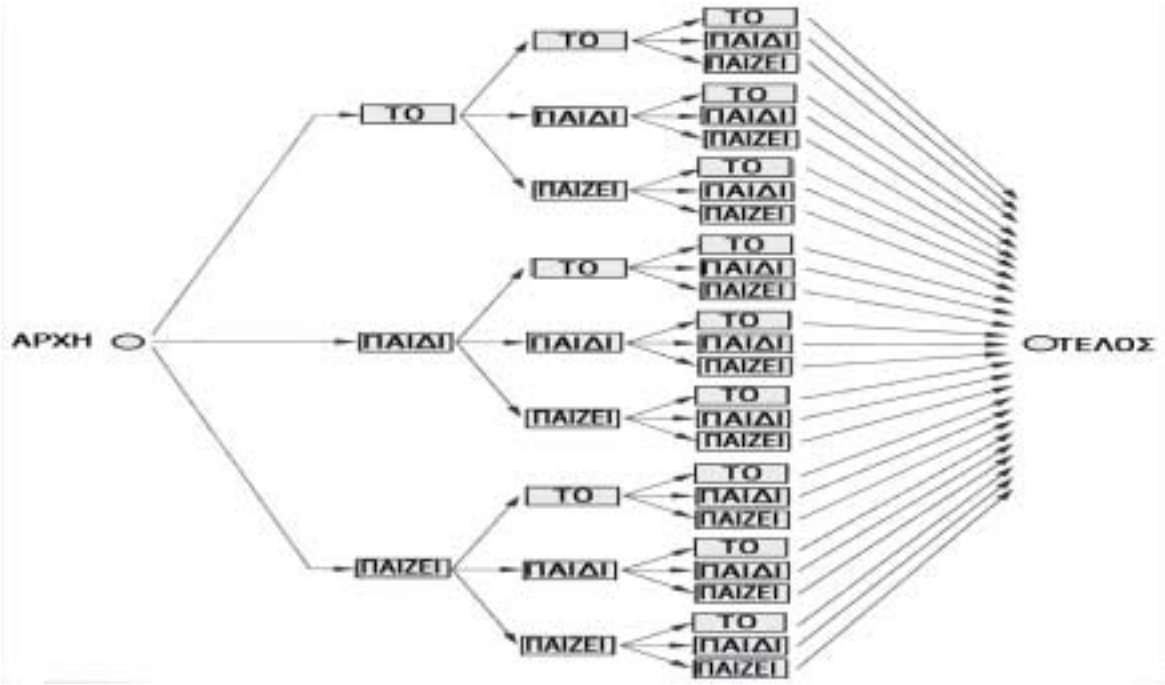
Αν και υπάρχουν προβλήματα με τη μετρική του *perplexity*, επιλέγουμε να το χρησιμοποιήσουμε στη δουλειά μας για τους ακόλουθους λόγους. Πρώτον, το πρώτο πρόβλημα που περιγράφηκε είναι πολύ σπάνιο και στη μεγάλη πλειοψηφία των περιπτώσεων η συσχέτιση ανάμεσα στο *perplexity* και στην ακρίβεια της αναγνώρισης είναι πολύ μεγάλη. Κατά δεύτερον, το δεύτερο πρόβλημα που περιγράφηκε είναι λιγότερο σοβαρό αν επιλέξουμε ως *test set* ένα κείμενο που να είναι αρκετά όμοιο με το κείμενο εκπαίδευσης, και σχετικό με το σκοπό για τον οποίο σχεδιάζεται ο αναγνωριστής. Τρίτον, η ποσότητα της εργασίας που απαιτείται για τη διερεύνηση του τρίτου προβλήματος είναι δυσανάλογη με τους σκοπούς της εργασίας μας. Τέλος, είναι σημαντικό να σημειώσουμε ότι το *perplexity* είναι η μόνη ευρέως αποδεκτή μετρική για την ποιότητα ενός γλωσσικού μοντέλου.

2.5 ***Word-error-rate***

Όπως ειπώθηκε και στην προηγούμενη παράγραφο, το *perplexity* είναι μία δημοφιλής μετρική της απόδοσης ενός γλωσσικού μοντέλου λόγω των μειωμένων υπολογιστικών απαιτήσεων σε σύγκριση με ένα πλήρες πείραμα αναγνώρισης. Ωστόσο, η μείωση στο *perplexity* δεν εγγυάται και μείωση του ποσοστού λάθους αναγνώρισης *(word-error-rate)*, το οποίο παραμένει το απόλυτο μέτρο της ποιότητας του γλωσσικού μοντέλου. Παρακάτω παρουσιάζουμε κάποιες μεθόδους με τις οποίες μπορούμε να ενσωματώσουμε τα γλωσσικά μοντέλα στην διαδικασία αναζήτησης *(search)* προκειμένου να γίνει η αναγνώριση.

Θεωρητικά ένας αναγνωριστής συνεχούς ομιλίας θα έπρεπε να αναζητεί στο πεδίο όλων των δυνατών αλληλουχιών μεταξύ λέξεων, τον πιο πιθανό συνδυασμό σε σχέση με τα ακουστικά δεδομένα. Ωστόσο, ακόμα και για μικρά λεξικά, αυτή η εξαντλητική *(exhaustive)* προσέγγιση είναι ανέφικτη, και έτσι είναι απαραίτητο να χρησιμοποιηθούν πιο εκλεπτυσμένες μέθοδοι. Είναι σημαντικό, από πρακτικής σκοπιάς, να καταλάβουμε τον μηχανισμό της διαδικασίας αποκωδικοποίησης, καθώς η φύση του γλωσσικού μοντέλου μπορεί να επηρεάσει σημαντικά την πολυπλοκότητα του προβλήματος αναζήτησης. Συγκεκριμένα, είναι δυνατό να χρησιμοποιήσουμε διάφορους τρόπους εφαρμογής των γλωσσικών μοντέλων, μερικοί από τους οποίους παρουσιάζονται στη συνέχεια. Ας θεωρήσουμε, σαν παράδειγμα, έναν αναγνωριστή που πρέπει να αναγνωρίσει οποιαδήποτε πρόταση που αποτελείται από τρεις λέξεις και περιέχει τις λέξεις «το», «παιδί», «παίζει». Το σύνολο των πιθανών προτάσεων μπορεί να παρασταθεί σαν δομή δέντρου, όπως απεικονίζεται στο σχήμα *(2.2)*

Σχήμα 2.2



Στη συνεχή ομιλία, οι στιγμές στις οποίες γίνονται οι μεταβάσεις μεταξύ των λέξεων δεν είναι γνωστές. Καθώς αυτές επηρεάζουν τις πιθανότητες που παίρνουμε από τα ακουστικά μοντέλα, προσδιορίζοντας ποια τμήματα από την ακολουθία παρατήρησης αναθέτονται σε ποιο μοντέλο, κάθε πιθανή επιλογή ορίων μεταξύ

λέξεων πρέπει να θεωρείται ως ξεχωριστή υπόθεση από τον αναγνωριστή. Έτσι, ο συνολικός αριθμός των διαφορετικών υποθέσεων που λαμβάνονται υπόψη στην αναζήτηση είναι πολύ μεγαλύτερος από τον αριθμό των προτάσεων που φαίνονται στο δένδρο του σχήματος *(2.2)*. Κατά συνέπεια δεν μας κάνει εντύπωση ότι, ακόμα και για μικρά λεξικά, το πεδίο αναζήτησης είναι στην πράξη πολύ ευρύ για να αντιμετωπιστεί με *exhaustive* αλγόριθμο. Το μέγεθος του πεδίου αναζήτησης περιορίζεται από πρακτικούς περιορισμούς, όπως ο χρόνος επεξεργασίας και ο διαθέσιμος χώρος αποθήκευσης, γι' αυτό οι δύο τεχνικές που ακολουθούν είναι χρήσιμες στο να γίνει εφαρμόσιμη η αναγνώριση φωνής.

1. Αποκλεισμός μονοπατιών *(path-pruning)* : Καθώς η αναζήτηση προχωράει, κάποιες υποθέσεις γίνονται πολύ απίθανες, και μπορούν να απορριφθούν, έτσι ώστε να εξοικονομηθεί τόσο χώρος, όσο και υπολογιστική ισχύς. Επειδή, όμως, δεν είναι δυνατό να μας εγγυηθεί ότι κάποιο από αυτά τα μονοπάτια δεν δίνει πράγματι το σωστό αποτέλεσμα (μπορεί αργότερα το μονοπάτι να γίνει πιο πιθανό), η μέθοδος *path-pruning* μπορεί να οδηγήσει σε λάθη αναζήτησης *(search errors)*.

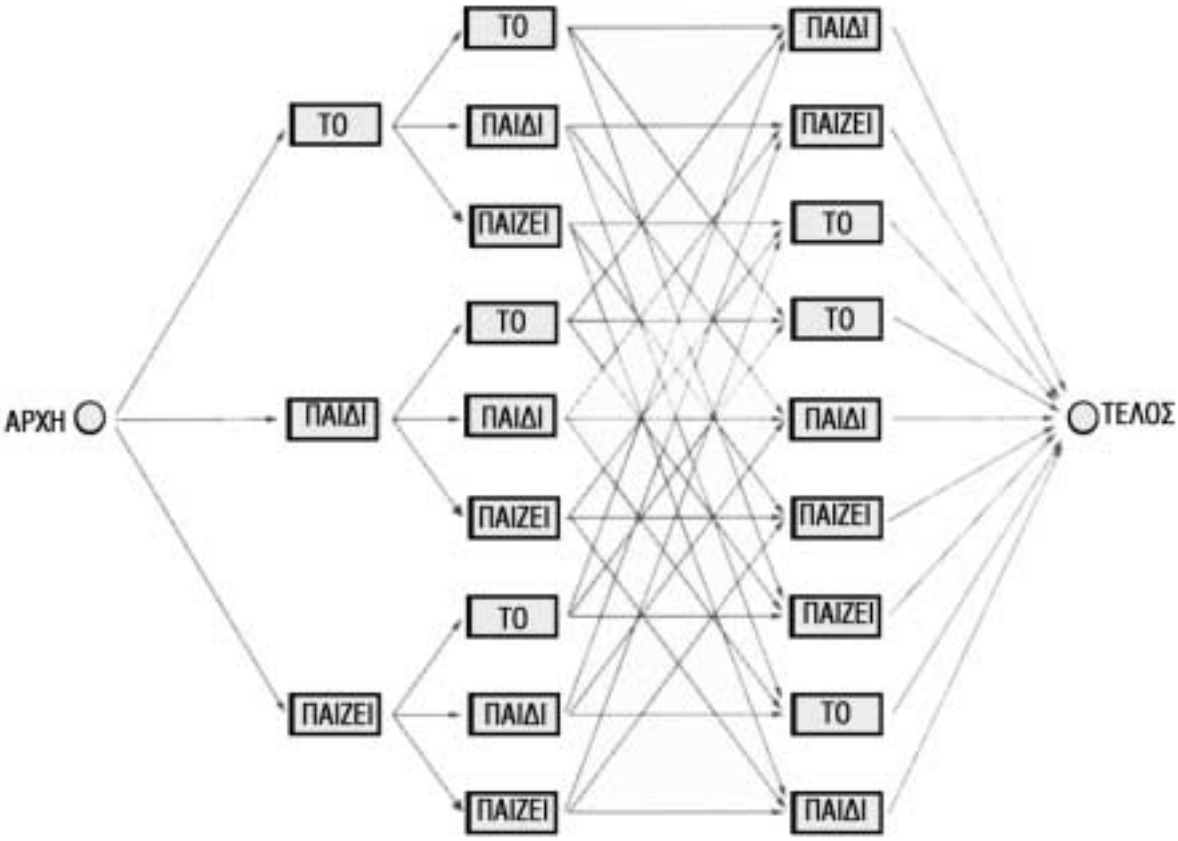
2. Συγχώνευση μονοπατιών *(path merging)*: Όταν δύο ή περισσότερα μονοπάτια συναντώνται, μπορεί να είναι πιθανό να συγχωνευθούν και να θεωρηθεί ότι έχουν μόνο μία κοινή συνέχεια. Ωστόσο, το αν αυτό είναι επιτρεπτό, εξαρτάται από το κατά πόσο η έκταση, στην οποία μεγαλώνει ο αριθμός των υπολογισμών για την επέκταση κάθε μονοπατιού, εξαρτάται από το ιστορικό. Συγκεκριμένα, μονοπάτια μπορούν να συγχωνευθούν μόνο όταν τα ιστορικά τους θεωρηθούν ισοδύναμα από υπολογιστικής άποψης. Καθώς τα γλωσσικά μοντέλα χρησιμοποιούν περισσότερο τα συμφραζόμενα από όσο τα ακουστικά μοντέλα, συχνά υπαγορεύουν τα σημεία στα οποία είναι δυνατό να υπάρξει συγχώνευση. Οι συγχωνεύσεις, μάλιστα, μπορούν να συμβούν όταν προκύψει ισοδυναμία από το γεγονός ότι δύο λέξεις κατατάσσονται στην ίδια κλάση *[23]*. Ας θεωρήσουμε ξανά το παράδειγμα του σχήματος *(2.2)*. Οι πιθανότητες του γλωσσικού μοντέλου υπολογίζονται στα όρια

μεταξύ των λέξεων, έτσι αν χρησιμοποιήσουμε *bigrams*, το πεδίο αναζήτησης που καταδεικνύεται στο σχήμα *(2.2)* μειώνεται σε αυτό που φαίνεται στο σχήμα *(2.3)*.

Σχήμα 2.3

Αν χρησιμοποιήσουμε *trigram*, ωστόσο, ο υπολογισμός της πιθανότητας του γλωσσικού μοντέλου βασίζεται στο πιο πρόσφατο ζεύγος λέξεων, και έτσι τα μονοπάτια μπορεί να συγχωνευθούν μόνο όταν καταλήγουν στην ίδια λέξη, όπως καταδεικνύεται στο σχήμα *(2.4)*. Η συγχώνευση μονοπατιών επιτρέπει πολύ ουσιώδεις μειώσεις στον αριθμό των διαφορετικών μονοπατιών που λαμβάνονται υπόψη σε κάθε στιγμή.

Για τα *trigram* και τα μεγαλύτερης τάξης γλωσσικά μοντέλα, ο αριθμός των μονοπατιών μπορεί να γίνει μη αντιμετωπίσιμα μεγάλος, ακόμα και με τη χρησιμοποίηση του αποδοτικού *path merging*. Για τον περιορισμό του εύρους αναζήτησης θα απαιτούνταν μεγάλης κλίμακας *pruning*, το οποίο μπορεί να αύξανε τον αριθμό των *search errors* σε μη επιτρεπτό βαθμό. Σε τέτοιες περιπτώσεις μπορούμε να υιοθετήσουμε τον ακόλουθο αλγόριθμο «διπλής προσπέλασης» *(two-pass)* :



Σχήμα 2.4

1. Εφαρμόζουμε ένα πιο απλό (π.χ. *bigram*) γλωσσικό μοντέλο κατά τη διάρκεια της αναζήτησης στην αναγνώριση, και εξάγουμε ένα υποσύνολο του πεδίου αναζήτησης, που περιέχει κάποια από τα πιο πιθανά μονοπάτια. Τα μονοπάτια αυτά ονομάζονται «ενδιάμεσες υποθέσεις» *(intermediate hypotheses)*.

2. Επεξεργαζόμαστε εκ των υστέρων *(rescore)* τις ενδιάμεσες υποθέσεις (*intermediate hypotheses)* εφαρμόζοντας το πιο πολύπλοκο γλωσσικό μοντέλο, και διαλέγοντας το πιο πιθανό μονοπάτι ως το τελικό αποτέλσμα αναγνώρισης.

Εφόσον το αποτέλεσμα που παίρνουμε μετά την αναζήτηση χρησιμοποιώντας το πιο πολύπλοκο γλωσσικό μοντέλο περιλαμβάνεται στις ενδιάμεσες υποθέσεις, το τελικό αποτέλεσμα θα παραμείνει ανεπηρέαστο από τη διαίρεση της διαδικασίας

αναγνώρισης σε δύο στάδια. Ωστόσο, αν η πρώτη προσπέλαση εξαλείψει αυτή την υπόθεση, προκύπτει λάθος αναζήτησης που η δεύτερη προσπέλαση δεν μπορεί να επανορθώσει. Η ορθότητα *(accuracy)* των ενδιάμεσων υποθέσεων επηρεάζεται από την ακρίβεια της πρώτης προσπέλασης, αλλά και από τον αριθμό των εναλλακτικών μονοπατιών στις ενδιάμεσες υποθέσεις. Η ορθότητα είναι ένα σημαντικό θέμα όταν προσεγγίζουμε το πρόβλημα της αναζήτησης με τη μέθοδο *two-pass*.

Στη συνέχεια περιγράφονται δύο τύποι *intermediate hypotheses*, οι «λίστες ν-καλύτερων» *(n-best lists)* και τα «πλέγματα» *(lattices)*.

2.5.1 ***N-best rescoring***

Η μέθοδος *N-best [24]* καθορίζει, κατά τη διάρκεια της αναζήτησης, μία λίστα που περιέχει τις *N* πιο πιθανές υποθέσεις, καθώς και τις πιθανότητες του ακουστικού και του γλωσσικού μοντέλου για κάθε λέξη. Το νέο γλωσσικό μοντέλο χρησιμοποιείται, είτε για να αντικαταστήσει, είτε για να τροποποιήσει τις υπάρχουσες πιθανότητες για κάθε υπόθεση. Στη συνέχεια η συνολική πιθανότητα υπολογίζεται εκ νέου. Το αποτέλεσμα της αναγνώρισης είναι η υπόθεση που καταλαμβάνει την υψηλότερη θέση στη νέα λίστα.

2.5.2 ***Lattice rescoring***

Αντί για λίστα, η αναζήτηση μπορεί να εξάγει ένα δίκτυο που περιέχει τα πιο πιθανά μονοπάτια στο πεδίο της αναζήτησης. Ένα πλέγμα *(lattice)* είναι ένας κατευθυνόμενος ακυκλικός γράφος στον οποίο οι κόμβοι αντιστοιχούν στα όρια των λέξεων στο χρόνο, και οι σύνδεσμοι ανάμεσα στους κόμβους αντιστοιχούν σε συγκεκριμένες υποθέσεις λέξεων. Οι πιθανότητες των ακουστικών και των γλωσσικών μοντέλων για κάθε λέξη αποθηκεύονται μέσα στο *lattice*, και κάθε μονοπάτι από τον αρχικό έως τον τελικό κόμβο αναπαριστά μία διακριτή υπόθεση. Το πιο πιθανό μονοπάτι του *lattice* μπορεί να βρεθεί χρησιμοποιώντας έναν κατάλληλο αλγόριθμο αναζήτησης *[25]*. Το *lattice* είναι μία πολύ πιο συμπαγής

αναπαράσταση ενός συνόλου εναλλακτικών μονοπατιών, από όσο η *N-best* λίστα, και σε κάποιες περιπτώσεις η αναζήτηση στο *lattice* μπορεί να γίνει πολύ αποδοτικά. Ωστόσο, στην περίπτωση του *lattice* προαπαιτείται να είναι δυνατός ο ανασυνδυασμός των σημαντικών μονοπατιών, αλλιώς ο αριθμός των διαφορετικών μονοπατιών που διατηρούνται κατά τη διάρκεια της αναζήτησης του *lattice* μπορεί να γίνει μη αντιμετωπίσιμος. Από την άλλη μεριά, για τις *N-best* λίστες οι απαιτήσεις σε χώρο, αλλά και σε υπολογιστική ισχύ, ορίζονται με ακρίβεια *a-priori*. Παρατηρούμε ότι οι *N-best* λίστες μπορούν να παραχθούν από *lattices*.

Κεφάλαιο 3

Αλγόριθμος Stemming για τα ελληνικά

3.1. Εισαγωγή

Όπως είδαμε στην παράγραφο *(§2.3)* είναι δυνατό να κατασκευάσουμε ένα *class-based* γλωσσικό μοντέλο, όπου κάθε κλάση λέξεων θα περιέχει λέξεις με την ίδια ρίζα *(stem)*, με σκοπό την μείωση των παραμέτρων του γλωσσικού μοντέλου και την κατασκευή, έτσι, ενός συμπαγούς και με καλύτερη εκτίμηση των χαρακτηριστικών της γλώσσας, γλωσσικού μοντέλου. Για την υλοποίηση αυτής της ταξινόμησης είναι απαραίτητο να χρησιμοποιηθούν τεχνικές αφαίρεσης κατάληξης *(stemming)*. Ένας αλγόριθμος *stemming* αναλαμβάνει να αφαιρέσει την κατάληξη μιας λέξης, αφήνοντας μόνο τη ρίζα της. Για παράδειγμα, οι λέξεις :

μπορώ
μπορώντας
μπορούσαν
μπορεί

θα περιορίζονταν στην ίδια ρίζα «μπορ-». Εφαρμόζοντας έναν αλγόριθμο *stemming* στο λεξικό που προκύπτει από τα δεδομένα εκπαίδευσης μπορούμε να πάρουμε την αντιστοιχία κάθε λέξης με τη ρίζα της. Στη συνέχεια, είναι εύκολο να ταξινομήσουμε τις λέξεις στις αντίστοιχες κλάσεις και να κατασκευάσουμε ένα γλωσσικό μοντέλο σύμφωνα που περιγράφεται από τη σχέση *(2.47)* :

$$P\big(w(i)\,|\,w(0,i-1)\big) \approx \sum_{\forall c: c \in C(w(i))} P\big(w(i)\,|\,C(w(i))\big)\cdot P\big(C(w(i))\,|\,H(w(i))\big) \qquad (3.1)$$

όπου $C(\cdot)$ η συνάρτηση που αντιστοιχίζει την κάθε λέξη στη ρίζα της, c οι κλάσεις των ριζών και $H(w_i)$ το ιστορικό κάθε λέξης.

Πολλοί αλγόριθμοι *stemming* έχουν αναπτυχθεί για διάφορες γλώσσες, κυρίως για τα αγγλικά, και αναφέρονται στη βιβλιογραφία *[26],[27],[28],[29],[30],[31]*. Οι αλγόριθμοι αυτοί ήταν αρχικά σχεδιασμένοι για εφαρμογή στο πεδίο της *ανάκτησης πληροφοριών (information retrieval)*. Είναι δυνατόν όμως, όπως περιγράψαμε πιο πάνω, να εφαρμοστεί ένας τέτοιος αλγόριθμος και στην ανάπτυξη *class-based* γλωσσικών μοντέλων. Ο περισσότερο απλός, αποδοτικός και πολυχρησιμοποιημένος είναι ο αλγόριθμος του *Porter [32]*. Η βασική προσέγγιση στο πρόβλημα του *stemming*, κατά τον αλγόριθμο του *Porter*, είναι ότι δεν εμπλέκει γλωσσολογικά θέματα *(linguistics)* στη διαδικασία του *stemming*. Πιο συγκεκριμένα, δεν είναι απαραίτητο η ρίζα της λέξης που θα υποστεί *stemming* να είναι και η γραμματικά σωστή ρίζα της. Αρκεί η ρίζα που θα μείνει να είναι τέτοια που να γίνεται καλύτερη ομαδοποίηση των λέξεων σε κλάσεις, με σκοπό την βελτίωση της απόδοσης του *information retrieval* συστήματος. Για την απομόνωση της ρίζας χρησιμοποιείται μία λίστα καταλήξεων που είναι πιθανό να αφαιρεθούν. Αντίστοιχα, στην περίπτωση της ταξινόμησης σε κλάσεις των λέξεων, με σκοπό την κατασκευή *class-based* γλωσσικού μοντέλου, μία τέτοια προσέγγιση είναι πολύ βολική. Δεχόμαστε αυτή την προσέγγιση, καθώς το όνομα μιας κλάσης λέξεων δεν μας ενδιαφέρει· μας αρκεί φυσικά να είναι διαφορετικό για κάθε κλάση.

Στα πλαίσια της εργασίας μας πάνω στα *class-based* γλωσσικά μοντέλα, αναπτύξαμε έναν απλό αλγόριθμο *stemming* για τα ελληνικά, βασισμένο στις βασικές αρχές του αλγορίθμου του *Porter*. Η μορφολογία της ελληνικής γλώσσας είναι τέτοια που επιτρέπει την εμφάνιση των λέξεων με πολλές παρόμοιες μορφές *(πολυμορφισμός)*. Το γεγονός ότι, αντίθετα π.χ. με την αγγλική γλώσσα, στην ελληνική υπάρχουν ξεχωριστές καταλήξεις για κάθε πρόσωπο, αριθμό, γένος και κλίση, παράγει αμέσως πολλαπλάσιο σε μέγεθος λεξικό. Ταυτόχρονα, όμως, γίνεται φανερό ότι μία ταξινόμηση των λέξεων σε κλάσεις ίδιων ριζών θα μπορούσε να εξάγει μία μεγαλύτερη ομαδοποίηση των λέξεων σε σχέση με τα αγγλικά, οδηγώντας έτσι σε μικρότερο, συγκριτικά, αριθμό κλάσεων.

Στις επόμενες παραγράφους περιγράφεται ο αλγόριθμος που αναπτύχθηκε για την υλοποίηση του *stemming* για τα ελληνικά, καθώς και τα αποτελέσματα που αυτός είχε στη μείωση του λεξικού.

3.2 ***Stemmer* για τα ελληνικά**

Για την παρουσίαση του αλγόριθμου *stemming* που υλοποιήσαμε είναι απαραίτητο να δώσουμε κάποιους ορισμούς:

Ένα *σύμφωνο* μέσα σε μια τυπωμένη λέξη είναι οποιοδήποτε γράμμα εκτός από τα «α», «ε», «η», «ι», «υ», «ο», «ω», «ά», «έ», «ή», «ί», «ϊ», «ΰ», «ΐ», «ύ», «ϋ», «ό», «ώ». Αν ένα γράμμα δεν είναι σύμφωνο, είναι *φωνήεν*.

Ένα σύμφωνο συμβολίζεται με *c*, ενώ ένα φωνήεν με *v*. Μία σειρά συμφώνων *ccc…* με μήκος μεγαλύτερο του μηδέν συμβολίζεται με *C*. Αντίστοιχα μία σειρά φωνηέντων *vvv…* με μήκος μεγαλύτερο του μηδέν συμβολίζεται με *V*. Κατά συνέπεια, οποιαδήποτε λέξη ή μέρος λέξης μπορεί να εκφραστεί με έναν από τους τέσσερις ακόλουθους τρόπους :

$$\begin{aligned} &CVCV \ldots C \\ &CVCV \ldots V \\ &VCVC \ldots C \\ &VCVC \ldots V \end{aligned} \qquad (3.2)$$

Το σύνολο των παραπάνω παραστάσεων μπορεί να εκφραστεί σε μία μόνο παράσταση, ως εξής :

$$[C]VCVC \ldots [V] \qquad (3.3)$$

όπου ο συμβολισμός *[·]* αναπαριστά την προαιρετική παρουσία του περιεχομένου των άγκιστρων.

Χρησιμοποιώντας το συμβολισμό *(VC){m}* για να παραστήσουμε ότι το πρότυπο *VC* επαναλαμβάνεται *m* φορές, η παράσταση *(3.3)* μπορεί να ξαναγραφεί ως εξής :

[C] (VC) {m} [V] *(3.4)*

Ο αριθμός *m* ορίζεται ως το *μέτρο* οποιασδήποτε λέξης ή μέρους λέξης όταν αυτή αναπαρίσταται στην παραπάνω μορφή. Η περίπτωση *m=0* καλύπτει τη μηδενική *(null)* λέξη. Ακολουθούν μερικά παραδείγματα λέξεων ή μερών λέξεων:

m=0	*μπ,*	*ώ,*	*μπώ,*	*ό,* *το*
m=1	*μπαίνω,*		*τόσο,*	*όσο,* *σπάζω*
m=2	*μπαίνουμε,*		*ωστόσο,*	*κόστος*

Οι *κανόνες* για να αφαιρεθεί μία κατάληξη δίνονται στη μορφή :

(υπόθεση) S1 → S2 *(3.5)*

Η παραπάνω σχέση σημαίνει ότι αν μία λέξη τελειώνει με την κατάληξη *S1*, και ικανοποιεί τη δεδομένη υπόθεση, η κατάληξη *S1* αντικαθίσταται με την κατάληξη *S2*. Η υπόθεση, όπως αυτή υλοποιήθηκε για τα ελληνικά, αφορά την τιμή που μπορεί να πάρει το μέτρο *m*, π.χ. :

(m > 2) μένος → NULL *(3.6)*

Στο παραπάνω παράδειγμα το *S1* είναι η κατάληξη *«-μένος»* και το *S2* η μηδενική λέξη. Σύμφωνα με τον παραπάνω κανόνα η λέξη *«χαμένος»* αντιστοιχίζεται στη ρίζα *«χα»*, καθώς για τη λέξη αυτή ισχύει *m=3*. Αντίθετα η λέξη *«μένος»* δεν επηρεάζεται από τον κανόνα αυτό, καθώς έχει μέτρο *m=2*.

Σύμφωνα με τον παραπάνω τρόπο, είναι δυνατόν τώρα να γράψουμε κανόνες που να αφαιρούν την κατάληξη μιας λέξης με τον τρόπο που επιθυμούμε. Οι άξονες πάνω στους οποίους κινηθήκαμε για την υλοποίηση των κανόνων του *stemming* για

τα ελληνικά ήταν δύο. Πρώτον, να χρησιμοποιήσουμε την ελληνική γραμματική *[33]* για να πάρουμε τις πλέον συνήθεις καταλήξεις που πρέπει να αφαιρεθούν, και δεύτερον, να διατηρήσουμε ολόκληρες μερικές μικρές λέξεις (π.χ. *«το», «ο», «του»* κλπ.).

Όσον αφορά τον πρώτο άξονα, βρήκαμε από τη γραμματική 174 ξεχωριστές καταλήξεις τις οποίες χρησιμοποιήσαμε για την κατασκευή αντίστοιχου αριθμού κανόνων. Πρέπει να σημειωθεί ότι για τη σωστή λειτουργία των κανόνων θα πρέπει αυτοί να εκτελούνται με κάποια προτεραιότητα. Το κριτήριο για το ποιος κανόνας θα δοκιμαστεί πρώτος είναι το μήκος του *string S1*. Ο κανόνας με το μεγαλύτερο μήκος του *string S1* δοκιμάζεται πρώτος *(Σχήμα 3.1)*. Για παράδειγμα, αν πρέπει να ελεγχθεί μία λέξη για το αν έχει κατάληξη *«-μένος»* ή *«-ος»*, τότε θα γίνει πρώτα ο έλεγχος για την κατάληξη *«-μένος»* και αν δεν γίνει *stemming* σε αυτή την περίπτωση, τότε μόνο θα ελεγχθεί και η περίπτωση της κατάληξης *«-ος»*. Έτσι, η λέξη *«χαμένος»* αντιστοιχίζεται στη ρίζα *«χα»,* ενώ η λέξη *«πόνος»* στη ρίζα *«πόν»*.

1. -μένος $\rightarrow$ *NULL*
2. -ος $\rightarrow$ *NULL*

Σχήμα 3.1

Το δεύτερο σημείο που μας ενδιαφέρει είναι η διατήρηση ολόκληρων των μικρών λέξεων, θεωρώντας ότι δεν έχουν κατάληξη. Επιλέγουμε μια τέτοια αντιμετώπιση, καθώς μία ομαδοποίηση στην ίδια ρίζα τέτοιων μικρών και συνηθισμένων λέξεων (π.χ. *«το»*, *«του»*, *«τα»*) θα είχε αρνητική επίπτωση στο γλωσσικό μοντέλο, αφού πλέον δε θα περιγραφόταν με σαφήνεια ένα μεγάλο κομμάτι του. Για να ικανοποιήσουμε αυτή την απαίτηση, δεχόμαστε ότι οι μονοσύλλαβες λέξεις δεν πρέπει να υφίστανται *stemming*. Ταυτόχρονα, μας ενδιαφέρει να διασαφηνίσουμε το πώς γίνεται το *stemming* σε κάποιες λέξεις που αποτελούν καταλήξεις από μόνες τους (π.χ. *«μένους»*). Για κάθε υπό εξέταση λέξη, λοιπόν, χρησιμοποιούμε το μέτρο της *m*, το οποίο θεωρούμε ικανοποιητική αναπαράσταση του αριθμού των συλλαβών της, για να αποφασίσουμε αν θα πρέπει να υποστεί τον εκάστοτε κανόνα *stemming*. Συγκεκριμένα, σε κάθε κανόνα, απαιτούμε να υπάρχει

μία τουλάχιστον επιπλέον συλλαβή στη λέξη, εκτός από τον αριθμό των συλλαβών της κατάληξης, ώστε να εφαρμοστεί ο κανόνας. Έτσι, οι λέξεις *«χαμένους»*, *«μένους»*, *«τους»*, σύμφωνα με το σύνολο κανόνων του *σχήματος 3.2* , (που αποτελεί και υποσύνολο των κανόνων που τελικά εφαρμόσαμε) αντιστοιχίζονται στις ρίζες *«χα»*, *«μέν»*, *«τους»*, αντίστοιχα.

1. (m>2) «μένους» → NULL
2. (m>1) «ους» → NULL

Σχήμα 3.2

Στη συνέχεια δίνονται μερικά παραδείγματα *stemming* σε ελληνικές λέξεις :

```
χθεσινή χθεσιν
ημέρες ημέρ
γύρω γύρ
ακριβώς ακριβ
ανακοίνωση ανακοίνωσ
συμμετοχή συμμετοχ
εβδομάδα εβδομάδ
βουλής βουλ
αυτών αυτ
τομέα τομ
παιχνίδι παιχνίδ
αυτής αυτ
θεσσαλονίκη θεσσαλονίκ
εργασίας εργασί
πολλοί πολλ
παρουσία παρουσί
πρωθυπουργό πρωθυπουργ
αρχή αρχ
σειρά σειρ
σημίτη σημίτ
κόμματα κόμμ
άλλους άλλ
ελληνικό ελληνικ
δημιουργία δημιουργί
σημίτης σημίτ
γνωστό γνωστ
απέναντι απέναντ
στόχο στόχ
δε δε
φυσικά φυσικ
καμία καμί
οικονομική οικονομικ
πολιτισμού πολιτισμ
εποχή εποχ
νέου νέου
αθήνας αθήν
εικόνα εικόν
```

```
ηγεσία ηγεσί
σημαίνει σημαίν
υγείας υγεί
εξελίξεις εξελίξ
γίνουν γίν
εταιρείας εταιρεί
αθηνών αθην
```

Ο κώδικας του αλγόριθμου που περιγράφηκε παραπάνω γράφτηκε σε *C++* και παρατίθεται στο *Παράρτημα Γ*.

3.3 Αποτελέσματα και αξιολόγηση του αλγορίθμου *Stemming* για τα ελληνικά

Ο αλγόριθμος *stemming* που αναπτύξαμε εφαρμόστηκε σε λεξικά που προήλθαν από τα κείμενα εκπαίδευσης των γλωσσικών μοντέλων. Το σύνολο των διαφορετικών λέξεων των κειμένων αυτών ανέρχεται σε 348955 λέξεις. Για την κατασκευή γλωσσικών μοντέλων βασισμένων στα κείμενα αυτά χρησιμοποιήσαμε λεξικά των 100000, 80000 και 64000 πιο συχνών λέξεων. Η εφαρμογή του *stemming* σε αυτά τα λεξικά έδωσε τα αποτελέσματα που φαίνονται στον *Πίνακα 3.1*.

Μέγεθος λεξικού *(λέξεις)*	***Μέγεθος stemmed λεξικού*** *(λέξεις)*	***Μείωση λεξικού*** *(%)*
348955	*189719*	*45,63*
100000	*47855*	*52,15*
80000	*38587*	*51,77*
64000	*31195*	*51,26*

Πίνακας 3.1

Θεωρητικά, το κριτήριο για να θεωρηθεί ότι δύο λέξεις *W1* και *W2* ανήκουν στην ίδια κλάση, θα ήταν ο τρόπος που χρησιμοποιούνται σε μία πρόταση να είναι παρόμοιος και το νόημα να είναι παραπλήσιο. Για παράδειγμα, θα ήταν πολύ λογικό για τις λέξεις *W1=«έρθει»* και *W2=«έρθετε»* στην πρόταση *«αν δεν έρθει/έρθετε θα φύγουμε»* να συγχωνευθούν στην ίδια ρίζα. Από την άλλη μεριά, αν *W1=«πιάνο»* και *W2=«πιάνομαι»*, τότε σε μια πρόταση που αρχίζει με τη φράση *«παίζω μπάλα και πιάνο/πιάνομαι ...»* είναι, σύμφωνα με τη λογική, λάθος να συγχωνευθούν αυτές οι δύο λέξεις. (Θα έπρεπε, ίσως, να αναφερθεί ότι σύμφωνα με το δικό μας αλγόριθμο οι δύο αυτές λέξεις συγχωνεύονται.) Ανάμεσα σε αυτές τις δύο ακραίες περιπτώσεις υπάρχει μία σειρά διαφορετικών περιπτώσεων, και δεδομένων δύο λέξεων *W1* και *W2* υπάρχει οπωσδήποτε μία διχογνωμία για το αν θα έπρεπε να συγχωνευθούν. Γι' αυτό και η αποτίμηση του αποτελέσματος του *stemming* είναι αντίστοιχα δύσκολη.

Στη διάρκεια της ανάπτυξης των κανόνων του αλγορίθμου φθάνουμε σε ένα σημείο όπου η πρόσθεση επιπλέον κανόνων για να καλύψουμε κάποιες εξεζητημένες περιπτώσεις καταλήξεων *(π.χ. «ψάρ-εμα»)*, θα είχε ως αποτέλεσμα την κακή απόδοση του αλγορίθμου σε άλλες περιπτώσεις *(π.χ. «ήρ-εμα»)*. Έτσι, χωρίς να δώσουμε εγκαίρως ιδιαίτερη προσοχή στις καταλήξεις που αποφασίζουμε να αφαιρέσουμε, είναι εύκολο να κάνουμε τον αλγόριθμο περισσότερο πολύπλοκο από όσο χρειάζεται. Η προσπάθειά μας ήταν, λοιπόν, να κρατήσουμε απλό και ταυτόχρονα αποδοτικό τον αλγόριθμό μας.

Το κριτήριο που εξετάζεται στην *υπόθεση* ενός κανόνα *stemming* είναι το *μέτρο m* της υπό εξέταση λέξης. Θωρήσαμε πως το μέτρο αυτό είναι μία καλή αναπαράσταση του αριθμού των συλλαβών της λέξης, αν και από γλωσσολογική άποψη δεν στέκει μία τέτοια προσέγγιση. Παρ' όλα αυτά, παρατηρήσαμε ότι το μέγεθος αυτό μπορούσε να χρησιμοποιηθεί πολύ αποδοτικά για να αποφασίσουμε αν θα ήταν σωστό να αφαιρεθεί ή όχι μία κατάληξη, και έτσι προχωρήσαμε στη χρησιμοποίησή του.

Ο αλγόριθμος για τα ελληνικά που αναπτύξαμε είναι μικρός, απλός και, σύμφωνα με τα αποτελέσματα που έδωσε, αποδοτικός. Είναι εύκολο να εφαρμοστεί σε οποιοδήποτε ελληνικό λεξικό δίνοντας ως έξοδο την αντιστοιχία κάθε λέξης με τη

ρίζα της. Παρατηρούμε ότι η συγχώνευση των λέξεων των εκάστοτε λεξικών είναι τόσο μεγάλη, ώστε το λεξικό που παίρνουμε να έχει μέγεθος μικρότερο και από το μισό του αρχικού. Σε σύγκριση με τα αποτελέσματα του *stemming* για τα αγγλικά *[32]*, όπου έχουμε μείωση του λεξικού κατά το 1/3 του αρχικού, μπορούμε να ισχυριστούμε ότι ο αλγόριθμος *stemming* για τα ελληνικά μπορεί με καλύτερο τρόπο να αποδώσει θετικά, τόσο όταν εφαρμόζεται για κατασκευή γλωσσικών μοντέλων, όσο και όταν εφαρμόζεται ως τεχνική στο πεδίο του *information retrieval.*

Κεφάλαιο 4

N-pos μοντέλα

Το σημαντικότερο πρόβλημα των *N-gram* μοντέλων είναι το μέγεθος των δεδομένων που απαιτούνται για την εκπαίδευση του μοντέλου. Επιπλέον, κάποιος μπορεί να υποστηρίξει ότι μερικοί από τους «τοπικούς» περιορισμούς σε μια ακολουθία λέξεων εξαρτώνται λιγότερο από την ταυτότητα των προηγούμενων λέξεων και περισσότερο από τις γραμματικές τους ιδιότητες. Αυτό οδηγεί στην ιδέα της ομαδοποίησης των λέξεων σε *κλάσεις (classes)* και του υπολογισμού των πιθανοτήτων με βάση αυτές τις κλάσεις. Μια επιλογή για αυτές τις κλάσεις είναι τα ονομαζόμενα *μέρη του λόγου (parts of speech – pos)* στη γλωσσολογία κάτι που εξηγεί και το όνομα των *N-pos μοντέλων*. Στις παραγράφους που ακολουθούν θα αναλύσουμε την βασική θεωρία των μοντέλων αυτών.

Έστω ότι $G=\{g_1,...,g_j,...,g_l\}$ είναι το σύνολο των κλάσεων και έστω ότι $g(w)$ είναι η κλάση μιας δοσμένης λέξης w. Επίσης ας θεωρήσουμε ότι $g(w_{i_1}^{i_2})$,

$1 \le i_1 \le i_2 \le n$ είναι μια συντομογραφία για το $g(w_{i_1})\ g(w_{i_1+1}),\ldots,g(w_{i_2})$. Στο *N-pos* μοντέλο, οι πιθανότητες εξαρτώνται από τις κλάσεις των *N-1* προηγούμενων λέξεων. Άρα το *context* καθορίζεται από τις *N-1* προηγούμενες κλάσεις:

$$P(w_t = w_l \mid c) = p(w_i = w_l \mid g(w_{i-N+1}^{i-1})) \qquad (4.1)$$

Αυτό το μοντέλο έχει $|G|^{N-1}$ κατανομές και απαιτεί τον υπολογισμό $|G|^{N-1}*|V|$ πιθανοτήτων. Για κάποιες ενδεικτικές τιμές $|G|=200$, $|V|=10000$, $N=3$, το *N-pos* μοντέλο *(tri-pos)* έχει $8*10^{10}$ πιθανότητες. Ο αριθμός αυτός είναι πολύ μικρότερος σε σχέση με τον αντίστοιχο αριθμό για ένα *tri-gram* μοντέλο.

Επιλέον, κάποιος μπορεί να υποστηρίξει ότι η κλάση της προηγούμενης λέξης περισσότερο περιορίζει την κλάση της επόμενης λέξης παρά την ταυτότητά της. Επομένως, μπορούμε να παράγουμε τις πιθανότητες με μια διαδικασία δύο βημάτων. Καταρχήν, υπολογίζουμε την πιθανότητα της κλάσης της επόμενης λέξης, δεδομένων των κλάσεων των προηγούμενων *N-1* λέξεων. Στη συνέχεια, υπολογίζουμε την πιθανότητα της λέξης δεδομένης της κλάσης της και ανεξάρτητα από τις κλάσεις των προηγούμενων λέξεων. Αυτό οδηγεί στον επόμενο τύπο:

$$P(w_i = w_l \mid c) = P(g(w_i) \mid g(w_{i-N+1}^{i-1})) \cdot P(w_i = w_l \mid g(w_i)) \qquad (4.2)$$

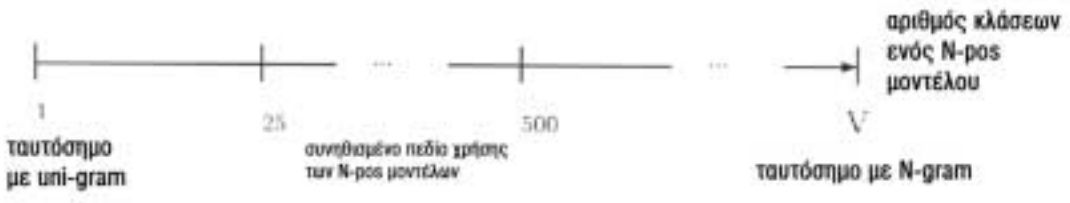
Αυτό το μοντέλο έχει τον ίδιο ορισμό για το *context* αλλά έχει μόνο $|G|^{N-1}*|G|+|G|*|V|$ παραμέτρους. Για τις ίδιες τιμές των $|G|$ και $|V|$ αυτό αντιστοιχεί σε 10^8 πιθανότητες, αριθμό πολύ μικρότερο συγκρινόμενο με αυτόν του προηγούμενου μοντέλου.

Το παραπάνω μοντέλο έχει τον περιορισμό κάθε λέξη να ανήκει σε μία και μόνο κλάση. Όμως, μία λέξη μπορεί να ανήκει σε παραπάνω από μία κλάσεις. Για παράδειγμα, έστω ότι μια λέξη ανήκει στις κλάσεις *C1*,*C2*,*C3*. Η πιθανότητα για να δούμε αυτή τη λέξη είναι ίση με την πιθανότητα να δούμε την λέξη και να ανήκει στην κλάση *C1* συν την πιθανότητα να δούμε την λέξη και να ανήκει στην κλάση *C2* συν την πιθανότητα να δούμε την λέξη και να ανήκει στην κλάση *C3*. Αυτό οδηγεί στον επόμενο τύπο, όπου οι πιθανότητες αθροίζονται πάνω σε όλες τις κλάσεις του συνόλου *G*:

$$P(w_i = w_l \mid c) = \sum_{g_j \in G} P(g(w_i)=g_j \mid g(w_{i-N+1}^{i-1})) \cdot P(w_i = w_l \mid g(w_i)=g_j). \qquad (4.3)$$

Αυτό είναι ισοδύναμο με το να αθροίσουμε πάνω σε όλες τις κλάσεις, στις οποίες η λέξη w_i ανήκει, αφού το δεύτερο μέλος του παραπάνω τύπου θα είναι μηδέν για τις κλάσεις που δεν περιέχουν την w_i.

Για να συσχετίσουμε τα *N-pos* μοντέλα με τα *N-gram* μοντέλα, είναι σημαντικό να δούμε τις ειδικές περιπτώσεις των *N-pos* μοντέλων, όπως για παράδειγμα ένα μοντέλο με μόνο μία λέξη και ένα μοντέλο με μία κλάση ανά λέξη *(Σχ. 4.1)*. Εάν ένα *N-pos* μοντέλο έχει μόνο μία κλάση, τότε το να γνωρίζουμε τις κλάσεις των *N-1* προηγούμενων λέξεων δεν περιέχει καμία πληροφορία για το context επειδή οι *N-1* τελευταίες λέξεις πάντα ανήκουν στην ίδια κλάση μοναδική κλάση. Ομοίως, η «πρόβλεψη» της κλάσης της λέξης δεν περιέχει καμιά πληροφορία για την «πρόβλεψη» της λέξης ή του context, επειδή όλες οι λέξεις ανήκουν σε αυτήν την κλάση. Επομένως, αφού και οι δύο παράγοντες στην εξίσωση *(4.3)* είναι ανεξάρτητοι του *context*, η πρόβλεψη της επόμενης λέξης θα είναι και αυτή ανεξάρτητη του *context*. Έτσι παράγεται ένα *μοντέλο ανεξάρτητο από το context (context independent model)* με μόνο μία κατανομή. Το μοντέλο αυτό είναι ισοδύναμο με το *unigram μοντέλο των word n-grams*. Στην άλλη ακραία περίπτωση, έχουμε ένα μοντέλο με μια ξεχωριστή κλάση ανά λέξη. Αν ένα μοντέλο έχει μία κλάση ανά λέξη, τότε η «πρόβλεψη» της λέξης δεδομένης της κλάσης της γίνεται ασήμαντη, αφού κάθε κλάση περιέχει μόνο μία λέξη. Σε αυτήν την περίπτωση, ο καθορισμός του *context* συναρτήσει των κλάσεων των *N-1* προηγούμενων λέξεων, είναι ουσιαστικά ο καθορισμός του *context* συναρτήσει της ταυτότητας των *N-1* προηγούμενων λέξεων. Αυτό σημαίνει ότι ο δεύτερος παράγοντας στην εξίσωση *(4.3)* θα είναι πάντα ίσος με ένα και ο πρώτος παράγοντας θα είναι η «πρόβλεψη» της επόμενης λέξης δεδομένων των *N-1* προηγούμενων λέξεων. Με άλλα λόγια, παράγουμε ουσιαστικά ένα καθαρά *word n-gram μοντέλο.* Από τις παρατηρήσεις αυτές συμπεραίνουμε ότι το *N-pos* μοντέλο βρίσκεται κάπου μεταξύ του *unigram* και του *N-gram* μοντέλου, ανάλογα με τον αριθμό των κλάσεων που χρησιμοποιεί.



Σχ. 4.1

Το πλεονέκτημα του *N-pos* μοντέλου είναι ότι απαιτεί πολύ λιγότερα δεδομένα εκπαίδευσης απ' ότι το *N-gram* μοντέλο, ενώ λαμβάνει υπ'όψιν την

πληροφορία για τις κλάσεις των *N-1* προηγούμενων λέξεων. Το μειονέκτημά του είναι ότι οι κατανομές του εξαρτώνται από κλάσεις και όχι από συγκεκριμένες λέξεις. Σαν παράδειγμα, ας θεωρήσουμε ότι η κλάση *«ΑΡΘΡΟ»* περιέχει το άρθρο *«το»* όπως και διάφορα άλλα άρθρα. Στην περίπτωση ενός *bi-pos* μοντέλου, θα έχουμε μία κατανομή, δεδομένου ότι η τελευταία λέξη ήταν άρθρο. Αν γνωρίζαμε, όμως, ότι η τελευταία λέξη ήταν το άρθρο *«το»*, η κατανομή θα ήταν σημαντικά διαφοροποιημένη αφού δεν θα περιείχε ουσιαστικά σε πληθυντικό βαθμό. Γενικά, η απόδοση του *N-pos* μοντέλου δεν είναι τόσο καλή όσο η απόδοση ενός *N-gram* μοντέλου που έχει εκπαιδευτεί με επαρκή δεδομένα, αλλά είναι καλύτερο από ένα *N-gram* μοντέλο εκπαιδευμένο με ανεπαρκή δεδομένα.

4.1 Ένα γενικευμένο *N-pos* μοντέλο

Στην παράγραφο αυτή θα εισάγουμε ένα πιο γενικευμένο *N-pos* γλωσσικό μοντέλο, γενικεύοντας τις δύο επόμενες ιδιότητες των *N-gram* και *N-pos μοντέλων*. Καταρχήν, αντί να έχουμε κατανομές βασισμένες στην τελευταία λέξη ή στην τελευταία κλάση, μπορούμε να βασίσουμε αυτές τις κατανομές σε κάθε πληροφορία που έχουμε για τις λέξεις που έχουμε δει μέχρι στιγμής. Θα κωδικοποιήσουμε αυτή την πληροφορία με μια μεταβλητή *X*. Για παράδειγμα, το *X* θα μπορούσε να εκφράζει την τελευταία λέξη, στην οποία περίπτωση το πεδίο τιμών του *X* θα ήταν όλες οι πιθανές λέξεις του λεξικού. Επίσης, το *X* θα μπορούσε να εκφράζει την κατάσταση ενός απλού *αναλυτή (parser)*, και θα μπορούσαμε έτσι να εκφράσουμε τις κατανομές του μοντέλου σε εξάρτηση με την πληροφορία αυτή.

Η δεύτερη ιδιότητα που μπορεί να γενικευτεί βασίζεται στο *N-pos* μοντέλο, του οποίου η πιθανότητα της επόμενης λέξης εξαρτάται μόνο από την υποτιθέμενη κλάση της λέξης $P(w_i| g_i)$. Επίσης, διαισθητικά είναι ξεκάθαρο ότι οι συχνότητες των ουσιαστικών ποικίλουν σημαντικά ανάλογα με το context της λέξης. Για παράδειγμα, στο context *«Ο Πέτρος μιλάει στον ΟΥΣΙΑΣΤΙΚΟ»* τα ουσιαστικά τα οποία τα οποία καταλαβαίνουν ομιλία είναι τα πιο πιθανά να εμφανιστούν. Επιπλέον, η διαίσθησή μας υποδεικνύει ότι το *«άμεσο context» (immediate context)* του *«ΟΥΣΙΑΣΤΙΚΟ»*, δηλαδή το *«στον»*, δεν περιορίζει των αριθμό των δυνατών επιλογών τόσο πολύ όσο

το γεγονός ότι στο *«ΟΥΣΙΑΣΤΙΚΟ»* μιλάει ο *«Πέτρος»*. Με άλλα λόγια, ακόμα και αν η αμέσως προηγούμενη λέξη είναι πολύ χρήσιμη στην «πρόβλεψη» της επόμενης κλάσης, φαίνεται πιθανό η χρήσιμη πληροφορία για την «πρόβλεψη» του πραγματικού ουσιαστικού να είναι πολύ πιο μακριά από την λέξη που θέλουμε να «προβλέψουμε». Το γενικευμένο μοντέλο θα πρέπει, επομένως, να επιτρέπει στην «πρόβλεψη» της πραγματικής λέξης να εξαρτάται από *contextual πληροφορία* και θα πρέπει να είναι πιθανό γι' αυτήν την πληροφορία να είναι διαφορετική από εκείνη που χρησιμοποιείται για την «πρόβλεψη» της επόμενης κλάσης.

Τώρα που έχουμε δει την διαισθητική ιδέα πίσω από το γενικευμένο *N-pos* μοντέλο, θα το εξετάσουμε με μεγαλύτερη λεπτομέρεια. Η πιθανότητα μιας ακολουθίας λέξεων *W* μπορεί να αναλυθεί σε ένα γινόμενο πιθανοτήτων για κάθε λέξη:

$$P(W) = \prod_{i=1}^{i=n} P(w_i \mid w_1^{i-1}) \qquad (4.4)$$

Επομένως, η πιθανότητα για κάθε λέξη μοντελοποιείται ως εξής:

$$P(w_i \mid w_1^{i-1}) = P(w_i \mid X_1,\ldots, X_{r+s}) =$$
$$\sum_{g_j \in G} P(g(w_i) \mid X_1,\ldots, X_r) \cdot P(w_i \mid g(w[i]) = g_j, X_{r+1},\ldots, X_{r+s}) \qquad (4.5)$$

όπου τα X_j, $1<=j<=r+s$, υποδηλώνουν μεταβλητές που κωδικοποιούν κάποια πληροφορία διαθέσιμη από τις λέξεις $w_1,\ldots,w_{i-1}$ που έχουμε δει μέχρι στιγμής. Είναι σημαντικό να επιβεβαιώσουμε ότι οι τελικές πιθανότητες αποτελούν μια κατανομή πιθανότητας. Με άλλα λόγια πρέπει να βεβαιωθούμε ότι:

$$\sum_{w \in V} P(w_i = w \mid X_1,\ldots, X_{r+s}) = 1. \qquad (4.6)$$

Σε ένα συγκεκριμένο σημείο την πρόταση, όλα τα X_j έχουν μια δεδομένη σταθερή τιμή. Αν οι δύο όροι πιθανοτήτων είναι πραγματικές κατανομές πιθανοτήτων για όλους τους συνδυασμούς των τιμών X_j (και αυτό επιβεβαιώνεται αν έχουν

δημιουργηθεί, ως συνήθως, από αριθμούς εμφάνισης των λέξεων στο κείμενο), θα αποδείξουμε στο *Παράρτημα Ε* ότι το παραπάνω άθροισμα ισούται με ένα.

Η ποσότητα των δεδομένων εκπαίδευσης που χρειάζεται για την εκπαίδευση του γενικευμένου *N-pos* μοντέλου εξαρτάται από την «γνώση» που κωδικοποιείται από τις διάφορες μεταβλητές. Επομένως, δεν μπορούμε να κάνουμε κάποια γενική αναφορά όσον αφορά την ποσότητα των δεδομένων εκπαίδευσης που χρειάζεται. Αλλά, όπως θα δούμε, το μοντέλο μπορεί να αναχθεί στο *N-gram* μοντέλο ή στο *N-pos* μοντέλο, και το μέγεθος των δεδομένων εκπαίδευσης σε αυτές τις περιπτώσεις θα είναι παρόμοιο με αυτό των δεδομένων που απαιτούνται από τα *N-gram* και *N-pos* μοντέλα.

Θα δείξουμε τώρα ότι το γενικευμένο μοντέλο ανάγεται στα *N-gram* και *N-pos* μοντέλα για κάποιες συγκεκριμένες επιλογές των τιμών των μεταβλητών X_j. Αν επιλέξουμε $s = 0$, $r = N-1$ και $X_j=g(w_{i-j})$, $j=1, ...,N-1$, τότε το γενικευμένο *N-pos* μοντέλο ανάγεται στο κλασσικό *N-pos* μοντέλο. Όπως θα αποδειχθεί στο *Παράρτημα ΣΤ* για $r = N-1$ και $X_{r+j}=X_j=w_{i-j}$, $j=1, ...,N-1$, το γενικευμένο μοντέλο ανάγεται στο *N-gram* μοντέλο. Για άλλες επιλογές των τιμών των μεταβλητών, προκύπτουν μοντέλα που μπορούν να κατασκευαστούν από το *N-gram* ή το *N-pos* μοντέλο. Αυτό δείχνει ότι πρόκειται πραγματικά για μια γενίκευση. Επιπλέον, κάποιες από τις μεταβλητές θα μπορούσαν να κωδικοποιήσουν γλωσσικά σχετικά μεταξύ τους γεγονότα που εκτείνονται σε μια μακρύτερη απόσταση μέσα στην πρόταση, για παράδειγμα το υποκείμενο της πρότασης ή το γεγονός αν το ρήμα είναι μεταβατικό. Το γενικευμένο *N-pos* μοντέλο είναι, λοιπόν, ένα μοντέλο που επιτρέπει τη σύλληψη περισσότερο γενικής γλωσσολογικής γνώσης.

Παρ' όλα αυτά, όμως, από πρακτικής πλευράς, το γενικευμένο μοντέλο είναι χρήσιμο μόνο αν υπάρχουν πηγές πληροφορίας για τις μεταβλητές X_j που πραγματικά βοηθούν στη βελτίωση της ποιότητας του μοντέλου σημαντικά. Τί πληροφορία θα έπρεπε, πραγματικά, να κωδικοποιήσουν οι μεταβλητές για να συμβάλλουν στην βελτίωση της ποιότητας του μοντέλου; Η έλλειψη της απάντησης σε αυτήν την ερώτηση αντιστοιχεί στην έλλειψη γνώσης σε αυτό το πεδίο έρευνας. Χρειάζεται αρκετή δουλειά για να βρεθεί τι είδους πληροφορία είναι χρήσιμη για αυτό το σκοπό.

4.2 Διαδικασία *POS tagging* στις λέξεις του κειμένου εκπαίδευσης *(training corpus)*

Λόγω έλλειψης ενός ελληνικού *part-of-speech λεξικού* δεν ήταν δυνατή η άμεση αντιστοίχιση *part-of-speech tags* στις λέξεις του κειμένου εκπαίδευσης. Δηλαδή η αντιστοίχιση των λέξεων του κειμένου εκπαίδευσης σε ένα αριθμό *part-of-speech κλάσεων*. Έτσι χρειάστηκε να ακολουθήσουμε μια ιδιαίτερα πολύπλοκη διαδικασία για το σκοπό αυτό.

Έχοντας στη διάθεσή μας τις λέξεις ενός αγγλικού λεξικού (50000 περίπου λέξεις) χρησιμοποιήσαμε ένα *online* αγγλοελληνικό λεξικό ενός ελληνικού web site για να εξάγουμε τελικά το ελληνικό λεξικό με *part-of-speech tags*. Το ηλεκτρονικό αγγλοελληνικό λεξικό περιείχε στην μετάφραση των αγγλικών λέξεων σε ελληνικές και το μέρος του λόγου των ελληνικών λέξεων.

Χρησιμοποιήσαμε ένα *script κατασκευασμένο σε γλώσσα Perl (Perl script),* η είσοδος του οποίου ήταν το αγγλικό λεξικό που είχαμε στη διάθεσή μας. Το συγκεκριμένο *script*, ουσιαστικά, δημιουργούσε έναν *εικονικό internet browser (virtual browser).* Για κάθε λέξη του αγγλικού λεξικού, ο *browser* έκανε μια *αίτηση (request)* στο *web site* που περιείχε το αγγλοελληνικό λεξικό και έπαιρνε σαν *απόκριση (respond)* τον κώδικα μιας *HTML σελίδας* που περιείχε και την μετάφραση της αγγλικής λέξης, για την οποία ο *browser* έκανε *request* στο αγγλοελληνικό λεξικό. Ο κώδικας της *HTML* σελίδας αποθηκεύονταν σε μια *ειδική δομή (object)* της γλώσσας *Perl*. Η επεξεργασία της δομής αυτής έγινε με κάποιες συναρτήσεις των πακέτων *LWP::UserAgent, HTTP::Request, HTTP::Resposne* της *Perl*.

Η μορφή της απόκρισης που λαμβάναμε για κάθε μία από τις αγγλικές λέξεις είναι αυτή που φαίνεται παρακάτω. Βλέπουμε ότι ανάμεσα στα *HTML tags* υπάρχει και το καθαρό κείμενο που αποτελεί τη μετάφραση για την αγγλική λέξη (στη συγκεκριμένη περίπτωση η λέξη *«array»*). Επίσης, μπορούμε να δούμε και το μέρος του λόγου της ελληνικής λέξης :

«

...

...

...

<TABLE WIDTH='710' cellpadding=5><TR><TH WIDTH='100%' COLSPAN=3 BGCOLOR=#E6D4C6><FONT FACE='ARIAL'>**Αποτελέσματα αναζήτησης για 'array':** </FONT></TH></TR><TR BGCOLOR=#CEBCAE><TD WIDTH = '4%' class='Number'>1</TD><td width='20%' class='Word'>array<TD WIDTH = '76%' class='Explanation'>[erEi]</p> **ουσ****. παράταξη, διάταξη, τάξη**: **in battle array** **σε διάταξη μάχης** # **συλλογή, σύνολο, σειρά, ποικιλία,** κν. **τακίμι**: **he showed me an impressive array of fabrics** **μου έδειξε μια εντυπωσιακή ποικιλία υφασμάτων** § **array of tools** **σειρά εργαλείων<**/font></TD></TR>

<TR BGCOLOR=#B6A496><TD WIDTH = '4%' class='Number'>2</TD><td width='20%' class='Word'>array<TD WIDTH = '76%' class='Explanation'>[erEi]</p> **ρ****. (παρα)τάσσω, αραδιάζω, παραθέτω, τακτοποιώ:** **books arrayed on the shelves** **βιβλία αραδιασμένα στα ράφια** # **διατάσσω, αναπτύσσω, βάζω σε διάταξη:** <i**> arrayed for battle** **διατεταγμένοι για μάχη** # **ενδύω, ντύνω,**

"περιβάλλω": **arrayed in ceremonial clothes** **ντυμένος με τελετουργική περιβολή**</TD></TR>
<TR BGCOLOR=#CEBCAE><TD WIDTH = '4%' class='Number'>3</TD><td width='20%' class='Word'>**array of tools**<TD WIDTH = '76%' class='Explanation'>**σειρά εργαλείων. Δείτε επίσης**: array</TD>
</TR>
<TR BGCOLOR=#B6A496><TD WIDTH = '4%' class='Number'>4</TD><td width='20%' class='Word'>**arrayed for battle**<TD WIDTH = '76%' class='Explanation'>**διατεταγμένοι για μάχη. Δείτε επίσης:** array</TD>
</TR>
<TR BGCOLOR=#CEBCAE><TD WIDTH = '4%' class='Number'>5</TD><td width='20%' class='Word'>**arrayed in ceremonial clothes**<TD WIDTH = '76%' class='Explanation'>**ντυμένος με τελετουργική περιβολή. Δείτε επίσης:** array</TD>
</TR>
</TABLE>

...
...
»

Είναι ξεκάθαρο πως το επόμενο βήμα ήταν ο «καθαρισμός» της *HTML* σελίδας από τα *HTML tags* και η εξαγωγή από αυτήν του *«καθαρού» κειμένου (plain text),* και στη συνέχεια η εξαγωγή από το «καθαρό» κείμενο της ελληνικής λέξης και του μέρους του λόγου της λέξης αυτής. Το «καθαρό» κείμενο φαίνεται με πιο έντονα γράμματα στην παραπάνω μορφή.

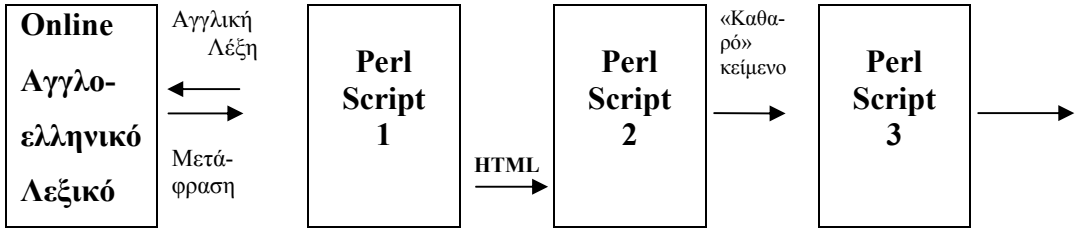
Ας πάρουμε, όμως, τα βήματα με τη σειρά. Το πρώτο βήμα, λοιπόν, από τα δύο, δηλαδή η εξαγωγή όλου του καθαρού κειμένου από τον *HTML* κώδικα,

υλοποιήθηκε με scripts σε γλώσσα *Perl*. Τα *scripts* αυτά έπαιρναν σαν είσοδο τον *HTML* κώδικα και η έξοδος τους ήταν της μορφής που φαίνεται παρακάτω:

«
...
...
Αποτελέσματα αναζήτησης για 'array':
ουσ. παράταξη, διάταξη, τάξη
in battle array
σε διάταξη μάχης
συλλογή, σύνολο, σειρά, ποικιλία,
τακίμι:
he showed me an impressive array of fabrics
μου έδειξε μια εντυπωσιακή ποικιλία υφασμάτων
array of tools
σειρά εργαλείων
ρ. (παρα)τάσσω, αραδιάζω, παραθέτω, τακτοποιώ:
books arrayed on the shelves
βιβλία αραδιασμένα στα ράφια
διατάσσω, αναπτύσσω, βάζω σε διάταξη:
arrayed for battle
διατεταγμένοι για μάχη
ενδύω, ντύνω, "περιβάλλω":
arrayed in ceremonial clothes
ντυμένος με τελετουργική περιβολή
array of tools
σειρά εργαλείων. Δείτε επίσης
arrayed for battle
διατεταγμένοι για μάχη. Δείτε επίσης:
arrayed in ceremonial clothes
ντυμένος με τελετουργική περιβολή. Δείτε επίσης:
...
»

Από το κείμενο αυτό, λοιπόν, θέλαμε να κρατήσουμε μόνο την ελληνική λέξη (μετάφραση της αγγλικής) και το *POS tag* της λέξης. Δηλαδή, η τελική έξοδός μας να είναι απλά της μορφής *«ουσ. παράταξη, διάταξη, τάξη» και «ρ. παρατάσσω, αραδιάζω, παραθέτω, τακτοποιώ»*. Το βήμα αυτό υλοποιήθηκε επίσης με *Perl scripts*. Οι κώδικες κάποιων από αυτά τα *scripts* φαίνονται στο *Παράρτημα Ζ*.

Η συνολική διαδικασία που ακολουθήθηκε μέχρι να καταλήξουμε σε μία έξοδο της μορφής *«POS tag – λέξη»*, μπορεί να αναπαρασταθεί σχηματικά ως εξής :



Σχ. 4.2

Με τον τρόπο αυτό έγινε δυνατό να περιγραφούν με *POS tag* γύρω στις 20000 λέξεις. Καθώς όμως οι απαιτήσεις μας για το λεξικό του γλωσσικού μοντέλου ήταν μεγαλύτερες (τουλάχιστον 60000 λέξεις), προχωρήσαμε στο επόμενο βήμα που ήταν ο εμπλουτισμός του *POS* λεξικού, βάσει στοιχείων της γραμματικής.

4.3 Εμπλουτισμός του *POS* λεξικού

Η παραπάνω διαδικασία, όπως αναφέραμε στην προηγούμενη παράγραφο, είχε ως αποτέλεσμα τη δημιουργία ενός *POS* λεξικού στα ελληνικά συνολικού μεγέθους περίπου 20000 λέξεων και 8 κλάσεων (ρήμα, ουσιαστικό, επίθετο, μετοχή, άρθρο, αντωνυμία, αριθμητικό). Το λεξικό αυτό βέβαια ήταν πολύ μικρό τόσο σε αριθμό λέξεων όσο και σε αριθμό κλάσεων.

Το επόμενό μας βήμα, λοιπόν, ήταν ο εμπλουτισμός του *POS* λεξικού σε λέξεις αλλά και σε κλάσεις. Στο βήμα αυτό εφαρμόσαμε κανόνες της ελληνικής

γραμματικής *[33]* πάνω στις ήδη υπάρχουσες λέξεις του λεξικού. Οι κανόνες αυτοί εφαρμόστηκαν μέσω *Perl scripts* που υλοποιήσαμε.

Ας δούμε ένα παράδειγμα. Θεωρούμε ότι αρχικά στο λεξικό υπήρχε η λέξη *«παίζω»* με *tag «ρήμα»*. Μετά την εφαρμογή των *scripts*, τα οποία έκλιναν το ρήμα *«παίζω»* σε όλους τους χρόνους, είχαμε στο λεξικό πολύ περισσότερες λέξεις από ότι αρχικά, όπως :

«παίζεις»	*«παίξω»*	*«έπαιξα»*
«παίζει»	*«παίξεις»*	*«έπαιξες»*
«παίζουμε»	*«παίξει»*	*«έπαιξε»*
«παίζετε»	*«παίξουμε»*	*«παίξαμε»*
«παίζουν»	*«παίξετε»*	*«παίξατε»*
	«παίξουν»	*«έπαιξαν»*

Βλέπουμε, λοιπόν, ότι από μία και μόνο λέξη, εφαρμόζοντας ένα απλό γραμματικό κανόνα πήραμε κάποιες επιπλέον δεκάδες λέξεων που προήλθαν από την αρχική. Παραδείγματα κανόνων γραμματικής που εφαρμόσαμε μέσω των *Perl scripts* ήταν :

«Τα ισοσύλλαβα αρσενικά ουσιαστικά της πρώτης τάξης, δηλαδή αυτά που τελειώνουν σε –ας –ης –ες –ους, σχηματίζουν την ονομαστική, αιτιατική και κλητική του πληθυντικού σε *–ες: ο ναύτης – οι ναύτες.* Τα ανισοσύλλαβα τις σχηματίζουν σε *–δες: ο καναπές – οι καναπέδες.*»

«Τα ρήματα: *απαντώ, βουτώ, γλεντώ, κεντώ, κολλώ, κυβερνώ, κυλώ, κυνηγώ, μελετώ, νικώ, ξενυχτώ, πηδώ, προσδοκώ, ρωτώ, σταματώ, τιμώ κ.ά.* κλίνονται κατά το *αγαπώ* » *[33 σελ.171]*

Οι παραπάνω γραμματικοί κανόνες είναι δύο από τους πολλούς που εφαρμόσαμε στα *scripts*.

Τελικά, μας προέκυψε ένα ελληνικό *POS* λεξικό συνολικού μεγέθους περίπου 300000 λέξεων, το οποίο χρησιμοποιήσαμε στα μετέπειτα πειράματά μας. Οι

συνολικές κλάσεις του λεξικού ήταν 172. Οι κλάσεις αυτές παρατίθενται αναλυτικά στο *Παράρτημα Ζ*.

Παρακάτω, μπορούμε να δούμε ένα δείγμα των κλάσεων *«ΜΤΧΕ»* (μετοχή ενεργητική) και *«ΡΥΑΑΠΛ»* (ρήμα υποτακτική αορίστου α΄ πληθυντικό).

```
«
…
ΜΤΧΕ έχοντας 2091
ΜΤΧΕ βεβαιώνοντας 15
ΜΤΧΕ αδειάζοντας 16
ΜΤΧΕ εγγράφοντας 12
ΜΤΧΕ βαδίζοντας 19
ΜΤΧΕ αγγίζοντας 50
ΜΤΧΕ βγαίνοντας 182
ΜΤΧΕ εγκαθιστώντας 11
ΜΤΧΕ αγκαλιάζοντας 14
ΜΤΧΕ εγκαινιάζοντας 102
ΜΤΧΕ εγκαταλείποντας 99
ΜΤΧΕ αδιαφορώντας 194
ΜΤΧΕ χαζεύοντας 23
ΜΤΧΕ αγκομαχώντας 12
ΜΤΧΕ εγκρίνοντας 17
ΜΤΧΕ βελτιώνοντας 54
…
»
```

```
«
…
ΡΥΑΑΠΛ έρθουμε 87
ΡΥΑΑΠΛ βεβαιωθούμε 15
ΡΥΑΑΠΛ αγαπήσουμε 25
ΡΥΑΑΠΛ βαδίσουμε 23
ΡΥΑΑΠΛ εγκαταλείψουμε 62
ΡΥΑΑΠΛ εγκαταστήσουμε 10
ΡΥΑΑΠΛ αδικήσουμε 11
ΡΥΑΑΠΛ βελτιώσουμε 67
ΡΥΑΑΠΛ γελάσουμε 29
ΡΥΑΑΠΛ δανειστούμε 16
ΡΥΑΑΠΛ αγνοήσουμε 42
ΡΥΑΑΠΛ αγοράσουμε 57
ΡΥΑΑΠΛ γεμίσουμε 24
ΡΥΑΑΠΛ γευτούμε 11
ΡΥΑΑΠΛ δείξουμε 194
…
»
```

Θα πρέπει, τέλος, να αναφερθεί ότι δεν ήταν δυνατόν να αντιστοιχήσουμε *POS tags* σε όλες τις λέξεις του κειμένου εκπαίδευσης. Οι λέξεις του κειμένου που δεν είχαν *POS tags* μετά την διαδικασία που περιγράψαμε στις δύο τελευταίες παραγράφους, θεωρήσαμε ότι ανήκουν σε μια κλάση *«ΑΓΝ»* (άγνωστη). Οι λέξεις με *POS tag* κάλυπταν περίπου το 70-75% του συνολικού αριθμού των λέξεων του κειμένου εκπαίδευσης,

Κεφάλαιο 5

Class-based Γλωσσικά Μοντέλα με χρήση Stemming και Part-of-Speech (POS)

5.1 Εισαγωγή

Στο κεφάλαιο αυτό θα περιγράψουμε την διαδικασία υλοποίησης των γλωσσικών μοντέλων που δημιουργήσαμε κατά τη διάρκεια της διπλωματικής μας εργασίας. Για το σκοπό αυτό χρησιμοποιήσαμε την *σειρά εργαλείων SRILM (SRILM Toolkit)*. Το *SRILM* είναι μια σειρά εργαλείων για την κατασκευή και την εφαρμογή *στατιστικών γλωσσικών μοντέλων (statistical language models)*. Θα περιγράψουμε στις επόμενες παραγράφους τα 3 πιο σημαντικά σε μας εργαλεία του *SRILM*.

5.1.1 *ngram-count*

Το εργαλείο *ngram-count* δημιουργεί και διαχειρίζεται αριθμούς εμφάνισης των *n-grams* μέσα στο κείμενο εκπαίδευσης και υπολογίζει από αυτούς γλωσσικά μοντέλα. Το πρόγραμμα αρχικά δημιουργεί ένα «εσωτερικό» σύνολο από αριθμούς εμφάνισης *n-grams*, είτε διαβάζοντας τους αριθμούς αυτούς από ένα αρχείο, είτε διαβάζοντας και αναλύοντας ένα κείμενο εισόδου. Στη συνέχεια το πρόγραμμα μπορεί να βγάλει στην έξοδό του τους αριθμούς εμφάνισης ή να τους χρησιμοποιήσει για να παράγει ένα γλωσσικό μοντέλο, ανάλογα με τις επιλογές που έχουμε κάνει πριν την εκτέλεση του προγράμματος.

Οι πιο βασικές επιλογές του *ngram-count* είναι οι ακόλουθες:

- **-order** *n*

  Θέτει τη μεγαλύτερη *τάξη (order)* των *n-grams*, οι αριθμοί εμφάνισης των οποίων θέλουμε να υπολογιστούν. Επίσης, καθορίζει και την τάξη του παραγόμενου γλωσσικού μοντέλου.

- **-vocab** *file*

  Διαβάζει ένα λεξικό από ένα αρχείο.

- **-text** *textfile*
  Διαβάζει τους αριθμούς εμφάνισης των *n-grams* από το αρχείο *textfile (training text).*

- **-lm** *lmfile*

  Υπολογίζει ένα *backoff γλωσσικό μοντέλο* από τους συνολικούς αριθμούς εμφάνισης και το αποθηκεύει στο αρχείο *lmfile.* Το γλωσσικό μοντέλο είναι σε μορφή *ngram-format (Παράρτημα Δ).*

- **-unk**

  Δημιουργεί ένα *«ανοιχτού λεξικού» γλωσσικό μοντέλο (open vocabulary language model)* δηλαδή ένα γλωσσικό μοντέλο που περιέχει στο λεξικό του το σύμβολο *«<unk>» (unknown word)* σαν κανονική λέξη.

- **-gt*n*min** *count*

  όπου *n=1,2,3,4,5* ή *6.* Θέτει τον ελάχιστο αριθμό εμφάνισης *n-grams* τάξεως *n* που θα συμπεριληφθούν στο γλωσσικό μοντέλο. Όλα τα *n-grams* με συχνότητα εμφάνισης μικρότερη από count αντιστοιχούνται σε αριθμό εμφάνισης 0.

- **-gt*n*max** *count*

  όπου *n=1,2,3,4,5* ή *6.* Θέτει τον μέγιστο αριθμό εμφάνισης *n-grams* τάξεως *n* των οποίων ο αριθμός εμφάνισης θα μειωθεί με την μέθοδο *Good-Turing.*

5.1.2 ***disambig***

Το πρόγραμμα *disambig* μεταφράζει μια *ροή από σύμβολα (stream of tokens)* από ένα λεξικό *V1,* σε μια αντίστοιχη ροή από σύμβολα από ένα λεξικό *V2*, σύμφωνα με μια πιθανοτική, *1-to-many*, αντιστοίχηση. Οι αμφιβολίες στην αντιστοίχιση επιλύονται βρίσκοντας την ακολουθία από το λεξικό *V2,* με την μεγαλύτερη *εκ των υστέρων πιθανότητα (posterior probability)* δεδομένης της ακολουθίας από το λεξικό *V1*.

Οι πιο βασικές επιλογές του *disambig* είναι οι ακόλουθες:

- **-text** *file*

  Καθορίζει το αρχείο που περιέχει τις προτάσεις του *V1*.

- **-map** *file*

  Καθορίζει το αρχείο που περιέχει την πληροφορία της *V1*-σε-*V2* αντιστοίχισης. Κάθε γραμμή του αρχείου περιέχει την αντιστοίχιση για μια λέξη του λεξικού *V1*:

  w1 w21 [p21] w22 [p22] ...

  όπου *w1* είναι μια λέξη από το λεξικό *V1*, που έχει σαν πιθανές αντιστοιχίσεις τις λέξεις *w21,w22 ...* από το λεξικό *V2*. Προαιρετικά, κάθε μία από αυτές τις λέξεις μπορεί να ακολουθηθεί από έναν αριθμό για τις πιθανότητες

p21,p22,…, οι οποίες έχουν σαν προκαθορισμένη τιμή το 1. Ο αριθμός p21 αντιστοιχεί στην πιθανότητα *P(w21|w1).*

- **-keep-unk**

  Δεν αντιστοιχεί τις άγνωστες λέξεις στην είσοδο στο σύμβολο <*unk*>. Αντί γι'αυτό βγάζει στην έξοδο του προγράμματος την λέξη εισόδου αναλλοίωτη.

5.1.3 ***ngram***

Η εντολή *ngram* εκτελεί διάφορες λειτουργίες πάνω σε *n-gram-based* και συγγενικά με *n-gram-based* γλωσσικά μοντέλα. Στις λειτουργίες αυτές εντάσσονται ο υπολογισμός του *perplexity*, η παραγωγή προτάσεων και διάφοροι τύποι παρεμβολής *(interpolation)* γλωσσικών μοντέλων. Τα *n-gram* γλωσσικά μοντέλα διαβάζονται από αρχεία της μορφής *ARPA n-gram-format* (*Παράρτημα Δ*).

Οι πιο βασικές επιλογές του *ngram* είναι οι ακόλουθες :

- **-order** *n*
  Θέτει τη μέγιστη *n-gram* τάξη που χρησιμοποιείται. Η τάξη του μοντέλου δεν τίθεται αυτόματα όταν διαβάζεται ένα μοντέλο, έτσι το ίδιο αρχείο μπορεί να χρησιμοποιηθεί για διάφορες τάξεις.

- **-lm** *file*
  Διαβάζει το κύριο *n-gram* μοντέλο από ένα αρχείο *(file)*.

- **-classes** *file*
  Ερμηνεύει το γλωσσικό μοντέλο σαν ένα *n-gram* μοντέλο που αποτελείται από κλάσεις. Οι επεκτάσεις των κλάσεων δίνονται στο αρχείο *file* σε μορφή *classes-format (Παράρτημα Δ)*. Τα σύμβολα του γλωσσικού μοντέλου που δεν ορίζονται σαν κλάσεις στο αρχείο *file* λαμβάνονται ως απλές λέξεις, έτσι που

το γλωσσικό μοντέλο να έχει τη δυνατότητα να περιέχει αναμεμειγμένα *n-grams* λέξεων και κλάσεων λέξεων.

- **-mix-lm** *file*
  Διαβάζει ένα δεύτερο *n-gram* μοντέλο με σκοπό την παρεμβολή. Το δεύτερο και οποιαδήποτε επιπλέον *interpolated* μοντέλα μπορούν να είναι επίσης *class n-grams* που χρησιμοποιούν τις ίδιες επεκτάσεις κλάσεων.

- **-lambda** *weight*
  Θέτει το βάρος *(weight)* του κύριου γλωσσικού μοντέλου όταν γίνεται παρεμβολή με την επιλογή -mix-lm.

- **-prune-lowprobs**
  Εφαρμόζει *pruning* στις πιθανότητες του *n-gram* που είναι μικρότερες από τις αντίστοιχες *backoff* εκτιμήσεις τους. Έτσι, παράγονται μοντέλα που μπορούν στη συνέχεια να μετασχηματιστούν σε στοχαστικά πεπερασμένα δίκτυα *(probabilistic finite-state networks)*.

- **-ppl** *textfile*
  Υπολογίζει τα επί μέρους, αλλά και το συνολικό *perplexity* για το μοντέλο, που προκύπτουν από τις προτάσεις του κειμένου δοκιμής *(test set)* που δίνεται στο αρχείο *textfile*.

5.2 Βασικό *Backoff Bigram* μοντέλο

Είναι το βασικό μας *word-based* μοντέλο. Περιγράφεται από την σχέση $P(W_2|W_1)$, δηλαδή την πιθανότητα της δεύτερης λέξης σε ένα *bigram* με δεδομένη την πρώτη λέξη. Αν για ένα *bigram* το μοντέλο δεν αντιστοιχίζει κάποια πιθανότητα, τότε γίνεται *backoff* στο *unigram* μοντέλο μεγίστης πιθανοφάνειας, $P(W_2)$. Τα *backoff βάρη (backoff weights)* υπολογίζονται με τον αλγόριθμο *Katz smoothing*.

Υλοποιήσαμε 5 μορφές του συγκεκριμένου μοντέλου, οι οποίες μπορούν να χωριστούν σε δύο κατηγορίες. Στα *μοντέλα «ανοιχτού» λεξικού (open-vocabulary models)* και στα *μοντέλα «κλειστού» λεξικού (closed-vocabulary models).* Στα μοντέλα «ανοιχτού» λεξικού οι λέξεις εκτός λεξικού *(Out-Of-Vocabulary words)* αντικαθίστανται με το σύμβολο *<unk>* και λαμβάνονται υπ' όψιν στον υπολογισμό των πιθανοτήτων των μοντέλων. Επομένως, τα μοντέλα αυτά περιλαμβάνουν στο λεξικό τους και το σύμβολο <unk>. Στα μοντέλα «κλειστού» λεξικού οι λέξεις εκτός λεξικού δεν λαμβάνονται υπ' όψιν και δεν συμμετέχουν στον υπολογισμό των πιθανοτήτων των μοντέλων.

Υλοποιήσαμε δύο μοντέλα «ανοιχτού» λεξικού. Το ένα με λεξικό μεγέθους 92000 λέξεων και το δεύτερο με λεξικό μεγέθους 63000 λέξεων. Στα δύο αυτά μοντέλα εφαρμόσαμε *Good-Turing discount* στα *bigrams* με *counts* από 1 εώς 7.

Τα δύο αυτά μοντέλα δημιουργήθηκαν με την εφαρμογή του προγράμματος *ngram-count* του *SRILM* στο κείμενο εκπαίδευσης *(Παράρτημα Α)* και με τιμές ορισμάτων που φαίνονται παρακάτω:

- `ngram-count -order 2 -text train.txt -gt2 discounts2 –unk –lm lm.2gram.GT1117.open.vocab-gt10-92K -vocab vocab-gt10-92K`

- `ngram-count -order 2 -text train.txt -gt2 discounts2 -unk –lm lm.2gram.GT1117.open.vocab-gt20-63K -vocab vocab-gt20-63K`

Τα μοντέλα «κλειστού» λεξικού που υλοποιήσαμε είχαν μέγεθος λεξικού 64000 λέξεις και 40000 λέξεις αντίστοιχα. Στα δύο αυτά μοντέλα, όπως και στα μοντέλα «κλειστού» λεξικού, εφαρμόσαμε *Good-Turing discount* στα *bigrams* με *counts* από 1 εώς 7.

Οι εντολές δημιουργίας των μοντέλων αυτών είναι:

- `ngram-count -order 2 -text train.txt -gt2 discounts2 -lm lm.2gram.GT1117.close.vocab-gt19-64K-noEnglish -vocab vocab-gt19-64K-noEnglish`

- `ngram-count -order 2 -text train.txt -gt2 discounts2 –lm lm.2gram.GT1117.close.vocab-gt19-40K -vocab vocab-gt19-40K`

Θα πρέπει να αναφερθεί ότι το μοντέλο *lm.2gram.GT1117.close.vocab-gt19-64K-noEnglish*, δεν περιείχε αγγλικές λέξεις.

Όλα τα παραπάνω μοντέλα χρησιμοποιήθηκαν για τον υπολογισμό της *αβεβαιότητας (perplexity)* του *test κειμένου (test text) (Παράρτημα Α)*. Ο υπολογισμός του *perplexity* πραγματοποιήθηκε με την εφαρμογή του προγράμματος *ngram* του *SRILM* στο *test* κείμενο. Για παράδειγμα, για να υπολογίσουμε το *perplexity* του *test* κειμένου με βάση το γλωσσικό μοντέλο *lm.2gram.GT1117.close.vocab-gt19-64K-noEnglish* χρησιμοποιήσαμε την εντολή:

```
ngram –lm lm.2gram.GT1117.close.vocab-gt19-64K-noEnglish –ppl
test.txt
```

Θα πρέπει, επίσης, να αναφέρουμε ότι τα γλωσσικά μοντέλα που δημιουργήσαμε με τα εργαλεία του *SRILM* έχουν την μορφή *ngram format (Παράρτημα Δ)*. Για να μπορέσουν τα μοντέλα αυτά να χρησιμοποιηθούν στον *αναγνωριστή της Nuance,* και να υπολογιστεί το *Word-Error-Rate (WER)* της αναγνώρισης, έπρεπε να τροποποιηθούν στην μορφή *PFSG (Παράρτημα Δ)*. Η τροποποίηση αυτή έγινε με ειδικά *Perl scripts*, ο κώδικας των οποίων δίνεται στο *(Παράρτημα Η*). Όλα τα αποτελέσματα παρουσιάζονται και αναλύονται στην παράγραφο *§6.1*.

5.3 *Class-based bigram* γλωσσικά μοντέλα με χρησιμοποίηση *stemming*

Για την κατασκευή *class-based* γλωσσικών μοντέλων χρησιμοποιήσαμε τον αλγόριθμο *stemming* για τα ελληνικά που περιγράψαμε στο κεφάλαιο 3, ομαδοποιώντας τις λέξεις του λεξικού με δύο διαφορετικούς τρόπους: Σε κλάσεις καταλήξεων και σε κλάσεις ριζών. Έτσι προέκυψαν δύο διαφορετικά *class-based* γλωσσικά μοντέλα που περιγράφονται στις επόμενες παραγράφους.

5.3.1 *Suffix-based* γλωσσικό μοντέλο

Με τη βοήθεια του *stemming* κατασκευάσαμε ένα *backoff bigram* γλωσσικό μοντέλο κλειστού λεξικού με κλάσεις καταλήξεων *(suffixes)*.

Η σχέση από την οποία παίρνουμε τις πιθανότητες των *bigrams* για την κατασκευή ενός *suffix-based bigram* γλωσσικού μοντέλου είναι:

$$P(w_2 \mid w_1) = P(w_2 \mid c_2^{suf}) \cdot P(c_2^{suf} \mid c_1^{suf}) \qquad (5.1)$$

όπου c_n^{suf} , $n \in \{1,2\}$ η κατάληξη της λέξης w_n. Όπως φαίνεται από την παραπάνω σχέση, θεωρούμε ότι η πιθανότητα κάθε *bigram* εξαρτάται μόνο από τη δεύτερη λέξη, την κατάληξή της και την κατάληξη της προηγούμενης λέξης. Στο μοντέλο αυτό γίνεται μια μεγάλη ομαδοποίηση των λέξεων σε μικρό αριθμό κλάσεων, αφού ο αριθμός των καταλήξεων είναι αρκετά μικρός.

Για την υλοποίηση του συγκεκριμένου μοντέλου, λοιπόν, πρέπει για κάθε *bigram* λέξεων του κειμένου εκπαίδευσης να υπολογιστούν οι δύο παράγοντες του δεύτερου μέλους της εξίσωσης *(5.1)*. Ο υπολογισμός των δύο αυτών παραγόντων έγινε με τον τρόπο που περιγράφεται παρακάτω.

Κατ' αρχήν, ήταν αναγκαία η ομαδοποίηση των λέξεων του λεξικού σε κλάσεις. Το γλωσσικό μοντέλο που κατασκευάσαμε εμπεριείχε ένα λεξικό των 64000 πιο συχνών λέξεων που συναντήθηκαν στα κείμενα εκπαίδευσης. Οι λέξεις αυτές ομαδοποιήθηκαν σε 175 κλάσεις καταλήξεων με τη βοήθεια του αλγόριθμου *stemming*. Έτσι, πήραμε μία αντιστοιχία κάθε λέξης με την κατάληξή της στο αρχείο *mapfile*. Για την υλοποίηση του δεύτερου παράγοντα του δεύτερου μέλους της σχέσης *(5.1)*, δηλαδή για να πάρουμε την πιθανότητα της κατάληξης κάθε λέξης δεδομένης της κατάληξης της προηγούμενης λέξης, αντικαταστήσαμε με τις καταλήξεις τους όλες τις λέξεις του λεξικού στο κείμενο εκπαίδευσης, χρησιμοποιώντας την εντολή *disambig* ως εξής:

```
disambig –text train.txt –map mapfile > train_suffix.txt
```

Στη συνέχεια υπολογίσαμε με τη βοήθεια του εργαλείου *ngram-count* τις πιθανότητες όλων των *bigrams* καταλήξεων. Το λεξικό μας πλέον ήταν οι 175 καταλήξεις, ενώ χρησιμοποιήσαμε και *Katz smoothing* για να υπολογίσουμε τα *backoff* βάρη τους. Έτσι, πήραμε σε *ARPA n-gram* μορφή τον παράγοντα $P(c_2^{suf} \mid c_1^{suf})$ για κάθε *bigram*, καθώς και τις πιθανότητες $P(c_2^{suf})$ των *unigrams* με τα αντίστοιχα *backoff* βάρη τους. Η εντολή που εκτελέσαμε για το σκοπό αυτό ήταν η εξής :

```
ngram-count –text train_suffix.txt –order 2 –lm lm.suffixes –
vocab vocab.suffix
```

Για τον υπολογισμό του παράγοντα $P(w_2 \mid c_2^{suf})$, δηλαδή την πιθανότητα κάθε λέξης μέσα στην κλάση της, σχηματίσαμε μία αντιστοίχιση κάθε κλάσης με τις λέξεις της και την πιθανότητα καθεμιάς λέξης μέσα στην κλάση, στη μορφή:

κλάση1	*λέξη1*	*P(λέξη1|κλάση1)*
κλάση1	*λέξη2*	*P(λέξη2|κλάση1)*
κλάση1	*λέξη3*	*P(λέξη3|κλάση1)*
κλάση2	*λέξη4*	*P(λέξη4|κλάση2)*
κλάση2	*λέξη5*	*P(λέξη5|κλάση2)*
...		

Η αντιστοίχιση αυτή ονομάζεται επέκταση των κλάσεων *(class expansion)*. Για την υλοποίηση της αντιστοίχισης αυτής, χρησιμοποιήσαμε τα κείμενα εκπαίδευσης στην αρχική μορφή τους. Από εκεί πήραμε τα *counts* των 64000 λέξεων και, αφού ομαδοποιήσαμε τις λέξεις με τα *counts* τους σε κλάσεις, με τη βοήθεια κατάλληλου *script (Παράρτημα Z*), υπολογίσαμε τις πιθανότητες των λέξεων μέσα σε κάθε κλάση, σύμφωνα με τη σχέση :

$$P(w_2 \mid c_2^{suf}) = \frac{n_{w_2}}{N_{c_2^{suf}}} \qquad (5.2)$$

όπου n_{w_2} το *count* της λέξης w_2 και $N_{c_2^{suf}}$ το άθροισμα των *counts* όλων των λέξεων μέσα στην κλάση c_2^{suf}.

Το σύνολο του *ARPA n-gram* μοντέλου καταλήξεων με τα *class expansions* τους μας δίνουν ένα ολοκληρωμένο γλωσσικό μοντέλο, για το οποίο είναι δυνατό να μετρήσουμε το *perplexity* του. Για τη μέτρηση του *perplexity* χρησιμοποιήσαμε το εργαλείο *ngram* και τα κείμενα δοκιμής *(test text)*, πάνω στα οποία έγινε η μέτρηση. Η εντολή με την οποία μετρήσαμε το *perplexity* είναι η εξής :

```
ngram  -lm lm.suffixes –ppl test.txt –classes suffix_expansions
```

Λόγω των φτωχών σε απόδοση αποτελεσμάτων στο *perplexity* (βλ. *§6.2*) αποφασίσαμε να μην προχωρήσουμε σε ενσωμάτωση του μοντέλου στον αναγνωριστή της *Nuance*.

5.3.2 ***Root-based* γλωσσικό μοντέλο**

Χρησιμοποιώντας, με τη βοήθεια του αλγόριθμου *stemming*, την αντιστοιχία των λέξεων με τις ρίζες τους κατασκευάσαμε ένα κλειστού λεξικού *backoff bigram* γλωσσικό μοντέλο με κλάσεις ριζών. Τα *backoff* βάρη υπολογίστηκαν με τον αλγόριθμο *Katz smoothing*.

Η αντίστοιχη με το *suffix-based bigram* γλωσσικό μοντέλο σχέση που δίνει την πιθανότητα των *bigrams*, είναι για το *root-based* μοντέλο η εξής:

$$P(w_2 \mid w_1) = P(w_2 \mid c_2^{root}) \cdot P(c_2^{root} \mid c_1^{root}) \qquad (5.3)$$

όπου c_n^{root} η ρίζα της λέξης w_n. Η πιθανότητα κάθε *bigram*, λοιπόν, εξαρτάται μόνο από τη δεύτερη λέξη, τη ρίζα της και τη ρίζα της προηγούμενης λέξης. Το μοντέλο αυτό, σε σχέση με το *suffix-based bigram* μοντέλο χρησιμοποιεί μικρότερη ομαδοποίηση των λέξεων, καθώς ο αριθμός των κλάσεων που προκύπτουν, δηλαδή των ριζών, είναι κατά πολύ μεγαλύτερος από αυτό των καταλήξεων.

Για τον υπολογισμό των δύο παραγόντων της σχέσης *(5.3)* εργαστήκαμε όπως και στην περίπτωση του *suffix-based bigram* γλωσσικού μοντέλου. Μετά την ομαδοποίηση των 64000 λέξεων δημιουργήθηκαν 30754 κλάσεις ριζών που αντικατέστησαν στα κείμενα εκπαίδευσης τις αντίστοιχές τους λέξεις. Στη συνέχεια, με τη βοήθεια του εργαλείου *ngram-count* υπολογίστηκε το *backoff bigram* μοντέλο των κλάσεων-ριζών που αναπαριστά τον παράγοντα $P(c_2^{root} \mid c_1^{root})$ της σχέσης *(5.3)* και τα αντίστοιχα *backoff* βάρη των *unigrams*. Επειδή κάποια *bigrams* ριζών είχαν μικρό αριθμό εμφάνισης στα κείμενα εκπαίδευσης, χρησιμοποιήσαμε *Good-Turing discount* για να τους δώσουμε μεγαλύτερη μάζα πιθανότητας. Έτσι, χρησιμοποιήσαμε *Good-Turing discount* για τα *bigrams* ριζών με *counts* από 1 έως 7. Η εντολή που τελικά εκτελέσαμε ήταν :

```
ngram-count –text train_root.txt –lm lm.root –vocab roots –order 2 –
gt1 discounts1 –gt2 discounts2
```

Με τρόπο αντίστοιχο με το *suffix-based bigram* γλωσσικό μοντέλο, σχηματίσαμε τα *class expansions* και προχωρήσαμε στη μέτρηση του *perplexity* του μοντέλου πάνω στο *test text*, χρησιμοποιώντας την εντολή :

```
ngram  -lm lm.roots –ppl test.txt –classes root_expansions
```

Στη συνέχεια ενσωματώσαμε το *root-based bigram* γλωσσικό μοντέλο στον αναγνωριστή της *Nuance.* Για το σκοπό αυτό εφαρμόσαμε *beam-pruning* στο μοντέλο, ώστε να αποκοπούν τα *bigrams* μικρής πιθανότητας και το μοντέλο να μπορεί να αναπαρασταθεί με τη μορφή *finite-state network*. Έτσι, μετρήσαμε και το *word error rate (WER)* της αναγνώρισης. Τα αποτελέσματα, τόσο στο *perplexity* όσο και στο *WER* δίνονται στην παράγραφο *§6.2*.

5.4 ***Class-based bigram* γλωσσικό μοντέλο με χρησιμοποίηση *POS***

Για την κατασκευή του *POS-based* γλωσσικού μοντέλου χρησιμοποιήσαμε την μέθοδο *POS-tagging* για τα ελληνικά που περιγράψαμε στο κεφάλαιο 4, ομαδοποιώντας τις λέξεις του λεξικού με βάση το μέρος του λόγου τους.

Η σχέση, η οποία περιγράφει το *POS-based* μοντέλο είναι:

$$P(w_2 \mid w_1) = P(w_2 \mid c_2^{POS}) \cdot P(c_2^{POS} \mid c_1^{POS}) \qquad (5.4)$$

όπου c_n^{POS} , $n \in \{1,2\}$ το μέρος του λόγου της λέξης w_n . Όπως και στο *suffix-based* γλωσσικό μοντέλο η πιθανότητα ενός *bigram* εξαρτάται από την δεύτερη λέξη του *bigram*, το μέρος του λόγου της λέξης αυτής και το μέρος του λόγου της πρώτης λέξης του *bigram*. Στο συγκεκριμένο μοντέλο γίνεται ομαδοποίηση των λέξεων σε αριθμό κλάσεων ανάλογο με τον αριθμό των κλάσεων στο *suffix-based* γλωσσικό μοντέλο. Συγκεκριμένα, ο αριθμός των *POS-κλάσεων* είναι 172.

Ο υπολογισμός των δύο παραγόντων του δεξιού μέλους της εξίσωσης *(5.4)* έγινε με παρόμοια μέθοδο με αυτή που περιγράφηκε στην παράγραφο *§5.3.1*.

Ομαδοποιήσαμε, λοιπόν, τις 64000 λέξεις του λεξικού στις 172 *POS*-κλάσεις Έτσι, πήραμε μία αντιστοιχία κάθε λέξης με το μέρος του λόγου της σε ένα αρχείο *mapfile*. Για τον υπολογισμό της πιθανότητας του μέρους του λόγου κάθε λέξης δεδομένου του μέρους του λόγου της προηγούμενης λέξης αντικαταστήσαμε όλες τις λέξεις του λεξικού με το μέρος του λόγου τους στο κείμενο εκπαίδευσης, χρησιμοποιώντας την εντολή *disambig* ως εξής:

```
disambig –text train.txt –map mapfile > train_pos.txt
```

Στη συνέχεια, με χρήση του προγράμματος ngram-count του SRILM υπολογίσαμε το *POS-bigram μοντέλο.* Το λεξικό του συγκεκριμένου μοντέλου περιελάμβανε τις 172 κλάσεις των λέξεων. Η εντολή που εκτελέσαμε για το σκοπό αυτό ήταν η εξής :

```
ngram-count –text train_pos.txt –order 2 –lm lm.pos –vocab vocab.pos
```

Υπολογίσαμε, λοιπόν, με την παραπάνω διαδικασία τον παράγοντα $P(c_2^{POS} \mid c_1^{POS})$ για κάθε *POS-bigram.*

Για τον υπολογισμό του παράγοντα $P(w_2 \mid c_2^{POS})$, δηλαδή την πιθανότητα κάθε λέξης μέσα στην *POS*-κλάση της, σχηματίσαμε μία αντιστοίχιση κάθε κλάσης με τις λέξεις της με μορφή παρόμοια με εκείνη της παραγράφου *§5.3.1*:

κλάση1	*λέξη2*	*P(λέξη2|κλάση1)*
κλάση1	*λέξη3*	*P(λέξη3|κλάση1)*
κλάση2	*λέξη4*	*P(λέξη4|κλάση2)*
κλάση2	*λέξη5*	*P(λέξη5|κλάση2)*

Δημιουργήσαμε, δηλαδή, τα *expansions* των POS-κλάσεων.

Για τη μέτρηση του *perplexity* χρησιμοποιήσαμε το πρόγραμμα *ngram* και το κείμενο δοκιμής *(test text).* Η εντολή με την οποία μετρήσαμε το *perplexity* είναι η εξής :

```
ngram  -lm lm.pos –ppl test.txt –classes pos_expansions
```

Λόγω των φτωχών σε απόδοση αποτελεσμάτων στο *perplexity* (βλ. *§6.3*) αποφασίσαμε να μην προχωρήσουμε σε ενσωμάτωση του μοντέλου στον αναγνωριστή της *Nuance,* όπως έγινε και για το μοντέλο με τις καταλήξεις των λέξεων.

5.5 *Class-based* γλωσσικό μοντέλο με συνδυασμό *stemming* και *POS*

Ένα άλλο *class-based* γλωσσικό μοντέλο που υλοποιήσαμε είναι αυτό στο οποίο η ομαδοποίηση σε κλάσεις προκύπτει από το συνδυασμό της ρίζας με το *POS* της λέξης. Κάθε λέξη του λεξικού, λοιπόν, χαρακτηρίζεται από τη ρίζα της και το μέρος του λόγου στο οποίο ανήκει. Έτσι, για παράδειγμα η λέξη *«διπλωματική»*, ως επίθετο με ρίζα *«διπλωματικ»* εντάσσεται στην κλάση *«Ε_διπλωματικ»* μαζί με τις λέξεις *«διπλωματικός», «διπλωματικού»* κλπ.

Για το κλειστού λεξικού *backoff bigram* συνδυασμένο μοντέλο που υλοποιήσαμε, η σχέση που δίνει την πιθανότητα των *bigrams* είναι :

$$P(w_2 \mid w_1) = P(w_2 \mid c_2^{comb}) \cdot P(c_2^{comb} \mid c_1^{comb}) \qquad (5.5)$$

όπου c_n^{comb} η κλάση που προκύπτει από το συνδυασμό της ρίζας με το *POS* της λέξης w_n. Για την υλοποίηση αυτού του συνδυασμού επιλέξαμε να χρησιμοποιήσουμε, από τα πολλά χαρακτηριστικά που προσδίδει στη λέξη το *POS* (π.χ. μέρος του λόγου, κλίση, πρόσωπο, αριθμός κλπ.), μόνο αυτό που αναφέρεται στο μέρος του λόγου στο οποίο η λέξη ανήκει. Έτσι, δημιουργήθηκαν 9 κλάσεις *POS*, που σε συνδυασμό με τη ρίζα κάθε λέξης απέδωσαν 35412 διαφορετικές κλάσεις στις οποίες ομαδοποιήθηκαν οι 64000 λέξεις του λεξικού. Ο λόγος που δε χρησιμοποιήσαμε τα επιπλέον χαρακτηριστικά του *POS tagging* είναι ότι σε μια τέτοια περίπτωση συνδυασμού

ρίζας και *POS*, ο αριθμός των παραγόμενων κλάσεων θα προσέγγιζε τον αριθμό των λέξεων του λεξικού και συνεπώς δε θα είχαμε ομαδοποίηση.

Για την υλοποίηση αυτού του μοντέλου αυξημένων κλάσεων εργαστήκαμε όπως και στις προηγούμενες περιπτώσεις. Χρησιμοποιήσαμε *Good-Turing discount* για τα *bigrams* κλάσεων με *counts* από 2 έως 7 και *Katz smoothing* για τον υπολογισμό των *backoff* βαρών. Η εντολή για την κατασκευή του *class-ngram* μοντέλου ήταν :

```
ngram-count –text train_comb.txt –lm lm.POS_roots.GT_1127
             -vocab roots –order 2 –gt1 discounts1 -gt2 discounts2
```

Στη συνέχεια δημιουργήσαμε τα *class expansions*, εφαρμόσαμε *pruning* και μετρήσαμε το *perplexity* και το *word error rate* του μοντέλου πάνω στο *test text* και στον αναγνωριστή της *Nuance* αντίστοιχα. Τα αποτελέσματα που πήραμε δίνονται στην παράγραφο *§6.4*.

5.6 Υβριδικά γλωσσικά μοντέλα με συνδυασμό κλάσεων και λέξεων

Στην προσπάθειά μας να μελετήσουμε τη συμπεριφορά των γλωσσικών μοντέλων υλοποιήσαμε μερικά μοντέλα υβριδικής μορφής. Η ιδιαιτερότητα αυτών των μοντέλων είναι ότι δεν αποτελούνται μόνο από κλάσεις ή μόνο από λέξεις, αλλά από το συνδυασμό τους. Έτσι, μέσα σ' αυτά υπάρχουν, σαν αυτούσιες λέξεις, οι λέξεις που απαρτίζουν τα πιο συχνά *bigrams* των κειμένων εκπαίδευσης, ενώ οι υπόλοιπες λέξεις του λεξικού ομαδοποιούνται σε κλάσεις. Το κριτήριο για την ένταξη μιας τέτοιας λέξης σε μια κλάση προκύπτει από το συνδυασμό της ρίζας και του *POS* της λέξης. Με τον τρόπο αυτό, δίνουμε ιδιαίτερη σημασία στην εκπαίδευση των πιο συχνών *bigrams* με σκοπό τη μείωση του *perplexity* και του *WER* σε σχέση με τα *class-based* μοντέλα, αφού αναμένεται συχνή εμφάνιση στο κείμενο δοκιμής των

συχνών *bigrams* των κειμένων εκπαίδευσης. Ταυτόχρονα, η πιθανότητα των πιο σπάνιων λέξεων μέσα στην κλάση τους αυξάνει, αφού πλέον η συχνή λέξη που πιθανόν να υπήρχε μέσα στην κλάση, τώρα δεν υπάρχει και η μάζα της πιθανότητάς της μέσα στην κλάση μοιράζεται στις υπόλοιπες λέξεις. Το τελικό αποτέλεσμα αυτού του μοντέλου ήταν η συνύπαρξη *bigram* λέξεων, *bigram* κλάσεων και *bigram* λέξεων και κλάσεων ταυτόχρονα. Για παράδειγμα, στο *bigram «διπλωματική εργασία»*, η συχνή λέξη *«εργασία»* παραμένει ως έχει, ενώ η σπάνια λέξη *«διπλωματική»* εντάσσεται στην κλάση *«Ε_διπλωματικ»* μαζί με τις επίσης σπάνιες λέξεις *«διπλωματικός», «διπλωματικού»* κλπ. Το αποτέλεσμα είναι να έχουμε ένα μεικτό *bigram* της μορφής *«Ε_διπλωματικ εργασία»*.

Υλοποιήσαμε υβριδικά γλωσσικά *backoff* μοντέλα κλειστού λεξικού για δύο περιπτώσεις λεξικών: 64000 λέξεων και 40000 λέξεων. Για κάθε λεξικό κατασκευάσαμε διάφορα μοντέλα. Στην *περίπτωση* του 64K λεξικού κρατήσαμε ως αυτούσιες, σε κάθε περίπτωση, 15K, 20K, 25K, 30K και 35K λέξεις , δηλαδή αυτές που σχηματίζουν τα *bigrams* με *count* μεγαλύτερο του 62, 42, 30, 23 και 18 αντίστοιχα. Στην περίπτωση του 40K λεξικού οι «καθαρές» λέξεις ήταν, σε κάθε περίπτωση, 10K, 17K, 20K και 23K, και τα κατώφλια στα *counts* των *bigrams* ήταν 100, 51, 41 και 34 αντίστοιχα. Οι υπόλοιπες λέξεις του κάθε λεξικού ομαδοποιήθηκαν, σε κάθε περίπτωση, σε κλάσεις συνδυασμένου *POS* και ρίζας, σχηματίζοντας έτσι μεικτά λεξικά κλάσεων και λέξεων. Τέλος, χρησιμοποιήθηκε *Katz smoothing* και *Good-Turing discount* στα *bigrams* με *counts* από 1 έως 7. Στον *πίνακα 5.1* φαίνονται συνοπτικά τα χαρακτηριστικά των μοντέλων που κατασκευάσαμε.

Στη συνέχεια αντικαταστήσαμε στα κείμενα εκπαίδευσης τις λέξεις που ανήκαν σε κλάση με την αντίστοιχη κλάση τους, και με τη βοήθεια του εργαλείου *ngram-count* κατασκευάσαμε τα μεικτά μοντέλα λέξεων και κλάσεων. Για παράδειγμα η εντολή που εκτελέσαμε για το μοντέλο *lm.hybrid_POS_roots.25K* ήταν η εξής:

```
ngram-count –text train_hybrid.txt –lm lm.hybrid_POS_roots.25K -vocab
hybrid_vocab.25K –order 2 –gt1 discounts1 -gt2 discounts2
```

Για τα *expansions* των κλάσεων εργαστήκαμε όπως και στις προηγούμενες περιπτώσεις *class-based* μοντέλων. Στη συνέχεια μετρήσαμε τα *perplexity* των μοντέλων πάνω στα κείμενα δοκιμής και, αφού εφαρμόστηκε *pruning* για τη μετατροπή των μοντέλων σε μορφή *PFSG*, ενσωματώσαμε τα μοντέλα στον αναγνωριστή της *Nuance* για να μετρήσουμε την απόδοσή τους σε ό,τι αφορά το *word error rate*. Τα αποτελέσματα για τα μοντέλα δίνονται στην παράγραφο *§6.5*.

Μοντέλα	**Λέξεις εκτός κλάσεων**	**Κατώφλι στα *counts* των *word-bigrams***
lm.hybrid.64K.15Kwords	*15K*	*62*
lm.hybrid.64K.20Kwords	*20K*	*42*
lm.hybrid.64K.25Kwords	*25K*	*30*
lm.hybrid.64K.30Kwords	*30K*	*23*
lm.hybrid.64K.35Kwords	*35K*	*18*
lm.hybrid.40K.10Kwords	*10K*	*100*
lm.hybrid.40K.17Kwords	*17K*	*72*
lm.hybrid.40K.20Kwords	*20K*	*51*
lm.hybrid.40K.23Kwords	*23k*	*41*

Πίνακας 5.1

5.7 Γλωσσικά μοντέλα μεγαλύτερης τάξης

Εκτός από τα *bigram* μοντέλα όλων των προηγουμένων παραγράφων, υλοποιήσαμε και γλωσσικά μοντέλα *μεγαλύτερης τάξης*, και συγκεκριμένα μοντέλα με *τάξη 3 (order 3)*. Ο τύπος που περιγράφει τα μοντέλα αυτά είναι:

$$P(W_3|W_2,W_1) \qquad (5.6)$$

δηλαδή η πιθανότητα της τρίτης λέξης ενός *trigram* δεδομένων των δύο προηγούμενων λέξεων. Αν για ένα *trigram* το μοντέλο δεν αντιστοιχίζει κάποια πιθανότητα, τότε γίνεται *backoff* στο *bigram* μοντέλο $P(W_2|W_1)$. Αν το μοντέλο δεν αντιστοιχίζει πιθανότητα ούτε στο *bigram* τότε γίνεται *backoff* στο *unigram* μοντέλο μεγίστης πιθανοφάνειας $P(W_3)$. Τα *backoff βάρη (backoff weights)* για τα *unigrams* και τα *bigrams* υπολογίζονται με τον αλγόριθμο *Katz smoothing*.

Υλοποιήσαμε 6 μορφές του *trigram* μοντέλου «κλειστού» λεξικού, η καθεμία με διαφορετικό μέγεθος λεξικού (64K,40K,35K,30K,25K). Στα μοντέλα αυτά εφαρμόσαμε *Good-Turing discount* στα *trigrams* με *counts* από 2 έως 7 και στα *bigrams* με *counts* από 1 έως 7. Επίσης, δημιουργήσαμε και ένα μοντέλο λεξικού μεγέθους 64K λέξεων με μόνη διαφορά ότι εφαρμόσαμε *Good-Turing discount* στα *trigrams* με *counts* από 3 έως 7. Τέλος, υλοποιήσαμε και ένα *trigram* μοντέλο «ανοιχτού» λεξικού με μέγεθος 64K λέξεων και *Good-Turing discount* στα *trigrams* με counts από 2 έως 7 και στα *bigrams* με *counts* από 1 έως 7. Τα 7 αυτά μοντέλα δημιουργήθηκαν με την εφαρμογή του προγράμματος *ngram-count* του *SRILM* στο κείμενο εκπαίδευσης (*Παράρτημα Α*) και με τιμές ορισμάτων που φαίνονται παρακάτω:

- `ngram-count -order 3 -text train.txt -gt1 discounts1 -gt2 discounts2 -gt3 discounts3 -lm lm.3gram.GT111727.close.vocab-gt19-64K-noEnglish –vocab vocab-gt19-64K-noEnglish`

- `ngram-count -order 3 -text train.txt -gt1 discounts1 -gt2 discounts2 -gt3 discounts3 -lm lm.3gram.GT111737.close.vocab-gt19-64K-noEnglish –vocab vocab-gt19-64K-noEnglish`

- `ngram-count -order 3 -text train.txt -gt1 discounts1 -gt2 discounts2 -gt3 discounts3 -lm lm.3gram.GT111727.close.vocab-gt40-40K-noEnglish –vocab vocab-gt40-40K-noEnglish`

- `ngram-count -order 3 -text train.txt -gt1 discounts1 -gt2 discounts2 -gt3 discounts3 -lm lm.3gram.GT111727.close.vocab-gt50-35K-noEnglish –vocab vocab-gt50-35K-noEnglish`

- `ngram-count -order 3 -text train.txt -gt1 discounts1 -gt2 discounts2 -gt3 discounts3 -lm lm.3gram.GT111727.close.vocab-gt65-30K-noEnglish –vocab vocab-gt65-30K-noEnglish`

- `ngram-count -order 3 -text train.txt -gt1 discounts1 -gt2 discounts2 -gt3 discounts3 -lm lm.3gram.GT111727.close.vocab-gt85-25K-noEnglish –vocab vocab-gt85-25K-noEnglish`

- `ngram-count -order 3 -text train.txt -gt1 discounts1 -gt2 discounts2 -gt3 discounts3 -unk –lm lm.3gram.GT111727.open.vocab-gt19-64K-noEnglish -vocab vocab-gt19-64K-noEnglish`

Θα πρέπει να αναφερθεί ότι τα *trigram* μοντέλα έχουν το μειονέκτημα ότι απαιτούν πολύ περισσότερο αποθηκευτικό χώρο από ότι τα *bigram* μοντέλα.

Όλα τα παραπάνω μοντέλα χρησιμοποιήθηκαν για τον υπολογισμό του *perplexity* του *test κειμένου,* μέσω του προγράμματος *ngram* του *SRILM*. Για παράδειγμα, για να υπολογίσουμε το *perplexity* του *test* κειμένου με βάση το γλωσσικό μοντέλο *lm.3gram.GT111727.close.vocab-gt19-64K-noEnglish* χρησιμοποιήσαμε την εντολή:

```
ngram -lm lm.3gram.GT111727.close.vocab-gt19-64K-noEnglish -ppl test.txt
```

Στη συνέχεια ενσωματώσαμε τα μοντέλα στον αναγνωριστή της *Nuance* και μετρήσαμε το *word error rate* τους. Τα αποτελέσματα δίνονται στην παράγραφο *§6.6*.

Κεφάλαιο 6

Αποτελέσματα και σχολιασμός

6.1 Βασικό *Backoff Bigram* μοντέλο

Όπως αναφέρθηκε στην παράγραφο *§5.1,* υλοποιήσαμε δύο word-bigram μοντέλα «ανοιχτού» λεξικού μεγέθους 92000 και 63000 λέξεων αντίστοιχα. Τα μοντέλα αυτά χρησιμοποιήθηκαν για τον υπολογισμό του perplexity του test κειμένου. Τα αποτελέσματα φαίνονται στον παρακάτω πίνακα. Είναι φανερό ότι όσο μειώνεται το μέγεθος του λεξικού μειώνεται και το perplexity του κειμένου.

Μοντέλο	***Perplexity στο test κείμενο (PPL)***	***Out Of Vocab words (OOV)***
lm.2gram.GT1117.open.vocab-gt10-92K	*253.53*	*0 (0%)*
lm.2gram.GT1117.open.vocab-gt20-63K	*227.937*	*0 (0%)*

Πίνακας 6.1

Επεξηγήσεις για τον πίνακα 6.1:

lm.2gram.GT1117.open.vocab-gt10-92K: word-bigram μοντέλο «ανοιχτού» λεξικού μεγέθους 92000 λέξεων και Good-Turing discount στα bigrams με counts από 1 εώς 7.

lm.2gram.GT1117.open.vocab-gt20-63K: word-bigram μοντέλο «ανοιχτού» λεξικού μεγέθους 63000 λέξεων και Good-Turing discount στα bigrams με counts από 1 εώς 7.

Υλοποιήσαμε, ακόμη, δύο μοντέλα «κλειστού» λεξικού μεγέθους 64000 και 40000 λέξεων αντίστοιχα, τα οποία χρησιμοποιήθηκαν για τον υπολογισμό του perplexity του test κειμένου. Επίσης, τα μοντέλα αυτά χρησιμοποιήθηκαν σαν γραμματική στον αναγνωριστή της Nuance, για τον υπολογισμό του Word-Error-Rate (WER) ενός συνόλου ηχογραφημένων προτάσεων από 10 διαφορετικούς ομιλητές. Τα αποτελέσματα συνοψίζονται στον παρακάτω πίνακα.

Μοντέλο	***Perplexity στο test κείμενο (PPL)***	***Word Error Rate (WER)***	***Out Of Vocabulary words (OOV) στο test***
lm.2gram.GT1117.close.vocab-gt19-64K-noEnglish	*242.402*	*22.43% (8.2647xcpuRT)*	*3,7%*
lm.2gram.GT1117.close.vocab-gt19-40K-noEnglish	*218.764*	*25.94% (6.15xcpuRT)*	*5.1%*

Πίνακας 6.2

Επεξηγήσεις για τον πίνακα 6.2:

lm.2gram.GT1117.close.vocab-gt19-64K-noEnglish: word-bigram μοντέλο «κλειστού» λεξικού μεγέθους 64000 λέξεων (χωρίς αγγλικές λέξεις) και Good-Turing discount στα bigrams με *counts* από 1 εώς 7.

lm.2gram.GT1117.close.vocab-gt19-40K-noEnglish: word-bigram μοντέλο «κλειστού» λεξικού μεγέθους 40000 λέξεων (χωρίς αγγλικές λέξεις) και Good-Turing discount στα bigrams με *counts* από 1 εώς 7.

6.2 *Class-based bigram* γλωσσικά μοντέλα με χρησιμοποίηση *stemming*

Με τη βοήθεια του αλγορίθμου *stemming* για τα ελληνικά, κατασκευάσαμε τα δύο *class-based* γλωσσικά μοντέλα που περιγράφηκαν στην *παράγραφο §5.3*. Και με τα δύο μοντέλα υπολογίσαμε το *perplexity* του *test* κειμένου, ενώ στον αναγνωριστή χρησιμοποιήσαμε μόνο το *root-based* μοντέλο, για τον υπολογισμό του *WER*. Ο λόγος που χρησιμοποιήσαμε μόνο αυτό το μοντέλο στον αναγνωριστή, είναι η πολύ υψηλή τιμή του perplexity του suffix-based μοντέλου, η οποία δεν μας επέτρεπε να περιμένουμε κάποιο ικανοποιητικό αποτέλεσμα στην μέτρηση του WER. Ο πίνακας 6.3 συνοψίζει τα αποτελέσματα για τα δύο αυτά μοντέλα.

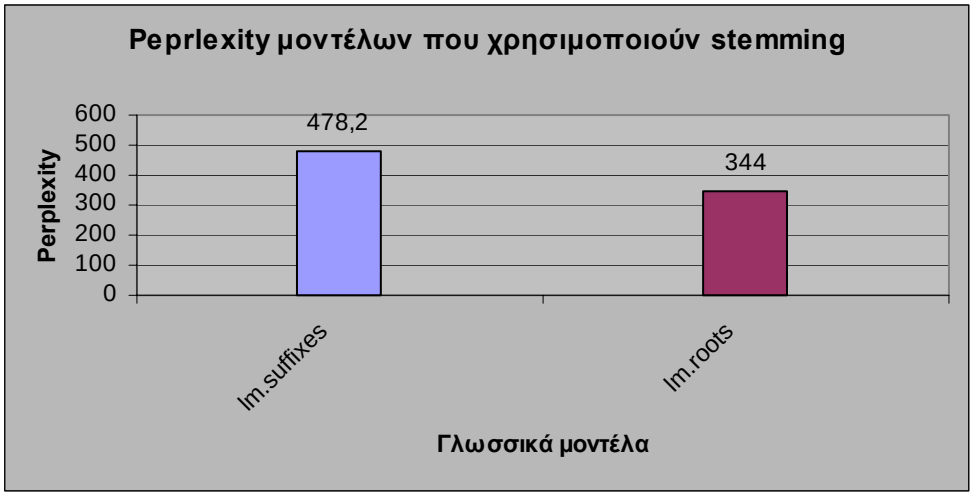
Μοντέλο	***Perplexity (PPL)***	***Word Error Rate (WER)***	***Out Of Vocab words (OOV)***
lm.suffixes	*478.231*	*---*	*3.7%*
lm.roots	*344.453*	*25.99%*	*3.7%*

Πίνακας 6.3

Επεξηγήσεις για τον πίνακα 6.3:

lm.suffixes: class-bigram μοντέλο 175 κλάσεων-καταλήξεων. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 175 κλάσεις ήταν 64000.
lm.roots: class-bigram μοντέλο 30754 κλάσεων-ριζών. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 30754 κλάσεις ήταν 64000.

Στην γραφική παράσταση που ακολουθεί φαίνεται και σχηματικά η διαφορά στο perplexity του test κειμένου με τη χρήση των δύο μοντέλων.



Γραφική παράσταση 6.1

Παρατηρούμε ότι η διαφορά στο perplexity των δύο stemmed μοντέλων είναι σημαντική. Αυτό οφείλεται στην μεγάλη διαφορά του αριθμού των κλάσεων, στις οποίες ομαδοποιήθηκαν οι λέξεις του λεξικού, στα δύο μοντέλα (175 και 30754 κλάσεις).

6.3 *Class-based bigram* γλωσσικό μοντέλο με χρησιμοποίηση *POS*

Στην παράγραφο αυτή αναλύονται τα αποτελέσματα του POS-based γλωσσικού μοντέλου που περιγράφηκε στην παράγραφο §5.4. Χρησιμοποιήσαμε το μοντέλο για την μέτρηση του perplexity στο test κείμενο, αλλά όχι για να εξάγουμε αποτέλεσμα αναγνώρισης, για τον ίδιο λόγο που περιγράψαμε στην προηγούμενη παράγραφο. Δηλαδή, λόγω της πολύ υψηλής τιμής του perplexity του μοντέλου στο test κείμενο.

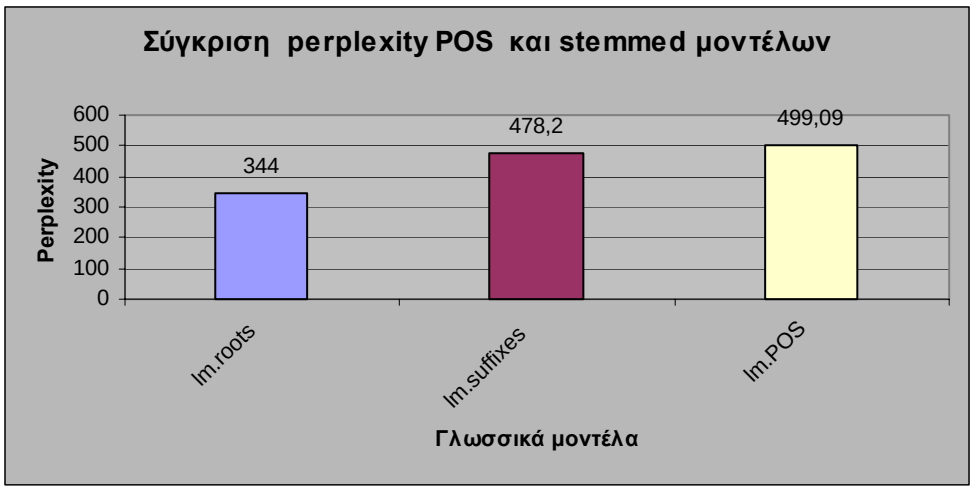
Μοντέλο	***Perplexity (PPL)***	***Word Error Rate (WER)***	***Out Of Vocab words (OOV)***
lm.POS	*499.098*	---	*3.7%*

Πίνακας 6.4

Επεξηγήσεις για τον πίνακα 6.4:

lm.POS: class-bigram μοντέλο 172 part-of-speech κλάσεων . Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 172 κλάσεις ήταν 64000.

Στη γραφική παράσταση που ακολουθεί έχουμε την σύγκριση των perplexities του συγκεκριμένου μοντέλου και των stemmed μοντέλων της προηγούμενης παραγράφου.



Γραφική παράσταση 6.2

6.4 Class-based γλωσσικό μοντέλο με συνδυασμό stemming και POS

Ένα μοντέλο, το οποίο χρησιμοποιήσαμε στον υπολογισμό τόσο του perplexity του test κειμένου όσο και στον υπολογισμό του WER, ήταν το class-based μοντέλο με συνδυασμό POS και stemming. Δηλαδή, το μοντέλο, στο οποίο η ομαδοποίηση σε κλάσεις προκύπτει από το συνδυασμό της ρίζας με το *POS* της λέξης, και το οποίο περιγράψαμε στην παράγραφο *§5.5*. Οι μετρήσεις για το συγκεκριμένο μοντέλο φαίνονται στον πίνακα που ακολουθεί.

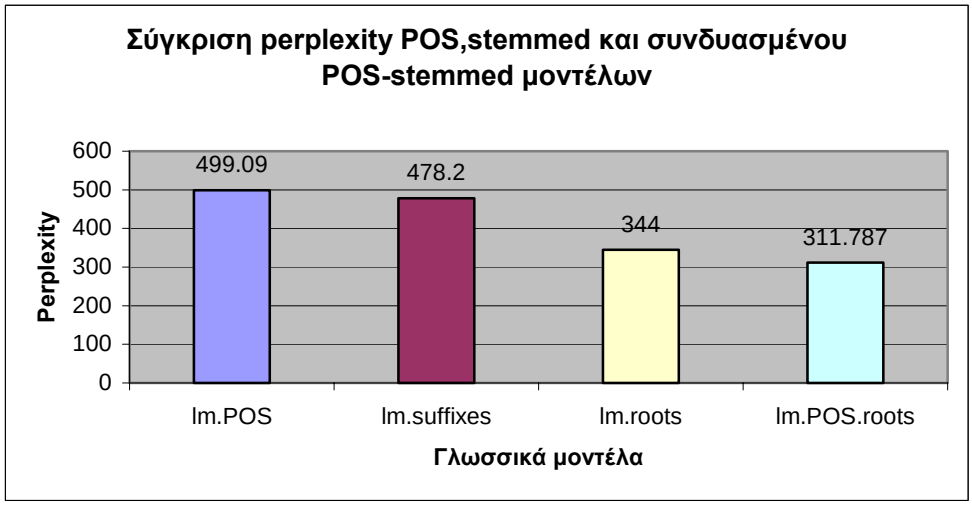
Μοντέλο	*Perplexity (PPL)*	*Word Error Rate (WER)*	*Out Of Vocab words (OOV)*
Lm.POS_roots_GT 1127	*311.787*	*25.86%*	*3.7%*

Πίνακας 6.5

Επεξηγήσεις για τον πίνακα 6.5:

lm.POS_roots_GT 1127: class-bigram μοντέλο 35412 POS_root-κλάσεων και Good-Turing discount στα bigrams με counts από 1 εώς 7. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 35412 κλάσεις ήταν 64000.

Ακολουθεί μία συγκριτική, ως προς το perplexity, γραφική παραστάση όλων των class-based μοντέλων που έχουμε περιγράψει μέχρι αυτήν την παράγραφο.



Γραφική παράσταση 6.3

Στις δύο γραφικές παραστάσεις παρατηρούμε ότι καταφέραμε να μειώσουμε μια τιμή του perplexity που προσεγγίζει το 500 (στα μοντέλα lm.POS και lm.suffixes), σε μια τιμή περίπου 300 (311,787 στο συνδυασμένο POS.root μοντέλο). Η διαφορά που παρατηρείται στο perplexity των μοντέλων lm.roots και lm.POS.roots οφείλεται στο ότι ο αριθμός των κλάσεων στο μοντέλο lm.POS.roots είναι μεγαλύτερος από τον αντίστοιχο αριθμό του μοντέλου lm.roots.

6.5 Υβριδικά γλωσσικά μοντέλα με συνδυασμό κλάσεων και λέξεων

Στους δύο πίνακες που ακολουθούν συνοψίζονται τα αποτελέσματα που εξάγαμε από την χρήση των υβριδικών γλωσσικών μοντέλων, που περιγράφονται στην παράγραφο *§5.6,* τόσο στην μέτρηση του perplexity του test κειμένου όσο και στην μέτρηση του WER του αναγνωριστή.

Υβριδικό Μοντέλο λεξικού 64000 λέξεων	***Perplexity (PPL)***	***Word Error Rate (WER)***	***Out Of Vocab words (OOV)***
lm.hybrid_POS_roots.35K.64	*242.832*	*22.47%*	*3.7%*
lm.hybrid_POS_roots.30K.64	*243.432*	*22.49%*	*3.7%*
lm.hybrid_POS_roots.25K.64	*244.643*	*22.62%*	*3.7%*
lm.hybrid_POS_roots.20K.64	*245.719*	*22.57%*	*3.7%*
lm.hybrid_POS_roots.15K.64	*247.986*	*22.58%*	*3.7%*

Πίνακας 6.6

Επεξηγήσεις για τον πίνακα 6.6:

lm.hybrid_POS_roots.25K.64: class-bigram μοντέλο 25431 POS_root-κλάσεων και 25124 λέξεων με Good-Turing discount στα bigrams με counts από 1 εώς 7. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 25431 κλάσεις ήταν 38876.

lm.hybrid_POS_roots.15K.64: class-bigram μοντέλο 30752 POS_root-κλάσεων και 15007 λέξεων με Good-Turing discount στα bigrams με counts από 1 εώς 7. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 30752 κλάσεις ήταν 48993.

lm.hybrid_POS_roots.20K.64: class-bigram μοντέλο 28607 POS_root-κλάσεων και 20092 λέξεων με Good-Turing discount στα bigrams με counts από 1 έως 7. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 28607 κλάσεις ήταν 43908.

lm.hybrid_POS_roots.30K.64: class-bigram μοντέλο 23467 POS_root-κλάσεων και 30418 λέξεων με Good-Turing discount στα bigrams με counts από 1 εώς 7. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 23467 κλάσεις ήταν 33582.

lm.hybrid_POS_roots.35K.64: class-bigram μοντέλο 20479 POS_root-κλάσεων και 35559 λέξεων με Good-Turing discount στα bigrams με counts από 1 έως 7. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 20479 κλάσεις ήταν 28441.

Υβριδικό Μοντέλο λεξικού 40000 λέξεων	***Perplexity (PPL)***	***Word Error Rate (WER)***	***Out Of Vocab words (OOV)***
lm.hybrid_POS_roots.10K.40	*225.252*	*25.01%*	*5.1%*
lm.hybrid_POS_roots.17K.40	*221.372*	*24.77%*	*5.1%*
lm.hybrid_POS_roots.20K.40	*220.461*	*24.68%*	*5.1%*
lm.hybrid_POS_roots.23K.40	*219.873*	*24.67%*	*5.1%*

Πίνακας 6.7

Επεξηγήσεις για τον πίνακα 6.7:

lm.hybrid_POS_roots.10K.40: class-bigram μοντέλο 20061 POS_root-κλάσεων και 10474 λέξεων με Good-Turing discount στα bigrams με counts από 1 έως 7. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 20061 κλάσεις ήταν 29526.
lm.hybrid_POS_roots.17K.40: class-bigram μοντέλο 16734 POS_root-κλάσεων και 17420 λέξεων με Good-Turing discount στα bigrams με counts από 1 έως 7. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 16734 κλάσεις ήταν 22580.

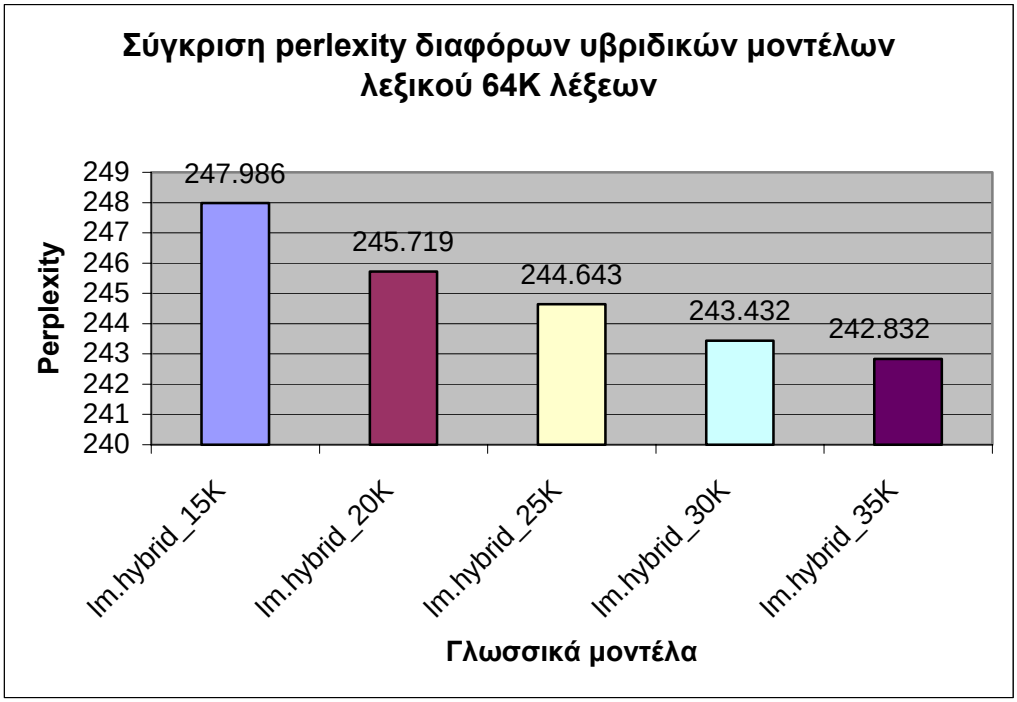
lm.hybrid_POS_roots.20K.40: class-bigram μοντέλο 15039 POS_root-κλάσεων και 20446 λέξεων με Good-Turing discount στα bigrams με counts από 1 έως 7. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 15039 κλάσεις ήταν 19554.

lm.hybrid_POS_roots.23K.40: class-bigram μοντέλο 13321 POS_root-κλάσεων και 23230 λέξεων με Good-Turing discount στα bigrams με counts από 1 έως 7. Οι συνολικές λέξεις που ομαδοποιήθηκαν στις 13321 κλάσεις ήταν 16770.

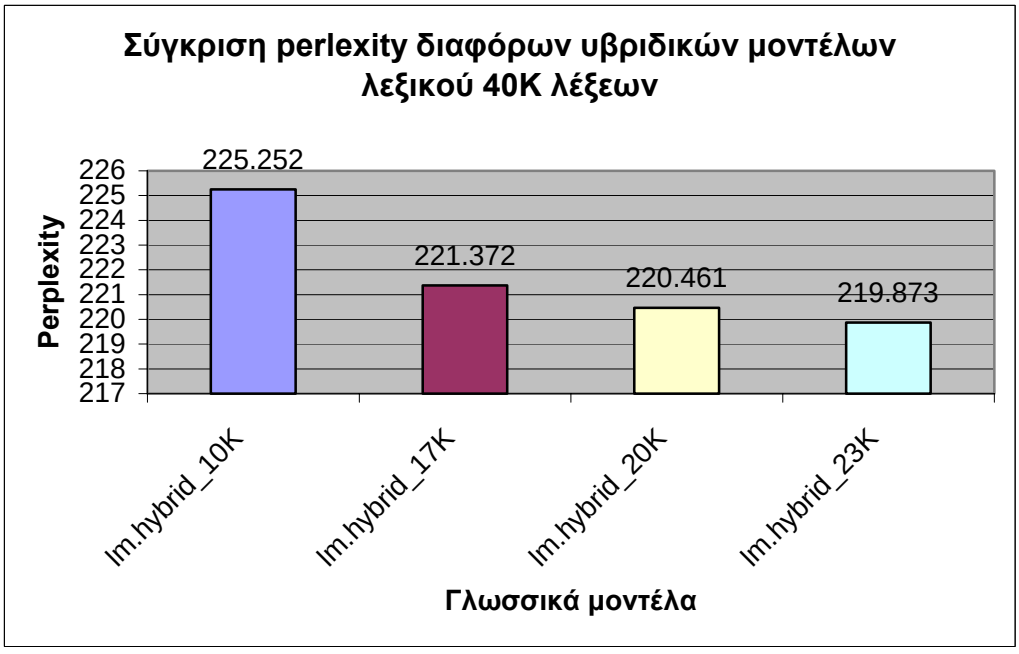
Παρατηρούμε ότι τα αποτελέσματα τόσο στις τιμές του perplexity όσο και στις τιμές του WER είναι αισθητά βελτιωμένα σε σχέση με τις μετρήσεις των

προηγούμενων μοντέλων, ενώ σε κάποιες περιπτώσεις προσεγγίζουν τις αντίστοιχες τιμές του baseline bigram μοντέλου.

Στις δύο γραφικές παραστάσεις που ακολουθούν βλέπουμε σχηματικά την διαφορά στις τιμές του perplexity των διαφόρων περιπτώσεων υβριδικών μοντέλων ,για λεξικό 64000 και 40000 λέξεων αντίστοιχα.



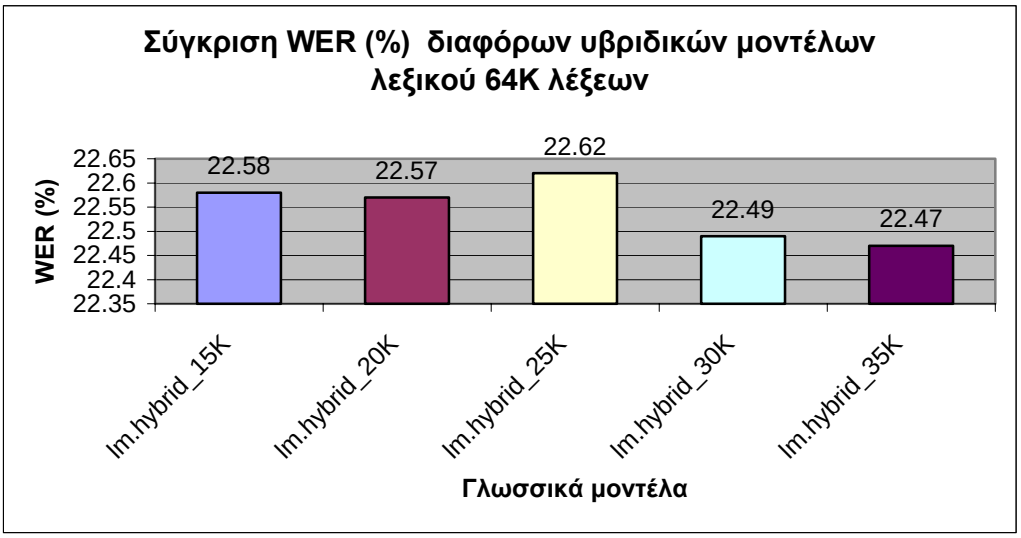
Γραφική παράσταση 6.4



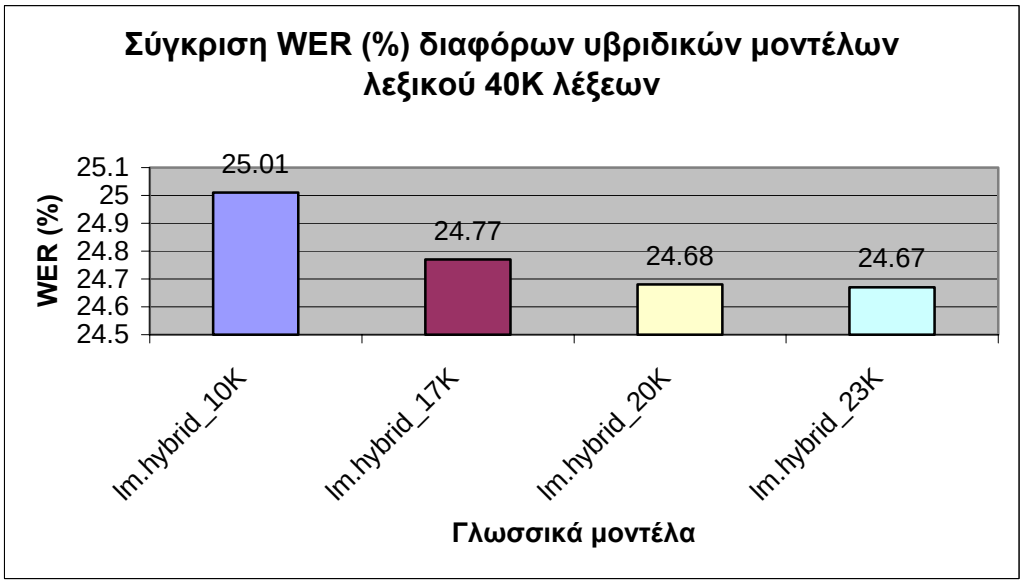
Γραφική παράσταση 6.5

Παρατηρούμε ότι στα υβριδικά μοντέλα και των δύο περιπτώσεων, όσο αυξάνεται ο αριθμός των λέξεων και μειώνεται αντίστοιχα ο αριθμός των κλάσεων στο λεξικό, μειώνεται ταυτόχρονα και το perplexity των μοντέλων.

Κάτι αντίστοιχο μπορούμε να παρατηρήσουμε και στις δύο επόμενες γραφικές παραστάσεις, που αφορούν τις τιμές του WER του αναγνωριστή για τα διάφορα μοντέλα.

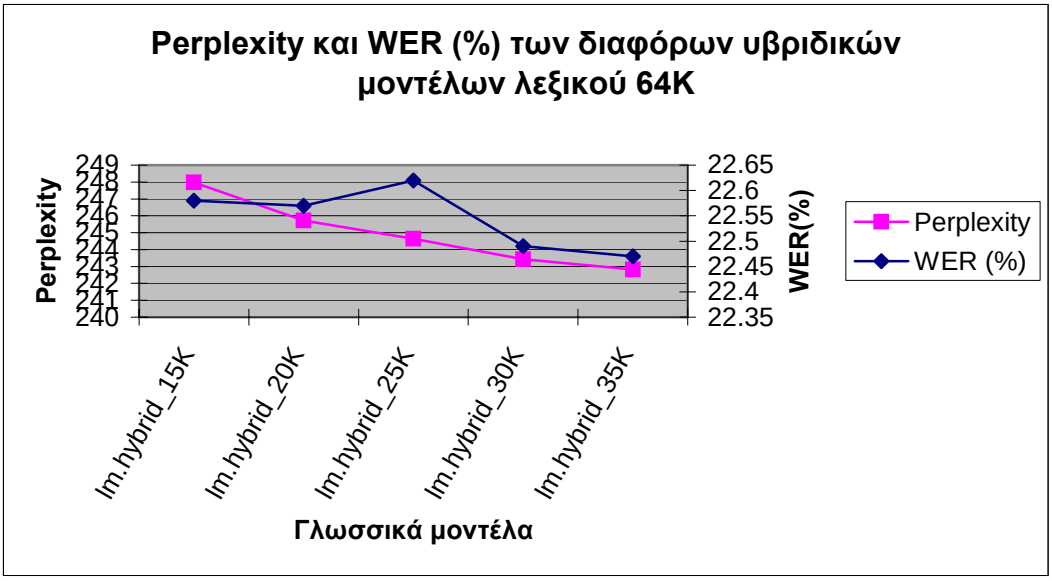


Γραφική παράσταση 6.6

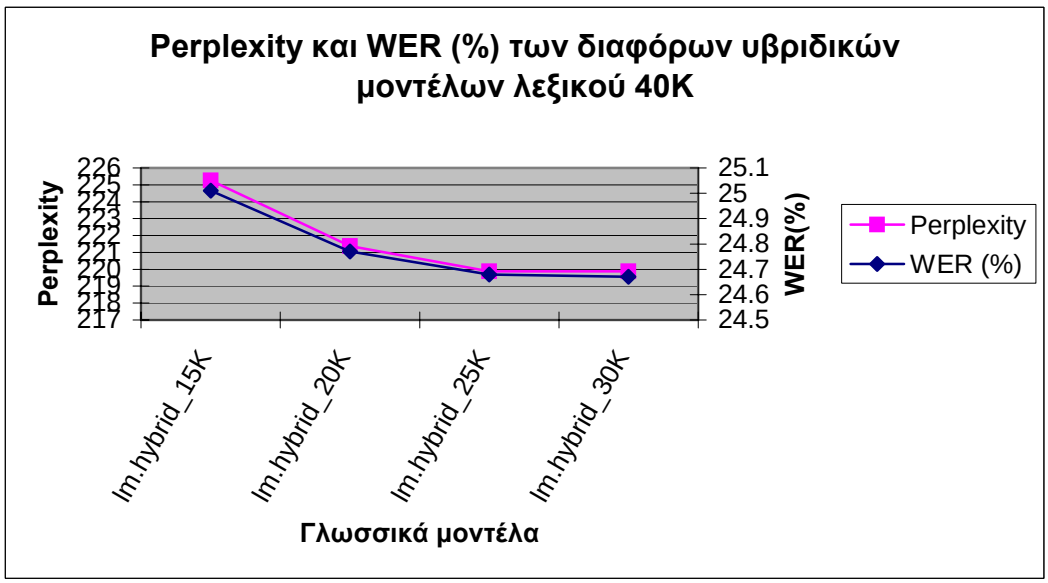


Γραφική παράσταση 6.7

Παρατηρούμε ότι, γενικά, τόσο στην περίπτωση του 64K λεξικού όσο και στην περίπτωση του 40K λεξικού, όσο αυξάνουμε τον αριθμό των λέξεων και μειώνουμε παράλληλα τον αριθμό των κλάσεων στο λεξικό των μοντέλων, μειώνεται και το WER. Η μόνη εξαίρεση παρατηρείται στην τιμή του WER για το μοντέλο lm.hybrid_25K (64K λεξικό), η οποία είναι ελάχιστα μεγαλύτερη από την αντίστοιχη τιμή για το μοντέλο lm.hybrid_20K.



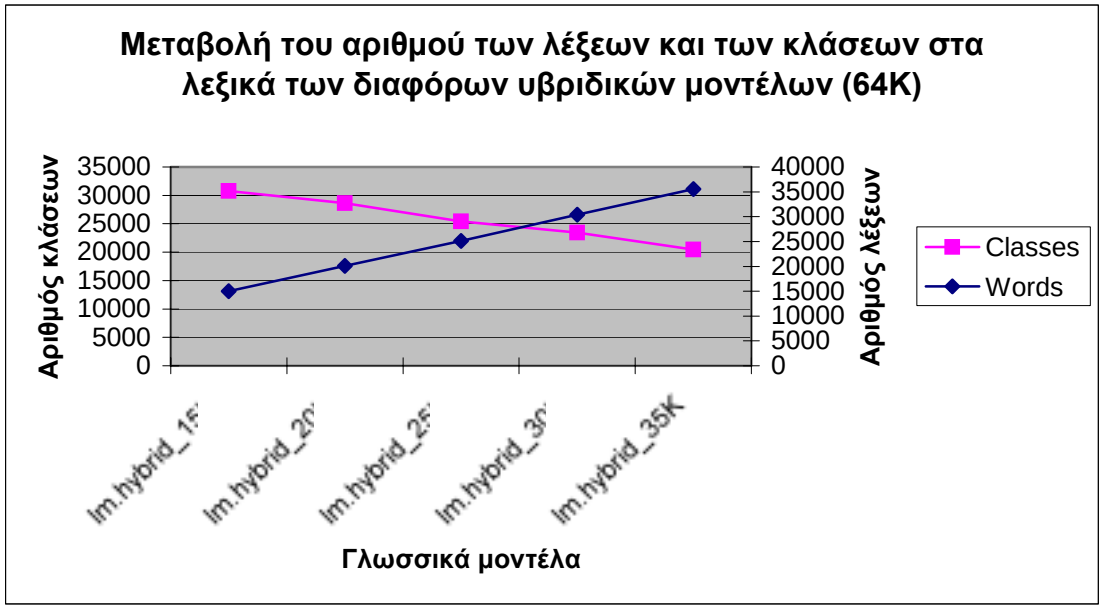
Γραφική παράσταση 6.8



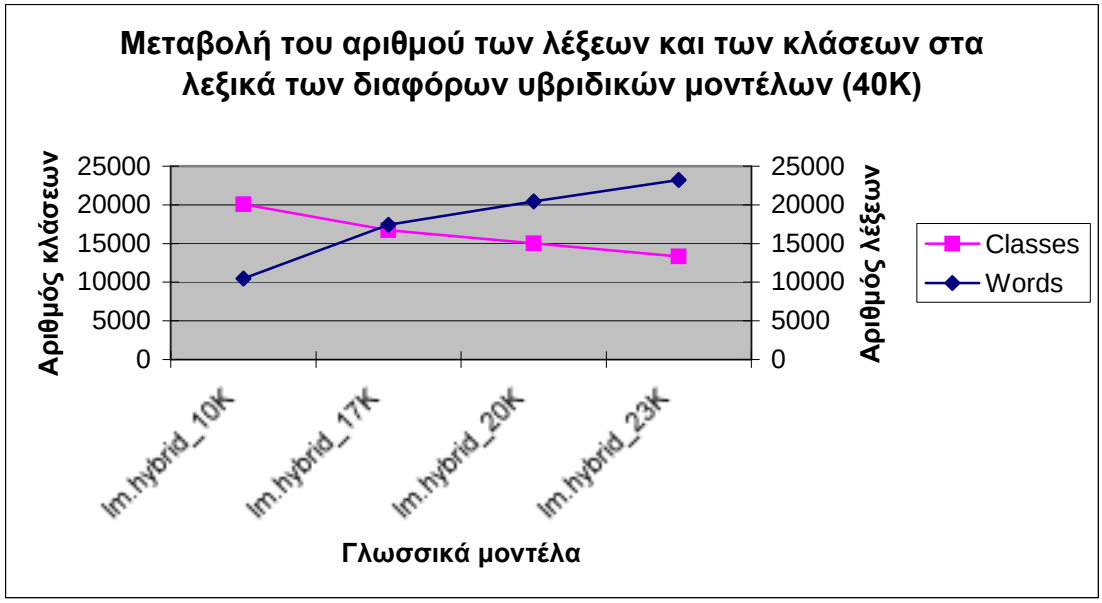
Γραφική παράσταση 6.9

Αυτό που μπορεί να εξαχθεί σαν συμπέρασμα από τις δύο παραπάνω γραφικές παραστάσεις είναι ότι, με εξαίρεση την περίπτωση του μοντέλου lm.hybrid_25K στην περίπτωση του 64K λεξικού, η μείωση του perplexity των μοντέλων συνεπάγεται και μείωση του WER.

Τέλος, ακολουθούν δύο γραφικές παραστάσεις που δείχνουν την μεταβολή του αριθμού των κλάσεων στο λεξικό, σε συνάρτηση με την μεταβολή του αριθμού των λέξεων στο αντίστοιχο λεξικό (για λεξικά 64K και 40K).



Γραφική παράσταση 6.10



Γραφική παράσταση 6.11

6.6 Γλωσσικά μοντέλα μεγαλύτερης τάξης

Όλα τα προηγούμενα αποτελέσματα αφορούσαν γλωσσικά μοντέλα με τάξη 2 (bigram γλωσσικά μοντέλα). Εκτός, όμως, από αυτά, όπως περιγράψαμε στην παράγραφο *§5.7,* υλοποιήσαμε και γλωσσικά μοντέλα μεγαλύτερης τάξης και συγκεκριμένα τάξης 3. Το ένα από τα μοντέλα αυτά ήταν «ανοιχτού» λεξικού μεγέθους 64000 λέξεων και τα αποτελέσματα που εξάγαμε γι' αυτό φαίνονται στον παρακάτω πίνακα.

Μοντέλο	***Perplexity (PPL)***	***Word Error Rate (WER)***	***Out Of Vocab words (OOV)***
lm.3gram.GT111727.open.vocab-gt19-64K-noEnglish	*163.235*	---	*0%*

Πίνακας 6.8

Επεξηγήσεις για τον πίνακα 6.8:

lm.3gram.GT111727.open.vocab-gt19-64K-noEnglish: word-trigram μοντέλο «ανοιχτού» λεξικού μεγέθους 64000 λέξεων και Good-Turing discount στα bigrams με counts από 1 εώς 7 και στα trigrams με counts από 2 εώς 7. Δηλαδή τα trigrams με count 1 απορρίπτονται.

Τα υπόλοιπα trigram μοντέλα που υλοποιήσαμε ήταν «κλειστού» λεξικού, με μέγεθος διαφορετικό σε κάθε περίπτωση. Τα μεγέθη των λεξικών που χρησιμοποιήσαμε ήταν 64K,40K,35K,30K,25K λέξεις αντίστοιχα, και τα αποτελέσματα, τόσο σε perplexity όσο και σε WER συνοψίζονται στον *πίνακα 6.9*.

Μοντέλο	***Perplexity (PPL)***	***Word Error Rate (WER)***	***Out Of Vocab words (OOV)***
lm.3gram.GT111737.close.vocab-gt19-64K-noEnglish	*181.498*	*21.00% (6.9343xcpuRT)*	*3.55%*
lm.3gram.GT111727.close.vocab-gt19-64K-noEnglish	*174.938*	*21.01% (6.1605xcpuRT)*	*3.55%*
lm.3gram.GT111727.close.vocab-gt40-40K-noEnglish	*158.604*	*23.23% (5.3794xcpuRT)*	*5.38%*
lm.3gram.GT111727.close.vocab-gt50-35K-noEnglish	*153.187*	*23.83% (5.0909xcpuRT)*	*6.08%*
lm.3gram.GT111727.close.vocab-gt65-30K-noEnglish	*146.759*	*25.13% (4.7984xcpuRT)*	*7.02%*
lm.3gram.GT111727.close.vocab-gt85-25K-noEnglish	*140.05*	*26.78% (4.4603xcpuRT)*	*8.10%*

Πίνακας 6.9

Επεξηγήσεις για τον πίνακα 6.9:

lm.3gram.GT111737.close.vocab-gt19-64K-noEnglish: word-trigram μοντέλο «κλειστού» λεξικού μεγέθους 64000 λέξεων και Good-Turing discount στα bigrams με counts από 1 έως 7 και στα trigrams με counts από 3 έως 7. Τα trigrams με count 1 ή 2 απορρίπτονται.

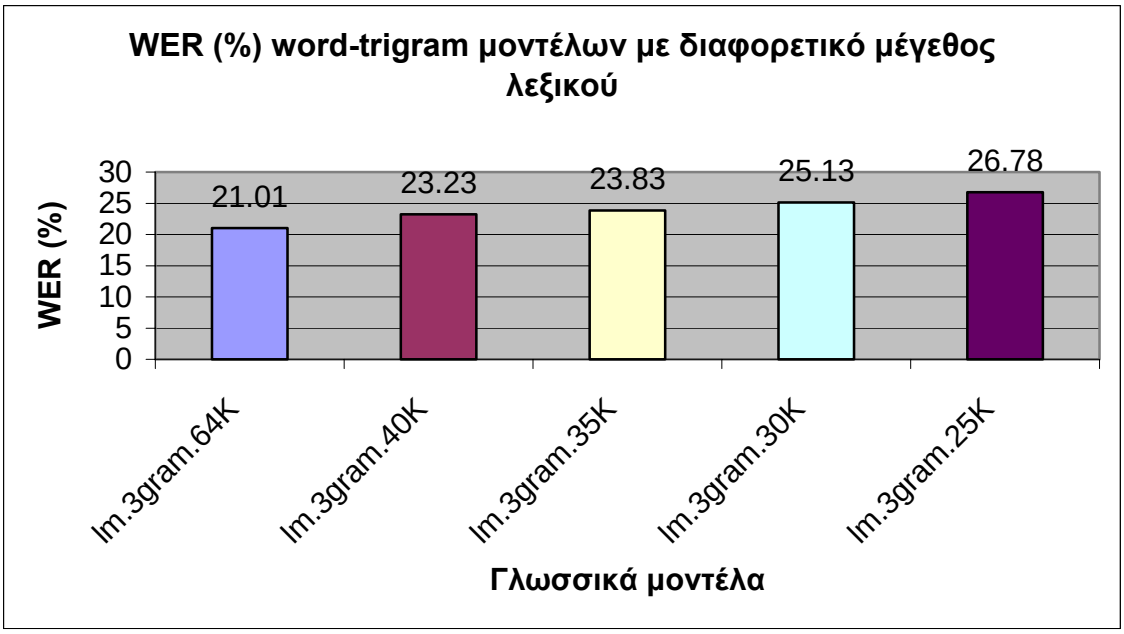
lm.3gram.GT111727.close.vocab-gt19-64K-noEnglish: word-trigram μοντέλο «κλειστού» λεξικού μεγέθους 64000 λέξεων και Good-Turing discount στα bigrams με counts από 1 έως 7 και στα trigrams με counts από 2 έως 7. Δηλαδή τα trigrams με count 1 απορρίπτονται.

lm.3gram.GT111727.close.vocab-gt19-40K-noEnglish: word-trigram μοντέλο «κλειστού» λεξικού μεγέθους 40000 λέξεων και Good-Turing discount στα bigrams με counts από 1 έως 7 και στα trigrams με counts από 2 έως 7. Τα trigrams με count 1 απορρίπτονται.

lm.3gram.GT111727.close.vocab-gt19-30K-noEnglish: word-trigram μοντέλο «κλειστού» λεξικού μεγέθους 30000 λέξεων και Good-Turing discount στα bigrams με counts από 1 έως 7 και στα trigrams με counts από 2 έως 7. Τα trigrams με count 1 απορρίπτονται.

lm.3gram.GT111727.close.vocab-gt19-25K-noEnglish: word-trigram μοντέλο «κλειστού» λεξικού μεγέθους 25000 λέξεων και Good-Turing discount στα bigrams με counts από 1 έως 7 και στα trigrams με counts από 2 έως 7. Τα trigrams με count 1 απορρίπτονται.

Αυτό που μπορούμε να παρατηρήσουμε από τον πίνακα 6.9 είναι ότι, όσο το μέγεθος του λεξικού των trigram μοντέλων μειώνεται, μειώνεται ταυτόχρονα και το perplexity των μοντέλων και αυξάνεται το WER. Επίσης, όσο μειώνουμε το μέγεθος του λεξικού, αυξάνεται, φυσικά, και το ποσοστό των OOV λέξεων, γεγονός που εξηγεί και την αύξηση στο WER.



Γραφική παράσταση 6.12

Κεφάλαιο 7

Συμπεράσματα - Μελλοντικές επεκτάσεις

7.1 Συμπεράσματα

Στη διπλωματική αυτή εργασία μελετήσαμε διάφορες περιπτώσεις *n-gram backoff* γλωσσικών μοντέλων βασισμένων σε λέξεις, κλάσεις ή το συνδυασμό τους. Παρατηρώντας τα αποτελέσματα που πήραμε από τις μετρικές απόδοσης (*WER* και *perplexity)* για τα μοντέλα αυτά, μπορούμε να συμπεράνουμε ότι για λεξικά όπου η σποραδικότητα των δεδομένων είναι μικρή, όπως αυτό των 64000 λέξεων, το *word-bigram* μοντέλο δίνει καλύτερα αποτελέσματα, τόσο στο *perplexity*, όσο και στο *WER*. Τα *class-based* γλωσσικά μοντέλα έδωσαν μεγαλύτερο *perplexity*, πράγμα αναμενόμενο, καθώς αποτελούν πιο γενικευμένα μοντέλα, αλλά και σε επίπεδο αναγνώρισης η απόδοσή τους δεν ήταν ικανοποιητική σε σχέση με τα αντίστοιχα *word-bigram* μοντέλα. Η απόδοση του αλγορίθμου *stemming* που αναπτύχθηκε για την ομαδοποίηση των λέξεων σε κλάσεις κρίνεται ικανοποιητική, καθώς παρατηρήθηκε ότι η αύξηση των κλάσεων επιφέρει καλύτερα αποτελέσματα στο *WER*. Έτσι, δεν ήταν αναγκαία μια περαιτέρω ομαδοποίηση των λέξεων στη ρίζα τους. Ο συνδυασμός δε *stemming* και *POS-tagging* έδωσε ακόμα καλύτερα αποτελέσματα. Τα υβριδικά μοντέλα κατάφεραν να προσεγγίσουν την απόδοση του *word-bigram* μοντέλου, χωρίς ωστόσο να την ξεπεράσουν. Τα γλωσσικά μοντέλα μεγαλύτερης τάξης έδωσαν καλύτερο *word-error rate* σε σχέση με το *word-bigram* μοντέλο, παρ' όλα αυτά ο αποθηκευτικός χώρος που αυτά απαιτούν εξακολουθεί να αποτελεί κριτήριο για τη χρησιμοποίησή τους ή μη.

7.2 Μελλοντικές επεκτάσεις

Σε αυτή τη διπλωματική εργασία έγινε μια πρώτη προσέγγιση για την κατασκευή γλωσσικών μοντέλων για τα ελληνικά. Ως μελλοντικές επεκτάσεις θα μπορούσαμε να προτείνουμε τις παρακάτω:

- Εμπλουτισμός του περιεχομένου των κλάσεων συνδυασμένου *part-of-speech* και *stemming* με λέξεις εκτός λεξικού, με σκοπό τη μείωση του *OOV rate*.

- Κατασκευή γλωσσικών μοντέλων βασισμένων στο κριτήριο μέγιστης εντροπίας

- Χρησιμοποίηση των γλωσσικών μοντέλων σε άλλες εφαρμογές, π.χ. *text parsing*, *tagging* κλπ.

- Χρησιμοποίηση των ελληνικών *stemmer* και *POS-tagger* σε εφαρμογές ανάκτησης πληροφορίας, π.χ. σε μηχανές αναζήτησης, βάσεις δεδομένων, κλπ.

Παράρτημα Α

Στατιστικά στοιχεία κειμένων

Ως κείμενα εκπαίδευσης και δοκιμής χρησιμοποιήθηκαν άρθρα της εφημερίδας «Ελευθεροτυπία» από το 1995 έως το 2000. Από το κείμενο που προεπεξεργάστηκε, το 90% χρησιμοποιήθηκε σαν κείμενο εκπαίδευσης *(training text)*, ενώ το υπόλοιπο 10% σαν κείμενο δοκιμής *(test text)*. Οι προτάσεις του κειμένου που χρησιμοποιήθηκαν ως κείμενο δοκιμής προέκυψαν αναθέτοντας κάθε δέκατη πρόταση του συνολικού κειμένου στο κείμενο δοκιμής, ενώ οι υπόλοιπες παρέμειναν ως κείμενο εκπαίδευσης. Στον επόμενο πίνακα φαίνονται περιληπτικά τα στατιστικά στοιχεία του κειμένου.

Στατιστικά κειμένου	**Συνολικό κείμενο**	**Κείμενο εκπαίδευσης**	**Κείμενο δοκιμής**
Αριθμός προτάσεων	1741637	1567474	174163
Αριθμός λέξεων	38130785	34304880	3825905
Μέγεθος λεξικού	445246	425525	159294

Πίνακας Α.1

Παράρτημα Β

Προεπεξεργασία κειμένων

Για να βελτιωθεί η ποιότητα των γλωσσικών μοντέλων χρειάστηκε μια προεπεξεργασία των κειμένων, καθώς η αρχική μορφή τους, δηλαδή αυτούσια τα άρθρα της «Ελευθεροτυπίας», περιείχαν αρκετό «θόρυβο». Έτσι, για να φέρουμε τα κείμενα σε μορφή που να μπορούμε να εκμεταλλευτούμε εφαρμόσαμε πάνω τους τις ακόλουθες αλλαγές:

- **Επικεφαλίδες:** Οι επικεφαλίδες των άρθρων αφαιρέθηκαν από τα κείμενα, καθώς, γενικά, δεν περιέχουν γραμματικά σωστές προτάσεις.

- **Άχρηστο κείμενο:** Αποτελέσματα αγώνων, ΠΡΟ-ΠΟ, λαχείων, καθώς και ορισμοί σταυρολέξων, σκάκι κλπ. αφαιρέθηκαν από τα κείμενα.

- **Χωρισμός σε προτάσεις:** Τα κείμενα χωρίστηκαν σε προτάσεις, έτσι ώστε να υπάρχει μία πρόταση ανά γραμμή. Μία πρόταση μπορεί να τελειώνει σε τελεία, ερωτηματικό, θαυμαστικό ή άνω κάτω τελεία.

- **Συντομογραφίες:** Γράφτηκαν ολογράφως οι περισσότερες συντομογραφίες των κειμένων και αφαιρέθηκαν τα μικρά ονόματα ανθρώπων που υποδηλώνονταν με κεφαλαίο γράμμα, για παράδειγμα :

  «Σ. Κόκκαλης» → *«Κόκκαλης»*
  «Δρ.» → *«διδάκτωρ»*

- **Αριθμοί:** Οι αριθμοί που βρέθηκαν στα κείμενα γράφτηκαν ολογράφως, π.χ. το *«349076»* μετατράπηκε σε *«τριακόσιες σαράντα εννιά χιλιάδες εβδομήντα έξι»*.

- **Ημερομηνίες:** Οι γραμμένες με αριθμό ημερομηνίες που βρέθηκαν στα κείμενα μετατράπηκαν σε κείμενο ως εξής:

  «12/4/96» → *«δώδεκα τετάρτου ενενήντα έξι»*

- **Τόνοι:** Λάθος τονισμένες λέξεις ή λέξεις με διπλούς τόνους μετατράπηκαν ώστε να είναι γραμμένες με το σωστό τονισμό, π.χ.:

  «απόλυτός» → *«απόλυτος»*
  «κατώ» → *«κάτω»*

- **Κεφαλαία:** Όλα τα κεφαλαία γράμματα μετατράπηκαν σε μικρά.

- **Στίξη:** Τέλος, αφαιρέθηκε κάθε σημείο στίξης εκτός από τους τόνους και τις αποστρόφους.

Οι παραπάνω αλλαγές υλοποιήθηκαν με *Perl scripts*, τα κυριότερα εκ των οποίων παρατίθενται στο *Παράρτημα Ζ*.

ΠΑΡΑΡΤΗΜΑ Γ

Κώδικας Stemmer

Ο κώδικας του ελληνικού *stemmer* που αναπτύξαμε γράφτηκε σε *C++* και δίνεται παρακάτω:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <locale.h>

#define FALSE  0
#define TRUE   1
#define EOS    '\0'

#define IsVowel(c)  ('α'==(c) || 'ε'==(c) || 'ι'==(c) || 'η'==(c) ||
'ο'==(c) || 'υ'==(c) || 'ω'==(c) || 'ά'==(c) || 'έ'==(c) || 'ί'==(c)
|| 'ή'==(c) || 'ό'==(c) || 'ύ'==(c) || 'ώ'==(c) || 'ϊ'==(c) ||
'ϋ'==(c) || 'ΐ'==(c) || 'ΰ'==(c))



typedef struct
{
      int id;
      char *old_end;
      char *new_end;
      int old_offset;
      int new_offset;
      int min_root_size;
      int (*condition)();
} RuleList;


static char LAMBDA[1]="";
static char *end;

#ifdef __STDC__

static int WordSize(char *word);
static int ContainsVowel(char *word);
static int ReplaceEnd(char *word, RuleList *rule);

#else

static int WordSize();
```

```
static int ContainsVowel();
static int ReplaceEnd();

#endif


static RuleList step1_rules[] =
          {
                    101, "όμασταν", LAMBDA, 6, -1, 0, NULL,
                    102, "όσασταν", LAMBDA, 6, -1, 0, NULL,
                      103, "ούμαστε", LAMBDA, 6, -1, 0, NULL,
                      104, "ούσαστε", LAMBDA, 6, -1, 0, NULL,
                      105, "μένους", LAMBDA, 5, -1, 0, NULL,
                      106, "όμαστε", LAMBDA, 5, -1, 0, NULL,
                      107, "όσαστε", LAMBDA, 5, -1, 0, NULL,
                      108, "ούσαμε", LAMBDA, 5, -1, 0, NULL,
                      109, "ούσατε", LAMBDA, 5, -1, 0, NULL,
                      110, "ούσανε", LAMBDA, 5, -1, 0, NULL,
                      111, "ήσουμε", LAMBDA, 5, -1, 0, NULL,
                      112, "ούνται", LAMBDA, 5, -1, 0, NULL,
                      113, "ούνταν", LAMBDA, 5, -1, 0, NULL,
                      114, "ούσουν", LAMBDA, 5, -1, 0, NULL,
                      115, "ούμουν", LAMBDA, 5, -1, 0, NULL,
                      116, "ούδες", LAMBDA, 4, -1, 0, NULL,
                      117, "ούδων", LAMBDA, 4, -1, 0, NULL,
                      118, "οντας", LAMBDA, 4, -1, 0, NULL,
                      119, "μένος", LAMBDA, 4, -1, -1, NULL,
                      120, "μένου", LAMBDA, 4, -1, -1, NULL,
                      121, "μένοι", LAMBDA, 4, -1, -1, NULL,
                      122, "μένων", LAMBDA, 4, -1, -1, NULL,
                      123, "μένης", LAMBDA, 4, -1, -1, NULL,
                      124, "μένες", LAMBDA, 4, -1, -1, NULL,
                      125, "ονται", LAMBDA, 4, -1, 0, NULL,
                      126, "όμουν", LAMBDA, 4, -1, 0, NULL,
                      127, "όσουν", LAMBDA, 4, -1, 0, NULL,
                      128, "ονταν", LAMBDA, 4, -1, 0, NULL,
                      129, "ήκαμε", LAMBDA, 4, -1, 0, NULL,
                      130, "ήκατε", LAMBDA, 4, -1, 0, NULL,
                      131, "ώντας", LAMBDA, 4, -1, 0, NULL,
                      132, "ούσες", LAMBDA, 4, -1, 0, NULL,
                      133, "ούσαν", LAMBDA, 4, -1, 0, NULL,
                      134, "ήσαμε", LAMBDA, 4, -1, 0, NULL,
                      135, "ήσατε", LAMBDA, 4, -1, 0, NULL,
                      136, "ήσεις", LAMBDA, 4, -1, 0, NULL,
                      137, "ήσομε", LAMBDA, 4, -1, 0, NULL,
                      138, "ήσετε", LAMBDA, 4, -1, 0, NULL,
                      139, "ήσουν", LAMBDA, 4, -1, 0, NULL,
                      140, "όνταν", LAMBDA, 4, -1, 0, NULL,
                      141, "ούμαι", LAMBDA, 4, -1, 0, NULL,
                      142, "είσαι", LAMBDA, 4, -1, 0, NULL,
                      143, "είται", LAMBDA, 4, -1, 0, NULL,
                      144, "είστε", LAMBDA, 4, -1, 0, NULL,
                      145, "ώνται", LAMBDA, 4, -1, 0, NULL,
                      146, "άδες", LAMBDA, 3, -1, 0, NULL,
                      147, "άδων", LAMBDA, 3, -1, 0, NULL,
                      148, "ηδες", LAMBDA, 3, -1, 0, NULL,
                      149, "ήδες", LAMBDA, 3, -1, 0, NULL,
                      150, "ήδων", LAMBDA, 3, -1, 0, NULL,
                      151, "ηδων", LAMBDA, 3, -1, 0, NULL,
                      152, "έδες", LAMBDA, 3, -1, 0, NULL,
                      153, "έδων", LAMBDA, 3, -1, 0, NULL,
```

```
154, "ατος", LAMBDA, 3, -1, 0, NULL,
155, "άτων", LAMBDA, 3, -1, 0, NULL,
156, "ώτων", LAMBDA, 3, -1, 0, NULL,
157, "ότων", LAMBDA, 3, -1, 0, NULL,
158, "ιούς", LAMBDA, 3, -1, 0, NULL,
159, "έους", LAMBDA, 3, -1, 0, NULL,
160, "ουμε", LAMBDA, 3, -1, 0, NULL,
161, "ουνε", LAMBDA, 3, -1, 0, NULL,
162, "είτε", LAMBDA, 3, -1, 0, NULL,
163, "μένο", LAMBDA, 3, -1, -1, NULL,
164, "μένη", LAMBDA, 3, -1, -1, NULL,
165, "μένα", LAMBDA, 3, -1, -1, NULL,
166, "ομαι", LAMBDA, 3, -1, 0, NULL,
167, "εσαι", LAMBDA, 3, -1, 0, NULL,
168, "εται", LAMBDA, 3, -1, 0, NULL,
169, "όταν", LAMBDA, 3, -1, 0, NULL,
170, "ηκες", LAMBDA, 3, -1, 0, NULL,
171, "ηκαν", LAMBDA, 3, -1, 0, NULL,
172, "ήκες", LAMBDA, 3, -1, 0, NULL,
173, "ήκαν", LAMBDA, 3, -1, 0, NULL,
174, "ούμε", LAMBDA, 3, -1, 0, NULL,
175, "είτε", LAMBDA, 3, -1, 0, NULL,
176, "ούσα", LAMBDA, 3, -1, 0, NULL,
177, "ούσε", LAMBDA, 3, -1, 0, NULL,
178, "ησαν", LAMBDA, 3, -1, 0, NULL,
179, "ήσει", LAMBDA, 3, -1, 0, NULL,
180, "ήστε", LAMBDA, 3, -1, 0, NULL,
181, "έμαι", LAMBDA, 3, -1, 0, NULL,
182, "έσαι", LAMBDA, 3, -1, 0, NULL,
183, "έται", LAMBDA, 3, -1, 0, NULL,
184, "έστε", LAMBDA, 3, -1, 0, NULL,
185, "άσαι", LAMBDA, 3, -1, 0, NULL,
186, "άται", LAMBDA, 3, -1, 0, NULL,
187, "ήσου", LAMBDA, 3, -1, 0, NULL,
188, "άστε", LAMBDA, 3, -1, 0, NULL,
189, "άμαι", LAMBDA, 3, -1, 0, NULL,
190, "ώμαι", LAMBDA, 3, -1, 0, NULL,
191, "άσθε", LAMBDA, 3, -1, 0, NULL,
192, "ώτος", LAMBDA, 3, -1, 0, NULL,
193, "ότος", LAMBDA, 3, -1, 0, NULL,
194, "ησες", LAMBDA, 3, -1, 0, NULL,
195, "ούς", LAMBDA, 2, -1, 0, NULL,
196, "ους", LAMBDA, 2, -1, 0, NULL,
197, "εως", LAMBDA, 2, -1, 0, NULL,
198, "εις", LAMBDA, 2, -1, 0, NULL,
199, "είς", LAMBDA, 2, -1, 0, NULL,
200, "εων", LAMBDA, 2, -1, 0, NULL,
201, "ιού", LAMBDA, 2, -1, 0, NULL,
202, "ιών", LAMBDA, 2, -1, 0, NULL,
203, "ατα", LAMBDA, 2, -1, 0, NULL,
204, "ιάς", LAMBDA, 2, -1, 0, NULL,
205, "ιές", LAMBDA, 2, -1, 0, NULL,
206, "ιών", LAMBDA, 2, -1, 0, NULL,
207, "ιοί", LAMBDA, 2, -1, 0, NULL,
208, "έος", LAMBDA, 2, -1, 0, NULL,
209, "έου", LAMBDA, 2, -1, 0, NULL,
210, "έοι", LAMBDA, 2, -1, 0, NULL,
211, "έων", LAMBDA, 2, -1, 0, NULL,
212, "ετε", LAMBDA, 2, -1, 0, NULL,
213, "ουν", LAMBDA, 2, -1, 0, NULL,
214, "ομε", LAMBDA, 2, -1, 0, NULL,
```

```
215, "αμε", LAMBDA, 2, -1, 0, NULL,
216, "ατε", LAMBDA, 2, -1, 0, NULL,
217, "άμε", LAMBDA, 2, -1, 0, NULL,
218, "ηκα", LAMBDA, 2, -1, 0, NULL,
219, "ηκε", LAMBDA, 2, -1, 0, NULL,
220, "ήκα", LAMBDA, 2, -1, 0, NULL,
221, "ήκε", LAMBDA, 2, -1, 0, NULL,
222, "ούν", LAMBDA, 2, -1, 0, NULL,
223, "άτε", LAMBDA, 2, -1, 0, NULL,
224, "ησα", LAMBDA, 2, -1, 0, NULL,
225, "ησε", LAMBDA, 2, -1, 0, NULL,
226, "ήσω", LAMBDA, 2, -1, 0, NULL,
227, "άνε", LAMBDA, 2, -1, 0, NULL,
228, "ειν", LAMBDA, 2, -1, 0, NULL,
229, "είν", LAMBDA, 2, -1, 0, NULL,
230, "ας", LAMBDA, 1, -1, 0, NULL,
231, "ων", LAMBDA, 1, -1, 0, NULL,
232, "ες", LAMBDA, 1, -1, 0, NULL,
233, "ών", LAMBDA, 1, -1, 0, NULL,
234, "ης", LAMBDA, 1, -1, 0, NULL,
235, "ής", LAMBDA, 1, -1, 0, NULL,
236, "ές", LAMBDA, 1, -1, 0, NULL,
237, "ών", LAMBDA, 1, -1, 0, NULL,
238, "άς", LAMBDA, 1, -1, 0, NULL,
239, "ού", LAMBDA, 1, -1, 0, NULL,
240, "ος", LAMBDA, 1, -1, 0, NULL,
241, "ός", LAMBDA, 1, -1, 0, NULL,
242, "ου", LAMBDA, 1, -1, 0, NULL,
243, "οί", LAMBDA, 1, -1, 0, NULL,
244, "οι", LAMBDA, 1, -1, 0, NULL,
245, "ον", LAMBDA, 1, -1, 0, NULL,
246, "ώς", LAMBDA, 1, -1, 0, NULL,
247, "ως", LAMBDA, 1, -1, 0, NULL,
248, "ια", LAMBDA, 1, -1, 0, NULL,
249, "ιά", LAMBDA, 1, -1, 0, NULL,
250, "έα", LAMBDA, 1, -1, 0, NULL,
251, "ύς", LAMBDA, 1, -1, 0, NULL,
252, "όν", LAMBDA, 1, -1, 0, NULL,
253, "ήν", LAMBDA, 1, -1, 0, NULL,
254, "ην", LAMBDA, 1, -1, 0, NULL,
255, "έο", LAMBDA, 1, -1, 0, NULL,
256, "ει", LAMBDA, 1, -1, 0, NULL,
257, "αν", LAMBDA, 1, -1, 0, NULL,
258, "τε", LAMBDA, 1, -1, 0, NULL,
259, "εί", LAMBDA, 1, -1, 0, NULL,
260, "με", LAMBDA, 1, -1, 0, NULL,
261, "νε", LAMBDA, 1, -1, 0, NULL,
262, "αι", LAMBDA, 1, -1, 0, NULL,
263, "α", LAMBDA, 0, -1, 0, NULL,
264, "η", LAMBDA, 0, -1, 0, NULL,
265, "ή", LAMBDA, 0, -1, 0, NULL,
266, "ά", LAMBDA, 0, -1, 0, NULL,
267, "έ", LAMBDA, 0, -1, 0, NULL,
268, "ο", LAMBDA, 0, -1, 0, NULL,
269, "ε", LAMBDA, 0, -1, 0, NULL,
270, "ώ", LAMBDA, 0, -1, 0, NULL,
271, "ω", LAMBDA, 0, -1, 0, NULL,
272, "ό", LAMBDA, 0, -1, 0, NULL,
273, "ί", LAMBDA, 0, -1, 0, NULL,
274, "ι", LAMBDA, 0, -1, 0, NULL,
275, "υ", LAMBDA, 0, -1, 0, NULL,
```

```
                    276, "ύ", LAMBDA, 0, -1, 0, NULL,
                    000, NULL,   NULL,   0,  0, 0, NULL,
                };


static int
WordSize(word)
  char *word;
{
      register int result;
      register int state;

      result=0;
      state=0;


      while (EOS!=*word)
      {
        switch (state)
        {
        case 0: state=(IsVowel(*word)) ? 1 : 2;
                  break;
            case 1: state=(IsVowel(*word)) ? 1 : 2;
                if (2==state) result++;
                  break;
        case 2: state=(IsVowel(*word)) ? 1 : 2;
                  break;
        }
        word++;
      }

      return (result);
}



static int
ContainsVowel (word)
   char *word;
{
      if (EOS==*word)
            return(FALSE);
      else
          return(IsVowel(*word)  ||
(NULL!=strpbrk(word+1,"αεηιουωάέήίόύώϊϋΐΰ")));
}



static int
ReplaceEnd(word,rule)
   char *word;
   RuleList *rule;
   {
         register char *ending;
         char tmp_ch;

         while (0!=rule->id)
         {
               ending=end-rule->old_offset;
```

```
               if (word<=ending)
                     if (0==strcmp(ending,rule->old_end))
                     {
                           tmp_ch= *ending;
                           *ending= EOS;
                           if (rule->min_root_size<WordSize(word))
                                 if (!rule->condition || (*rule-
>condition)(word))
                                 {

                                       (void)strcat(word,rule-
>new_end);
                                       end=ending+rule->new_offset;
                                       break;
                                 }
                              *ending=tmp_ch;
                     }
                  rule++;
         }
      return (rule->id);
   }


int
Stem(word)
   char *word;
{
      int rule;



      for (end=word;*end!=EOS;end++)
            if (!isalpha(*end)) return(FALSE);
            else *end=tolower(*end);
      end--;


      (void)ReplaceEnd(word,step1_rules);

      return(TRUE);
}

int main()
{
      char *input,*in,*suf;
  int i;
if (setlocale(LC_ALL, "el")==NULL) printf("failed to set locale\n");



  while     (gets(input))
      {

       in=strdup(input);
       Stem(input);
      suf=in + strlen(input);
       printf("%s%s %s\n",input,suf,input);
            free(in);
      }

return(TRUE);}
```

Παράρτημα Δ

Μορφή γλωσσικών μοντέλων

Δ.1 *ngram-format (ARPA format)*

Είναι η μορφή των *ARPA backoff N-gram μοντέλων.*

\data
ngram 1=*n1*
nrgam 2=*n2*
…
ngram N=*nN*
\1-grams:
p w [bow]
…
\2-grams:
p w1 w2 [bow]
…
\N-grams
p w1 … nN
…
\end

Το *ARPA format* για *N-gram backoff μοντέλα* αρχίζει με μια επικεφαλίδα, η οποία αποτελείται από τη λέξη-κλειδί **\data\,** και συνεχίζει με την παράθεση του αριθμού των *Ngrams* κάθε τάξης. Στη συνέχεια, τα *Ngrams* παρατίθενται ένα σε κάθε

σειρά, ομαδοποιημένα σε τμήματα ανάλογα με το μήκος τους. Κάθε τέτοιο τμήμα αρχίζει με τη λέξη-κλειδί \N**-gram:,** όπου *N* είναι το μήκος των *Ngrams* που ακολουθούν. Κάθε γραμμή με *Ngrams* αρχίζει με τον δεκαδικό λογάριθμο της υπό συνθήκη πιθανότητας *p* του συγκεκριμένου *Ngram*, ακολουθούμενη από τις λέξεις *w1 ... wN,* από τις οποίες αποτελείται το *Ngram*. Οι λέξεις ακολουθούνται, προαιρετικά από τον δεκαδικό λογάριθμο του *backoff βάρους (backoff weight*) του συγκεκριμένου *Ngram*. Η λέξη-κλειδί **\end** ολοκληρώνει την μορφή του μοντέλου.

Τα *backoff weights* απαιτούνται μόνο για εκείνα τα *Ngrams* που σχηματίζουν ένα *πρόθεμα (prefix*) μακρύτερων *Ngrams* μέσα στο μοντέλο. Τα *Ngrams* της μεγαλύτερης τάξης στο μοντέλο δεν χρειάζονται *backoff βάρη.*

Αφού το log(0) (πλην άπειρο) δεν έχει κάποια συγκεκριμένη αναπαράσταση, τιμές σαν κι αυτό συμβολίζονται με ένα μεγάλο αρνητικό αριθμό (-99) μέσα στο μοντέλο.

Δ.2 ***pfsg-format***

Είναι η μορφή των αρχείων για τις *πιθανοτικές, πεπερασμένων καταστάσεων γραμματικές (probabilistic finite-state grammars-PFSG)* του αναγνωριστή.

name *name*
nodes *N w1 ... wN*
initial *i*
final *f*
transitions *T*
n1 n2 p
...

Οι *PFSGs* είναι μια μορφή ενός *αυτομάτου πεπερασμένων καταστάσεων (finite state automaton)* που χρησιμοποιείται από τον αναγνωριστή. Οι *PFSGs* επιτρέπουν λέξεις στους κόμβους και όχι στις ακμές Συγκεκριμένες μορφές γλωσσικών μοντέλων, κατασκευασμένα από το *SRILM*, μπορούν να μεταφραστούν κατευθείαν σε μορφή *ι* και να χρησιμοποιηθούν στον αναγνωριστή.

Κάθε *PFSG* έχει ένα όνομα *name*. Η γραμμή των *nodes* δίνει τον αριθμό των κόμβων στον γράφο καταστάσεων και ακολουθούν οι συμβολοσειρές των λέξεων που αντιστοιχούν σε κάθε node. Αν ο κόμβος αναπαριστά μια *κατηγορία (category)* που είναι *expansion* μιας άλλης *PFSG*, τότε το *όνομα (name*) αυτής της *PFSG* δίνεται σε αυτό το σημείο. Το σύμβολο **NULL** έχει ειδική σημασία και ορίζει τον αντίστοιχο κόμβο σαν μη εκπέμπων (*non-emitting*). Είναι, επίσης, βολικό να χρησιμοποιούμε μικρά γράμματα για τις λέξεις και κεφαλαία για τις κατηγορίες και τα ονόματα των *PFSG*.

Οι γραμμές **initial** και **final** καθορίζουν την αρχική και τελική κατάσταση μιας *PFSG*, αντίστοιχα. Η αρίθμηση των κόμβων αρχίζει από το μηδέν (0).

Η γραμμή των *μεταβάσεων (***transitions***)* δίνει τον αριθμό των μεταβάσεων μεταξύ των καταστάσεων. Ακολουθείται από πολλές γραμμές, καθεμιά από τις οποίες καθορίζει μία μετάβαση από μια αρχική κατάσταση *n1* σε μια κατάσταση *n2* με κόστος *p*. Το κόστος μετάβασης συνήθως μεταφράζεται ως 10000,5 φορές τον φυσικό αλγόριθμο μιας πιθανότητας και πρέπει να κανονικοποιηθεί ανάλογα.

Δ.3 ***classes-format***

Είναι η μορφή των αρχείων για τον καθορισμό των κλάσεων λέξεων.

class [p] word1 word2 ...

Πολλά προγράμματα που διαχειρίζονται κλάσεις λέξεων χρησιμοποιούν αυτή τη μορφή για τον καθορισμό των πιθανών expansions των κλάσεων και τις αντίστοιχες πιθανότητές τους. Κάθε expansion εμφανίζεται σε ξεχωριστή γραμμή, όπως φαίνεται παραπάνω, όπου το *class* είναι το όνομα μιας κλάσης, το p δίνει την πιθανότητα της expansion της κλάσης και τα *word1 word2 ...*καθορίζουν την συμβολοσειρά λέξεων στις οποίες η κλάση γίνεται expand. Αν το p παραλειφθεί θεωρείται 1. Το άθροισμα των πιθανοτήτων p πρέπει να αθροίζει στη μονάδα.

Παράρτημα Ε

Το γενικευμένο N-pos μοντέλο – Μέρος Α

Σ'αυτό το παράρτημα θα δείξουμε ότι το γενικευμένο N-pos μοντέλο που εισαγάγαμε στην παράγραφο *§4.1* εξασφαλίζει ότι το άθροισμα των πιθανοτήτων όλων των λέξεων είναι ίσο με το 1.

Όπως είδαμε στο *Κεφάλαιο 4,* η πιθανότητα της λέξης w_i δεδομένων των μεταβλητών $X_1,...,X_{r+s}$ είναι:

$$P(w_i \mid X_1,...,X_{r+s}) =$$

$$\sum_{g_j \in G} P(g(w_i) \mid X_1,...,X_r) \cdot P(w_i \mid g(w[i]) = g_j, X_{r+1},...,X_{r+s})$$

Στη συνέχεια θα χρησιμοποιήσουμε το $P_1(g_i \mid X_1^r)$ σαν συντομογραφία του $P(g(w_i) = g_i \mid X_1,...,X_s)$ και το $P_2(w_i \mid g_i, X_{r+1}^{r+s})$ σαν συντομογραφία του $P(w_i = w_l \mid g(w_i) = g_i, X_{r+1},...,X_{r+s})$.

Θεωρούμε ότι τα $P_1(g_i|X_r)$ και $P_2(w_l|g_i,X_{r+s})$ είναι κατανομές πιθανοτήτων για όλους τους συνδυασμούς των τιμών των μεταβλητών X_k, $1<=k<=r+s$ και των κλάσεων g_i, $1<=i<=t$. Με άλλα λόγια:

$$\sum_{i=1}^{i=t} P_1(gi \mid X_r) = 1$$

και

$$\sum_{l=1}^{l=m} P_2(w_l \mid g_i, X_{r+s}) = 1$$

Αυτή για παράδειγμα είναι η περίπτωση που τα $P_1(g_i|X_r)$ και $P_2(w_l|g_i,X_{r+s})$ δημιουργούνται από δεδομένα συχνοτήτων και είναι γενικά μια λογική παραδοχή.

Μπορούμε τώρα να δείξουμε ότι το άθροισμα S των πιθανοτήτων όλων των λέξεων ισούται με 1:

$$S = \sum_{w_l \in V} \sum_{g_i \in G} P_1(g_i \mid X_r) \cdot P_2(w_l \mid g_i, X_{r+s})$$

$$= \sum_{g_i \in G} P_1(g_i \mid X_r) \cdot \sum_{w_l \in V} P_2(w_l \mid g_i, X_{r+s})$$

$$= \sum_{g_i \in G} P_1(g_i \mid X_r)$$

$$= 1$$

Παράρτημα ΣΤ

Το γενικευμένο N-pos μοντέλο – Μέρος Β

Στο παράρτημα αυτό, θέλουμε να δείξουμε ότι το γενικευμένο *N-pos* μοντέλο ανάγεται στο *N-gram* μοντέλο. Για ευκολία στην αναφορά, επαναλαμβάνουμε την σχέση που περιγράφει το γενικευμένο μοντέλο:

$$P(w[i] = w_k \mid X_1,\ldots, X_m) =$$

$$\sum_{g_j \in G} P(g(w[i]) = g_j \mid X_1,\ldots, X_n) \cdot P(w[i] = w_k \mid g(w[i]) = g_j, X_{n+1},\ldots, X_{n+m})$$

Αν επιλέξουμε $n=N-1$, $X_{n+l}=X_l=w_{i-1}$, $l=1,\ldots,N-1$ παίρνουμε:

$$P(w[i] = w_k \mid X_1,\ldots, X_m) =$$

$$\sum_{g_j \in G} P(g(w[i]) = g_j \mid w[i-N+1:i-1]) \cdot P(w[i] = w_k \mid g(w[i]) = g_j, w[i-N+1:i-1]).$$

Εάν, επιπλέον, θεωρήσουμε ότι οι πιθανότητες έχουν υπολογιστεί από συχνότητες αριθμών εμφάνισης των λέξεων, οι οποίες υποδηλώνονται από τη συνάρτηση *f( )*, έχουμε:

$$P(w[i] = w_k \mid X_1,\ldots, X_m) =$$

$$= \sum_{g_j \in G} P(g(w[i]) = g_j \mid w[i-N+1:i-1]) \cdot P(w[i] = w_k \mid g(w[i]) = g_j, w[i-N+1:i-1]) =$$

$$= \sum_{g_j \in G} \frac{f(g(w[i]) = g_j, w[i-N+1:i-1])}{f(w[i-N+1:i-1])} \cdot \frac{f(w[i] = w_k, g(w[i]) = g_j, w[i-N+1:i-1])}{f(g(w[i]) = g_j, w[i-N+1:i-1])} =$$

$$= \sum_{g_j \in G} \frac{f(w[i] = w_k, g(w[i]) = g_j, w[i-N+1:i-1])}{f(w[i-N+1:i-1])} =$$

$$= \frac{f(w[i] = w_k, w[i-N+1:i-1])}{f(w[i-N+1:i-1])} =$$

$$= P(w[i] = w_k \mid w[i-N+1:i-1]).$$

Αφού οι αριθμοί εμφάνισης συνήθως *ομαλοποιούνται (smoothed)* πριν χρησιμοποιηθούν για την εκτίμηση πιθανοτήτων, το γενικευμένο N-pos μοντέλο δεν θα είναι ακριβώς ίδιο με το N-gram μοντέλο. Όμως, όπως δείχνουν και οι προηγούμενοι υπολογισμοί, θα είναι μια προσέγγιση του N-gram μοντέλου, που βασίζεται στις ίδιες εξαρτήσεις.

Παράρτημα Ζ

Ορισμοί POS κλάσεων

Όνομα κλάσης	*Περιγραφή*
ΑΑ	Αντωνυμία Αναφορική (Άκλιτη)
ΑΑΑΕΝΘ	Αντωνυμία Αναφορική Αιτιατική Ενικού Θηλυκό
ΑΑΓΕΝΘ	Αντωνυμία Αναφορική Γενική Ενικού Θηλυκό
ΑΑΓΕΝΟ	Αντωνυμία Αναφορική Γενική Ενικού Ουδέτερο
ΑΑΑΠΛΑ	Αντωνυμία Αναφορική Αιτιατική Πληθυντικού Αρσενικό
ΑΑΓΠΛΟ	Αντωνυμία Αναφορική Γενική Πληθυντικού Ουδέτερο
ΑΑΟ	Αντωνυμία Αόριστη (Άκλιτη)
ΑΑΟΑΕΝΘ	Αντωνυμία Αόριστη Αιτιατική Ενικού Θηλυκό
ΑΑΟΓΕΝΘ	Αντωνυμία Αόριστη Γενική Ενικού Θηλυκό
ΑΑΟΓΕΝΟ	Αντωνυμία Αόριστη Γενική Ενικού Ουδέτερο
ΑΑΟΑΠΛΑ	Αντωνυμία Αόριστη Αιτιατική Πληθυντικού Αρσενικό
ΑΑΟΓΠΛΟ	Αντωνυμία Αόριστη Γενική Πληθυντικού Ουδέτερο
ΑΑΟΕΝΑ	Αντωνυμία Αναφορική Ονομαστική Ενικού Αρσενικό
ΑΑΟΕΝΘ	Αντωνυμία Αναφορική Ονομαστική Ενικού Θηλυκό
ΑΑΟΕΝΟ	Αντωνυμία Αναφορική Ονομαστική Ενικού Ουδέτερο
ΑΑΟΠΛΑ	Αντωνυμία Αναφορική Ονομαστική Πληθυντικού Αρσενικό
ΑΑΟΠΛΘ	Αντωνυμία Αναφορική Ονομαστική Πληθυντικού Θηλυκό
ΑΑΟΟΕΝΑ	Αντωνυμία Αόριστη Ονομαστική Ενικού Αρσενικό
ΑΑΟΟΕΝΘ	Αντωνυμία Αόριστη Ονομαστική Ενικού Θηλυκό
ΑΑΟΟΕΝΟ	Αντωνυμία Αόριστη Ονομαστική Ενικού Ουδέτερο
ΑΑΟΟΠΛΑ	Αντωνυμία Αόριστη Ονομαστική Πληθυντικού Αρσενικό
ΑΑΟΟΠΛΘ	Αντωνυμία Αόριστη Ονομαστική Πληθυντικού Θηλυκό
ΑΔΑΕΝΑ	Αντωνυμία Δεικτική Αιτιατική Ενικού Αρσενικό
ΑΔΑΕΝΘ	Αντωνυμία Δεικτική Αιτιατική Ενικού Θηλυκό
ΑΔΓΕΝΘ	Αντωνυμία Δεικτική Γενική Ενικού Θηλυκό
ΑΔΓΕΝΟ	Αντωνυμία Δεικτική Γενική Ενικού Ουδέτερο
ΑΔΑΠΛΑ	Αντωνυμία Δεικτική Αιτιατική Πληθυντικού Αρσενικό
ΑΔΑΠΛΟ	Αντωνυμία Δεικτική Αιτιατική Πληθυντικού Ουδέτερο
ΑΔΓΠΛΟ	Αντωνυμία Δεικτική Γενική Πληθυντικού Ουδέτερο
ΑΔΟΕΝΑ	Αντωνυμία Δεικτική Ονομαστική Ενικού Αρσενικό
ΑΔΟΕΝΘ	Αντωνυμία Δεικτική Ονομαστική Ενικού Θηλυκό
ΑΔΟΕΝΟ	Αντωνυμία Δεικτική Ονομαστική Ενικού Ουδέτερο
ΑΔΟΠΛΑ	Αντωνυμία Δεικτική Ονομαστική Πληθυντικού Αρσενικό
ΑΔΟΠΛΘ	Αντωνυμία Δεικτική Ονομαστική Πληθυντικού Θηλυκό
ΑΕ	Αντωνυμία Ερωτηματική (Άκλιτη)
ΑΕΑΕΝΘ	Αντωνυμία Ερωτηματική Αιτιατική Ενικού Θηλυκό
ΑΕΓΕΝΟ	Αντωνυμία Ερωτηματική Γενική Ενικού Ουδέτερο
ΑΕΑΠΛΑ	Αντωνυμία Ερωτηματική Αιτιατική Πληθυντικού Αρσενικό
ΑΕΟΕΝΑ	Αντωνυμία Ερωτηματική Ονομαστική Ενικού Αρσενικό
ΑΕΟΕΝΟ	Αντωνυμία Ερωτηματική Ονομαστική Ενικού Ουδέτερο
ΑΕΟΠΛΑ	Αντωνυμία Ερωτηματική Ονομαστική Πληθυντικού Αρσενικό
ΑΕΟΠΛΘ	Αντωνυμία Ερωτηματική Ονομαστική Πληθυντικού Θηλυκό
ΑΟΓΕΝΘ	Αντωνυμία Οριστική Γενική Ενικού Θηλυκό
ΑΟΓΕΝΟ	Αντωνυμία Οριστική Γενική Ενικού Ουδέτερο
ΑΟΑΠΛΑ	Αντωνυμία Οριστική Αιτιατική Πληθυντικού Αρσενικό
ΑΟΓΠΛΟ	Αντωνυμία Οριστική Γενική Πληθυντικού Ουδέτερο
ΑΟΟΕΝΑ	Αντωνυμία Οριστική Ονομαστική Ενικού Αρσενικό

Όνομα κλάσης	**Περιγραφή**
ΑΟΟΕΝΘ	Αντωνυμία Οριστική Ονομαστική Ενικού Θηλυκό
ΑΟΟΕΝΟ	Αντωνυμία Οριστική Ονομαστική Ενικού Ουδέτερο
ΑΟΟΠΛΑ	Αντωνυμία Οριστική Ονομαστική Πληθυντικού Αρσενικό
ΑΟΟΠΛΘ	Αντωνυμία Οριστική Ονομαστική Πληθυντικού Θηλυκό
ΑΠΑΓΕΝ	Αντωνυμία Προσωπική Γενική Ενικού
ΑΠΔΓΕΝ	Αντωνυμία Προσωπική και Δεικτική Γενική Ενικού
ΑΠΔΑΕΝΑ	Αντωνυμία Προσωπική και Δεικ. Αιτιατική Ενικού Αρσενικό
ΑΠΔΑΕΝΘ	Αντωνυμία Προσωπική και Δεικτική Αιτιατική Ενικού Θηλυκό
ΑΠΔΓΕΝΘ	Αντωνυμία Προσωπική και Δεικτική Γενική Ενικού Θηλυκό
ΑΠΔΑΠΛΑ	Αντωνυμία Προσωπ. και Δεικ. Αιτιατ. Πληθυντικού Αρσενικό
ΑΠΔΟΕΝ	Αντωνυμία Προσωπική και Δεικτική Ονομαστική Ενικού
ΑΠΑΟΕΝΑ	Αντωνυμία Προσωπική Ονομαστική Ενικού Αρσενικό
ΑΠΔΟΕΝΑ	Αντωνυμία Προσωπική και Δεικ. Ονομαστική Ενικού Αρσενικό
ΑΠΑΟΕΝΘ	Αντωνυμία Προσωπική Ονομαστική Ενικού Θηλυκό
ΑΠΔΟΕΝΘ	Αντωνυμία Προσωπική και Δεικ. Ονομαστική Ενικού Θηλυκό
ΑΠΑΟΠΛΑ	Αντωνυμία Προσωπική Ονομαστική Πληθυντικού Αρσενικό
ΑΠΔΟΠΛΑ	Αντωνυμία Προσωπ. Και Δεικ. Ονομασ. Πληθυντικού Αρσενικό
ΑΠΑΟΠΛΘ	Αντωνυμία Προσωπική Ονομαστική Πληθυντικού Θηλυκό
ΑΠΔΟΠΛΘ	Αντωνυμία Προσωπ. και Δεικτ. Ονομαστ. Πληθυντικού Θηλυκό
ΕΑΕΝΑ	Επίθετο Αιτιατική Ενικού Αρσενικό
ΕΓΕΝΑ	Επίθετο Γενική Ενικού Αρσενικό
ΕΓΕΝΘ	Επίθετο Γενική Ενικού Θηλυκό
ΕΓΕΝΟ	Επίθετο Γενική Ενικού Ουδέτερο
ΕΑΠΛΑ	Επίθετο Αιτιατική Πληθυντικού Αρσενικό
ΕΓΠΛΑ	Επίθετο Γενική Πληθυντικού Αρσενικό
ΕΓΠΛΘ	Επίθετο Γενική Πληθυντικού Θηλυκό
ΕΓΠΛΟ	Επίθετο Γενική Πληθυντικού Ουδέτερο
ΕΟΕΝΑ	Επίθετο Ονομαστική Ενικού Αρσενικό
ΕΟΕΝΘ	Επίθετο Ονομαστική Ενικού Θηλυκό
ΕΟΕΝΟ	Επίθετο Ονομαστική Ενικού Ουδέτερο
ΕΟΠΛΑ	Επίθετο Ονομαστική Πληθυντικού Αρσενικό
ΕΟΠΛΘ	Επίθετο Ονομαστική Πληθυντικού Θηλυκό
ΕΟΠΛΟ	Επίθετο Ονομαστική Πληθυντικού Ουδέτερο
ΑΡΑΠΓΠΛΟ	Αριθμητικό Απόλυτο Γενική Πληθυντικού Ουδέτερο
ΑΡΑΠΟΠΛΘ	Αριθμητικό Απόλυτο Ονομαστική Πληθυντικού Θηλυκό
ΑΡΑΠΟΠΛΟ	Αριθμητικό Απόλυτο Ονομαστική Πληθυντικού Ουδέτερο
ΑΡΑΝΓΕΝΘ	Αριθμητικό Αναλογικό Γενική Ενικού Θηλυκό
ΑΡΑΝΓΕΝΟ	Αριθμητικό Αναλογικό Γενική Ενικού Ουδέτερο
ΑΡΑΝΓΠΛΟ	Αριθμητικό Αναλογικό Γενική Πληθυντικού Ουδέτερο
ΑΡΑΝΟΕΝΑ	Αριθμητικό Αναλογικό Ονομαστική Ενικού Αρσενικό
ΑΡΑΝΟΕΝΟ	Αριθμητικό Αναλογικό Ονομαστική Ενικού Ουδέτερο
ΑΡΑΝΟΠΛΑ	Αριθμητικό Αναλογικό Ονομαστική Πληθυντικού Αρσενικό
ΑΡΑΝΟΠΛΘ	Αριθμητικό Αναλογικό Ονομαστική Πληθυντικού Θηλυκό
ΑΡΑΝΟΠΛΟ	Αριθμητικό Αναλογικό Ονομαστική Πληθυντικού Ουδέτερο
ΑΡΠΓΕΝΘ	Αριθμητικό Περιληπτικό Γενική Ενικού Θηλυκό
ΑΡΠΓΠΛΘ	Αριθμητικό Περιληπτικό Γενική Πληθυντικού Θηλυκό
ΑΡΠΛΓΕΝΘ	Αριθμητικό Πολλαπλασιαστικό Γενική Ενικού Θηλυκό
ΑΡΠΛΓΕΝΟ	Αριθμητικό Πολλαπλασιαστικό Γενική Ενικού Ουδέτερο
ΑΡΠΛΓΠΛΟ	Αριθμητικό Πολλαπλασιαστικό Γενική Πληθυντικού Ουδέτερο
ΑΡΠΛΟΕΝΑ	Αριθμητικό Πολλαπλασιαστικό Ονομαστική Ενικού Αρσενικό
ΑΡΠΛΟΕΝΘ	Αριθμητικό Πολλαπλασιαστικό Ονομαστική Ενικού Θηλυκό
ΑΡΠΛΟΕΝΟ	Αριθμητικό Πολλαπλασιαστικό Ονομαστική Ενικού Ουδέτερο
ΑΡΠΛΟΠΛΑ	Αριθμητικό Πολλαπλασιαστ. Ονομαστ. Πληθυντικού Αρσενικό
ΑΡΠΛΟΠΛΘ	Αριθμητικό Πολλαπλασιαστ. Ονομαστική Πληθυντικού Θηλυκό
ΑΡΠΟΕΝΘ	Αριθμητικό Περιληπτικό Ονομαστική Ενικού Θηλυκό
ΑΡΠΟΠΛΘ	Αριθμητικό Περιληπτικό Ονομαστική Πληθυντικού Θηλυκό
ΑΡΘΑΓΕΝ	Άρθρο Αόριστο Γενική Ενικού
ΑΡΚΓΕΝΘ	Αριθμητικό Κλασματικό Γενική Ενικού Θηλυκό
ΑΡΘΑΟΕΝ	Άρθρο Αόριστο Ονομαστική Ενικού
ΑΡΘΟΑΕΝ	Άρθρο Οριστικό Αιτιατική Ενικού

Όνομα κλάσης	*Περιγραφή*
ΑΡΘΟΓΕΝ	Άρθρο Οριστικό Γενική Ενικού
ΑΡΘΟΑΠΛ	Άρθρο Οριστικό Αιτιατική Πληθυντικού
ΑΡΘΟΓΠΛ	Άρθρο Οριστικό Γενική Πληθυντικού
ΑΡΘΟΟΕΝ	Άρθρο Οριστικό Ονομαστική Ενικού
ΑΡΤΓΕΝΘ	Αριθμητικό Ταχτικό Γενική Ενικού Θηλυκό
ΑΡΤΓΕΝΟ	Αριθμητικό Ταχτικό Γενική Ενικού Ουδέτερο
ΑΡΤΓΠΛΟ	Αριθμητικό Ταχτικό Γενική Πληθυντικού Ουδέτερο
ΑΡΤΟΕΝΑ	Αριθμητικό Ταχτικό Ονομαστική Ενικού Αρσενικό
ΑΡΤΟΕΝΘ	Αριθμητικό Ταχτικό Ονομαστική Ενικού Θηλυκό
ΑΡΤΟΕΝΟ	Αριθμητικό Ταχτικό Ονομαστική Ενικού Ουδέτερο
ΑΡΤΟΠΛΑ	Αριθμητικό Ταχτικό Ονομαστική Πληθυντικού Αρσενικό
ΑΡΤΟΠΛΘ	Αριθμητικό Ταχτικό Ονομαστική Πληθυντικού Θηλυκό
ΑΡΤΟΠΛΟ	Αριθμητικό Ταχτικό Ονομαστική Πληθυντικού Ουδέτερο
ΠΑΡΣΓΕΝΑΟ	Παραθετικό Συγκριτικός Γενική Ενικού Αρσενικό Ουδέτερο
ΠΑΡΥΓΕΝΑΟ	Παραθετικό Υπερθετικός Γενική Ενικού Αρσενικό Ουδέτερο
ΠΑΡΣΓΕΝΘ	Παραθετικό Συγκριτικός Γενική Ενικού Θηλυκό
ΠΑΡΥΓΕΝΘ	Παραθετικό Υπερθετικός Γενική Ενικού Θηλυκό
ΠΑΡΣΑΠΛΑ	Παραθετικό Συγκριτικός Αιτιατική Πληθυντικού Αρσενικό
ΠΑΡΥΑΠΛΑ	Παραθετικό Υπερθετικός Αιτιατική Πληθυντικού Αρσενικό
ΠΑΡΣΓΠΛΑΟΘ	Παραθ. Συγκριτ. Γενική Πληθυντ. Αρσενικό Ουδέτερο Θηλυκό
ΠΑΡΥΓΠΛΑΟΘ	Παραθ. Υπερθετ. Γενική Πληθυντ. Αρσενικό Ουδέτερο Θηλυκό
ΠΑΡΣΑΟΕΝΑ0	Παραθ. Συγκριτ. Αιτ. και Ονομ. Ενικού Αρσενικό Ουδέτερο
ΠΑΡΥΑΟΕΝΑ0	Παραθ. Υπερθετ. Αιτ. και Ονομ. Ενικού Αρσενικό Ουδέτερο
ΠΑΡΣΑΟΠΛ0	Παραθ. Συγκριτ. Αιτ. και Ονομαστική Πληθυντικού Ουδέτερο
ΠΑΡΥΑΟΠΛ0	Παραθ. Υπερθετ. Αιτ. και Ονομαστική Πληθυντικού Ουδέτερο
ΠΑΡΣΟΑΠΛΘ	Παραθ. Συγκριτ. Αιτ. και Ονομαστική Πληθυντικού Θηλυκό
ΠΑΡΥΟΑΠΛΘ	Παραθ. Υπερθετ. Αιτ. και Ονομαστική Πληθυντικού Θηλυκό
ΠΑΡΣΟΕΝΑ	Παραθετικό Συγκριτικός Ονομαστική Ενικού Αρσενικό
ΠΑΡΥΟΕΝΑ	Παραθετικό Υπερθετικός Ονομαστική Ενικού Αρσενικό
ΠΑΡΣΟΕΝΘ	Παραθετικό Συγκριτικός Ονομαστική Ενικού Θηλυκό
ΠΑΡΥΟΕΝΘ	Παραθετικό Υπερθετικός Ονομαστική Ενικού Θηλυκό
ΠΑΡΣΟΠΛΑ	Παραθετικό Συγκριτικός Ονομαστική Πληθυντικού Αρσενικό
ΠΑΡΥΟΠΛΑ	Παραθετικό Υπερθετικός Ονομαστική Πληθυντικού Αρσενικό
ΟΓΕΝ	Ουσιαστικό Γενική Ενικού
ΟΓΠΛ	Ουσιαστικό Γενική Πληθυντικού
ΟΟΕΝ	Ουσιαστικό Ονομαστική Ενικού
ΟΟΠΛ	Ουσιαστικό Ονομαστική Πληθυντικού
ΜΤΧΕ	Μετοχή Ενεργητική
ΜΤΧΠ	Μετοχή Παθητική
ΑΓΝ	Άγνωστη
ΡΟΑΑΕΝ	Ρήμα Ονομαστική Αορίστου Πρώτο Ενικό Πρόσωπο
ΡΟΑΑΠΛ	Ρήμα Ονομαστική Αορίστου Πρώτο Πληθυντικό Πρόσωπο
ΡΟΑΒΕΝ	Ρήμα Ονομαστική Αορίστου Δεύτερο Ενικό Πρόσωπο
ΡΟΑΒΠΛ	Ρήμα Ονομαστική Αορίστου Δεύτερο Πληθυντικό Πρόσωπο
ΡΟΑΓΕΝ	Ρήμα Ονομαστική Αορίστου Τρίτο Ενικό Πρόσωπο
ΡΟΑΓΠΛ	Ρήμα Ονομαστική Αορίστου Τρίτο Πληθυντικό Πρόσωπο
ΡΟΠΑΕΝ	Ρήμα Ονομαστική Παρατατικού Πρώτο Ενικό Πρόσωπο
ΡΟΠΑΠΛ	Ρήμα Ονομαστική Παρατατικού Πρώτο Πληθυντικό Πρόσωπο
ΡΟΠΒΕΝ	Ρήμα Ονομαστική Παρατατικού Δεύτερο Ενικό Πρόσωπο
ΡΟΠΒΠΛ	Ρήμα Ονομαστική Παρατατικού Δεύτερο Πληθυντικό Πρόσωπο
ΡΟΠΓΕΝ	Ρήμα Ονομαστική Παρατατικού Τρίτο Ενικό Πρόσωπο
ΡΟΠΓΠΛ	Ρήμα Ονομαστική Παρατατικού Τρίτο Πληθυντικό Πρόσωπο
ΡΟΥΕΑΕΝ	Ρήμα Ονομαστ. και Υποτ. Ενεστώτα Πρώτο Ενικό Πρόσωπο
ΡΟΥΕΑΠΛ	Ρήμα Ονομαστ. και Υποτ. Ενεστ. Πρώτο Πληθυντικό Πρόσωπο
ΡΟΥΕΒΕΝ	Ρήμα Ονομαστ. και Υποτ. Ενεστώτα Δεύτερο Ενικό Πρόσωπο
ΡΟΥΕΒΠΛ	Ρήμα Ονομαστ. και Υποτ. Ενεστ. Δεύτερο Πληθυντ. Πρόσωπο
ΡΟΥΕΓΕΝ	Ρήμα Ονομαστ. και Υποτ. Ενεστώτα Τρίτο Ενικό Πρόσωπο
ΡΟΥΕΓΠΛ	Ρήμα Ονομαστ. και Υποτ. Ενεστ. Τρίτο Πληθυντικό Πρόσωπο
ΡΠΑΒΕΝ	Ρήμα Προστακτική Αορίστου Δεύτερο Ενικό Πρόσωπο
ΡΠΑΒΠΛ	Ρήμα Προστακτική Αορίστου Δεύτερο Πληθυντικό Πρόσωπο

Όνομα κλάσης	*Περιγραφή*
ΡΠΕΒΕΝ	Ρήμα Προστακτική Ενεστώτα Δεύτερο Ενικό Πρόσωπο
ΡΥΑΑΕΝ	Ρήμα Υποτακτική Αορίστου Πρώτο Ενικό Πρόσωπο
ΡΥΑΑΠΛ	Ρήμα Υποτακτική Αορίστου Πρώτο Πληθυντικό Πρόσωπο
ΡΥΑΒΕΝ	Ρήμα Υποτακτική Αορίστου Δεύτερο Ενικό Πρόσωπο
ΡΥΑΒΠΛ	Ρήμα Υποτακτική Αορίστου Δεύτερο Πληθυντικό Πρόσωπο
ΡΥΑΓΕΝ	Ρήμα Υποτακτική Αορίστου Τρίτο Ενικό Πρόσωπο
ΡΥΑΓΠΛ	Ρήμα Υποτακτική Αορίστου Τρίτο Πληθυντικό Πρόσωπο

Παράρτημα Η

Κώδικες χαρακτηριστικών scripts

Η.1 *Κώδικες από scripts που χρησιμοποιήθηκαν στην προεπεξεργασία του corpus*

Script 1: Ανάλυση συντομογραφιών στο corpus

```
#!/usr/local/net/bin/perl -w

 use POSIX;
 setlocale(POSIX::LC_ALL,"el");


while(<>){
  #s/([^\W0-9_])/\l$1/g;
  if(!/^<[sp]/ && !/^<\/[sp]/)
  {

s/ Ακ\. / Ακάκιος /g;
s/ Αν\. / Αναστάσιος /g;
s/ Υβ\. / Υβόννη /g;
s/ Ε\.Ε\.ς / Ε\.Ε\. /g;
s/ η Ε\.Ε\. / η ευρωπαϊκή ένωση /g;
s/ της Ε\.Ε\. / της ευρωπαϊκής ένωσης /g;
s/ την Ε\.Ε\. / την ευρωπαϊκή ένωση /g;
s/ στην Ε\.Ε\. / στην ευρωπαϊκή ένωση /g;
s/ Ε\.Ε\. / ευρωπαϊκή ένωση /g;
s/ Σχ\. / σχολές /g;
s/ σχ\. / σχήμα /g;
s/ Στ\. / στέλιος /g;
s/ ΑΠ\. / απάντηση /g;
s/ Κ\.Κ\. / κομμουνιστικό κόμμα /g;
s/ γ\. / γενικό /g;
s/ Σ\.σ\. / σημείωση συγγραφέα /g;
s/ σ\.σ\. / σημείωση συγγραφέα /g;
s/ π\. / πόντοι /g;
s/ ΕΡ\. / ερώτηση /g;
s/ Ηρ\. / Ηρακλής /g;
s/ Ηλ\. / Ηλίας /g;
```

```
s/ Σπ\. / Σπύρος /g;
s/ Κλ\. / Κλεάνθης /g;
s/ Ευ\. / Ευάγγελος /g;
s/ Αλ\. / Αλέξανδρος /g;
s/ Χρ\. / Χρήστος /g;
s/ Αθ\. / Αθανάσιος /g;
s/ Βλ\. / Βλάσης /g;
s/ Χ\. Τρικούπη / χαριλάου τρικούπη /g;
s/ Χ\. / χι /g;
s/ Ελ\. / Ελένη /g;
s/ Απ\. / Απόστολος /g;
s/ ο Αγ\. / ο άγιος /g;
s/ του Αγ\. / του αγίου /g;
s/ τον Αγ\. / τον άγιο /g;
s/ στον Αγ\. / στον άγιο /g;
s/ οι Αγ\. / οι άγιοι /g;
s/ των Αγ\. / των αγίων /g;
s/ τους Αγ\. / τους αγίους /g;
s/ των Αγ\. / των αγίων /g;
s/ στους Αγ\. / στους αγίους /g;
s/ Αγ\. / άγιος /g;
s/ Ν\.Α\. / νοτιοανατολική /g;
s/ Α\.Ο\. / αθλητικός όμιλος /g;
s/ ΠΑΛ\. / παλαιά /g;
s/ ΑΓ\. / άγιος /g;
s/^Κ\. Μπ\. / κάπα μπι /g;
s/ Α\.Ε\. / ανώνυμος εταιρία /g;
s/ Α\.Τ\.Ε\. / αγροτική τράπεζα της ελλάδος /g;
s/ Γ\.Ο\.Κ\. / γοκ /g;
s/ « Ε » / « Ελευθεροτυπία » /g;
s/ ΕΟΚ / εόκ /g;
s/ η Ν\.Δ\. / η νέα δημοκρατία /g;
s/ της Ν\.Δ\. / της νέας δημοκρατίας /g;
s/ την Ν\.Δ\. / την νέα δημοκρατία /g;
s/ τη Ν\.Δ\. / τη νέα δημοκρατία /g;
s/ στην Ν\.Δ\. / στην νέα δημοκρατία /g;
s/ στη Ν\.Δ\. / στη νέα δημοκρατία /g;
s/ Ν\.Δ\. / νέα δημοκρατία /g;
s/ R\. / ρο /g;
s/ Π\.Μ\.Κ\. / πι μι κάπα /g;
s/ Σ\.Ο\. / σίγμα όμικρον /g;
s/ Τρ\. / τράπεζα /g;
s/ Δ\.Α\.Σ\. / δας /g;
s/ Π\.Κ\.Δ\. / πι κάπα δέλτα /g;
s/ Γ\.Σ\. / γενική συνέλευση /g;
s/ ΒΑΣ\. / βασίλης /g;
s/ π\. Πίος / πάπας Πίος /g;
s/ Ν\.Μ\. / νι μι /g;
s/ δ\. / δίποντα /g;
s/ τρ\. / τρίποντα /g;
s/ β\. / βαθμοί /g;
s/ ρ\. / ριμπάουντ /g;
s/ δισ\. / δισεκατομμύρια /g;
s/ Τ\.Α\. / τοπική αυτοδιοίκηση /g;
s/ κ\.Πλεινεβό / κύριο Πλεινεβό /g;
s/ Ντ\. Χ\. / Ντι Χι /g;
s/ σ\. / σελίδα /g;
s/ π\.Χ\. / προ Χριστού /g;
s/ μ\.Χ\. / μετά Χριστόν /g;
s/ J\. / τζέι /g;
s/ ΑΠΕ / αθηναϊκό πρακτορείο είδησεων /g;
```

```
s/ J\.P\. / τζέι πι /g;
s/ χιλ\. δραχμές / χιλιάδες δραχμές /g;
s/ χιλ\. / χιλιάδες /g;
s/ Κ\.Κ\.Ε\. / ΚΚΕ /g;
s/ το ΚΚΕ / το κομμουνιστικό κόμμα ελλάδας /g;
s/ του ΚΚΕ / του κομμουνιστικού κόμματος ελλάδας /g;
s/ στο ΚΚΕ / στο κομμουνιστικό κόμμα ελλάδας /g;
s/ ΚΚΕ / κομμουνιστικό κόμμα ελλάδας /g;
s/ ΦΠΑ / φόρος προστιθέμενης αξίας /g;
s/ Φιλαδ\. / Φιλαδέλφεια /g;
s/ Ι\.Α\. / γιώτα άλφα /g;
s/ Θ\. / θήτα /g;
s/ αι\. / αιώνα /g;
s/ το ΥΠΕΘΟ / το υπουργείο εθνικής οικονομίας /g;
s/ του ΥΠΕΘΟ / του υπουργείου εθνικής οικονομίας /g;
s/ στο ΥΠΕΘΟ / στο υπουργείο εθνικής οικονομίας /g;
s/ ΥΠΕΘΟ / υπουργείο εθνικής οικονομίας /g;
s/ Ν\.Φ\. / νέας φιλαδέλφειας /g;
s/ τηλ\. / τηλέφωνο /g;
s/ Ε\.Δ\.Η\.Ν / εδήν /g;
s/ το Σ\.Τ\.Ε\. / το συμβούλιο της επικρατείας /g;
s/ του Σ\.Τ\.Ε\. / του συμβουλίου της επικρατείας /g;
s/ στο Σ\.Τ\.Ε\. / στο συμβούλιο της επικρατείας /g;
s/ Σ\.Τ\.Ε\. / συμβούλιο της επικρατείας /g;
s/ ΡΚΚ / πε κα κα /g;
s/ Σ\.Ε\. / συντονιστική επιτροπή /g;
s/ Ε\.Ι\.Π\. / έψιλον γιώτα πι /g;
s/ Πειρ\. / Πειραιάς /g;
s/ Σημ\. / σημείωση /g;
s/^ΥΓ\. / υστερόγραφο /g;
s/ ΥΓ\. / υστερόγραφο /g;
s/ ΔΣ / διοικητικό συμβούλιο /g;
s/ \. , / , /g;
s/ Γ\.Κεφαλογιάννη / γιάννη κεφαλογιάννη /g;
s/ φ\. / φι /g;
s/ ΔΕΗ / δεή /g;
s/ ΔΟΕ / δόε /g;
s/ ΕΟΑ / εόα /g;
s/ ΕΠΑΕ / επαέ /g;
s/ ΝΕ\.ΧΑ / νε χα /g;
s/ Ντ\. / Ντι /g;
s/ ΑΓΕΤ / αγέτ /g;
s/ Π\. Τζ\. / πι τζέι /g;
s/ πολύχρ\. / πολύχρωμο /g;
s/ ολοσελ\. / ολοσέλιδο /g;
s/ εικονογραφ\. / εικονογραφημένο /g;
s/ δεμ\. / δεμένο /g;
s/ συμμ\. / συμμετέχει /g;
s/ Μ\.Τ\.Σ\. / μι ταυ σίγμα /g;
s/ D\.N\.A\. / ντι εν έι /g;
s/ G\.C\.E\. / τζι σι ι /g;
s/ G\.T\.I\. / τζι τι άι /g;
s/ P\.C\.A\. / πι σι έι /g;
s/ Καλ\. / καλ /g;
s/ Ντ\. / ντι /g;
s/ Σ\.Ε\.Φ\. / στάδιο ειρήνης και φιλίας /g;
s/ και άλλα\. / και άλλα /g;
s/ Ν\.Κ\. / νι κάπα /g;
s/ ΣΕΤΕ / σέτε /g;
s/ · / ·\n /g;
s/ ν\. / νόμος /g;
```

```
s/ 1β / 1 βήτα /g;
s/ ΕΟΤ / ελληνικός οργανισμός τουρισμού /g;
s/ Τρ\.Κρήτης / Τράπεζα Κρήτης /g;
s/ « Κ\.Ε\. » / « Κυριακάτικη Ελευθεροτυπία » /g;
s/ Γ\.-Α\. / γεώργιο αλέξανδρο /g;
s/ ο υπ\. / ο υπουργός /g;
s/ του υπ\. / του υπουργού /g;
s/ τον υπ\. / τον υπουργό /g;
s/ στον υπ\. / στον υπουργό /g;
s/ οι υπ\. / οι υπουργοί /g;
s/ των υπ\. / των υπουργών /g;
s/ τους υπ\. / τους υπουργούς /g;
s/ στους υπ\. / στους υπουργούς /g;
s/ το υπ\. / το υπουργείο /g;
s/ τα υπ\. / τα υπουργεία /g;
s/ στα υπ\. / στα υπουργεία /g;
s/ στο υπ\. / στο υπουργείο /g;
s/ υπ\. / υπουργός /g;
s/ C\.I\.A\. / σι άι έι /g;
s/ ΑΕ / ανώνυμος εταιρία /g;
s/ Κ\.Ε\. / κεντρική επιτροπή /g;
s/ πλ\. / πλατεία /g;
s/ ΓΣΕΕ / γε σε ε /g;
s/ Πατ\. / πατ /g;
s/ Ιατ\. / ιατρός /g;
s/ πολ\.μηχ\. / πολιτικός μηχανικός /g;
s/ Ευγ\. / ευγενία /g;
s/^Α\. / άλφα /g;
s/ γεν\. / γενική /g;
s/ S\.A\.S\. / ες έι ες /g;
s/ Αρθ\. / άρθρο /g;
s/ ΕΡ\. / ερώτηση /g;
s/ Τηλ\. / τηλέφωνο /g;
s/^Ν\. / νι /g;
s/ ΝΔ\. / ΝΔ /g;
s/ η ΝΔ / η νέα δημοκρατία /g;
s/ της ΝΔ / της νέας δημοκρατίας /g;
s/ την ΝΔ / την νέα δημοκρατία /g;
s/ τη ΝΔ / τη νέα δημοκρατία /g;
s/ στην ΝΔ / στην νέα δημοκρατία /g;
s/ στη ΝΔ / στη νέα δημοκρατία /g;
s/ ΝΔ / νέα δημοκρατία /g;
s/ P\.S\. / πι ες /g;
s/ ABS / έι μπι ες /g;
s/ Π\.Γ\. / πολιτικό γραφείο /g;
s/ χρ\. / χρονών /g;
s/ Ιορ\. / Ιορδανία /g;
s/ Αγγλ\. / Αγγλία /g;
s/ Ουγγ\. / Ουγγαρία /g;
s/ Ο\.Α\. / ολυμπιακή αεροπορία /g;
s/ ΕΡΤ / ερτ /g;
s/^Κ\. / κάπα /g;
s/ Οδ\. / Οδυσσέα /g;
s/ Μάρτ\. / μάρτιος /g;
s/ Νοέμ\. / νοέμβριος /g;
s/ Δεκ\. / δεκέμβριος /g;
s/ Αχ\. / αχιλλέας /g;
s/ Πρ\. / πρόδρομος /g;
s/ Co\. / κορπορέισιον /g;
s/ Ρ\.Κ\. / ρεπουμπλικανικό κόμμα /g;
s/ Μπ\. / μπάμπης /g;
```

```
s/^Ρ\/Σ / ραδιοφωνικός σταθμός /g;
s/ Ρ\/Σ / ραδιοφωνικός σταθμός /g;
s/ ΕΣΗΕΑ / εσηέα /g;
s/ c\.d\. / σι ντι /g;
s/ ο Γ\.Γ\. / ο γενικός γραμματέας /g;
s/ του Γ\.Γ\. / του γενικού γραμματέα /g;
s/ τον Γ\.Γ\. / τον γενικό γραμματέα /g;
s/ στον Γ\.Γ\. / στον γενικό γραμματέα /g;
s/ στο Γ\.Γ\. / στο γενικό γραμματέα /g;
s/ η Γ\.Γ\. / η γενική γραμματέας /g;
s/ της Γ\.Γ\. / της γενικής γραμματέως /g;
s/ την Γ\.Γ\. / την γενική γραμματέα /g;
s/ στην Γ\.Γ\. / στην γενική γραμματέα /g;
s/ τη Γ\.Γ\. / τη γενική γραμματέα /g;
s/ στη Γ\.Γ\. / στη γενική γραμματέα /g;
s/ Γ\.Γ\. / γενικός γραμματέας /g;
s/ Κ\.Πυλαρινός / κάπα πυλαρινός /g;
s/ ΕΓ / εκτελεστικό γραφείο /g;
s/ στ\. / στάση /g;
s/ κιν\/φος / κινηματογράφος /g;
s/ Β\.Δ\. / βορειοδυτικής /g;
s/ καθ\. / καθαρεύουσα /g;
s/ αντισ\. / αντίστροφα /g;
s/ αρ\. / αριθμός /g;
s/ Ι\. / γιώτα /g;
s/ Ν\.Ε\. / νομαρχιακή επιτροπή /g;
s/ ενν\.Ανδρέας / εννοείται Ανδρέας /g;
s/ ΚΕΠΠΑ / κοινή εξωτερική πολιτική  και πολιτική ασφάλειας /g;
s/ Γ\.Α\. / γάμμα άλφα /g;
s/ ΗΠΑ / ήπα /g;
s/ Ιω\. / ιωάννης /g;
s/ Ν\.Τ\.V\. / εν τι βι /g;
s/ D\.P\.A\. / ντι πι έι /g;
s/ Τζ\. Τζανετάκης / τζανής τζανετάκης /g;
s/ Τζ\. / τζέι /g;
s/^Μ\. / μι /g;
s/ α\' / άλφα /g;
s/ β\' / βήτα /g;
s/ γ\' / γάμμα /g;
s/ Π\.Σ\.Ε\. / προγράμματα σπουδών επιλογής /g;
s/ φωτ\. / φωτογραφία /g;
s/ Σεπτ\. / Σεπτέμβρης /g;
s/ ΕΦΚ / έψιλον φι κάπα /g;
s/ Ο\.Η\.Μ\.S\. / ο έιτς εμ ες /g;
s/ Ε\.Δ\. / έψιλον δέλτα /g;
s/ λοιπά\. / λοιπά /g;
s/ ΕΥΔΑΠ / ευδάπ /g;
s/ W\.R\. / ντάμπλιγιου αρ /g;
s/ ^Γ\. / γάμμα /g;
s/ Παν\. / παναγιώτης /g;
s/ Ε\.Ο\.ΠΕ\. / εοπέ /g;
s/ η ΕΛ\.ΑΣ\. / η ελληνική αστυνομία /g;
s/ της ΕΛ\.ΑΣ\. / της ελληνικής αστυνομίας /g;
s/ την ΕΛ\.ΑΣ\. / την ελληνική αστυνομία /g;
s/ στην ΕΛ\.ΑΣ\. / στην ελληνική αστυνομία /g;
s/ ΕΛ\.ΑΣ\. / ελληνική αστυνομία /g;
s/ Ν-ΝΔ / νότιοι νοτιοδυτικοί /g;
s/ ΒΔ\. / ΒΔ \. /g;
s/ ΒΔ / βορειοδυτικοί /g;
s/ δολ / δολάρια /g;
s/ δολ\. / δολάρια /g;
```

```
s/ ΕΕ\. / ΕΕ \. /g;
s/ η ΕΕ / η ευρωπαϊκή ένωση /g;
s/ της ΕΕ / της ευρωπαϊκής ένωσης /g;
s/ την ΕΕ / την ευρωπαϊκή ένωση /g;
s/ στην ΕΕ / στην ευρωπαϊκή ένωση /g;
s/ ΕΕ / ευρωπαϊκή ένωση /g;
s/ « ΚΕ » / « Κυριακάτικη Ελευθεροτυπία » /g;
s/ ΑΓΕΤ / αγέτ /g;
s/ Κ\.Μητσοτάκης / κώστας μητσοτάκης /g;
s/ Α\.Ανδριανόπουλος / ανδρέας ανδριανόπουλος /g;
s/ ΣΤ / έκτο /g;
s/ Κων\/νου / κωνσταντίνου /g;
s/ ΠΛ\. / πλατεία /g;
s/ Χ\.Δ\. / χι δέλτα /g;
s/ Ερ\. / ερυθρός /g;
s/ Α1 / άλφα ένα /g;
s/ Α2 / άλφα δύο /g;
s/ ΥΠΑ / ύπα /g;
s/ FIR / φιρ /g;
s/^Β\. / βητα /g;
s/^Ε\. / έψιλον /g;
s/ ΕΜΣΕ / εψιλον μι σίγμα γιώτα /g;
s/ Π\.Δ\. / προεδρικό διάταγμα /g;
s/ μμ\. / μετά μεσημβρίαν /g;
s/ δρ\. / δόκτωρ /g;
s/ St\. / σεντ /g;
s/ μ\. / μέτρα /g;
s/ ν\.μ / ναυτικά μίλια /g;
s/ ε\. / έψιλον /g;
s/ τ\. / ταυ /g;
s/^Λ\. / λάμδα /g;
s/ τ\.μ\. / τετραγωνικά μέτρα /g;
s/^Ε\.Φ\. / έψιλον φι /g;
s/ ΠΚΔ / πι κάπα δέλτα /g;
s/ ΑΚ / άλφα κάπα /g;
s/ ΑΟ / αθλητικός όμιλος /g;
s/ ΦΕΣ / φι έψιλον σίγμα /g;
s/ ΠΣ / πι σίγμα /g;
s/ Κ\. / κάπα /g;
s/ ΣΟ / σίγμα όμικρον /g;
s/ Ο\.Τζ\. / ο τζέι /g;
s/ Γ\.Ι\. / γάμμα γιώτα /g;
s/ Α\.Α\. / άλφα άλφα /g;
s/ λ\. / λεωφόρος /g;
s/ ΟΠΑΠ / οπάπ /g;
s/ ΣΕΓΑΣ / σέγας /g;
s/ Δ\.ΚΩΝ /  /g;
s/ ΑΠΘ / αριστοτέλειο πανεπιστήμιο θεσσαλονίκης /g;
s/ Μ-Λ ΚΚΕ / μι λάμδα κου κου ε /g;
s/ π\. Αρσένιος / πατήρ αρσένιος /g;
s/ παράγρ / παράγραφος /g;
s/ « ΚΕ  » / « Κυριακάτικη Ελευθεροτυπία » /g;
s/ Π\.Κ\. / πι κάπα /g;
s/ τα Η\.Ε\. / τα ηνωμένα έθνη /g;
s/ των Η\.Ε\. / των ηνωμένων εθνών /g;
s/ στα Η\.Ε\. / στα ηνωμένα έθνη /g;
s/ Η\.Ε\. / ηνωμένα έθνη /g;
s/^Θ\.Π / θήτα πι /g;
s/ ΣΥΝ / συνασπισμός /g;
s/ KGB / κα γκε μπε /g;
s/ ΚΚΕ-ΕΣ-ΑΝ\.ΑΡ\. / Κου Κου Ε εσωτερικού ανανεωτική αριστερά /g;
```

```
s/ Κ\. / κάπα /g;
s/ Ο\.Α\. ΧΑΝΙΩΝ / όμιλος αντισφαίρισης χανίων /g;
s/ Ο\.Α\. / ολυμπιακή αεροπορία /g;
s/ Π\.Σ\.Φ\. / πι σίγμα φι /g;
s/ Γ\.Ε\. / γάμμα έψιλον /g;
s/^ΑΛ\. / αλέξανδρος /g;
s/ Γ\.Κ\. / γάμμα κάπα /g;
s/ κά\. / και άλλα /g;
s/ Π\.Ρ\. / πι ρο /g;
s/ Γερμ\. / γερμανία /g;
s/ ε\.σ\. / έψιλον σίγμα /g;
s/ ΙΧ / γιώτα χι /g;
s/ Ο\.ΚΑ\.ΝΑ\. / οργανισμός κατά των ναρκωτικών /g;
s/ Τ\.Τ\. / ταχυδρομικό ταμιευτήριο /g;
s/ Β\. / βήτα /g;
s/ Γ\.Ε\. / γάμμα έψιλον /g;
s/ Κρ\. / κρέζιμιρ /g;
s/ Φλ\. / φλώρας /g;
s/ Χρ\.Μούρτζης / χρήστος μούρτζης /g;
s/ Δρ\. / δόκτωρ /g;
s/ οκ\. / ο κύριος /g;
s/ ΑΣΤ\. / αστέρας /g;
s/^ΑΠΑΝΤ\. / απάντηση /g;
s/ ΠΟΛΑ / πολιτική άνοιξη /g;
s/ ΤτΕ  / τράπεζα της ελλάδος /g;
s/ το ΔΝΤ  / το διεθνές νομισματικό ταμείο /g;
s/ του ΔΝΤ  / του διεθνούς νομισματικού ταμείου /g;
s/ στο ΔΝΤ  / στο διεθνές νομισματικό ταμείο /g;
s/ ΔΝΤ  / διεθνές νομισματικό ταμείο /g;
s/ το ΑΕΠ  / το ακαθάριστο εθνικό προϊόν /g;
s/ του ΑΕΠ  / του ακαθάριστου εθνικού προϊόντος /g;
s/ στο ΑΕΠ  / στο ακαθάριστο εθνικό προϊόν /g;
s/ ΑΕΠ  / ακαθάριστο εθνικό προϊόν /g;
s/ Π-Σ\.Σ\. / πι σίγμα σίγμα /g;
s/ Π\.Κ\.Σ\. / πι κάπα σίγμα /g;
s/ ΣΕΛ / σελίδα /g;
s/ Μιχ\. / μιχάλης /g;
s/ Φωτ\. / φώτης /g;
s/ Νικ\. / νίκος /g;
s/ Δημ\. / δημήτρης /g;
s/ Αναστ\. / αναστάσιος /g;
s/ Αντ\. / αντώνης /g;
s/ Ευάγγ\. / ευάγγελος /g;
s/ Γεωργ\. / γεώργιος /g;
s/ Ναζ\. / ναζ /g;
s/ Αικ\. / αικατερίνη /g;
s/ Μαρ\. / μαρία /g;
s/ Αθαν\. / αθανάσιος /g;
s/ Σταμ\. / σταμάτης /g;
s/ Σοφ\. / σοφία /g;
s/ Θεόδ\. / θεόδωρος /g;
s/ Νικ / νικόλαος /g;
s/ Γη\. / γη \.\n /g;
s/ αντίσ\. / αντίστροφα /g;
s/ = / ίσον /g;
s/ Ο\.π\. / όρα παράγραφο /g;
s/ ό\.π\. / όρα παράγραφο /g;
s/^Δ\. / δέλτα /g;
s/ ΜΜΕ / μέσα μαζικής ενημέρωσης /g;
s/ εκατ\. / εκατομμύρια /g;
s/ ΓΝΑ / γάμμα νι άλφα /g;
```

```
s/ ΕΛ\.ΠΕ\. / ελληνικά πετρέλαια /g;
s/ ΟΦΘ / όμιλος φίλων θαλάσσης /g;
s/ AIDS / έιτζ /g;
s/^ΑΝΤ\. / αντώνης /g;
s/ ξεν / ξενικά /g;
s/ καθ\. / καθηγητής /g;
s/ καθ / καθαρεύουσα /g;
s/ Β\.Ι\. / βήτα γιώτα /g;
s/^F\.A\.T / εφ έι τι /g;
s/ AGB / έι τζι μπι /g;
s/^ΚΩΝ\. / κωνσταντίνου /g;
s/ Γρ\. / γρηγόρη /g;
s/^κ\. / κύριος /g;
s/ ΣΤ / έκτη /g;
s/ ΔΟΥ / δόυ /g;
s/ ΔΗ\.ΠΕ\.ΘΕ\. / δη πε θε /g;
s/ Αι\. / αικατερίνη /g;
s/ Υ\.Δ\.Τ / υπουργείου δημόσιας τάξης /g;
s/ Π\.ΕΝ\.Α\.Α\. / πανελλήνιας ένωσης απόστρατων αξιωματικών /g;
s/ έν\. / έναντι /g;
s/^Ν\.Α\.Κ\. / νι αλφα κάπα /g;
s/ ΕΣΡ / εθνικό συμβούλιο ραδιοτηλεόρασης /g;
s/ ο υφ\. / ο υφυπουργός /g;
s/ του υφ\. / του υφυπουργού /g;
s/ τον υφ\. / τον υφυπουργό /g;
s/ στον υφ\. / στον υφυπουργό /g;
s/ οι υφ\. / οι υφυπουργοί /g;
s/ των υφ\. / των υφυπουργών /g;
s/ τους υφ\. / τους υφυπουργούς /g;
s/ στους υφ\. / στους υφυπουργούς /g;
s/ το υφ\. / το υφυπουργείο /g;
s/ τα υφ\. / τα υφυπουργεία /g;
s/ στα υφ\. / στα υφυπουργεία /g;
s/ στο υφ\. / στο υφυπουργείο /g;
s/ υφ\. / υφυπουργός /g;
s/^Π\.χ\. / παραδείγματος χάριν /g;
s/ Θεσ\/νίκης / θεσσαλονίκης /g;
s/ Α\.Τ\. / άλφα ταυ /g;
s/ γ\.γ\.Α\. / γενικός γραμματέας αθλητισμού /g;
s/Α\./ άλφα /g;
s/Β\./ βήτα /g;
s/Γ\./ γάμμα /g;
s/Δ\./ δέλτα /g;
s/Ε\./ έψιλον /g;
s/Ζ\./ ζήτα /g;
s/Η\./ ήτα /g;
s/Θ\./ θήτα /g;
s/Ι\./ γιώτα /g;
s/Κ\./ κάπα /g;
s/Λ\./ λάμδα /g;
s/Μ\./ μι /g;
s/Ν\./ νι /g;
s/Ξ\./ ξι /g;
s/Ο\./ όμικρον /g;
s/Π\./ πι /g;
s/Ρ\./ ρο /g;
s/Σ\./ σίγμα /g;
s/Τ\./ ταυ /g;
s/Υ\./ ύψιλον /g;
s/Φ\./ φι /g;
s/χ\./ χι /g;
```

```
s/Ψ\./ ψι /g;
s/Ω\./ ωμέγα /g;
s/A\./ έι /g;
s/B\./ μπι /g;
s/C\./ σι /g;
s/D\./ ντι /g;
s/E\./ ι /g;
s/F\./ εφ /g;
s/G\./ τζι /g;
s/H\./ έιτς /g;
s/I\./ άι /g;
s/J\./ τζέι /g;
s/K\./ κέι /g;
s/L\./ ελ /g;
s/M\./ εμ /g;
s/N\./ εν /g;
s/O\./ όου /g;
s/P\./ πι /g;
s/Q\./ κιου /g;
s/R\./ αρ /g;
s/S\./ ες /g;
s/T\./ τι /g;
s/U\./ γιου /g;
s/V\./ βι /g;
s/W\./ ντάμπλιγιου /g;
s/X\./ έξ /g;
s/Y\./ γουάι /g;
s/Z\./ ζεντ /g;

   }
    print $_;
  }
```

Script 2: Διόρθωση πτώσεων στους αριθμούς του corpus

```
#!/usr/local/net/bin/perl -w

 use POSIX;
 setlocale(POSIX::LC_ALL,"el");



%numbers = (
            "ένα" => undef,
                                    "τρία" => undef,
                                    "τέσσερα" => undef,
                                    "διακόσια" => undef,
                                    "τριακόσια" => undef,
            "τετρακόσια" => undef,
            "πεντακόσια" => undef,
            "εξακόσια" => undef,
            "επτακόσια" => undef,
            "οκτακόσια" => undef,
            "εννιακόσια" => undef,
            "χίλια" => undef,
            "χιλιάδες" => undef,
            "εκατομμύριο" => undef,
            "εκατομμύρια" => undef,
            "δραχμές" => undef
           );



$numbers{"ένα"} =
['ένας','ενός','ένα+','ένα+','ένα+','μία','μίας','ένα+'];
$numbers{"τρία"} =
['τρία+','τρία+','τρεις','τριών','τρεις','τρία+','τρία+','τρεις'];
$numbers{"τέσσερα"} =
['τέσσερα+','τέσσερα+','τέσσερις','τεσσάρων','τέσσερις','τέσσερα+','τ
έσσερα+','τέσσερις'];
$numbers{"διακόσια"} =
['διακόσια+','διακόσια+','διακόσιοι','διακοσίων','διακόσιους','διακόσ
ια+','διακόσια+','διακόσιες'];
$numbers{"τριακόσια"} =
['τριακόσια+','τριακόσια+','τριακόσιοι','τριακοσίων','τριακόσιους','τ
ριακόσια+','τριακόσια+','τριακόσιες'];
$numbers{"τετρακόσια"} =
['τετρακόσια+','τετρακόσια+','τετρακόσιοι','τετρακοσίων','τετρακόσιου
ς','τετρακόσια+','τετρακόσια+','τετρακόσιες'];
$numbers{"πεντακόσια"} =
['πεντακόσια+','πεντακόσια+','πεντακόσιοι','πεντακοσίων','πεντακόσιου
ς','πεντακόσια+','πεντακόσια+','πεντακόσιες'];
$numbers{"εξακόσια"} =
['εξακόσια+','εξακόσια+','εξακόσιοι','εξακοσίων','εξακόσιους','εξακόσ
ια+','εξακόσια+','εξακόσιες'];
$numbers{"επτακόσια"} =
['επτακόσια+','επτακόσια+','επτακόσιοι','επτακοσίων','επτακόσιους','ε
πτακόσια+','επτακόσια+','επτακόσιες'];
```

```
$numbers{"οκτακόσια"} =
['οκτακόσια+','οκτακόσια+','οκτακόσιοι','οκτακοσίων','οκτακόσιους','ο
κτακόσια+','οκτακόσια+','οκτακόσιες'];
$numbers{"εννιακόσια"} =
['εννιακόσια+','εννιακόσια+','εννιακόσιοι','εννιακοσίων','εννιακόσιου
ς','εννιακόσια+','εννιακόσια+','εννιακόσιες'];
$numbers{"χίλια"} =
['χίλια+','χίλια+','χίλιοι','χιλίων','χίλιους','χίλια+','χίλια+','χίλ
ιες'];
$numbers{"χιλιάδες"} =
['χιλιάδες+','χιλιάδες+','χιλιάδες+','χιλιάδων','χιλιάδες+','χιλιάδες
+','χιλιάδες+','χιλιάδες+'];
$numbers{"εκατομμύριο"} =
['εκατομμύριο+','εκατομμυρίου','εκατομμύριο+','εκατομμύριο+','εκατομμ
ύριο+','εκατομμύριο+','εκατομμύριο+','εκατομμύριο+'];
$numbers{"εκατομμύρια"} =
['εκατομμύρια+','εκατομμυρία+','εκατομμύρια+','εκατομμυρίων','εκατομμ
ύρια+','εκατομμύρια+','εκατομμύρια+','εκατομμύρια+'];
$numbers{"δραχμές"} =
['δραχμές+','δραχμές+','δραχμές+','δραχμών','δραχμές+','δραχμή','δραχ
μής','δραχμές+'];
```

```
local $/ = "\n";
while(<>){

  #s/([^\W0-9_])/\l$chunk/g;
  if(!/^<[sp]/ && !/^<\/[sp]/)
  {
#chomp;
     while  (/([Α-Ωα-ωάέίόύώήΐΰ]+)/g)
               {

                    $chunk=$1;
                    if (exists $numbers{$chunk})
                         {

                    if ((/ ο $chunk / ) || (/ [Α-Ωα-ωάέίόύώήΐΰ]+ος
$chunk /) || (/ [Α-Ωα-ωάέίόύώήΐΰ]+ός $chunk /) || (/ $chunk [Α-Ωα-
ωάέίόύώήΐΰ]+ος /) || (/ $chunk [Α-Ωα-ωάέίόύώήΐΰ]+ός /))
                                        {
                                          foreach $key (keys %numbers)
                                                {
                                                      if ($key =~
$chunk)                 # AOE
                                                           {

                                                                s/
$chunk / @{ $numbers{$key} }[0] /;

                                                                    }
                                                }
```

```
                                        }

elsif ((/ του $chunk / ) || (/ [Α-Ωα-ωάέίόύώήϊΐϋΰ]+ου $chunk /) || (/
[Α-Ωα-ωάέίόύώήϊΐϋΰ]+ού $chunk /) || (/ $chunk [Α-Ωα-ωάέίόύώήϊΐϋΰ]+ου /)
|| (/ $chunk [Α-Ωα-ωάέίόύώήϊΐϋΰ]+ού /))
                                    {
                                      foreach $key (keys %numbers)
                                            {
                                                  if ($key =~
$chunk)                 # AGE
                                                     {
                                                           s/
$chunk / @{ $numbers{$key} }[1] /;

                                                     }
                                               }
                                        }

                                        elsif ((/ οι $chunk / ) ||
(/ [Α-Ωα-ωάέίόύώήϊΐϋΰ]+οι $chunk /) || (/ [Α-Ωα-ωάέίόύώήϊΐϋΰ]+οί $chunk
/) || (/ $chunk [Α-Ωα-ωάέίόύώήϊΐϋΰ]+οι /) || (/ $chunk [Α-Ωα-
ωάέίόύώήϊΐϋΰ]+οί /))
                                    {
                                      foreach $key (keys %numbers)
                                            {
                                                  if ($key =~
$chunk)                 # AOP
                                                     {
                                                           s/
$chunk / @{ $numbers{$key} }[2] /;

                                                     }
                                               }
                                        }
                                        elsif ((/ των $chunk / ) ||
(/ [Α-Ωα-ωάέίόύώήϊΐϋΰ]+ων $chunk /) || (/ [Α-Ωα-ωάέίόύώήϊΐϋΰ]+ών $chunk
/) || (/ $chunk [Α-Ωα-ωάέίόύώήϊΐϋΰ]+ων /) || (/ $chunk [Α-Ωα-
ωάέίόύώήϊΐϋΰ]+ών /))
                                    {
                                      foreach $key (keys %numbers)
                                            {
                                                  if ($key =~
$chunk)                 # AGP
                                                     {
                                                           s/
$chunk / @{ $numbers{$key} }[3] /;

                                                     }
                                               }
                                        }
                                        elsif ((/ τους $chunk / )
|| (/ [Α-Ωα-ωάέίόύώήϊΐϋΰ]+ους $chunk /) || (/ [Α-Ωα-ωάέίόύώήϊΐϋΰ]+ούς
$chunk /) || (/ $chunk [Α-Ωα-ωάέίόύώήϊΐϋΰ]+ους /) || (/ $chunk [Α-Ωα-
ωάέίόύώήϊΐϋΰ]+ούς /) || (/ $chunk [Α-Ωα-ωάέίόύώήϊΐϋΰ]+εις /) || (/ $chunk
[Α-Ωα-ωάέίόύώήϊΐϋΰ]+είς /))
                                    {
                                      foreach $key (keys %numbers)
                                            {
                                                  if ($key =~
$chunk)                 # AAP
                                                     {
```

```
                                                                s/
$chunk / @{ $numbers{$key} }[4] /;

                                                                  }
                                                            }
                                                        }
                                                        elsif ((/ στην $chunk / )
|| (/ την $chunk / ) || (/ η $chunk / ) || (/ [Α-Ωα-ωάέίόύώήΐΰ]+η
$chunk /) || (/ [Α-Ωα-ωάέίόύώήΐΰ]+ή $chunk /) || (/ $chunk [Α-Ωα-
ωάέίόύώήΐΰ]+η /) || (/ $chunk [Α-Ωα-ωάέίόύώήΐΰ]+ή /))
                                                {
                                                  foreach $key (keys %numbers)
                                                        {
                                                              if (($key =~
$chunk))                # 8OE
                                                                 {
                                                                      s/
$chunk / @{ $numbers{$key} }[5] /;

                                                                  }
                                                            }
                                                        }
                                                elsif ((/ της $chunk / ) || (/
[Α-Ωα-ωάέίόύώήΐΰ]+ας $chunk /) || (/ [Α-Ωα-ωάέίόύώήΐΰ]+άς $chunk /)
|| (/ $chunk [Α-Ωα-ωάέίόύώήΐΰ]+ας /) || (/ $chunk [Α-Ωα-
ωάέίόύώήΐΰ]+άς /) || (/ [Α-Ωα-ωάέίόύώήΐΰ]+ης $chunk /) || (/ [Α-Ωα-
ωάέίόύώήΐΰ]+ής $chunk /) || (/ $chunk [Α-Ωα-ωάέίόύώήΐΰ]+ης /) || (/
$chunk [Α-Ωα-ωάέίόύώήΐΰ]+ής /))
                                                {
                                                  foreach $key (keys %numbers)
                                                        {
                                                              if ($key =~
$chunk)                 # 8GE
                                                                 {
                                                                      s/
$chunk / @{ $numbers{$key} }[6] /;

                                                                  }
                                                            }
                                                        }
                                                elsif ((/ στις $chunk / ) || (/
τις $chunk / ) || (/ [Α-Ωα-ωάέίόύώήΐΰ]+ες $chunk /) || (/ [Α-Ωα-
ωάέίόύώήΐΰ]+ές $chunk /) || (/ $chunk [Α-Ωα-ωάέίόύώήΐΰ]+ες /) || (/
$chunk [Α-Ωα-ωάέίόύώήΐΰ]+ές /))
                                                {
                                                  foreach $key (keys %numbers)
                                                        {
                                                              if ($key =~
$chunk)                 # 8OP
                                                                 {
                                                                      s/
$chunk / @{ $numbers{$key} }[7] /;

                                                                  }
                                                            }
                                                        }


                                                }
```

```
                }

            s/\+//g;

   }
 print $_;  }
```

Script 3: Επεξεργασία αριθμών στο corpus

```
#!/usr/bin/perl -w

use POSIX;
setlocale(POSIX::LC_ALL,"el");

$/="\n";



while (<>) {
      s/([μΜ]ε ψήφους [0-9]+)\s*\-\s*([0-9]+)/$1 προς $2/g;
      s/([0-9]+)\s*\-\s*([0-9]+\/[0-9]+)/$1 εώς $2/g;
      s/(τάξεως
του|τριετία|διετία|πενταετία|δεκαετία|στα|διαρκεί|διάστημα|περίοδο|πε
ρίοδος|δεκάδα|τάξη|σελίδες|ηλικίες|ηλικία|σειρά|ομάδες|ομάδα|αναλογία
|περίπου|Σελ\.|σελ\.|οποία|οποίους|οποίες|οποίο|οποία|οποίο|οι)(\s*[0
-9]+)\s*\-\s*([0-9]+)/$1$2 εώς $3/g;
      s/(γύρω)(\s*)(στο|στη|στους|στις|στα)(\s*[0-9]+)\s*\-\s*([0-
9]+)/$1$2$3$4 με $5/g;
      s/(τηλ\.|τηλέφωνο|αριθμό)(\s*[0-9]+)\s*\-\s*([0-9]+)/$1$2$3/g;
      s/(για [0-9]+)\s*\-\s*([0-9]+)/$1 εώς $2/g;
      s/(ισοφάριζε|ισοφαρίζει|μειώσει|μείωσε|ισοφάρισαν|ισοφάρισε|ισο
φάρισα|ισοφαρίσει)(\s*σε)(\s*[0-9]+)\s*\-\s*([0-9]+)/$1$2 $3/g;
      s/(κάθε| σε|στις|στους|επί)(\s*[0-9]+)\s*\-\s*([0-
9]+)([\s*\)\:\.\,\'\-][^(0-9)])/$1$2 με $3$4/g;
      s/(από τις [0-9]+)\s*\-\s*([0-9]+)/$1 εώς τις $2/g;
      s/(από τις [0-9]+[\.\:][0-9]+)\s*\-\s*([0-9]+[\.\:][0-9]+)/$1
εώς τις $2/g;
      s/(τηλέφων[αο]\s*)([0-9]+[0-9]+[0-9]+[0-9]+)\s*\-\s*([0-9]+[0-
9]+[0-9]+[0-9]+)/$1$2 $3/g;
      s/([0-9][0-9][0-9][0-9])\s*\-\s*([0-9][0-9][0-9][0-
9])([^0123456789])/$1 εώς $2$3/g;
      s/(\([0-9][0-9][0-9][0-9])\s*\-\s*([0-9][0-9]\))/$1 εώς $2/g;
      s/(περίπου [0-9]+)\s*\-\s*([0-9]+)/$1 εώς $2/g;
      s/([Αα]πό [0-9]+)\s*\-\s*([0-9]+)/$1 εώς $2/g;
      s/(μεταξύ|ανάμεσα σε)(\s*[0-9]+)\s*\-\s*([0-9]+)/$1$2 και $3/g;
      s/(μεταξύ)(\s*[0-9]+\.[0-9]+)\s*\-\s*([0-9]+\.[0-9]+)/$1$2 και
$3/g;
      s/([α-ωά-ώΑ-Ω¶-Ω ]+\s*\s*\-\s*\s*[α-ωά-ώΑ-Ω¶-Ω ]+\s*[0-
9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/([0-9]+\,*[0-9]*)\s*\-\s*([0-9]+\,*[0-9]*\s*\%)/$1 εώς $2/g;
      s/([0-9]+)\s*\-\s*([0-9]+\s*\%)/$1 εώς $2/g;
      s/(το σκορ ήταν [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(τελειώσει [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(τελείωσε|έχασε)([0-9]+)\s*\-\s*([0-9]+)/$1$2 $3/g;
      s/(έμεινε στο [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(το αποτέλεσμα ήταν [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/([Αα]ποτέλεσμα [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(σκορ [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(έληξε [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(έληξε με [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(πρωί|καθημερινά|βράδυ|απόγευμα|μεσημέρι)([0-9]+\.[0-
9]+)\s*\-\s*([0-9]+\.[0-9]+)/$1$2 με $3/g;
```

```
      s/([0-9]+\.[0-9]+)\s*\-\s*([0-9]+\.[0-
9]+\s*)(δρχ|δραχμές|δραχμών)/$1 εώς $2$3/g;
      s/([0-9]+\.[0-9]+)\s*\-\s*([0-9]+\.[0-9]+)/$1 εώς $2/g;
      s/([0-9]+)\s*\-\s*([0-9]+\.[0-9]+)/$1 εώς $2/g;
      s/([0-9]+\.[0-9]+)\s*\-\s*([0-9]+)/$1 εώς $2/g;
      s/([0-9]+\,[0-9]+)\s*\-\s*([0-9]+\,[0-9]+)/$1 εώς $2/g;
      s/([0-9]+)\s*\-\s*([0-9]+\,[0-9]+)/$1 εώς $2/g;
      s/([0-9]+\,[0-9]+)\s*\-\s*([0-9]+)/$1 εώς $2/g;
      s/([0-9]+\:[0-9]+)\s*\-\s*([0-9]+\:[0-9]+)/$1 εώς $2/g;
      s/([0-9]+)\s*\-\s*([0-9]+\:[0-9]+)/$1 εώς $2/g;
      s/([0-9]+\:[0-9]+)\s*\-\s*([0-9]+)/$1 εώς $2/g;
      s/(ισόπαλ)(ο|η|α|οι|ες)(\s*)([0-9]+)\s*\-\s*([0-9]+)/$1$2$3$4
$5/g;
      s/(τελικό [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(ισοπαλία [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(έφερε [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/([0-9]+)\s*\-\s*([0-9]+\s*με\s*)(την|τον|το|τη)/$1 $2$3/g;
      s/(έμεινε στο [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(φιλικό [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(έφερε [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(κρατήσει το [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(κράτησε το [0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;
      s/(προηγήθηκε|προηγήθηκε με|προηγείται|προηγείται με)(\s*[0-
9]+)\s*\-\s*([0-9]+)/$1$2 $3/g;
      s/([0-9]+)\s*\-\s*([0-9]+\s*)(το βράδυ|βράδυ)/$1 με $2$3/g;
      s/([0-9]+)\s*\-\s*([0-9]+\s*)(το πρωί|πρωί)/$1 με $2$3/g;
      s/([0-9]+\:[0-9]+)\s*\-\s*([0-9]+\:[0-9]+)/$1 με $2/g;
      s/([0-9]+)\s*\-\s*([0-
9]+\s*)(χιλ|χιλιάδες|χιλιάδων|εκ|εκατ|εκατομ|δισ|τρισ|εκατομμυρίων|δι
σεκατομμυρίων|τρισεκατομμυρίων|εκατομμύρια|δισεκατομμύρια|τρισεκατομμ
ύρια)/$1 εώς $2$3/g;
s/([0-9]+)\s*\-\s*([0-
9]+\s*)(Ιανουαρίου|Φεβρουαρίου|Μαρτίου|Απριλίου|Μαΐου|Ιουνίου|Ιουλίου
|Αυγούστου|Σεπτεμβρίου|Οκτωβρίου|Νοεμβρίου|Δεκεμβρίου)/$1 εώς $2$3/g;
s/([0-9]+)\s*\-\s*([0-9]+\s*)(των)/$1 $2$3/g;
s/([0-9]+)\s*\-\s*([0-9]+\s*)([α-ωά-ώ]ων|Μποφόρ|του          μηνός|του
μήνα|μποφόρ|ο                                                    C|
C|C|οC|θύματα|πλειστηριασμοί|αγρότες|δενδρύλλια|τόνων|τόνοι|Ρίχτερ|ρί
χτερ|κιλά|π\.μ\.|μ\.μ\.|κλινών|κλίνες|τ\.μ\|.σπουδαστές|σπουδαστών|με
γαμπάιτ|κιλά|αγώνες|νομοί|νομαρχίες|μέτρα|μέτρων|ανδρών|άνδρες|ίπποι|
π\.Χ\.|μ\.Χ\.|μίλια|χλμ|χιλιομέτρων|χιλιόμετρα|πόντους|πόντοι|λεπτά|λ
επτών|εκατοστών|εκατοστά|εκ|μεγαβάτ|γραμμάρια|δολάρια|δρχ|δραχμών|δρα
χμές|μίλια|ετών|έτη|χρόνων|χρονών|χρόνια|μέρες|ημέρες|ημερών|άτομα|ατ
όμων|μέτρα|τοις εκατό|του μήνα)/$1 εώς $2$3/g;
while (m/([0-9]+)\s*\-\s*([0-9]+)/g){
s/([0-9]+)\s*\-\s*([0-9]+)/$1 $2/g;  }
#**********************#
s/(είναι)(\s*)([0-9]+)\s*\:\s*([0-9]+)/$1$2$3 προς $4/g;
s/(\(\s*)([0-9]+)\s*\:\s*([0-9]+)(\s*\))/$1$2 προς $3$4/g;
s/([0-9]+)\s*\:\s*([0-9][0-9])/$1 και $2/g;
s/([0-9]+)(\.)([0-9][0-9])(\s*)(π\.μ\.|μ\.μ\.)(\s*)(\-)(\s*)([0-
9]+)(\.)([0-9][0-9])(\s*)(π\.μ\.|μ\.μ\.)/$1 και $3$4$5$6 εώς $8$9 και
$11$12$13/g; # 10.00 π.μ. - 11.00 μ.μ.
s/([0-9]+)(\.)([0-9][0-9])(\s*)(π\.μ\.|μ\.μ\.)(\s*)(εώς|και)(\s*)([0-
9]+)(\.)([0-9][0-9])(\s*)(π\.μ\.|μ\.μ\.)/$1  και  $3$4$5$6$7$8$9  και
$11$12$13/g; # 10.00 π.μ. εώς|και 11.00 μ.μ.
            s/([0-9]+)(\.)([0-9][0-9])(\s*)(\-)(\s*)([0-9]+)(\.)([0-
9][0-9])(\s*)(π\.μ\.|μ\.μ\.)/$1 και $3$4 εώς $6$7 και $9$10$11/g; #
10.00 - 11.00 μ.μ.
```

```
      s/([0-9]+)(\.)([0-9][0-9])(\s*)(εώς|και)(\s*)([0-9]+)(\.)([0-
9][0-9])(\s*)(π\.μ\.|μ\.μ\.)/$1 και $3$4$5$6$7 και $9$10$11/g; #
10.00 εώς|και 11.00 μ.μ.
            s/([0-9]+)(\.)([0-9][0-9])(\s*)(εώς|και)(\s*)([0-
9]+)(\s*)(π\.μ\.|μ\.μ\.)/$1 και $3$4$5$6$7$8$9/g;     # 10.00 εώς|και
11 μ.μ.
            s/([0-9]+)(\s*)(εώς|και)(\s*)([0-9]+)(\.)([0-
9]+)(\s*)(π\.μ\.|μ\.μ\.)/$1$2$3$4$5 και $7$8$9/g;     # 10 εώς|και
11.00 μ.μ.
            s/([0-9]+)(\s*)(π\.μ\.|μ\.μ\.)(\s*)(\-)(\s*)([0-
9]+)(\s*)(π\.μ\.|μ\.μ\.)/$1$2$3$4 εώς $6$7$8$9/g;     # 10 π.μ. - 11
μ.μ.

            s/([0-9])(\.)([0-9][0-9])(\s*)(εώς|και)(\s*)([0-
9])(\.)([0-9][0-9])([\s*\)\:\.\,\'\-][^(0-9)])/$1 και $3$4$5$6$7 και
$9$10/g;    # 1.00 εώς|και 4.00
            s/([0-9]+)(\s*)(εώς|και)(\s*)([0-9])(\.)([0-9][0-
9])([\s*\)\:\.\,\'\-][^(0-9)])/$1$2$3$4$5 και $7$8/g; # 10 εώς|και
1.00
            s/([^0123456789])([0-9])(\.)([0-9][0-
9])(\s*)(εώς|και)(\s*)([0-9]+)(\s*)/$1$2 και $4$5$6$7$8$9/g;      #
1.00 εώς|και 11
      s/(στ[αο]
τελευταί[αο]|[Σσ]τη|[Σσ]το|[Ττ]ις|Ωρα|Στις|ώρα|στις)(\s*)([0-
9]+)(\.)([0-9][0-9])([\s*\)\:\.\,\'\-][^(0-9)])/$1$2$3 και $5$6/g;
      s/([0-9]+)(\.)([0-9][0-9])(\s*)(πριν|το πρωί|τα ξημερώματα|το
απόγευμα|ώρες|το βράδυ)/$1 και $3$4$5/g;
            s/([0-9]+)(\s*)(\-)(\s*)([0-9]+)(\s*)(π\.μ\.|μ\.μ\.)/$1$2
εώς $4$5$6$7/g;   # 10 - 11 μ.μ.
            s/([0-9]+)(\.)([0-9]+)(\s*)(π\.μ\.|μ\.μ\.)/$1 και
$3$4$5/g;   #  11.00 μ.μ.
      s/([\-\s*\(])([0-9][0-9])\s*\.\s*([0-9][0-9])([\s*\)\:\.\,\'\-
][^(0-9)])/$1$2 και $3$4/g;
      s/και 00(\s*)//g;
      s/και 00([\)\:\.\,\'])/$1/g;


      print $_;
}
```

Script 4: Επεξεργασία ημερομηνιών στο corpus

```
#!/usr/bin/perl

use POSIX;
setlocale(POSIX::LC_ALL,"el");


local $/;


%days=(
           "1" => "πρώτη",
           "2" => "δύο",
           "3" => "τρεις",
           "4" => "τέσσερις",
           "5" => "πέντε",
           "6" => "έξι",
           "7" => "επτά",
           "8" => "οκτώ",
           "9" => "εννιά",
           "10" => "δέκα",
           "11" => "έντεκα",
           "12" => "δώδεκα",
           "13" => "δέκα τρεις",
           "14" => "δέκα τέσσερις",
           "15" => "δέκα πέντε",
           "16" => "δέκα έξι",
           "17" => "δέκα επτά",
           "18" => "δέκα οκτώ",
           "19" => "δέκα εννιά",
           "20" => "είκοσι",
           "21" => "είκοσι μία",
           "22" => "είκοσι δύο",
           "23" => "είκοσι τρεις",
           "24" => "είκοσι τέσσερις",
           "25" => "είκοσι πέντε",
           "26" => "είκοσι έξι",
           "27" => "είκοσι επτά",
           "28" => "είκοσι οκτώ",
           "29" => "είκοσι εννιά",
           "30" => "τριάντα",
           "31" => "τριάντα μία"



      );


%months=(
           "1" => "πρώτου",
           "2" => "δευτέρου",
           "3" => "τρίτου",
           "4" => "τετάρτου",
```

```
            "5" => "πέμπτου",
            "6" => "έκτου",
            "7" => "εβδόμου",
            "8" => "ογδόου",
            "9" => "ενάτου",
            "10" => "δεκάτου",
            "11" => "ενδεκάτου",
            "12" => "δωδεκάτου"

      );



$file=<>;

$file=~ s/([0-9]+)(\/)([0-9]+)(\/)([0-9]+)/$days{$1} $months{$3}
$5/g;
$file=~ s/([στις|τις|την|ως|εώς|μέχρι])(\s*)([0-9]+)(\/)([0-
9]+)/$1$2$days{$3} $months{$5}/g;

print $file;
```

H.2 *Κώδικες από scripts που χρησιμοποιήθηκαν στην δημιουργία του ελληνικού POS λεξικού*

Script 5: Αίτηση στο online λεξικό

```
#!/usr/local/net/bin/perl -w

use LWP::UserAgent;
#$ua = new LWP::UserAgent;

use LWP::Simple;
use URI::URL;
use HTTP::Request;

open(IN,"< ./vocab_br1");
open(OUT,">> ./vocab_OUT_html1.10");

$/="\n";
@array = <IN>;
my ($url,$req);

select(OUT);

foreach $word (@array) {
    print
"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\nWORD=$word\n\n\n\n\n\n\n\n\n\n\n\n\n\n
";

    $url = url("http://www.in.gr/dictionary/lookup.asp?Word=$word");
    $req = get($url);
    print $req;
}

close(IN);
close(OUT);
```

Script 6: Εμπλουτισμός του ελληνικού POS λεξικού με επίθετα

```
#!/usr/local/net/bin/perl -w

use POSIX;
setlocale(POSIX::LC_ALL,"el");

open(IN,"< ./$ARGV[0]");
open(OUT,"> ./output");


$/="\n";
@array = <IN>;
close(IN);

select(OUT);

foreach $line (@array) {
      print $line;


      if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)επής\n/g) {
                  print "επίθ. ονομ. ενικ. αρσε. $1επής\n";
                  print "επίθ. γενι. ενικ. αρσε. $1επούς\n";
                  print "επίθ. αιτι. ενικ. αρσε. $1επή\n";
                  print "επίθ. ονομ. πληθ. αρσε. $1επείς\n";
                  print "επίθ. γενι. πληθ. αρσε. $1επών\n";
                  print "επίθ. αιτι. πληθ. αρσε. $1επείς\n";
                  print "επίθ. ονομ. ενικ. θηλ. $1επής\n";
            print "επίθ. γενι. ενικ. θηλ. $1επούς\n";
                  print "επίθ. αιτι. ενικ. θηλ. $1επή\n";
                  print "επίθ. ονομ. πληθ. θηλ. $1επείς\n";
                  print "επίθ. γενι. πληθ. θηλ. $1επών\n";
                  print "επίθ. αιτι. πληθ. θηλ. $1επείς\n";
                  print "επίθ. ονομ. ενικ. ουδ. $1επές\n";
                  print "επίθ. γενι. ενικ. ουδ. $1επών\n";
                  print "επίθ. αιτι. ενικ. ουδ. $1επές\n";
                  print "επίθ. ονομ. πληθ. ουδ. $1επή\n";
                  print "επίθ. γενι. πληθ. ουδ. $1επών\n";
                  print "επίθ. αιτι. πληθ. ουδ. $1επή\n";
                  next;                              }


      if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ώδης\n/g) {
                  print "επίθ. ονομ. ενικ. αρσε. $1ώδης\n";
                  print "επίθ. γενι. ενικ. αρσε. $1ώδους\n";
                  print "επίθ. αιτι. ενικ. αρσε. $1ώδη\n";
                  print "επίθ. ονομ. πληθ. αρσε. $1ώδεις\n";
                  print "επίθ. γενι. πληθ. αρσε. $1ώδων\n";
                  print "επίθ. αιτι. πληθ. αρσε. $1ώδεις\n";
                  print "επίθ. ονομ. ενικ. θηλ. $1ώδης\n";
                  print "επίθ. γενι. ενικ. θηλ. $1ώδους\n";
                  print "επίθ. αιτι. ενικ. θηλ. $1ώδη\n";
                  print "επίθ. ονομ. πληθ. θηλ. $1ώδεις\n";
                  print "επίθ. γενι. πληθ. θηλ. $1ώδων\n";
```

```
            print "επίθ. αιτι. πληθ. θηλ. $1ώδεις\n";
            print "επίθ. ονομ. ενικ. ουδ. $1ώδες\n";
            print "επίθ. γενι. ενικ. ουδ. $1ώδων\n";
            print "επίθ. αιτι. ενικ. ουδ. $1ώδες\n";
            print "επίθ. ονομ. πληθ. ουδ. $1ώδη\n";
            print "επίθ. γενι. πληθ. ουδ. $1ώδων\n";
            print "επίθ. αιτι. πληθ. ουδ. $1ώδη\n";
            next;                                }

if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ειδής\n/g) {
            print "επίθ. ονομ. ενικ. αρσε. $1ειδής\n";
            print "επίθ. γενι. ενικ. αρσε. $1ειδούς\n";
            print "επίθ. αιτι. ενικ. αρσε. $1ειδή\n";
            print "επίθ. ονομ. πληθ. αρσε. $1ειδείς\n";
            print "επίθ. γενι. πληθ. αρσε. $1ειδών\n";
            print "επίθ. αιτι. πληθ. αρσε. $1ειδείς\n";
            print "επίθ. ονομ. ενικ. θηλ. $1ειδής\n";
            print "επίθ. γενι. ενικ. θηλ. $1ειδούς\n";
            print "επίθ. αιτι. ενικ. θηλ. $1ειδή\n";
            print "επίθ. ονομ. πληθ. θηλ. $1ειδείς\n";
            print "επίθ. γενι. πληθ. θηλ. $1ειδών\n";
            print "επίθ. αιτι. πληθ. θηλ. $1ειδείς\n";
            print "επίθ. ονομ. ενικ. ουδ. $1ειδές\n";
            print "επίθ. γενι. ενικ. ουδ. $1ειδών\n";
            print "επίθ. αιτι. ενικ. ουδ. $1ειδές\n";
            print "επίθ. ονομ. πληθ. ουδ. $1ειδή\n";
            print "επίθ. γενι. πληθ. ουδ. $1ειδών\n";
            print "επίθ. αιτι. πληθ. ουδ. $1ειδή\n";
            next;                                }


if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ής\n/g) {
            print "επίθ. ονομ. ενικ. αρσε. $1ής\n";
            print "επίθ. γενι. ενικ. αρσε. $1ιού\n";
            print "επίθ. αιτι. ενικ. αρσε. $1ή\n";
            print "επίθ. ονομ. πληθ. αρσε. $1ιοί\n";
            print "επίθ. γενι. πληθ. αρσε. $1ιών\n";
            print "επίθ. αιτι. πληθ. αρσε. $1ιούς\n";
            print "επίθ. ονομ. ενικ. θηλ. $1ιά\n";
            print "επίθ. γενι. ενικ. θηλ. $1ιάς\n";
            print "επίθ. αιτι. ενικ. θηλ. $1ιά\n";
            print "επίθ. ονομ. πληθ. θηλ. $1ιές\n";
            print "επίθ. γενι. πληθ. θηλ. $1ιών\n";
            print "επίθ. αιτι. πληθ. θηλ. $1ιές\n";
            print "επίθ. ονομ. ενικ. ουδ. $1ί\n";
            print "επίθ. γενι. ενικ. ουδ. $1ιού\n";
            print "επίθ. αιτι. ενικ. ουδ. $1ί\n";
            print "επίθ. ονομ. πληθ. ουδ. $1ιά\n";
            print "επίθ. γενι. πληθ. ουδ. $1ιών\n";
            print "επίθ. αιτι. πληθ. ουδ. $1ιά\n";
            next;                                }

if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ων\n/g) {
            print "επίθ. ονομ. ενικ. αρσε. $1ων\n";
            print "επίθ. γενι. ενικ. αρσε. $1οντος\n";
            print "επίθ. αιτι. ενικ. αρσε. $1ων\n";
            print "επίθ. ονομ. πληθ. αρσε. $1οντες\n";
            print "επίθ. γενι. πληθ. αρσε. $1όντων\n";
            print "επίθ. αιτι. πληθ. αρσε. $1οντες\n";
            print "επίθ. ονομ. ενικ. θηλ. $1ουσα\n";
            print "επίθ. γενι. ενικ. θηλ. $1ουσας\n";
```

```
            print "επίθ. αιτι. ενικ. θηλ. $1ουσα\n";
            print "επίθ. ονομ. πληθ. θηλ. $1ουσες\n";
            print "επίθ. γενι. πληθ. θηλ. $1ουσών\n";
            print "επίθ. αιτι. πληθ. θηλ. $1ουσες\n";
            print "επίθ. ονομ. ενικ. ουδ. $1ον\n";
            print "επίθ. γενι. ενικ. ουδ. $1οντος\n";
            print "επίθ. αιτι. ενικ. ουδ. $1ον\n";
            print "επίθ. ονομ. πληθ. ουδ. $1οντα\n";
            print "επίθ. γενι. πληθ. ουδ. $1όντων\n";
            print "επίθ. αιτι. πληθ. ουδ. $1οντα\n";
            next;                              }


if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίος\n/g) {
            print "επίθ. ονομ. ενικ. αρσε. $1αίος\n";
            print "επίθ. γενι. ενικ. αρσε. $1αίου\n";
            print "επίθ. αιτι. ενικ. αρσε. $1αίο\n";
            print "επίθ. ονομ. πληθ. αρσε. $1αίοι\n";
            print "επίθ. γενι. πληθ. αρσε. $1αίων\n";
            print "επίθ. αιτι. πληθ. αρσε. $1αίους\n";
            print "επίθ. ονομ. ενικ. θηλ. $1αία\n";
            print "επίθ. γενι. ενικ. θηλ. $1αίας\n";
            print "επίθ. αιτι. ενικ. θηλ. $1αία\n";
        print "επίθ. ονομ. πληθ. θηλ. $1αίες\n";
            print "επίθ. γενι. πληθ. θηλ. $1αίων\n";
            print "επίθ. αιτι. πληθ. θηλ. $1αίες\n";
            print "επίθ. ονομ. ενικ. ουδ. $1αίο\n";
            print "επίθ. γενι. ενικ. ουδ. $1αίου\n";
            print "επίθ. αιτι. ενικ. ουδ. $1αίο\n";
            print "επίθ. ονομ. πληθ. ουδ. $1αία\n";
            print "επίθ. γενι. πληθ. ουδ. $1αίων\n";
            print "επίθ. αιτι. πληθ. ουδ. $1αία\n";
            next;                              }

if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ούχος\n/g) {
            print "επίθ. ονομ. ενικ. αρσε. $1ούχος\n";
            print "επίθ. γενι. ενικ. αρσε. $1ούχου\n";
            print "επίθ. αιτι. ενικ. αρσε. $1ούχο\n";
            print "επίθ. ονομ. πληθ. αρσε. $1ούχοι\n";
            print "επίθ. γενι. πληθ. αρσε. $1ούχων\n";
            print "επίθ. αιτι. πληθ. αρσε. $1ούχους\n";
            print "επίθ. ονομ. ενικ. θηλ. $1ούχα\n";
            print "επίθ. γενι. ενικ. θηλ. $1ούχας\n";
            print "επίθ. αιτι. ενικ. θηλ. $1ούχα\n";
            print "επίθ. ονομ. πληθ. θηλ. $1ούχες\n";
            print "επίθ. γενι. πληθ. θηλ. $1ούχων\n";
            print "επίθ. αιτι. πληθ. θηλ. $1ούχες\n";
            print "επίθ. ονομ. ενικ. ουδ. $1ούχο\n";
            print "επίθ. γενι. ενικ. ουδ. $1ούχου\n";
            print "επίθ. αιτι. ενικ. ουδ. $1ούχο\n";
            print "επίθ. ονομ. πληθ. ουδ. $1ούχα\n";
            print "επίθ. γενι. πληθ. ουδ. $1ούχων\n";
            print "επίθ. αιτι. πληθ. ουδ. $1ούχα\n";
            next;          }


if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)οίος\n/g) {
            print "επίθ. ονομ. ενικ. αρσε. $1οίος\n";
            print "επίθ. γενι. ενικ. αρσε. $1οίου\n";
            print "επίθ. αιτι. ενικ. αρσε. $1οίο\n";
```

```
            print "επίθ. ονομ. πληθ. αρσε. $1οίοι\n";
            print "επίθ. γενι. πληθ. αρσε. $1οίων\n";
            print "επίθ. αιτι. πληθ. αρσε. $1οίους\n";
            print "επίθ. ονομ. ενικ. θηλ. $1οία\n";
            print "επίθ. γενι. ενικ. θηλ. $1οίας\n";
            print "επίθ. αιτι. ενικ. θηλ. $1οία\n";
        print "επίθ. ονομ. πληθ. θηλ. $1οίες\n";
            print "επίθ. γενι. πληθ. θηλ. $1οίων\n";
            print "επίθ. αιτι. πληθ. θηλ. $1οίες\n";
            print "επίθ. ονομ. ενικ. ουδ. $1οίο\n";
            print "επίθ. γενι. ενικ. ουδ. $1οίου\n";
            print "επίθ. αιτι. ενικ. ουδ. $1οίο\n";
            print "επίθ. ονομ. πληθ. ουδ. $1οία\n";
            print "επίθ. γενι. πληθ. ουδ. $1οίων\n";
            print "επίθ. αιτι. πληθ. ουδ. $1οία\n";
            next;          }

    if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)είος\n/g) {
            print "επίθ. ονομ. ενικ. αρσε. $1είος\n";
            print "επίθ. γενι. ενικ. αρσε. $1είου\n";
            print "επίθ. αιτι. ενικ. αρσε. $1είο\n";
            print "επίθ. ονομ. πληθ. αρσε. $1είοι\n";
            print "επίθ. γενι. πληθ. αρσε. $1είων\n";
            print "επίθ. αιτι. πληθ. αρσε. $1είους\n";
            print "επίθ. ονομ. ενικ. θηλ. $1εία\n";
            print "επίθ. γενι. ενικ. θηλ. $1είας\n";
            print "επίθ. αιτι. ενικ. θηλ. $1εία\n";
            print "επίθ. ονομ. πληθ. θηλ. $1είες\n";
            print "επίθ. γενι. πληθ. θηλ. $1είων\n";
            print "επίθ. αιτι. πληθ. θηλ. $1είες\n";
            print "επίθ. ονομ. ενικ. ουδ. $1είο\n";
            print "επίθ. γενι. ενικ. ουδ. $1είου\n";
            print "επίθ. αιτι. ενικ. ουδ. $1είο\n";
            print "επίθ. ονομ. πληθ. ουδ. $1εία\n";
            print "επίθ. γενι. πληθ. ουδ. $1είων\n";
            print "επίθ. αιτι. πληθ. ουδ. $1εία\n";
            next;          }

    if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ειος\n/g) {
            print "επίθ. ονομ. ενικ. αρσε. $1ειος\n";
            print "επίθ. γενι. ενικ. αρσε. $1ειου\n";
            print "επίθ. αιτι. ενικ. αρσε. $1ειο\n";
            print "επίθ. ονομ. πληθ. αρσε. $1ειοι\n";
            print "επίθ. γενι. πληθ. αρσε. $1ειων\n";
            print "επίθ. αιτι. πληθ. αρσε. $1ειους\n";
            print "επίθ. ονομ. ενικ. θηλ. $1εια\n";
            print "επίθ. γενι. ενικ. θηλ. $1ειας\n";
            print "επίθ. αιτι. ενικ. θηλ. $1εια\n";
            print "επίθ. ονομ. πληθ. θηλ. $1ειες\n";
            print "επίθ. γενι. πληθ. θηλ. $1ειων\n";
            print "επίθ. αιτι. πληθ. θηλ. $1ειες\n";
            print "επίθ. ονομ. ενικ. ουδ. $1ειο\n";
            print "επίθ. γενι. ενικ. ουδ. $1ειου\n";
            print "επίθ. αιτι. ενικ. ουδ. $1ειο\n";
            print "επίθ. ονομ. πληθ. ουδ. $1εια\n";
            print "επίθ. γενι. πληθ. ουδ. $1ειων\n";
            print "επίθ. αιτι. πληθ. ουδ. $1εια\n";
            next;          }

    if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιος\n/g) {
            print "επίθ. ονομ. ενικ. αρσε. $1ιος\n";
```

```
            print "επίθ. γενι. ενικ. αρσε. $1ιου\n";
            print "επίθ. αιτι. ενικ. αρσε. $1ιο\n";
            print "επίθ. ονομ. πληθ. αρσε. $1ιοι\n";
            print "επίθ. γενι. πληθ. αρσε. $1ιων\n";
            print "επίθ. αιτι. πληθ. αρσε. $1ιους\n";
            print "επίθ. ονομ. ενικ. θηλ. $1ια\n";
            print "επίθ. γενι. ενικ. θηλ. $1ιας\n";
            print "επίθ. αιτι. ενικ. θηλ. $1ια\n";
            print "επίθ. ονομ. πληθ. θηλ. $1ιες\n";
            print "επίθ. γενι. πληθ. θηλ. $1ιων\n";
            print "επίθ. αιτι. πληθ. θηλ. $1ιες\n";
            print "επίθ. ονομ. ενικ. ουδ. $1ιο\n";
            print "επίθ. γενι. ενικ. ουδ. $1ιου\n";
            print "επίθ. αιτι. ενικ. ουδ. $1ιο\n";
            print "επίθ. ονομ. πληθ. ουδ. $1ια\n";
            print "επίθ. γενι. πληθ. ουδ. $1ιων\n";
        print "επίθ. αιτι. πληθ. ουδ. $1ια\n";
            next;          }

if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ικος\n/g) {
            print "επίθ. ονομ. ενικ. αρσε. $1ικος\n";
            print "επίθ. γενι. ενικ. αρσε. $1ικου\n";
            print "επίθ. αιτι. ενικ. αρσε. $1ικο\n";
            print "επίθ. ονομ. πληθ. αρσε. $1ικοι\n";
            print "επίθ. γενι. πληθ. αρσε. $1ικων\n";
            print "επίθ. αιτι. πληθ. αρσε. $1ικους\n";
            print "επίθ. ονομ. ενικ. θηλ. $1ικη\n";
            print "επίθ. γενι. ενικ. θηλ. $1ικης\n";
            print "επίθ. αιτι. ενικ. θηλ. $1ικη\n";
            print "επίθ. ονομ. πληθ. θηλ. $1ικες\n";
            print "επίθ. γενι. πληθ. θηλ. $1ικων\n";
            print "επίθ. αιτι. πληθ. θηλ. $1ικες\n";
            print "επίθ. ονομ. ενικ. ουδ. $1ικο\n";
            print "επίθ. γενι. ενικ. ουδ. $1ικου\n";
            print "επίθ. αιτι. ενικ. ουδ. $1ικο\n";
            print "επίθ. ονομ. πληθ. ουδ. $1ικα\n";
            print "επίθ. γενι. πληθ. ουδ. $1ικων\n";
            print "επίθ. αιτι. πληθ. ουδ. $1ικα\n";
            next;          }

if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ικός\n/g) {
            print "επίθ. ονομ. ενικ. αρσε. $1ικός\n";
            print "επίθ. γενι. ενικ. αρσε. $1ικού\n";
            print "επίθ. αιτι. ενικ. αρσε. $1ικό\n";
            print "επίθ. ονομ. πληθ. αρσε. $1ικοί\n";
            print "επίθ. γενι. πληθ. αρσε. $1ικών\n";
            print "επίθ. αιτι. πληθ. αρσε. $1ικούς\n";
            print "επίθ. ονομ. ενικ. θηλ. $1ική\n";
            print "επίθ. γενι. ενικ. θηλ. $1ικής\n";
            print "επίθ. αιτι. ενικ. θηλ. $1ική\n";
            print "επίθ. ονομ. πληθ. θηλ. $1ικές\n";
            print "επίθ. γενι. πληθ. θηλ. $1ικών\n";
            print "επίθ. αιτι. πληθ. θηλ. $1ικές\n";
            print "επίθ. ονομ. ενικ. ουδ. $1ικό\n";
            print "επίθ. γενι. ενικ. ουδ. $1ικού\n";
            print "επίθ. αιτι. ενικ. ουδ. $1ικό\n";
            print "επίθ. ονομ. πληθ. ουδ. $1ικά\n";
            print "επίθ. γενι. πληθ. ουδ. $1ικών\n";
            print "επίθ. αιτι. πληθ. ουδ. $1ικά\n";
            next;          }
```

```
            if ($line =~ m/επίθ. ([Α-Ωα-
ωαάεέηήιίοόυύωώΰϋϋΐΐ]*)ένος\n/g) {
                  print "επίθ. ονομ. ενικ. αρσε. $1ένος\n";
                  print "επίθ. γενι. ενικ. αρσε. $1ένου\n";
                  print "επίθ. αιτι. ενικ. αρσε. $1ένο\n";
                  print "επίθ. ονομ. πληθ. αρσε. $1ένοι\n";
                  print "επίθ. γενι. πληθ. αρσε. $1ένων\n";
                  print "επίθ. αιτι. πληθ. αρσε. $1ένους\n";
                  print "επίθ. ονομ. ενικ. θηλ. $1ένη\n";
                  print "επίθ. γενι. ενικ. θηλ. $1ένης\n";
                  print "επίθ. αιτι. ενικ. θηλ. $1ένη\n";
                  print "επίθ. ονομ. πληθ. θηλ. $1ένες\n";
                  print "επίθ. γενι. πληθ. θηλ. $1ένων\n";
                  print "επίθ. αιτι. πληθ. θηλ. $1ένες\n";
                  print "επίθ. ονομ. ενικ. ουδ. $1ένο\n";
                  print "επίθ. γενι. ενικ. ουδ. $1ένου\n";
                  print "επίθ. αιτι. ενικ. ουδ. $1ένο\n";
                  print "επίθ. ονομ. πληθ. ουδ. $1ένα\n";
                  print "επίθ. γενι. πληθ. ουδ. $1ένων\n";
                  print "επίθ. αιτι. πληθ. ουδ. $1ένα\n";
                  next;          }

            if ($line =~ m/επίθ. ([Α-Ωα-
ωαάεέηήιίοόυύωώΰϋϋΐΐ]*)ητος\n/g) {
                  print "επίθ. ονομ. ενικ. αρσε. $1ητος\n";
                  print "επίθ. γενι. ενικ. αρσε. $1ητου\n";
                  print "επίθ. αιτι. ενικ. αρσε. $1ητο\n";
                  print "επίθ. ονομ. πληθ. αρσε. $1ητοι\n";
                  print "επίθ. γενι. πληθ. αρσε. $1ητων\n";
                  print "επίθ. αιτι. πληθ. αρσε. $1ητους\n";
                  print "επίθ. ονομ. ενικ. θηλ. $1ητη\n";
                  print "επίθ. γενι. ενικ. θηλ. $1ητης\n";
                  print "επίθ. αιτι. ενικ. θηλ. $1ητη\n";
                  print "επίθ. ονομ. πληθ. θηλ. $1ητες\n";
                  print "επίθ. γενι. πληθ. θηλ. $1ητων\n";
                  print "επίθ. αιτι. πληθ. θηλ. $1ητες\n";
                  print "επίθ. ονομ. ενικ. ουδ. $1ητο\n";
                  print "επίθ. γενι. ενικ. ουδ. $1ητου\n";
                  print "επίθ. αιτι. ενικ. ουδ. $1ητο\n";
                  print "επίθ. ονομ. πληθ. ουδ. $1ητα\n";
                  print "επίθ. γενι. πληθ. ουδ. $1ητων\n";
                  print "επίθ. αιτι. πληθ. ουδ. $1ητα\n";
                  next;          }

            if ($line =~ m/επίθ. ([Α-Ωα-
ωαάεέηήιίοόυύωώΰϋϋΐΐ]*)ητός\n/g) {
                  print "επίθ. ονομ. ενικ. αρσε. $1ητός\n";
                  print "επίθ. γενι. ενικ. αρσε. $1ητού\n";
                  print "επίθ. αιτι. ενικ. αρσε. $1ητό\n";
                  print "επίθ. ονομ. πληθ. αρσε. $1ητοί\n";
                  print "επίθ. γενι. πληθ. αρσε. $1ητών\n";
                  print "επίθ. αιτι. πληθ. αρσε. $1ητούς\n";
                  print "επίθ. ονομ. ενικ. θηλ. $1ητή\n";
                  print "επίθ. γενι. ενικ. θηλ. $1ητής\n";
                  print "επίθ. αιτι. ενικ. θηλ. $1ητή\n";
                  print "επίθ. ονομ. πληθ. θηλ. $1ητές\n";
                  print "επίθ. γενι. πληθ. θηλ. $1ητών\n";
                  print "επίθ. αιτι. πληθ. θηλ. $1ητές\n";
                  print "επίθ. ονομ. ενικ. ουδ. $1ητό\n";
                  print "επίθ. γενι. ενικ. ουδ. $1ητού\n";
                  print "επίθ. αιτι. ενικ. ουδ. $1ητό\n";
```

```
                print "επίθ. ονομ. πληθ. ουδ. $1ητά\n";
                print "επίθ. γενι. πληθ. ουδ. $1ητών\n";
                print "επίθ. αιτι. πληθ. ουδ. $1ητά\n";
                next;          }


          if ($line =~ m/επίθ. ([Α-Ωα-
ωαάεέηήιίοόυύωώΰϋϊΐ]*)τυπος\n/g) {
                print "επίθ. ονομ. ενικ. αρσε. $1τυπος\n";
                print "επίθ. γενι. ενικ. αρσε. $1τυπου\n";
                print "επίθ. αιτι. ενικ. αρσε. $1τυπο\n";
                print "επίθ. ονομ. πληθ. αρσε. $1τυποι\n";
                print "επίθ. γενι. πληθ. αρσε. $1τυπων\n";
                print "επίθ. αιτι. πληθ. αρσε. $1τυπους\n";
                print "επίθ. ονομ. ενικ. θηλ. $1τυπη\n";
                print "επίθ. γενι. ενικ. θηλ. $1τυπης\n";
                print "επίθ. αιτι. ενικ. θηλ. $1τυπη\n";
                print "επίθ. ονομ. πληθ. θηλ. $1τυπες\n";
                print "επίθ. γενι. πληθ. θηλ. $1τυπων\n";
                print "επίθ. αιτι. πληθ. θηλ. $1τυπες\n";
                print "επίθ. ονομ. ενικ. ουδ. $1τυπο\n";
                print "επίθ. γενι. ενικ. ουδ. $1τυπου\n";
                print "επίθ. αιτι. ενικ. ουδ. $1τυπο\n";
                print "επίθ. ονομ. πληθ. ουδ. $1τυπα\n";
                print "επίθ. γενι. πληθ. ουδ. $1τυπων\n";
                print "επίθ. αιτι. πληθ. ουδ. $1τυπα\n";
                next;          }

          if ($line =~ m/επίθ. ([Α-Ωα-
ωαάεέηήιίοόυύωώΰϋϊΐ]*)τατος\n/g) {
                print "επίθ. ονομ. ενικ. αρσε. $1τατος\n";
                print "επίθ. γενι. ενικ. αρσε. $1τατου\n";
                print "επίθ. αιτι. ενικ. αρσε. $1τατο\n";
                print "επίθ. ονομ. πληθ. αρσε. $1τατοι\n";
                print "επίθ. γενι. πληθ. αρσε. $1τατων\n";
                print "επίθ. αιτι. πληθ. αρσε. $1τατους\n";
                print "επίθ. ονομ. ενικ. θηλ. $1τατη\n";
                print "επίθ. γενι. ενικ. θηλ. $1τατης\n";
                print "επίθ. αιτι. ενικ. θηλ. $1τατη\n";
                print "επίθ. ονομ. πληθ. θηλ. $1τατες\n";
                print "επίθ. γενι. πληθ. θηλ. $1τατων\n";
                print "επίθ. αιτι. πληθ. θηλ. $1τατες\n";
                print "επίθ. ονομ. ενικ. ουδ. $1τατο\n";
                print "επίθ. γενι. ενικ. ουδ. $1τατου\n";
                print "επίθ. αιτι. ενικ. ουδ. $1τατο\n";
                print "επίθ. ονομ. πληθ. ουδ. $1τατα\n";
                print "επίθ. γενι. πληθ. ουδ. $1τατων\n";
                print "επίθ. αιτι. πληθ. ουδ. $1τατα\n";
                next;          }

          if ($line =~ m/επίθ. ([Α-Ωα-
ωαάεέηήιίοόυύωώΰϋϊΐ]*)([ια])μος\n/g) {
                print "επίθ. ονομ. ενικ. αρσε. $1$2μος\n";
                print "επίθ. γενι. ενικ. αρσε. $1$2μου\n";
                print "επίθ. αιτι. ενικ. αρσε. $1$2μο\n";
                print "επίθ. ονομ. πληθ. αρσε. $1$2μοι\n";
                print "επίθ. γενι. πληθ. αρσε. $1$2μων\n";
                print "επίθ. αιτι. πληθ. αρσε. $1$2μους\n";
                print "επίθ. ονομ. ενικ. θηλ. $1$2μη\n";
                print "επίθ. γενι. ενικ. θηλ. $1$2μης\n";
                print "επίθ. αιτι. ενικ. θηλ. $1$2μη\n";
```

```
                print "επίθ. ονομ. πληθ. θηλ. $1$2μες\n";
                print "επίθ. γενι. πληθ. θηλ. $1$2μων\n";
                print "επίθ. αιτι. πληθ. θηλ. $1$2μες\n";
                print "επίθ. ονομ. ενικ. ουδ. $1$2μο\n";
                print "επίθ. γενι. ενικ. ουδ. $1$2μου\n";
                print "επίθ. αιτι. ενικ. ουδ. $1$2μο\n";
                print "επίθ. ονομ. πληθ. ουδ. $1$2μα\n";
                print "επίθ. γενι. πληθ. ουδ. $1$2μων\n";
                print "επίθ. αιτι. πληθ. ουδ. $1$2μα\n";
                next;          }


        if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ος\n/g)
{
                print "επίθ. ονομ. ενικ. αρσε. $1ος\n";
                print "επίθ. γενι. ενικ. αρσε. $1ου\n";
                print "επίθ. αιτι. ενικ. αρσε. $1ο\n";
                print "επίθ. ονομ. πληθ. αρσε. $1οι\n";
                print "επίθ. γενι. πληθ. αρσε. $1ων\n";
                print "επίθ. αιτι. πληθ. αρσε. $1ους\n";
                print "επίθ. ονομ. ενικ. θηλ. $1η\n";
                print "επίθ. γενι. ενικ. θηλ. $1ης\n";
                print "επίθ. αιτι. ενικ. θηλ. $1η\n";
                print "επίθ. ονομ. πληθ. θηλ. $1ες\n";
                print "επίθ. γενι. πληθ. θηλ. $1ων\n";
                print "επίθ. αιτι. πληθ. θηλ. $1ες\n";
                print "επίθ. ονομ. ενικ. ουδ. $1ο\n";
                print "επίθ. γενι. ενικ. ουδ. $1ου\n";
                print "επίθ. αιτι. ενικ. ουδ. $1ο\n";
                print "επίθ. ονομ. πληθ. ουδ. $1α\n";
                print "επίθ. γενι. πληθ. ουδ. $1ων\n";
                print "επίθ. αιτι. πληθ. ουδ. $1α\n";
                next;          }

        if ($line =~ m/επίθ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ός\n/g)
{
                print "επίθ. ονομ. ενικ. αρσε. $1ός\n";
                print "επίθ. γενι. ενικ. αρσε. $1ού\n";
                print "επίθ. αιτι. ενικ. αρσε. $1ό\n";
                print "επίθ. ονομ. πληθ. αρσε. $1οί\n";
                print "επίθ. γενι. πληθ. αρσε. $1ών\n";
                print "επίθ. αιτι. πληθ. αρσε. $1ούς\n";
                print "επίθ. ονομ. ενικ. θηλ. $1ή\n";
                print "επίθ. γενι. ενικ. θηλ. $1ής\n";
                print "επίθ. αιτι. ενικ. θηλ. $1ή\n";
                print "επίθ. ονομ. πληθ. θηλ. $1ές\n";
                print "επίθ. γενι. πληθ. θηλ. $1ών\n";
                print "επίθ. αιτι. πληθ. θηλ. $1ές\n";
                print "επίθ. ονομ. ενικ. ουδ. $1ό\n";
                print "επίθ. γενι. ενικ. ουδ. $1ού\n";
                print "επίθ. αιτι. ενικ. ουδ. $1ό\n";
                print "επίθ. ονομ. πληθ. ουδ. $1ά\n";
                print "επίθ. γενι. πληθ. ουδ. $1ών\n";
                print "επίθ. αιτι. πληθ. ουδ. $1ά\n";
                next;          }


                }

close(OUT);
```

Script 7: Εμπλουτισμός του ελληνικού POS λεξικού με ρήματα

```
#!/usr/bin/perl -w

use POSIX;
setlocale(POSIX::LC_ALL,"el");

open(IN,"< ./$ARGV[0]");
open(OUT,"> ./p_more");

my $pat;
$/="\n";
@array = <IN>;
close(IN);

select(OUT);

my ($line2,$flag);


foreach $line (@array) {
print $line;

if ($line =~ m/
(συν|υπερ|μετα|ξανα|κατα|προσ|ψευτο|ριψο|μυξο|περι|παρα|καλο|κακο|πυκ
νο|πυρο|παρτσ|πανικο|δυσ|αυτο|ασημο|χαρτο|βρομο|βροντο|αργο|αντι|απορ
|διαρ)([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)/g){
      print "ρ. $2\n";
      next;
      }

if ($line =~ m/
(υπο|αντ|αξιο|ανα|ξερο|γλυκο|ακρο|χειρο|αλκαλιο|ζητω|αλληλο|αλαφρο|αε
ρο|βαρυ|επανα|επαν|επι|απο|δια)([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)/g){
      print "ρ. $2\n";
      next;
      }

if ($line =~ m/
(υπ|μετ|ξε|κατ|προ|εμ|εν|αν|εξ|εισ|απ|επ|δι|εκ|αλληλ|χειρ)([Α-Ωα-
ωαάεέηήιίοόυύωώΰϋϊΐ]*)/g){
      print "ρ. $2\n";
      next;
      }


}

close(OUT);



open(OUT,"< ./p_more");
@array1 = <OUT>;
close(OUT);
```

```
open(OUT,"> ./p_more");
select(OUT);



foreach $line (@array1) {
 print "$line";
        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)(ίζ)ομαι\n/g) {
            print "ρ. $1ίζω\n";
            next;}


       if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ίζω\n/g) {
            print "ρ. $1ίζομαι\n";
            next;}


       if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ύζω\n/g) {
            print "ρ. $1ύζομαι\n";
            next;}


         if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)(ύζ)ομαι\n/g) {
            print "ρ. $1ύζω\n";
            next;}

         if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ώζω\n/g) {
            print "ρ. $1ώζομαι\n";
            next;}


         if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ώζομαι\n/g) {
            print "ρ. $1ώζω\n";
            next;}


         if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίζομαι\n/g) {
            print "ρ. $1αίζω\n";
            next;}


         if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίζω\n/g) {
            print "ρ. $1αίζομαι\n";
            next;}


         if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)σσομαι\n/g) {
            print "ρ. $1σσω\n";
            next;}


         if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)σσω\n/g) {
            print "ρ. $1σσομαι\n";
            next;}


         if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)χνομαι\n/g) {
            print "ρ. $1χνω\n";
            next;}

         if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)χνω\n/g) {
```

```
    print "ρ. $1χνομαι\n";
    next;}

if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίνομαι\n/g) {
    print "ρ. $1αίνω\n";
    next;}

if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίνω\n/g) {
    print "ρ. $1αίνομαι\n";
    next;}



if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)νομαι\n/g) {
    print "ρ. $1νω\n";
    next;}

if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)νω\n/g) {
    print "ρ. $1νομαι\n";
    next;}

if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)δομαι\n/g) {
    print "ρ. $1δω\n";
    next;}

if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)δω\n/g) {
    print "ρ. $1δομαι\n";
    next;}


if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)όζομαι\n/g) {
    print "ρ. $1όζω\n";
    next;}


if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)όζω\n/g) {
    print "ρ. $1όζομαι\n";
    next;}

if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ήζομαι\n/g) {
    print "ρ. $1ήζω\n";
    next;}

if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ήζω\n/g) {
    print "ρ. $1ήζομαι\n";
    next;}

if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)κομαι\n/g) {
    print "ρ. $1κω\n";
    next;}

if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)κω\n/g) {
    print "ρ. $1κομαι\n";
    next;}


if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίνομαι\n/g) {
    print "ρ. $1αίνω\n";
    next;}
```

```
        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίνω\n/g) {
            print "ρ. $1αινομαι\n";
            next;}

        if ($line =~ m/([Α-Ωα-
ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιά([κγζ])ομαι\n/g) {
            print "ρ. $1ιά$2ω\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιά([κγζ])ω\n/g)
{
            print "ρ. $1ιά$2ομαι\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)άζομαι\n/g) {
            print "ρ. $1άζω\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)άζω\n/g) {
            print "ρ. $1άζομαι\n";
            next;}



        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιέζομαι\n/g) {
            print "ρ. $1ιέζω\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιέζω\n/g) {
            print "ρ. $1ιέζομαι\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)([κγζ])ομαι\n/g)
{
            print "ρ. $1$2ω\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)([κγζ])ω\n/g) {
            print "ρ. $1$2ομαι\n";
            next;}

        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιέμαι\n/g) {
            print "ρ. $1ώ\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ομαι\n/g) {
            print "ρ. $1ω\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ω\n/g) {
            print "ρ. $1ομαι\n";
            next;}
```

```
      if ($line =~ /ρ\.
(αγαπ|απαντ|βουτ|γλεντ|κεντ|κολλ|κυβερν|κυλ|κυνηγ|μελετ|νικ|ξενυχτ|πη
δ|προσδοκ|ρωτ|σταματ)ώ\n/) {
              print "ρ. $1ιέμαι\n";
            next;}

      if ($line =~ /ρ\.
(τιμ|τρυπ|τσιμπ|φυσ|χαιρετ|χτυπ|βαστ|γελ|διψ|δρ|κρεμ|ξεχν|περν|πετ|ρο
υφ|σκουντ|σπ|τραβ|χαλ)ώ\n/) {
                  print "ρ. $1ιέμαι\n";
            next;}

       if ($line =~ /ρ\.
(αγνο|αδιαφορ|αδικ|αντιστοιχ|απειλ|απορ|αργ|δημιουργ|διεκδικ|δικαιολο
γ|διοικ|ενεργ|εξαντλ|επιθυμ|επιχειρ|ευνο|ευτυχ|ζ|θεωρ|θρην|ικανοποι|κ
αλλιεργ)ώ\n/) {
                  print "ρ. $1ούμαι\n";
            next;}

         if ($line =~ /ρ\.
(κατοικ|κιν|ναυαγ|ναυπηγ|νομοθετ|πλειοδοτ|πληροφορ|ποθ|προσπαθ|προχωρ
|συγκιν|συμμαχ|συνομιλ|τηλεγραφ|τιμωρ|υμν|υπηρετ|φιλ|φορολογ|φρον|φρο
υρ|φωτογραφ|ωφελ|αποτελ|αφαιρ|εκτελ|θαρρ|καλ|μπορ|προκαλ|συναιρ)ώ\n/)
{
                  print "ρ. $1ούμαι\n";
            next;}


}



close(OUT);







open(OUT,"< ./p_more");
@array2 = <OUT>;
close(OUT);

open(OUT,"> ./p_more");
select(OUT);



foreach $line (@array2) {
 print $line;
         if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)(ίζ)ομαι\n/g) {
             print "ρ. $1ίζω\n";
             next;}


       if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ίζω\n/g) {
             print "ρ. $1ίζομαι\n";
             next;}
```

```
if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ύζω\n/g) {
      print "ρ. $1ύζομαι\n";
      next;}


  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)(ύζ)ομαι\n/g) {
      print "ρ. $1ύζω\n";
      next;}

  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ώζω\n/g) {
      print "ρ. $1ώζομαι\n";
      next;}


  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ώζομαι\n/g) {
      print "ρ. $1ώζω\n";
      next;}


  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίζομαι\n/g) {
      print "ρ. $1αίζω\n";
      next;}


  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίζω\n/g) {
      print "ρ. $1αίζομαι\n";
      next;}


  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)σσομαι\n/g) {
      print "ρ. $1σσω\n";
      next;}


  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)σσω\n/g) {
      print "ρ. $1σσομαι\n";
      next;}


  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)χνομαι\n/g) {
      print "ρ. $1χνω\n";
      next;}

  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)χνω\n/g) {
      print "ρ. $1χνομαι\n";
      next;}

  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίνομαι\n/g) {
      print "ρ. $1αίνω\n";
      next;}

  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίνω\n/g) {
      print "ρ. $1αίνομαι\n";
      next;}



  if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)νομαι\n/g) {
      print "ρ. $1νω\n";
      next;}
```

```
        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)νω\n/g) {
            print "ρ. $1νομαι\n";
            next;}

        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)δομαι\n/g) {
            print "ρ. $1δω\n";
            next;}

        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)δω\n/g) {
            print "ρ. $1δομαι\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)όζομαι\n/g) {
            print "ρ. $1όζω\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)όζω\n/g) {
            print "ρ. $1όζομαι\n";
            next;}

        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ήζομαι\n/g) {
            print "ρ. $1ήζω\n";
            next;}

        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ήζω\n/g) {
            print "ρ. $1ήζομαι\n";
            next;}

        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)κομαι\n/g) {
            print "ρ. $1κω\n";
            next;}

        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)κω\n/g) {
            print "ρ. $1κομαι\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίνομαι\n/g) {
            print "ρ. $1αίνω\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αίνω\n/g) {
            print "ρ. $1αινομαι\n";
            next;}

        if ($line =~ m/([Α-Ωα-
ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιά([κγζ])ομαι\n/g) {
            print "ρ. $1ιά$2ω\n";
            next;}


        if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιά([κγζ])ω\n/g)
{
            print "ρ. $1ιά$2ομαι\n";
            next;}
```

```
          if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)άζομαι\n/g) {
              print "ρ. $1άζω\n";
              next;}


          if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)άζω\n/g) {
              print "ρ. $1άζομαι\n";
              next;}



          if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιέζομαι\n/g) {
              print "ρ. $1ιέζω\n";
              next;}


          if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιέζω\n/g) {
              print "ρ. $1ιέζομαι\n";
              next;}


          if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)([κγζ])ομαι\n/g)
{
              print "ρ. $1$2ω\n";
              next;}


          if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)([κγζ])ω\n/g) {
              print "ρ. $1$2ομαι\n";
              next;}

          if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιέμαι\n/g) {
              print "ρ. $1ώ\n";
              next;}


          if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ομαι\n/g) {
              print "ρ. $1ω\n";
              next;}


          if ($line =~ m/([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ω\n/g) {
              print "ρ. $1ομαι\n";
              next;}


      if ($line =~ /ρ\.
(αγαπ|απαντ|βουτ|γλεντ|κεντ|κολλ|κυβερν|κυλ|κυνηγ|μελετ|νικ|ξενυχτ|πη
δ|προσδοκ|ρωτ|σταματ)ώ\n/) {
                print "ρ. $1ιέμαι\n";
              next;}

      if ($line =~ /ρ\.
(τιμ|τρυπ|τσιμπ|φυσ|χαιρετ|χτυπ|βαστ|γελ|διψ|δρ|κρεμ|ξεχν|περν|πετ|ρο
υφ|σκουντ|σπ|τραβ|χαλ)ώ\n/) {
                    print "ρ. $1ιέμαι\n";
              next;}

       if ($line =~ /ρ\.
(αγνο|αδιαφορ|αδικ|αντιστοιχ|απειλ|απορ|αργ|δημιουργ|διεκδικ|δικαιολο
```

```
γ|διοικ|ενεργ|εξαντλ|επιθυμ|επιχειρ|ευνο|ευτυχ|ζ|θεωρ|θρην|ικανοποι|κ
αλλιεργ)ώ\n/) {
                    print "ρ. $1ούμαι\n";
             next;}

         if ($line =~ /ρ\.
(κατοικ|κιν|ναυαγ|ναυπηγ|νομοθετ|πλειοδοτ|πληροφορ|ποθ|προσπαθ|προχωρ
|συγκιν|συμμαχ|συνομιλ|τηλεγραφ|τιμωρ|υμν|υπηρετ|φιλ|φορολογ|φρον|φρο
υρ|φωτογραφ|ωφελ|αποτελ|αφαιρ|εκτελ|θαρρ|καλ|μπορ|προκαλ|συναιρ)ώ\n/)
{
                    print "ρ. $1ούμαι\n";
             next;}


}

close(OUT);
```

Script 8: Εμπλουτισμός του ελληνικού POS λεξικού με ουσιαστικά

```
#!/usr/bin/perl -w

use POSIX;
setlocale(POSIX::LC_ALL,"el");

open(IN,"< $ARGV[0]");
open(OUT,"> ./output3");


$/="\n";
@array = <IN>;
close(IN);

select(OUT);

foreach $line (@array) {
      print $line;

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)αν\n/g) {
                  print "ουσ. ονομ. πληθ. $1αντα\n";
                  print "ουσ. γενι. πληθ. $1άντων\n";
                  print "ουσ. αιτι. πληθ. $1αντα\n";
#********** KLHSEIS *************
                  print "ουσ. γενι. ενικ. $1αντος\n";
                  print "ουσ. αιτι. ενικ. $1αν\n";

                  next;                              }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)άν\n/g) {
                  print "ουσ. ονομ. πληθ. $1άντα\n";
                  print "ουσ. γενι. πληθ. $1άντων\n";
                  print "ουσ. αιτι. πληθ. $1άντα\n";
#********** KLHSEIS *************
                  print "ουσ. γενι. ενικ. $1αντός\n";
                  print "ουσ. αιτι. ενικ. $1άν\n";

                  next;                              }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ον\n/g) {
                  print "ουσ. ονομ. πληθ. $1οντα\n";
                  print "ουσ. γενι. πληθ. $1όντων\n";
                  print "ουσ. αιτι. πληθ. $1οντα\n";
#********** KLHSEIS *************
                  print "ουσ. γενι. ενικ. $1οντος\n";
                  print "ουσ. αιτι. ενικ. $1ον\n";

                  next;                              }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)όν\n/g) {
                  print "ουσ. ονομ. πληθ. $1όντα\n";
                  print "ουσ. γενι. πληθ. $1όντων\n";
                  print "ουσ. αιτι. πληθ. $1όντα\n";
#********** KLHSEIS *************
                  print "ουσ. γενι. ενικ. $1όντος\n";
```

```
                print "ουσ. αιτι. ενικ. $1όν\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ού\n/g) {
                print "ουσ. ονομ. πληθ. $1οϋδες\n";
                print "ουσ. γενι. πληθ. $1ούδων\n";
                print "ουσ. αιτι. πληθ. $1ούδες\n";
#********** KLHSEIS *************
                print "ουσ. ονομ. ενικ. $1ούς\n";
                print "ουσ. αιτι. ενικ. $1ού\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ούς\n/g) {
                print "ουσ. ονομ. πληθ. $1ούδες\n";
                print "ουσ. γενι. πληθ. $1ούδων\n";
                print "ουσ. αιτι. πληθ. $1ούδες\n";
#********** KLHSEIS *************
                print "ουσ. ονομ. ενικ. $1ού\n";
                print "ουσ. αιτι. ενικ. $1ού\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ες\n/g) {
                print "ουσ. ονομ. πληθ. $1ηδες\n";
                print "ουσ. γενι. πληθ. $1ηδων\n";
                print "ουσ. αιτι. πληθ. $1ηδες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1η\n";
                print "ουσ. αιτι. ενικ. $1η\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ές\n/g) {
                print "ουσ. ονομ. πληθ. $1έδες\n";
                print "ουσ. γενι. πληθ. $1έδων\n";
                print "ουσ. αιτι. πληθ. $1έδες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1έ\n";
                print "ουσ. αιτι. ενικ. $1έ\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)έας\n/g) {
                print "ουσ. ονομ. πληθ. $1είς\n";
                print "ουσ. γενι. πληθ. $1έων\n";
                print "ουσ. αιτι. πληθ. $1είς\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1έα\n";
                print "ουσ. αιτι. ενικ. $1έα\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ίας\n/g) {
                print "ουσ. ονομ. πληθ. $1ίες\n";
                print "ουσ. γενι. πληθ. $1ιών\n";
                print "ουσ. αιτι. πληθ. $1ίες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ία\n";
```

```
                print "ουσ. αιτι. ενικ. $1ία\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)άς\n/g) {
                print "ουσ. ονομ. πληθ. $1άδες\n";
                print "ουσ. γενι. πληθ. $1άδων\n";
                print "ουσ. αιτι. πληθ. $1άδες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ά\n";
                print "ουσ. αιτι. ενικ. $1ά\n";
                next;                                   }


      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ας\n/g) {
                print "ουσ. ονομ. πληθ. $1ες\n";
                print "ουσ. γενι. πληθ. $1ων\n";
                print "ουσ. αιτι. πληθ. $1ες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1α\n";
                print "ουσ. αιτι. ενικ. $1α\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ούρα\n/g) {
                print "ουσ. ονομ. πληθ. $1ούρες\n";
                print "ουσ. γενι. πληθ. $1ουρών\n";
                print "ουσ. αιτι. πληθ. $1ούρες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ούρας\n";
                print "ουσ. αιτι. ενικ. $1ούρα\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ίλα\n/g) {
                print "ουσ. ονομ. πληθ. $1ίλες\n";
                print "ουσ. γενι. πληθ. $1ιλών\n";
                print "ουσ. αιτι. πληθ. $1ιλες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ίλας\n";
                print "ουσ. αιτι. ενικ. $1ιλα\n";

                next;                                   }


      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ίτσα\n/g) {
                print "ουσ. ονομ. πληθ. $1ίτσες\n";
                print "ουσ. γενι. πληθ. $1ιτσών\n";
                print "ουσ. αιτι. πληθ. $1ίτσες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ίτσας\n";
                print "ουσ. αιτι. ενικ. $1ίτσα\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ούδα\n/g) {
                print "ουσ. ονομ. πληθ. $1ούδες\n";
                print "ουσ. γενι. πληθ. $1ουδών\n";
                print "ουσ. αιτι. πληθ. $1ούδες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ούδας\n";
```

```
                print "ουσ. αιτι. ενικ. $1ούδα\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ούλα\n/g) {
                print "ουσ. ονομ. πληθ. $1ούλες\n";
                print "ουσ. γενι. πληθ. $1ουλών\n";
                print "ουσ. αιτι. πληθ. $1ούλες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ούλας\n";
                print "ουσ. αιτι. ενικ. $1ούλα\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)τρα\n/g) {
                print "ουσ. ονομ. πληθ. $1τρες\n";
                print "ουσ. γενι. πληθ. $1τρών\n";
                print "ουσ. αιτι. πληθ. $1τρες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1τρας\n";
                print "ουσ. αιτι. ενικ. $1τρα\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ίνα\n/g) {
                print "ουσ. ονομ. πληθ. $1ίνες\n";
                print "ουσ. γενι. πληθ. $1ινών\n";
                print "ουσ. αιτι. πληθ. $1ινες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ίνας\n";
                print "ουσ. αιτι. ενικ. $1ίνα\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ισσα\n/g) {
                print "ουσ. ονομ. πληθ. $1ισσες\n";
                print "ουσ. γενι. πληθ. $1ισσών\n";
                print "ουσ. αιτι. πληθ. $1ισσες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ισσας\n";
                print "ουσ. αιτι. ενικ. $1ισσα\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)τρια\n/g) {
                print "ουσ. ονομ. πληθ. $1τριες\n";
                print "ουσ. γενι. πληθ. $1τριών\n";
                print "ουσ. αιτι. πληθ. $1τριες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1τριας\n";
                print "ουσ. αιτι. ενικ. $1τρια\n";

                next;                                   }

          if ($line =~ m/ουσ. ([Α-Ωα-
ωαάεέηήιίοόυύωώΰϋϊΐ]*)αινα\n/g) {
                print "ουσ. ονομ. πληθ. $1αινες\n";
                print "ουσ. γενι. πληθ. $1αινών\n";
                print "ουσ. αιτι. πληθ. $1αινες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1αινας\n";
```

```
                print "ουσ. αιτι. ενικ. $1αινα\n";

                next;                                   }


      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ευμα\n/g) {
                print "ουσ. ονομ. πληθ. $1εύματα\n";
                print "ουσ. γενι. πληθ. $1ευμάτων\n";
                print "ουσ. αιτι. πληθ. $1εύματα\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1εύματος\n";
                print "ουσ. αιτι. ενικ. $1ευμα\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ίδα\n/g) {
                print "ουσ. ονομ. πληθ. $1ίδες\n";
                print "ουσ. γενι. πληθ. $1ίδων\n";
                print "ουσ. αιτι. πληθ. $1ίδες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ίδας\n";
                print "ουσ. αιτι. ενικ. $1ίδα\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ώνας\n/g) {
                print "ουσ. ονομ. πληθ. $1ώνες\n";
                print "ουσ. γενι. πληθ. $1ώνων\n";
                print "ουσ. αιτι. πληθ. $1ώνες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ώνα\n";
                print "ουσ. αιτι. ενικ. $1ώνα\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ότητα\n/g) {
                print "ουσ. ονομ. πληθ. $1ότητες\n";
                print "ουσ. γενι. πληθ. $1οτήτων\n";
                print "ουσ. αιτι. πληθ. $1ότητες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ότητας\n";
                print "ουσ. αιτι. ενικ. $1οτητα\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)όλη\n/g) {
                print "ουσ. ονομ. πληθ. $1όλες\n";
                print "ουσ. γενι. πληθ. $1όλων\n";
                print "ουσ. αιτι. πληθ. $1όλες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1όλης\n";
                print "ουσ. αιτι. ενικ. $1όλη\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ολή\n/g) {
                print "ουσ. ονομ. πληθ. $1ολές\n";
                print "ουσ. γενι. πληθ. $1ολών\n";
                print "ουσ. αιτι. πληθ. $1ολές\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ολής\n";
                print "ουσ. αιτι. ενικ. $1ολή\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ίνη\n/g) {
```

```
                print "ουσ. ονομ. πληθ. $1ίνες\n";
                print "ουσ. γενι. πληθ. $1ινών\n";
                print "ουσ. αιτι. πληθ. $1ίνες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ίνης\n";
                print "ουσ. αιτι. ενικ. $1ίνη\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέήιίοόυύωώΰϋϊΐ]*)ίστης\n/g) {
                print "ουσ. ονομ. πληθ. $1ίστες\n";
                print "ουσ. γενι. πληθ. $1ιστών\n";
                print "ουσ. αιτι. πληθ. $1ίστες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ίστη\n";
                print "ουσ. αιτι. ενικ. $1ίστη\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέήιίοόυύωώΰϋϊΐ]*)ιστής\n/g) {
                print "ουσ. ονομ. πληθ. $1ιστές\n";
                print "ουσ. γενι. πληθ. $1ιστών\n";
                print "ουσ. αιτι. πληθ. $1ιστές\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ιστή\n";
                print "ουσ. αιτι. ενικ. $1ιστή\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέήιίοόυύωώΰϋϊΐ]*)ίτης\n/g) {
                print "ουσ. ονομ. πληθ. $1ίτες\n";
                print "ουσ. γενι. πληθ. $1ιτών\n";
                print "ουσ. αιτι. πληθ. $1ίτες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ίτη\n";
                print "ουσ. αιτι. ενικ. $1ίτη\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέήιίοόυύωώΰϋϊΐ]*)άκης\n/g) {
                print "ουσ. ονομ. πληθ. $1άκηδες\n";
                print "ουσ. γενι. πληθ. $1άκηδων\n";
                print "ουσ. αιτι. πληθ. $1άκηδες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1άκη\n";
                print "ουσ. αιτι. ενικ. $1άκη\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέήιίοόυύωώΰϋϊΐ]*)άρης\n/g) {
                print "ουσ. ονομ. πληθ. $1άρηδες\n";
                print "ουσ. γενι. πληθ. $1άρηδων\n";
                print "ουσ. αιτι. πληθ. $1άρηδες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1άρη\n";
                print "ουσ. αιτι. ενικ. $1άρη\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέήιίοόυύωώΰϋϊΐ]*)τζής\n/g) {
                print "ουσ. ονομ. πληθ. $1τζήδες\n";
                print "ουσ. γενι. πληθ. $1τζήδων\n";
                print "ουσ. αιτι. πληθ. $1τζήδες\n";
#********** KLHSEIS *************
```

```
                print "ουσ. γενι. ενικ. $1τζή\n";
                print "ουσ. αιτι. ενικ. $1τζή\n";

                next;                                   }


      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ης\n/g) {
                print "ουσ. ονομ. πληθ. $1ες\n";
                print "ουσ. γενι. πληθ. $1ών\n";
                print "ουσ. αιτι. πληθ. $1ες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1η\n";
                print "ουσ. αιτι. ενικ. $1η\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ής\n/g) {
                print "ουσ. ονομ. πληθ. $1ές\n";
                print "ουσ. γενι. πληθ. $1ών\n";
                print "ουσ. αιτι. πληθ. $1ές\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ή\n";
                print "ουσ. αιτι. ενικ. $1ή\n";

                next;                                   }


      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)([ψξσ])η\n/g)
{
                print "ουσ. ονομ. πληθ. $1$2εις\n";
                print "ουσ. γενι. πληθ. $1$2εων\n";
                print "ουσ. αιτι. πληθ. $1$2εις\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1$2ης\n";
                print "ουσ. γενι. ενικ. $1$2εως\n";
                print "ουσ. αιτι. ενικ. $1$2η\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-
ωαάεέηήιίοόυύωώΰϋϊΐ]*)([ξψσ])ιμο\n/g) {
                print "ουσ. ονομ. πληθ. $1$2ίματα\n";
                print "ουσ. γενι. πληθ. $1$2ιμάτων\n";
                print "ουσ. αιτι. πληθ. $1$2ίματα\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1$2ίματος\n";
                print "ουσ. αιτι. ενικ. $1$2ιμάτων\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)μα\n/g) {
                print "ουσ. ονομ. πληθ. $1ματα\n";
                print "ουσ. γενι. πληθ. $1μάτων\n";
                print "ουσ. αιτι. πληθ. $1ματα\n";

#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ματος\n";
                print "ουσ. αιτι. ενικ. $1μα\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ο\n/g) {
                print "ουσ. ονομ. πληθ. $1α\n";
                print "ουσ. γενι. πληθ. $1ων\n";
```

```
                print "ουσ. αιτι. πληθ. $1α\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ου\n";
                print "ουσ. αιτι. ενικ. $1ο\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ό\n/g) {
                print "ουσ. ονομ. πληθ. $1ά\n";
                print "ουσ. γενι. πληθ. $1ών\n";
                print "ουσ. αιτι. πληθ. $1ά\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ού\n";
                print "ουσ. αιτι. ενικ. $1ό\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ι\n/g) {
                print "ουσ. ονομ. πληθ. $1ια\n";
                print "ουσ. γενι. πληθ. $1ιών\n";
                print "ουσ. αιτι. πληθ. $1ια\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ιού\n";
                print "ουσ. αιτι. ενικ. $1ι\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ί\n/g) {
                print "ουσ. ονομ. πληθ. $1ιά\n";
                print "ουσ. γενι. πληθ. $1ιών\n";
                print "ουσ. αιτι. πληθ. $1ιά\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ιού\n";
                print "ουσ. αιτι. ενικ. $1ί\n";
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ος\n/g) {
                print "ουσ. ονομ. πληθ. $1οι\n";
                print "ουσ. γενι. πληθ. $1ων\n";
                print "ουσ. αιτι. πληθ. $1ους\n";
#********** KLHSEIS *************
                print "ουσ. ονομ. ενικ. $1ου\n";
                print "ουσ. αιτι. ενικ. $1ο\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ός\n/g) {
                print "ουσ. ονομ. πληθ. $1οί\n";
                print "ουσ. γενι. πληθ. $1ών\n";
                print "ουσ. αιτι. πληθ. $1ούς\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ού\n";
                print "ουσ. αιτι. ενικ. $1ό\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ύτητα\n/g) {
                print "ουσ. ονομ. πληθ. $1ύτητες\n";
                print "ουσ. γενι. πληθ. $1υτήτων\n";
                print "ουσ. αιτι. πληθ. $1ύτητες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ύτητας\n";
                print "ουσ. αιτι. ενικ. $1ύτητα\n";
```

```
                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ή\n/g) {
                print "ουσ. ονομ. πληθ. $1ές\n";
                print "ουσ. γενι. πληθ. $1ών\n";
                print "ουσ. αιτι. πληθ. $1ές\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ής\n";
                print "ουσ. αιτι. ενικ. $1ή\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)η\n/g) {
                print "ουσ. ονομ. πληθ. $1ες\n";
                print "ουσ. γενι. πληθ. $1ών\n";
                print "ουσ. αιτι. πληθ. $1ες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ης\n";
                print "ουσ. αιτι. ενικ. $1η\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ία\n/g) {
                print "ουσ. ονομ. πληθ. $1ίες\n";
                print "ουσ. γενι. πληθ. $1ιών\n";
                print "ουσ. αιτι. πληθ. $1ίες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ίας\n";
                print "ουσ. αιτι. ενικ. $1ία\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ιά\n/g) {
                print "ουσ. ονομ. πληθ. $1ιές\n";
                print "ουσ. γενι. πληθ. $1ιών\n";
                print "ουσ. αιτι. πληθ. $1ιές\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ιάς\n";
                print "ουσ. αιτι. ενικ. $1ιά\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ού\n/g) {
                print "ουσ. ονομ. πληθ. $1ούδες\n";
                print "ουσ. γενι. πληθ. $1ούδων\n";
                print "ουσ. αιτι. πληθ. $1ούδες\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1ούς\n";
                print "ουσ. αιτι. ενικ. $1ού\n";

                next;                                   }

      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)ά\n/g) {
                print "ουσ. ονομ. πληθ. $1ές\n";
                print "ουσ. γενι. πληθ. $1ών\n";
                print "ουσ. αιτι. πληθ. $1ά\n";
#********** KLHSEIS *************
                print "ουσ. γενι. ενικ. $1άς\n";
                print "ουσ. αιτι. ενικ. $1ά\n";

                next;                                   }
```

```
      if ($line =~ m/ουσ. ([Α-Ωα-ωαάεέηήιίοόυύωώΰϋϊΐ]*)α\n/g) {
                  print "ουσ. ονομ. πληθ. $1ες\n";
                  print "ουσ. γενι. πληθ. $1ών\n";
                  print "ουσ. αιτι. πληθ. $1ες\n";
#********** KLHSEIS *************
                  print "ουσ. γενι. ενικ. $1ας\n";
                  print "ουσ. αιτι. ενικ. $1α\n";

                  next;                              }



                  }

close(OUT);
```

H.3 *Κώδικες από scripts που χρησιμοποιήθηκαν στην δημιουργία των μοντέλων σε συνδυασμό με το SRILM toolkit*

Script 9: Υπολογισμός των πιθανοτήτων των expansions των κλάσεων

```
#!/usr/local/net/bin/perl

use POSIX;
setlocale(POSIX::LC_ALL,"el");


open(IN2,"< ./class_file_23K"); #class_words_uniq


$/="\n";
@array = <IN2>;

close(IN2);

%allclasses=();

            foreach $line(@array) {

                                ($class,$word,$count)=split(" ",$line);
                                chomp($count);

                                                        if
(!exists($allclasses{$class})) {

      $allclasses{$class}=0;
                                                                  }

      $allclasses{$class}=$allclasses{$class}+$count;


                  }

open(IN2,"< ./class_file_23K"); #class_words_uniq
@array2 = <IN2>;
close(IN2);


foreach $line2(@array2) {
                                ($class2,$word2,$count2)=split("
",$line2);
                                      chomp $count2;

$div=($count2/$allclasses{$class2});
                                      print "$class2 $div $word2\n";

                  }
```

Script 10: Δημιουργία PFSG μοντέλου από ARPA μοντέλο

```
#!/usr/local/bin/gawk -f
#
# make-ngram-pfsg --
#     Create a Decipher PFSG from an ARPA language model
#
# usage: make-ngram-pfsg [debug=1] [check_bows=1] [maxorder=N]
backoff-lm > pfsg
#
# $Header: /home/srilm/devel/utils/src/RCS/make-ngram-pfsg,v 1.19
2000/07/11 06:12:13 stolcke Exp $
#

##########################################
#
# Output format specific code
#

BEGIN {
      logscale = 2.30258509299404568402 * 10000.5;
      round = 0.5;
      start_tag = "<s>";
      end_tag = "</s>";

      getline pid < "/dev/pid";
      close("/dev/pid");
      tmpfile = "tmp.pfsg.trans." pid;
      debug = 0;
}

function rint(x) {
      if (x < 0) {
          return int(x - round);
      } else {
          return int(x + round);
      }
}

function scale_log(x) {
      return rint(x * logscale);
}

function output_for_node(name) {
      if (name == start_tag || name == end_tag) {
          return "NULL";
      } else if (name == bo_name || match(name, bo_name " ") == 1) {
          return "NULL";
      } else if (k = index(name, " ")) {
          name2 = substr(name, k+1);
          if (name2 == end_tag) {
            return "NULL";
          } else {
            return name2;
          }
```

```
      } else {
          return name;
      }
}

function node_exists(name) {
      return (name in node_num);
}

function node_index(name) {
      i = node_num[name];
      if (i == "") {
          i = num_nodes ++;
          node_num[name] = i;
          node_string[i] = output_for_node(name);

          if (debug) {
            print "node " i " = " name ", output = " node_string[i] \
                        > "/dev/stderr";
          }
      }
      return  i;
}

function start_grammar(name) {
      num_trans = 0;
      num_nodes = 0;
      return;
}

function end_grammar(name) {
      if (!node_exists(start_tag)) {
            print start_tag " tag undefined in LM" > "/dev/stderr";
            exit(1);
      } else if (!node_exists(end_tag)) {
            print end_tag " tag undefined in LM" > "/dev/stderr";
            exit(1);
      }

      printf "%d pfsg nodes\n", num_nodes > "/dev/stderr";
      printf "%d pfsg transitions\n", num_trans > "/dev/stderr";

      print "name " name;
      printf "nodes %s", num_nodes;
      for (i = 0; i < num_nodes; i ++) {
            printf " %s", node_string[i];
      }
      printf "\n";

      print "initial " node_index(start_tag);
      print "final " node_index(end_tag);
      print "transitions " num_trans;
      fflush();

      close(tmpfile);
      system("/bin/cat " tmpfile "; /bin/rm -f " tmpfile);
}

function add_trans(from, to, prob) {
      num_trans ++;
```

```
      print node_index(from), node_index(to), scale_log(prob) >
tmpfile;
}

##########################################
#
# Generic code for parsing backoff file
#

BEGIN {
      maxorder = 3;
      grammar_name = "PFSG";
      bo_name = "BO";
      check_bows = 0;
      epsilon = 1e-5;         # tolerance for lowprob detection
}

NR == 1 {
      start_grammar(grammar_name);
}

NF == 0 {
      next;
}

/^ngram *[0-9][0-9]*=/ {
      num_grams = substr($2,index($2,"=")+1);
      if (num_grams > 0) {
          order = substr($2,1,index($2,"=")-1);
          if (order > maxorder) {
            print "ngram order exceeds maximum (" maxorder ")" \
                        > "/dev/stderr";
            exit(1);
          }
          if (order == 2) {
            grammar_name = "BIGRAM_PFSG";
          } if (order == 3) {
            grammar_name = "TRIGRAM_PFSG";
          }
      }
      next;
}

/^\\[0-9]-grams:/ {
      currorder=substr($0,2,1);
      next;
}
/^\\/ {
      next;
}

#
# unigrams
#
currorder == 1 {
      uniprob = $1;
      word = $2;
      bow = $3;

      uni_prob[word] = uniprob;
```

```
      if (word != end_tag) {
          #
          # we always need a transition out of a unigram node
          # (implicit bow = 1)
          #
          if (bow == "") {
            bow = 0;
          } else {
            uni_bows[word] = bow;
          }

          if (order > 2) {
            add_trans(bo_name " " word, bo_name, bow);
              add_trans(word, bo_name " " word, 0);
          } else {
            add_trans(word, bo_name, bow);
          }
      }

      if (word != start_tag) {
          add_trans(bo_name, word, uniprob);
      }
}

#
# bigrams
#
currorder == 2 {
      biprob = $1;
      word1 = $2;
      word2 = $3;
      bow = $4;

      if (word2 == start_tag) {
          printf "warning: ignoring bigram into start tag %s ->
%s\n", \
                  word1, word2 > "/dev/stderr";
          next;
      }

      word12 = $2 " " $3;

      if (check_bows && order > 2) {
          bi_prob[word12] = biprob;
      }

      if (bow != "" && order > 2) {
          bi_bows[word12] = bow;

          add_trans(word12, bo_name " " word2, bow);
          add_trans(bo_name " " word1, word12, biprob);
      } else {
          if (order > 2) {
            add_trans(bo_name " " word1, word2, biprob);
          } else {
            add_trans(word1, word2, biprob);
          }
      }

      if (check_bows && \
          word2 in uni_prob && \
```

```
          uni_prob[word2] + uni_bows[word1] - biprob > epsilon)
      {
          printf "warning: bigram loses to backoff %s -> %s\n", \
                      word1, word2 > "/dev/stderr";
      }
}

#
# trigrams
#
currorder == 3 {
      triprob = $1;
      word1 = $2;
      word2 = $3;
      word3 = $4;

      if (word3 == start_tag) {
          printf "warning: ignoring trigram into start tag %s %s ->
%s\n", \
                  word1, word2, word3 > "/dev/stderr";
          next;
      }

      word12 = word1 " " word2;
      word23 = word2 " " word3;

      if (word12 in bi_bows) {
          if (word23 in bi_bows) {
            add_trans(word12, word23, triprob);
          } else {
            add_trans(word12, word3, triprob);
          }

          if (check_bows && \
            word23 in bi_prob && \
            bi_prob[word23] + bi_bows[word12] - triprob > epsilon)
          {
            printf "warning: trigram loses to backoff %s %s -> %s\n",
\
                        word1, word2, word3 > "/dev/stderr";
          }
      }
}

END {
      end_grammar(grammar_name);
}
```

Script 11: Εκτέλεση του αναγνωριστή σε μια λίστα από wavs αρχεία

```
#!/bin/csh -fx
#
# Example usage:  go.batchrec WTW GPW PRUNE
#

if ($#argv  != 3) then
    echo "Usage: $0 WTW GPW PRUNE"
    exit -1
endif

set WTW   = $1
set GPW   = $2
set PRUNE = $3

setenv NUANCE /speech_disks/speech26/nuance/v7.0.4
source $NUANCE/SETUP -quiet

echo machine is `hostname`

set PACKAGE     = /home/pratsdim/take/enet
setenv GREEK        /speech_disks/speech26/enet/master.package.960
set TESTSET = /speech_disks/speech26/enet/tables/wavlist
set REFS    = /speech_disks/speech26/enet/tables/transcriptions-good-
new-text

# specify test parameters

#kickme run batchrec                  \
batchrec\
      -debug_level_during_init 4         \
      -debug_level 4                       \
      -memory_accounting                 \
      -package $PACKAGE            \
      -testset $TESTSET            \
      -transcriptions $REFS                \
      rec.NuanceInternal_8392=FALSE        \
      rec.PrintStats=all                   \
      rec.BacktraceFinalsOnly=FALSE        \
      rec.pass1.gp.WTW=$WTW                \
      rec.GrammarWeight=$GPW               \
      rec.Pruning=$PRUNE                   \
      rec.ConfidenceRejectionThreshold=-1  \
      -print_confidence_scores             \
      ep.PrepareForBargeIn=FALSE           \
      rec.SkipFrames=FALSE                 \
        rec.GenEpFeedback=FALSE
#this is necessary for v7.0.3 or
#v7.0.2 + Endpointer extensions due to
#a bug in Endpointer extensions
#see NUANCE-7-0-2-SP4-NT.readme for more

exit 0
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Rabiner, L.R, Juang, B.H: *Fundamentals of Speech recognition*, Prentice-Hall, 1993

[2] Lidstone, G.J.: *Note on the general case of the Bayes-Laplace formula for inductive or a-posteriori probabilities*, Transactions of the Faculty of Actuaries, 8:182-192, 1920

[3] Johnson, W.E.: *Probability: deductive and inductive problems.* Mind, 41:421-423, 1932

[4] Jeffreys, H.: *Theory of probability*, Clarendon Press, Oxford, second edition, 1948

[5] Gale, William A. and Kenneth W. Church: *Estimation procedures for language context: poor estimates are worse than none.* In COMPSTAT, Proceedings in Computational Statistics, 9th symposium, pages 69-74, Dubrovnik, Yugoslavia, September, 1990

[6] Gale, William A. and Kenneth W. Church: *What's wrong with adding one?* In N. Oostdijk and P. de Haan, editors. Corpus Based Research into Language, Rodolpi, Amsterdam, 1994

[7] Good, I.J.: *The population frequencies of species and the estimation of population parameters.* Biometrica, 40(3 and 4):237-264, 1953

[8] Gale, William A. And Geoffrey Sampson: *Good-Turing frequency estimation without tears.* Journal of Quantitative Linguistics, 2(3). To appear. 1995

[9] Jelinek, Frederick and Robert L. Mercer: *Interpolated estimation of Markov source parameters from sparse data.* In Proceedings of the Workshop on Pattern Recognition in Practice, Amsterdam, The Netherlands: North-Holland, May, 1980

[10] Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai. and Robert L. Mercer: *An estimate of an upper bound for the entropy of English.* Computational Linguistics, 18(1):31-40, March, 1992

[11] Baum, L.E.: *An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process.* Inequalities, 3:1-8, 1972

[12] Bahl, Lalit R., Frederick Jelinek, and Robert L. Mercer: *A maximum likelihood approach to continuous speech recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-5(2):179-190, March, 1983

[13] Chen, Stanley F.: *Building Probabilistic Models for Natural Language.* Ph.D. thesis, Harvard University, June, 1996

[14] Katz, Slava M. 1987: *Estimation of probabilities from sparse data for the language model component of a speech recognizer.* IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-35(3):400-401, March, 1987

[15] Ney, Hermann, Ute Essen, and Reinhard Kneser: *On structuring probabilistic dependences in stochastic language modeling*. Computer, Speech, and Language, 8:1-38, 1994

[16] Kneser, Reinhard and Hermann Ney: *Improved backing-off for m-gram language modeling.* In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, volume 1, pages 181-184, 1995

[17] C.E. Shannon and W. Weaver: *The Mathematical Theory of Communication.* The University of Illinois Press, Urbana, IL, 1964

[18] R. Tolman: *Principles of Statistical Mechanics.* Oxford, Clarendon, England, 1938

[19] F. Jelinek: *Self-organized language modeling for speech recognition*. In A. Waibel and K.-F. Lee, editors, Readings in Speech Recognition, pages 450-506. Morgan Kaufmann, San Mateo, CA, 1990

[20] M. Codogno, L. Fissore, A. Martelli, G. Pirani, and G. Volpi : *Experimental evaluation of Italian language models for large-dictionary speech recognition.* In Proceedings of the European Conference on Speech Technology, pages vol.1, 159-162, Edinburgh, September, 1987

[21] K.-S. Fu and T. L. Booth: *Grammatical inference: Introduction and survey - part 1.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 5(1):95-110, 1975

[22] K.-S. Fu and T. L. Booth: *Grammatical inference: Introduction and survey - part 2.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 5(4):409-423, 1975

[23] Thomas Niesler: *Category-based statistical language models*, Ph.D. Thesis, St. John's College, June, 1997

[24] Swhartz, R, Chow, Y-L: *The N-best algorithm: an efficient and exact procedure for finding the N most likely sentence hypotheses*, Proceedings of the International Conference on Acoustics, Speech and Signal Processing, pp. 81-84, Albuquerque, April, 1990

[25] Odell, J.J.: *The use of context in large vocabulary speech recognition*, Ph.D. Thesis, Department of Engineering, University of Cambridge, 1995

[26] Lovins, J.B.: *Development of a Stemming Algorithm.* Mechanical Translation and computation Linguistics, pp23-31March, 1968

[27] Andrews, K.: *The Development of a Fast Conflation Algorithm for English.* Dissertation for the Diploma in Computer Science, Computer Laboratory, University of Cambridge, 1971

[28] Petrarca, A.E. and Lay W.M.: *Use of an automatically generated authority list to eliminate scattering caused by some singular and plural main index terms.* Proceedings of the American Society for Information Science, pp. 277-282, June, 1969

[29] Dattola, Robert T.: *FIRST: Flexible Information Retrieval System for Text.* Webster N.Y: Xerox Corporation, December, 1975

[30] Colombo, D.S. and Niehoff R.T.: *Final report on improved access to scientific and technical information through automated vocabulary switching.* NSF Grant No. SIS75-12924 to the National Science Foundation

[31] Dawson, J.L.: *Suffix Removal and Word Conflation.* ALLC Bulletin, pp. 33-46, Michaelmas, 1974

[32] Porter, M.F.: *An algorithm for suffix stripping*, Program, 14 No. 3, pp. 130-137, July, 1980

[33] Κέντρο Εκπαιδευτικών Μελετών και Επιμόρφωσης (ΚΕΜΕ): *Νεοελληνική Γραμματική, Αναπροσαρμογή της Μικρής Νεοελληνικής Γραμματικής του Μανόλη Τριανταφυλλίδη*, ΟΕΔΒ, Αθήνα