Σχολη Τεχνολογικων Εφαρμογων

Τμημα Ηλεκτρονικων Μηχανικων

Πτυχιακη Εργασια

# Επεξεργασία του Φυσικού Λόγου

Μωυσίδης Παύλος Α.Μ.: 4390

Δέσης-Λουκάς Στέφανος Α.Μ.: 4206

*Υπεύθυνος Καθηγητής: Δρ. Δημήτριος Πλιάκης*

# Περιεχόμενα

# Κεφάλαιο 1

# Εισαγωγή

## 1.1 Τι είναι η επεξεργασία του φυσικού λόγου

Με τον όρο «Επεξεργασία του Φυσικού Λόγου» (Natural Language Processing) εννοούμε τον κλάδο εκείνο της Επιστήμης των υπολογιστών, της Γλωσσολογίας και της Τεχνητής Νοϋμοσύνης που ασχολείται με την αλληλεπίδραση μεταξύ υπολογιστών και ανθρώπινων (φυσικών) γλωσσών. Η Επεξεργασία του φυσικού λόγου, όπως είναι προφανές, είναι στενά συνδεδεμένη με την «Αλληλεπίδραση ανθρώπου-υπολογιστή».

### 1.1.1 Ιστορία της επεξεργασίας του φυσικού λόγου

Η ιστορία του NLP ξεκινάει τη δεκαετία του 1950, αν και στοιχεία του συναντάμε και σε προγενέστερες περιόδους. Το 1950 ο Alan Turing δημοσίευσε ένα άρθρο με τίτλο "Computing Machinery and Intelligence", το οποίο αφορούσε αυτό που σήμερα είναι γνωστό ως Turing Test. Αυτό το τεστ είχε ως σκοπό να ελέγξει αν ένα Turing machine [1] είναι ικανό να παρουσιάσει συμπεριφορά ίδια (ή μη διαφέρουσα) με την ανθρώπινη.

Επίσης, το πείραμα Georgetown το 1954 περιελάμβανε πλήρως αυτοματοποιημένη μετάφραση από τα ρώσικα στα αγγλικά, για πάνω από 60 φράσεις. Οι συγγραφείς τότε, υποστήριζαν πως το πρόβλημα της αυτοματοποιημένης μετάφρασης θα είχε λυθεί τα επόμενα τρία με πέντε χρόνια.

Ωστόσο, η ουσιαστική πρόοδος ήταν πολύ αργή και μετά την έκθεση ALPAC [2] το 1966 (κατά την οποία διαπιστώθηκε πως μετά από 10 χρόνια μακράς

[1] Μία υποθετική συσκευή που χειρίζεται σύμβολα πάνω σε μία λωρίδα ταινίας σύμφωνα με ένα πίνακα κανόνων. Alan Turing - Computing Machinery and Intelligence, 1950

[2] Automatic Language Processing Advisory Committee (Συμβουλευτική Επιτροπή Αυ-

έρευνας, οι προσδοκίες δεν εκπληρώθηκαν), η χρηματοδότηση για την αυτοματοποιημένη μετάφραση μειώθηκε δραματικά.

Μέχρι τα τέλη της δεκαετίας του 1980, διεξάχθηκε μικρή περαιτέρω έρευνα πάνω στην αυτοματοποιημένη μετάφραση. Τότε όμως αναπτύχθηκαν τα πρώτα συστήματα για τη στατιστική αυτοματοποιημένη μετάφραση. Κάποια ιδιαίτερα επιτυχημένα συστήματα NLP, αναπτύχθηκαν τη δεκαετία του 1960. Τέτοια ήταν το SHRDLU [3] , ένα σύστημα φυσικού λόγου του οποίου η λειτουργία βασιζόταν σε συγκεκριμένα "block worlds" με περιορισμένα λεξιλόγια, καθώς και το ELIZA μια προσομοίωση ενός ψυχοθεραπευτή Rogerian, που γράφτηκε απο τον Joseph Weizenbaum μεταξύ 1964-1966.

Μη χρησιμοποιώντας σχεδόν καμία πληροφορία σε σχέση με την ανθρώπινη σκέψη ή το συναίσθημα, το ELIZA[4] μπορούσε να παρέχει μια εκπληκτική αλληλεπίδραση που προσομοίαζε στην ανθρώπινη. Όταν ο «ασθενής» ξεπερνούσε την πολύ μικρή βάση δεδομένων του ELIZA, το σύστημα έδινε μια γενική απάντηση, για παράδειγμα απάνταγε στο «Το κεφάλι μου πονάει» με το «Γιατί λες ότι πονάει το κεφάλι σου;».

Κατά τη διάρκεια της δεκαετίας του 1970 πολλοί προγραμματιστές άρχισαν να γράφουν «εννοιολογικές οντότητες», οι οποίες διαρθρώνονται μεταξύ των πληροφοριών του πραγματικού κόσμου και των δεδομένων που μπορούν να γίνουν κατανοητά από τον υπολογιστή.

Τέτοια παραδείγματα είναι τα MARGIE (Schank, 1975), SAM (Cullingford, 1978), PAM (Wilensky, 1978), TaleSpin (Meehan, 1976), QUALM (Lehnert, 1977), Politics (Carbonell, 1979), και Plot Units (Lehnert 1981).

Κατά τη διάρκεια αυτής της περιόδου επίσης, γράφτηκαν πολλά προγράμματα που σαν σκοπό είχαν τη μίμηση της ανθρώπινης συνομιλίας (chatterbox), όπως το PARRY [5] , τα Racter και Jabberwacky .

Μέχρι τη δεκαετία του 1980, τα περισσότερα συστήματα NLP , βασιζόντουσαν σε περίπλοκα σετ από χειροποίητους κανόνες. Ξεκινώντας από τα τέλη της δεκαετίας του 1980 όμως, αναπτύσσεται μια επανάστατική μέθοδος στο NLP, με την εισαγωγή μια αυτοματοποιημένης τεχνικής για τη μάθηση των αλγορίθμων που χρησιμοποιούνται στην επεξεργασία του λόγου.

Αυτό οφείλεται, τόσο στη σταθερή αύξηση της υπολογιστικής ισχύος που

τόματης Επεξεργασίας Γλωσσών), ήταν μια επιτροπή από επτά επιστήμονες με επικεφαλής τον John R. Pierce, που ιδρύθηκε το 1964 από την κυβέρνηση των ΗΠΑ, προκειμένου να αξιολογήσει την πρόοδο στην υπολογιστική γλωσσολογία γενικά και την αυτόματη μετάφραση ειδικότερα.

[3] Πρόγραμμα κατανόησης της φυσικής γλώσσας γραμμένο σε Micro Planner και Lisp από τον Terry Winograd στο MIT μεταξύ 1968–1970.

[4] Αρχέτυπο πρόγραμμα Επεξεργασίας του Φυσικού Λόγου γραμμένο στο MIT από τον Joseph Weizenbaum μεταξύ 1964–1966

[5] Ένα πρόγραμμα εξομοίωσης συνομιλίας που γράφτηκε το 1972 από τον ψυχίατρο Kenneth Colby στο πανεπιστήμιο του Stanford

προκύπτει από το νόμο του Moore, όσο και στην σταδιακή αποδυνάμωση της κυριαρχίας των θεωριών γλωσσολογίας του Chomsky(πχ. μετασχηματιστική γραμματική), των οποίων το θεωρητικό υπόβαθρο αποθάρρυνε το είδος του κλάδου των "corpus linguistics" [6] που βρίσκεται πίσω από την αυτοματοποιημένη μάθηση για την προσέγγιση της επεξεργασίας του λόγου. Κάποιοι από αυτούς τους αλγόριθμους που χρησιμοποιήθηκαν πρώτοι για την αυτοματοποιημένη μάθηση, ήταν τα «δένδρα απόφασης» (decision trees) τα οποία ήταν συστήματα που παράχθηκαν και περιλάμβαν αυστηρούς κανόνες if-then παρόμοιους με τους ήδη τότε υπάρχοντες που ήταν χειροποίητοι. Με το πέρας του χρόνου, η έρευνα επικεντρώθηκε σε στατιστικά μοντέλα, τα οποία έπαιρναν ήπιου τύπου, πιθανολογικές αποφάσεις βασιζόμενα στην προσάρτηση πραγματικών τιμών σαν χαρακτηριστικά που αποτελούσαν τα δεδομένα εισόδου. Τα γλωσσικά μοντέλα cache (cache language models)[7], πάνω στα οποία στηρίζοντα πολλά προγράμματα ομιλίας τώρα, είναι παραδείγματα τέτοιων στατιστικών μοντέλων.

Τέτοιου είδους μοντέλα είναι γενικά πιο αποδοτικά όταν χορηγείται μια άγνωστη είσοδος, ειδικά όταν αυτή περιέχει λάθη (όπως πολύ συχνά συμβαίνει στα δεδομένα του πραγματικού κόσμου), ενώ είναι ικανά να παράξουν πιο αξιόπιστα αποτελέσματα όταν ενσωματώνονται σε ένα ευρύτερο σύστημα που περιλαμβάνει πολλαπλές υποδραστηριότητες.

Πολλές από τις πρώτες επιτυχίες που σημειώθηκαν στον τομέα της αυτοματοποιημένης μετάφρασης, οφείλονται στην έρευνα της IBM Research, όπου και σταδιακά αναπτύξε πιο περίπλοκα στατιστικά μοντέλα. Τα συστήματα αυτά ήταν σε θέση να επωφεληθούν από τους ήδη υπάρχοντες πολύγλωσσους γραπτούς κώδικες, οι οποίοι και είχαν παραχθεί απο το κοινοβούλιο του Καναδά και την Ευρωπαϊκή Ένωση, επειδή υπήρχε η νομική ανάγκη για τη μετάφραση όλων των κυβερνητικών διαδικασιών σε όλες τις επίσημες γλώσσες των αντίστοιχων κυβερνήσεων.

Ωστόσο, πολλά άλλα συστήματα εξαρτιόντουσαν από κώδικες ειδικά σχεδιασμένους για τις εργασίες που θα είχαν εφαρμογή αυτά τα συστήματα (και συχνά αυτό συνεχίζεται μέχρι σήμερα). Αυτή η διαδικασία είχε σαν αποτέλεσμα, να επέλθει ένας σημαντικός περιορισμός στην επιτυχία αυτών των συστημάτων. Έτσι η έρευνα στράφηκε σε μεθόδους για την ακόμα πιο αποτελεσματική μάθηση, σε σχέση με τις τότε ποσότητες δεδομένων που είχαν περιορισμένες δυνατότητες.

[6] Μελέτη δηλαδή μιας γλώσσας εκφρασμένη σε δείγματα (corpora). Η μέθοδος αυτή, αντιπροσωπεύει μία προσέγγιση, η οποία έχει ως αποτέλεσμα ένα σύνολο κανόνων από τους οποίους διέπεται μία φυσική γλώσσα ή η σχέση της με μία άλλη.

[7] Στατιστικά μοντέλα, τα οποία συναντώνται στην Επεξεργασία του φυσικού λόγου τα οποία αναθέτουν πιθανότητες σε δοθείσες ακολουθίες από λέξεις με την έννοια της κατανομής πιθανοτήτων.

Τις τελευταίες δεκαετίες η έρευνα έχει επικεντρωθεί κυρίως, στους αλγόριθμους ανεξέλεγκτης και ημι-εποπτευόμενης μάθησης. Τέτοιοι αλγόριθμοι είναι σε θέση να μάθουν και να αντλήσουν πληροφορίες από δεδομένα, που είτε δεν είναι σχολιασμένα από τον άνθρωπο με τις επιθυμητές απαντήσεις, είτε συνδιάζουν τα σχολιασμένα με τα μη σχολιασμένα δεδομένα.

Σε γενικές γραμμές το έργο αυτό είναι πολύ πιο δύσκολο από την εποπτεύομενη μάθηση, και συνήθως παράγει λιγότερο ακριβή αποτελέσματα για ένα ορισμένο ποσό των δεδομένων της εισόδου. Παρ΄ όλα αυτα υπάρχει ακόμα ένας τεράστιος όγκος των μη σχολιασμένων δεδομένων (συμπεριλαμβανομένου του Internet), το οποίο συχνά μπορεί να επιφέρει αποτέλεσματα κατώτερα του επιθυμητού.

### 1.1.2 Η επεξεργασία του φυσικού λόγου σήμερα

Οι σύγχρονοι αλγόριθμοι NLP βασίζονται στην αυτοματοποιημένη μάθηση, με επίκεντρο τη στατιστική μηχανική μάθηση (Statistical Machine Learning). Το παράδειγμα της αυτοματοποιημένης μάθησης είναι διαφορετικό σε σχέση με τις προηγούμενες προσπάθειες που είχαν γίνει στον τομέα της επεξεργασίας τους λόγου. Πριν από την υλοποίηση της επεξεργασίας του λόγου, διάφορες υποδιαδικασίες τυπικά εμπλέκονταν στην απευθείας χειροποίητη κωδικοποίηση του λόγου, κάτι που αφορούσε μεγάλες λίστες κανόνων.

Η αυτοματοποιημένη μάθηση για παράδειγμα, αντί να ζητάει τη χρήση γενικών αλγορίθμων μάθησης (συχνά, αλλά όχι πάντα βασίζεται στη στατιστική συμπερασματολογία), αυτόματα κατανοεί τους κανόνες οι οποίοι έχουν προκύψει από την ανάλυση κωδικών σε παραδείγματα αληθινού κόσμου. Ο κώδικας είναι ένα σύνολο εγγράφων (ή κάποιες φορές και μεμονωμένες προτάσεις) που έχουν χειροποίητα σχολιασμένες τις σωστές τιμές που πρέπει να μάθει.

Στις υποδιαδικασίες του NLP, έχουν εφαρμοστεί πολλές διαφορετικές τάξεις αλγορίθμων αυτοματοποιημένης μάθησης. Αυτοί οι αλγόριθμοι παίρνουν σαν είσοδο μια μεγάλη λίστα από χαρακτηριστικά, τα οποία προκύπτουν από τα δεδομένα εισόδου. Κάποιοι από τους πρώτους αλγόριθμους, όπως τα «δένδρα απόφασης», οδήγησαν στην παραγωγή συστημάτων με κανόνες if-then σκληρού τύπου, οι οποίοι ήταν τότε παρόμοιοι με τους χειροποίητους κανόνες των αντίστοιχων συστημάτων.

Ολοένα και περισσότερο όμως, η έρευνα πλέον επικεντρώνεται σε στατιστικά μοντέλα που παίρνουν ήπιου τύπου, πιθανολογικές αποφάσεις, οι οποίες στηρίζονται στη σύνδεση μεταξύ των πραγματικών τιμών και των χαρακτηριστικών της εισόδου. Τέτοια μοντέλα έχουν το πλεονέκτημα ότι μπορούν να εκφράσουν με σχετική βεβαιότητα, όχι μόνο μία, αλλά πολλές διαφορετικές απαντήσεις, παράγοντας έτσι πιο αξιόπιστα αποτελέσματα, ειδικά όταν ένα τέτοιο μοντέλο αποτελεί κομμάτι ενός μεγαλύτερου συστήματος.

Τα συστήματα που βασίζονται στους αλγορίθμους αυτοματοποιημένης μάθησης έχουν αρκετά πλεονεκτήματα σε σχέση με τα αντίστοιχα που χρησιμοποιούν χειροποίητους κανόνες. Τα πλεονέκτηματα αυτά είναι:

1. Οι μαθησιακές διαδικασίες που χρησιμοποιούνται κατά τη διάρκεια της αυτοματοποιημένης μετάφρασης, επικεντρώνονται αυτόματα στις πιο συνηθισμένες περιπτώσεις, ενώ κατά τη σύνταξη των χειροποίητων κανόνων συχνά δεν είναι καθόλου προφανές το προς τα που πρέπει να κατευθυνθεί η προσπάθεια.

2. Οι αυτόματες διαδικασίες μάθησης, μπορούν να κάνουν χρήση των στατιστικών συμπερασματικών αλγορίθμων, για να παράξουν μοντέλα τα οποία είναι ικανά να αντέξουν σε άγνωστες(πχ. λέξεις ή δομές που δεν έχουν ξαναδεί) ή σε λανθασμένες (πχ. ανορθόγραφες λέξεις) εισόδους. Σε γενικές γραμμές, ο χειρισμός τέτοιων εισόδων, στα συστήματα χειροποίητων κανόνων ή ακόμα πιο γενικά στα συστήματα ήπιου τύπου αποφάσεων, είναι εξαιρετικά δύσκολος και χρονοβόρος, ενώ είναι επιρρεπής στα λάθη.

3. Τα συστήματα που βασίζονται στην αυτόματη εκμάθηση κανόνων μπορούν να γίνουν ακόμα πιο ακριβή, απλά με την παροχή περισσοτέρων δεδομένων εισόδου. Από την άλλη ωστόσο, τα συστήματα που βασίζονται στους χειροποίητους κανόνες, μπορούν να αποκτήσουν μεγαλύτερη ακρίβεια μόνο με την αύξηση της περιπλοκότητας των κανόνων, το οποίο σαν έργο ειναι και πολύ δυσκολότερο. Πιο συγκεκριμένα, υπάρχει ένα όριο στην πολυπλοκότητα των συστημάτων που βασίζονται στους χειροποίητους κανόνες, πέρα από το οποίο τα συστήματα αυτά γίνονται όλο και πιο δύσχρηστα. Στα συστήματα αυτοματοποιημένης μάθησης όμως, η δημιουργία περισσότερων δεδομένων ως είσοδο, απαιτεί απλά μια αντίστοιχη αύξηση του αριθμού των ανθρωποωρών εργασίας, το ποίο συνήθως δεν περιλαμβάνει σημαντικές αυξήσεις στην πολυπλοκότητα της διαδικασίας σχολιασμού.

### 1.1.3 Εφαρμογές της επεξεργασίας του φυσικού λόγου

Η λίστα που ακολουθεί περιλαμβάνει ενδεικτικά κάποιες διαδικασίες στις οποίες ερευνάται ο τομέας NLP. Ορισμένες από αυτές τις διαδικασίες έχουν άμεσες ρεαλιστικές εφαρμογές, ενώ άλλες χρησιμεύουν ως επιμέρους τμήματα για βοήθεια στην επίλυση μεγαλύτερων εργασιών. Αυτό που ξεχωρίζει τις εργασίες αυτές από τις άλλες που είναι πιθανές και πραγματικές στο NLP, δεν είναι μόνο η έκταση της έρευνας, αλλά το γεγονός πως για κάθε διαδικασία υπάρχει συνήθως μια καλά καθορισμένη ρύθμιση του προβλήματος, μια τυπική μέτρηση για

την αξιολόγηση της διαδικασίας, κάποιοι πρώτυποι κώδικες βάσει των οποίων μπορεί να αξιολογηθεί η διαδικασία, καθώς και οι διαγωνισμοί που προκηρύσσονται για το συγκεκριμένο θέμα.

**Εφαρμογές**

- Αυτόματη περίληψη: Δημιουργία μίας ευανάγνωστης σύνοψης ενός κειμένου. Χρησιμοποιείται συχνά για την παροχή περιλήψεων σε κείμενα γνωστού τύπου, όπως τα άρθρα στο οικονομικό τμήμα μίας εφημερίδας.

- Ανάλυση Coreference: Καθορισμός των λέξεων οι οποίες αναφέρονται στα αντικείμενα, βάσει μιας πρότασης ή ενός μεγαλύτερου κομματιού του κειμένου. Χαρακτηριστικό παράδειγμα αυτής της διαδικασίας είναι η αντιστοίχιση των αντωνυμιών με τα ουσιαστικά και τα ονόματα που αναφέρονται. Η πιο γενική λειτουργία της ανάλυσης Coreference περιλαμβάνει επίσης τον εντοπισμό της λεγόμενης «γεφύρωσης σχέσεων», με την εμπλοκή των εκφράσεων αναφοράς. Για παράδειγμα, σε μια πρόταση όπως «Μπήκε στο σπίτι του Πετρίδη από τη μπροστινή πόρτα», η «πόρτα» είναι μία έκφραση αναφοράς και η γεφύρωση της σχέσης που πρέπει να προσδιοριστεί, είναι το γεγονός πως η πόρτα που αναφέρεται είναι η μπροστινή πόρτα του σπιτιού του Πετρίδη (και όχι κάποια άλλη κατασκευή που θα μπορούσε επίσης να αναφέρεται).

- Ανάλυση ομιλίας: Περιλαμβάνει μια σειρά από σχετικές υποκατηγορίες. Μια από αυτές είναι ο προσδιορισμός της δομής της ομιλίας σε ένα κείμενο, πχ. η φύση των σχέσεων ενός διαλόγου μεταξύ των προτάσεων (πχ. κατάρτηση, επεξήγηση, αντίθεση). Μία άλλη πιθανή υποκατηγορία, είναι η αναγνώριση και η ταξινόμηση των πράξεων ομιλίας που βρίσκονται σε ένα κείμενο (πχ. απάντηση ναι-όχι, ερώτηση περιεχομένου, δήλωση, ισχυρισμός κτλ.).

- Αυτόματη μετάφραση: Αυτόματη μετάφραση κειμένου από μία γλώσσα σε μία άλλη. Είναι ένα από τα πιο δύσκολα προβλήματα, και είναι μέρος μιας κατηγορίας προβλημάτων που ονομάζεται « AI-complete », δηλαδή απαιτούνται όλα τα διαφορετικά είδη γνώσης του ανθρώπου (γραμματική, σημασιολογία, γεγονότα σχετικά με τον πραγματικό κόσμο, κτλ.), προκειμένου να επιλυθεί σωστά.

- Μορφολογική κατάτμηση: Ξεχωρίζει λέξεις σε επιμέρους μορφήματα και προσδιορίζει το είδος αυτών. Η δυσκολία αυτής της διαδικασίας, εξαρτάται σε μεγάλο βαθμό από την πολυπλοκότητα της μορφολογίας (δηλαδή τη δόμη των λέξεων) της γλώσσας που εξετάζεται. Τα αγγλικά έχουν

μία αρκετά απλή μορφολογία, κι έτσι είναι συχνή η αγνόηση αυτής της διαδικασίας με την μοντελοποίηση όλων των πιθανών μορφών μίας λέξης (πχ. "open, opens, opened, opening") σαν ξεχωριστές. Σε γλώσσες όμως όπως τα τουρκικά, μία τέτοια προσέγγιση δεν είναι δυνατή από τη στιγμή που κάθε καταχώρηση στο λεξικό έχει χιλιάδες πιθανές μορφές.

- Ονομαστική αναγνώριση καταχώρησης: Καθορισμός για το ποια στοιχεία του κειμένου αναφέρονται σε κύρια ονόματα, όπως είναι τα άτομα ή οι τόποι, καθώς και το είδος αυτών (πρόσωπο, θέση, οργάνωση). Αν και η κεφαλαιοποίηση μπορεί να βοηθήσει στην αναγνώριση των επώνυμων καταχωρήσεων σε γλώσσες σαν τα αγγλικά, η πληροφορία αυτή δεν μπορεί να βοηθήσει στον καθορισμό του τύπου της ονομαστικής καταχώρησης, και σε κάθε περίπτωση είναι συχνά ανακριβής ή ανεπαρκής. Για παράδειγμα, η πρώτη λέξη μιας πρότασης είναι επίσης κεφαλαιοποιημένη, ενώ οι ονομαστικές καταχωρήσεις συχνά περιλαμβάνουν πολλές λέξεις, από τις οποίες μόνο κάποιες κεφαλαιοποιούνται. Επιπλέον, πολλές άλλες μη δυτικές γλώσσες (πχ. κινέζικα, αραβικά), δεν έχουν καμία κεφαλαιοποίηση, ενώ ακόμα και γλώσσες με κεφαλαιοποίηση, μπορεί να μην τη χρησιμοποιούν για την περιγραφή ονομάτων. Για παράδειγμα, στα γερμανικά κεφαλαιοποιούνται όλα τα ουσιαστικά, ανεξάρτητα αν αναφέρονται σε ονόματα, ενώ τα γαλλικά και τα ισπανικά δεν κεφαλαιοποιούν ονόματα που χρησιμοποιούνται σαν επίθετα.

- Παραγωγή φυσικού λόγου: Μετατροπή πληροφοριών από βάσεις δεδομένων υπολογιστή σε ανθρώπινη γλώσσα.

- Κατανόηση του φυσικού λόγου: Μετατροπή κομματιών κειμένου σε πιο επίσημη μορφή, όπως είναι η πρώτης τάξεως λογικές δομές που είναι πιο εύκολο να κατανοηθούν απο τα προγράμματα των ηλεκτρονικών υπολογιστών. Η κατανόηση του φυσικού λόγου περιλαμβάνει τον προσδιορισμό της σημασιολογικής πρόβλεψης από τις πολλαπλές πιθανές σημασιολογίες, οι οποίες μπορεί να προέρχονται από την έκφραση του φυσικού λόγου που συνήθως παίρνει τη μορφή οργανωμένων συμβολισμών των εννοιών της φυσικής γλώσσας. Η εισαγωγή και η δημιουργία μεταμοντέλων γλώσσας είναι αποτελεσματική, ωστόσο πρόκειται για μια εμπειρική λύση.

- Οπτική αναγνώριση χαρακτήρων: Μετατροπή μιας εικόνας που απεικονίζει κείμενο, σε χαρακτήρες.

- Καθορισμός των μερών του λόγου: Καθορισμός του είδους του μέρους του λόγου για κάθε λέξη. Πολλές λέξεις, ειδικά κάποιες κοινές, μπορεί να χρησιμοποιούνται σαν πολλά μέρη του λόγου. Για παράδειγμα, στα

αγγλικά η λέξη "book" μπορεί να είναι ουσιαστικό ("the book is on the table" , μτφ. το βιβλίο είναι πάνω στο τραπέζι) ή ρήμα ( "to book a flight", μτφ. για να κάνεις κράτηση σε μία πτήση), η λέξη "set" μπορεί να χρησιμοποιηθεί σαν ουσιαστικό, ρήμα ή επίθετο, ενώ η λέξη "out" σαν οτιδήποτε από τα προηγούμενα. Μερικές γλώσσες, έχουν περισσότερες τέτοιες ασάφειες από άλλες. Γλώσσες με κλιτική μορφολογία, όπως τα αγγλικά, είναι ιδιαίτερα επιρρεπείς σε τέτοιου είδους ασάφειες. Τα κινέζικα επειδή είναι μια τονική γλώσσα κατα τη διάρκεια της ομιλίας, είναι επίσης επιρρεπείς.

- Ερμηνεία: Δημιουργία ερμηνευτικού δενδροδιαγράμματος (γραμματική ανάλυση) μιας δοσμένης πρότασης. Η γραμματική του φυσικού λόγου είναι συχνά ασαφής, ενώ κάποιες προτάσεις είναι δυνατόν να έχουν διαφορετικές πιθανές ερμηνείες. Στην πραγματικότητα, για κάθε πρόταση μπορεί να υπάρχουν χιλιάδες πιθανές ερμηνείες (οι περισσότερες εκ των οποίων θα φαινόντουσαν εντελώς παράλογες σε έναν άνθρωπο).

- Απάντηση στην ερώτηση: Παραγωγή ερωτήσεων φυσικού λόγου, για τον καθορισμό της απάντησης. Οι πιο πολλές ερωτήσεις έχουν μια συγκεκριμένη σωστή απάντηση (όπως «Ποιά είναι η πρωτεύουσα του Καναδα;»), αλλά ενίοτε υπάρχουν και ανοιχτού τύπου ερωτήσεις (όπως «Πότε θα αυξηθεί η ποσότητα της μερίδας του φαγητού στη λεσχη;»).

- Αντίληψη σχέσης: Προσδιορισμός των σχέσεων μεταξύ των επώνυμων καταχωρήσεων (πχ. ποιος ειναι σύζηγος με ποιον).

- Διαχωρισμός πρότασης (επίσης γνωστή και σαν αποσαφήνιση ορίου διαχώρισης): Εύρεση των ορίων διαχώρισης. Τα όρια διαχώρισης συχνά καθορίζονται από περιόδους ή άλλα σημεία στίξης, αλλά κάποιες φορές μπορεί να χρησιμοποιούνται για διαφορετικούς σκοπούς (πχ. σήμανση για συντομογραφίες).

- Ανάλυση συναισθήματος: Απόσπαση υποκειμενικής πληροφορίας, συνήθως από ένα σύνολο εγγράφων. Χρησιμοποιείται συχνά σε σχόλια του διαδικτύου για τον προσδιορισμό τάσεων συγκεκριμένων αντικειμένων. Είναι ιδιαίτερα χρήσιμη για την κατανόηση της αντίληψης της κοινής γνώμης στα social media, με σκοπό το marketing .

- Αναγνώριση ομιλίας: Καταγραφή κειμένου που προέρχεται από ένα πρόσωπο ή άνθρώπους που μιλάνε. Είναι το ακριβώς αντίθετο, από τη μετατροπή κειμένου σε ομιλία και είναι ένα από τα πιο δύσκολα προβλήματα. Ανήκει στην κατηγορία «AI-complete». Στη φυσική ομιλία, δεν υπάρχουν σχεδόν καθόλου παύσεις μεταξύ των διαδοχικών λέξεων, κι έτσι ο

διαχωρισμός του λόγου είναι απαραίτητη διαδικασία για την αναγνώριση της ομιλίας. Στις πιο διαδεδομένες γλώσσες, οι ήχοι που αντιπροσωπεύουν διαδοχικά γράμματα μπλέκονται μεταξύ τους (συνάρθρωση), κι έτσι η μετατροπή του αναλογικού σήματος σε διακριτούς χαρακτήρες είναι μια πολύ δύσκολη διαδικασία.

- Διαχωρισμός ομιλίας: Διαχωρισμός των λέξεων που προέρχονται από ένα αρχείο ήχου, από ένα πρόσωπο ή από ανθρώπους που μιλάνε. Είναι υποκατηγορία της αναγνώρισης ομιλίας, και συνήθως ομαδοποιούνται μαζί.

- Διαχωρισμός θέματος και αναγνώριση: Διαχωρισμός ενός κειμένου σε τμήματα αναλογως το θέμα, καθώς και προσδιορισμός αυτών.

- Διαχωρισμός λέξεων: Διαχωρισμός ενός μεγάλου κομματιού συνεχούς κειμένου σε χωριστές λέξεις. Σε μία γλώσσα όπως τα αγγλικά είναι κάτι εύκολο, αφού οι λέξεις συνήθως χωρίζονται με κενά. Ωστόσο, σε ορισμένες γραπτές γλώσσες όπως τα κινέζικα, τα ιαπωνικά, και τα ταϋλανδικά, τα όρια των λέξεων δεν σηματοδοτούνται με τέτοιο τρόπο, και σε αυτές τις περιπτώσεις ο διαχωρισμός των λέξεων αποτελεί μια δύσκολη εργασία που απαιτεί τη γνώση του λεξιλογίου και της γλωσσικής μορφολογίας των λέξεων.

- Λεξική αποσαφήνιση έννοιας: Επιλογή της έννοιας που βγάζει περισσότερο νόημα για τις λέξεις που έχουν πιο πολλές από μία ερμηνείες. Για την επίλυση αυτού του προβλήματος συνήθως χρησιμοποιείται μια λίστα με λέξεις και συναφείς ερμηνείες

Σε ορισμένες περιπτώσεις, τα σύνολα των σχετικών διαδικασιών ομαδοποιούνται σε υποκατηγορίες NLP, που συχνά εξετάζονται ξεχωριστά από το συνολικό NLP . Τέτοια παραδείγματα αφορούν:

- Ανάκτηση πληροφοριών: Αποθήκευση, αναζήτηση και ανάκτηση πληροφοριών. Πρόκειται για ένα ξεωριστό πεδίο της επιστήμης των υπολογιστών (πιο κοντά στις βάσεις δεδομένων), αλλά η ανάκτηση πληροφοριών βασίζεται σε κάποιες μεθόδους NLP. Μερικές τωρινές έρευνες και εφαρμογές προσπαθούν να γεφυρώσουν το χάσμα μεταξύ της ανάκτησης πληροφοριών και του NLP.

- Εξαγωγή πληροφοριών: Εξαγωγή σημασιολογικής πληροφορίας από ένα κείμενο. Καλύπτει διαδικασίες όπως η ονομαστική αναγνώριση καταχώρησης, η ανάλυση Coreference, η αντίληψη σχέσης, κ.α..

- Επεξεργασία ομιλίας: Αναγνώριση ομιλίας, μετατροπή κειμένου σε ομιλία και σχετικές διαδικασίες.

Κάποιες ακόμα διαδικασίες περιλαμβάνουν:

- Απλούστευση κειμένου
- Κείμενο σε ομιλία
- Αναζήτηση φυσικού λόγου
- Επέκταση ερώτησης
- Αυτόματη βαθμολόγηση

### 1.1.4 Στατιστική επεξεργασία του φυσικού λόγου

Η στατιστική NLP(Statistical NLP) χρησιμοποιεί στοχαστική, πιθανολογία και στατιστικές μεθόδους για την επίλυση ορισμένων από τις δυσκολίες που αναφέρθηκαν παραπάνω, ιδιαίτερα εκείνων που προκύπτουν επειδή οι μεγαλύτερες προτάσεις γίνονται ασαφείς όταν επεξεργάζονται με πραγματική γραμματική, αποδίδοντας έτσι χιλιάδες ή και εκατομμύρια πιθανές αναλύσεις. Οι μέθοδοι για τη συγκεκριμένη αποσαφήνιση συχνά περιλαμβάνουν τη χρήση των μοντέλων Markov.

Η στατιστική NLP, περιλαμβάνει όλες τις ποσοτικές προσεγγίσεις για την αυτοματοποιημένη επεξεργασία του λόγου, όπως είναι η πιθανολογική μοντελοποίηση, η θεωρία της πληροφορίας και η γραμμική άλγεβρα. Η τεχνολογία της στατιστικής NLP , κυρίως προέρχεται από την αυτοματοποιημένη μάθηση και από την εξαγωγή δεδομένων. Και τα δύο αυτά πεδία, ασχολούνται με τη μάθηση από τα δεδομένα και ανήκουν στον τομέα της τεχνητής νοημοσύνης.

### 1.1.5 Τυποποίηση στην επεξεργασία του φυσικού λόγου

Μια ISO υποεπιτροπή εργάζεται προκειμένου να διευκολύνει τη διαλειτουργικότητα μεταξύ των λεξικών πόρων και των προγραμμάτων NLP. Η υποεπιτροπή αυτή είναι μέρος του ISO/TC37 και ονομάζεται ISO/TC37/SC4. Μερικά πρότυπα ISO έχουν ήδη δημοσιευθεί, αλλά τα περισσότερα εξ αυτών είναι υπό κατασκευή.

#### Το μέλλον της επεξεργασίας του φυσικού λόγου

Το NLP είναι ένα πρόβλημα της κατηγορίας « AI-complete ». Είναι δηλαδή ισοδύναμο με την επίλυση κεντρικών προβλημάτων της τεχνητής νοημοσύνης, όπως είναι η διαμόρφωσή της έτσι ώστε οι ηλεκτρονικοί υπολογιστές να πλησιάσουν κάποια στιγμή στο μέλλον, την ανθρώπινη σκέψη. Συνεπώς, το μέλλον

του NLP συνδέεται άμεσα με την ανάπτυξη στον τομέα της τεχνητής νοημοσύνης.

Όσο η κατανόηση του φυσικού λόγου θα βελτιώνεται, τόσο και οι υπολογιστές θα έρχονται σε θέση να μάθουν από τις πληροφορίες που τους δίνονται, καθώς και να προχωρήσουν στις αντίστοιχες εφαρμογές στον πραγματικό κόσμο. Σε συνδυασμό με την παραγωγή του φυσικού λόγου, σταδιακά οι υπολογιστές θα γίνονται όλο και πιο ικανοί στο να δίνουν και να λαμβάνουν οδηγίες.

# Κεφάλαιο 2

# Βασικές έννοιες

## 2.1 Απόσταση επεξεργασίας

Ένας τρόπος για να καθορίσουμε πόσο όμοιες είναι δύο συμβολοσειρές (strings), είναι η «απόσταση επεξεργασίας». Σαν απόσταση επεξεργασίας ορίζεται ο αριθμός των δραστηριοτήτων που πρέπει να πραγματοποιηθούν για να μετατρέψουμε ένα string σε ένα άλλο. [1]

### 2.1.1 Ελάχιστη απόσταση επεξεργασίας

Μολονότι υπάρχουν διαφορετικές ακολουθίες για την μετατροπή μίας συμβολοσειράς σε μία άλλη, αυτό που έχει πραγματική σημασία είναι να επιτευχθεί το ζητούμενο με τον ελάχιστο δυνατό αριθμό ενεργειών.

Ο υπολογισμός του αριθμού αυτού αρχικά μοίαζει υπολογιστικά ακριβός, όμως χρησιμοποιώντας $nxm$ συγκρίσεις, όπου n, m τα μήκη των δύο συμβολοσειρών, μπορούμε να κάνουμε τον υπολογισμό αυτό με χρήση αλγορίθμων δυναμικού προγραμματισμού (dynamic programming).

#### Αλγόριθμος Levenshstein

Έστω ότι έχουμε δύο strings, X και Y, με μήκος n και m αντίστοιχα. Ορίζουμε σαν «ελάχιστη απόσταση επεξεργασίας», minimum edit distance ή Levenshtein distance το ελάχιστο δυνατό κόστος για να μετατρέψουμε την συμβολοσειρά X στην συμβολοσειρά Y. Για την διαδικασία της μετατροπής αυτής, έχουμε διαθέσιμα τρία εργαλεία:

- Εισαγωγή Χαρακτήρα (Insertion)

[1] Taming Text, Grant S. Ingersoll, Thomas S. Morton, Andrew S. Farris

- Σβήσιμο Χαρακτήρα (Deletion)
- Αντικατάσταση Χαρακτήρα (Substitution)[2]

Σύμφωνα με τον αλγόριθμο Levenshtein λοιπόν, δημιουργούμε έναν πίνακα που περιέχει αρχικά τις δύο συμβολοσειρές και σταδιακά συμπληρώνουμε τις τιμές στον πίνακα, σύμφωνα με την παρακάτω σχέση:

$$D(i,j) = min \begin{cases} D(i-1,j)+1 \\ D(i,j-1)+1 \\ D(i-1,j-1)+k \end{cases}$$

Όπου

$$k = \begin{cases} 0 & : x(i) = y(i) \\ 2 & : x(i) \neq y(i) \end{cases}$$

Table 2.1: Minimum edit distance

| | | | | | |
|---|---|---|---|---|---|
| g | 3 | 4 | 5 | 6 | 7 |
| o | 2 | 3 | 4 | 5 | 6 |
| d | 1 | 2 | 3 | 4 | 5 |
| # | 0 | 1 | 2 | 3 | 4 |
| | # | c | a | t | s |

Επίσης μπορούμε να ξαναγράψουμε τον πίνακα και σε κάθε κελί του να φαίνεται κάθε βήμα, δηλαδή από ποια συμβολοσειρά πάμε σε ποια:

Table 2.2: Minimum edit distance

| | | | | | |
|---|---|---|---|---|---|
| g | ””→”dog” | ”c”→”dog” | ”ca”→”dog” | ”cat”→”dog” | ”cats”→”dog” |
| o | ””→”do” | ”c”→”do” | ”ca”→”do” | ”cat”→”dog” | ”cats”→”dog” |
| d | ””→”d” | ”c”→”d” | ”ca”→”d” | ”cat”→ ”d” | ”cats”→”d” |
| # | ””→”” | ””→”c” | ””→”ca” | ””→”cat” | ””→”cats” |
| | # | c | a | t | s |

Όπως είναι προφανές και από τον πίνακα, για δυο συμβολοσειρές X και Y οι οποίες έχουν μήκος n και m αντίστοιχα, για τα κελιά $D(0,0) - D(0,n)$ και $D(0,0) - D(m,0)$ δεν χρειάζεται να γίνει κάποιος υπολογισμός για να συμπληρωθούν και επιπλέον με βάση αυτά χτίζεται ο υπόλοιπος πίνακας.

[2] Η οποία μπορεί να αναχθεί σε ένα σβήσιμο και μία εισαγωγή χαρακτήρα.

### Υλοποίηση αλγορίθμου Levenshstein

Για να υπολογίσουμε το ελάχιστο κόστος επεξεργασίας μεταξύ δύο συμβολοσειρών με βάση την μαθηματική περιγραφή του αλγορίθμου, μπορούμε να τον υλοποιήσουμε με λίγες γραμμές στην γλώσσα προγραμματισμού Python.

```
#!/usr/bin/env python

# Levenshtein distance, based on mathematical description of the algo

def lev(s1, s2):
    s1 = ' ' + s1
    s2 = ' ' + s2
    d = {}
    S=len(s1)
    T=len(s2)
    for i in range(S):
        d[i,0] = i
    for j in range(T):
        d[0,j] = j
    for j in range(1,T):
        for i in range(1,S):
            if s1[i] == s2[j]:
                d[i,j] = d[i-1, j-1]
            else:
                 d[i, j] = min(d[i-1, j] + 1, d[i, j-1] + 1, d[i-1, j-1] + 2)
    return d[S-1, T-1]
```

Υποθέτοντας ότι το script έχει το όνομα lev.py, αφού τρέξουμε τον interpreter στο linux shell μας, μπορούμε να φορτώσουμε το script μας και να δούμε αν δουλεύει:

```
$ python # run the python interpreter
Python 3.3.4 (default, Feb 11 2014, 15:56:08)
[GCC 4.8.2 20140206 (prerelease)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import lev # load the script
a = lev("intention", "execution") # run the lev function and store result to a
print(a) # print the levenshtein distance
8
```

### Σταθμισμένη απόσταση επεξεργασίας

Όσον αφορά την διόρθωση ορθογραφίας, κάποιοι χαρακτήρες έχουν περισσότερες πιθανότητες να μπερδευτούν από κάποιους άλλους καθώς ο χρήστης πληκτρολογεί κάποιο κείμενο. Έτσι μιλάμε πλέον για να σταθμισμένη απόσταση επεξεργασίας και μπορούμε να την περιγράψουμε όπως φαίνεται παρακάτω.

Αρχικοποίηση:

$$D(0,0) = 0$$
$$D(i,0) = D(i-1,j) + del[x(i)]$$
$$D(0,j) = D(0,j-1) + ins[y(j)]$$

Όπου:

$$1 < i \leq n$$
$$1 < j \leq m$$

Μετά την τροποποίηση του αλγορίθμου, βλέπουμε ότι έχουμε ένα επιπλέον κόστος για κάθε σβήσιμο, είσαγωγή και αντικατάσταση χαρακτήρα. Αντί να προσθέσουμε 1 σε κάθε σβήσιμο το κόστος για να σβήσουμε όλους τους χαρακτήρες.

Και για τα υπόλοιπα κελιά του πίνακα:

$$D(i,j) = min \begin{cases} D(i-1,j) + del[x(i)] \\ D(i,j-1) + ins[y(j) \\ D(i-1,j-1) + sub[x(i),y(j)] \end{cases}$$

Δηλαδή έχουμε ένα ξεχωριστό κόστος για κάθε εισαγωγή, σβήσιμο ή αντικατάσταση χαρακτήρα και στη συνέχεια μπορούμε να δημιουργούμε νέους πίνακες με το κόστος εισαγωγής, σβησίματος ή αντικατάστασης χαρακτήρα. Προφανώς το τελικό κόστος είναι στο κελί D(n,m).

Παρακάτω ακολουθεί ο πίνακας που δείχνει τα συχνά λάθη όσον αφορά την πληκτρολόγηση των χαρακτήρων.

**sub[X, Y] = Substitution of X (incorrect) for Y (correct)**

| X \ Y (correct) | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | 0 | 0 | 7 | 1 | 342 | 0 | 0 | 2 | 118 | 0 | 1 | 0 | 0 | 3 | 76 | 0 | 0 | 1 | 35 | 9 | 9 | 0 | 1 | 0 | 5 | 0 |
| b | 0 | 0 | 9 | 9 | 2 | 2 | 3 | 1 | 0 | 0 | 0 | 5 | 11 | 5 | 0 | 10 | 0 | 0 | 2 | 1 | 0 | 0 | 8 | 0 | 0 | 0 |
| c | 6 | 5 | 0 | 16 | 0 | 9 | 5 | 0 | 0 | 0 | 1 | 0 | 7 | 9 | 1 | 10 | 2 | 5 | 39 | 40 | 1 | 3 | 7 | 1 | 1 | 0 |
| d | 1 | 10 | 13 | 0 | 12 | 0 | 5 | 5 | 0 | 0 | 2 | 3 | 7 | 3 | 0 | 1 | 0 | 43 | 30 | 22 | 0 | 0 | 4 | 0 | 2 | 0 |
| e | 388 | 0 | 3 | 11 | 0 | 2 | 2 | 0 | 89 | 0 | 0 | 3 | 0 | 5 | 93 | 0 | 0 | 14 | 12 | 6 | 15 | 0 | 1 | 0 | 18 | 0 |
| f | 0 | 15 | 0 | 3 | 1 | 0 | 5 | 2 | 0 | 0 | 0 | 3 | 4 | 1 | 0 | 0 | 0 | 6 | 4 | 12 | 0 | 0 | 2 | 0 | 0 | 0 |
| g | 4 | 1 | 11 | 11 | 9 | 2 | 0 | 0 | 0 | 1 | 1 | 3 | 0 | 0 | 2 | 1 | 3 | 5 | 13 | 21 | 0 | 0 | 1 | 0 | 3 | 0 |
| h | 1 | 8 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 12 | 14 | 2 | 3 | 0 | 3 | 1 | 11 | 0 | 0 | 2 | 0 | 0 | 0 |
| i | 103 | 0 | 0 | 0 | 146 | 0 | 1 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 49 | 0 | 0 | 0 | 2 | 1 | 47 | 0 | 2 | 1 | 15 | 0 |
| j | 0 | 1 | 1 | 9 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| k | 1 | 2 | 8 | 4 | 1 | 1 | 2 | 5 | 0 | 0 | 0 | 0 | 5 | 0 | 2 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 4 | 0 | 0 | 3 |
| l | 2 | 10 | 1 | 4 | 0 | 4 | 5 | 6 | 13 | 0 | 1 | 0 | 0 | 14 | 2 | 5 | 0 | 11 | 10 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| m | 1 | 3 | 7 | 8 | 0 | 2 | 0 | 6 | 0 | 0 | 4 | 4 | 0 | 180 | 0 | 6 | 0 | 0 | 9 | 15 | 13 | 3 | 2 | 2 | 3 | 0 |
| n | 2 | 7 | 6 | 5 | 3 | 0 | 1 | 19 | 1 | 0 | 4 | 35 | 78 | 0 | 0 | 7 | 0 | 28 | 5 | 7 | 0 | 0 | 1 | 2 | 0 | 2 |
| o | 91 | 1 | 1 | 3 | 116 | 0 | 0 | 0 | 25 | 0 | 2 | 0 | 0 | 0 | 0 | 14 | 0 | 2 | 4 | 14 | 39 | 0 | 0 | 0 | 18 | 0 |
| p | 0 | 11 | 1 | 2 | 0 | 6 | 5 | 0 | 2 | 9 | 0 | 2 | 7 | 6 | 15 | 0 | 0 | 1 | 3 | 6 | 0 | 4 | 1 | 0 | 0 | 0 |
| q | 0 | 0 | 1 | 0 | 0 | 0 | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| r | 0 | 14 | 0 | 30 | 12 | 2 | 2 | 8 | 2 | 0 | 5 | 8 | 4 | 20 | 1 | 14 | 0 | 0 | 12 | 22 | 4 | 0 | 0 | 1 | 0 | 0 |
| s | 11 | 8 | 27 | 33 | 35 | 4 | 0 | 1 | 0 | 1 | 0 | 27 | 0 | 6 | 1 | 7 | 0 | 14 | 0 | 15 | 0 | 0 | 5 | 3 | 20 | 1 |
| t | 3 | 4 | 9 | 42 | 7 | 5 | 19 | 5 | 0 | 1 | 0 | 14 | 9 | 5 | 5 | 6 | 0 | 11 | 37 | 0 | 0 | 2 | 19 | 0 | 7 | 6 |
| u | 20 | 0 | 0 | 0 | 44 | 0 | 0 | 0 | 64 | 0 | 0 | 0 | 0 | 2 | 43 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 2 | 0 | 8 | 0 |
| v | 0 | 0 | 7 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 8 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| w | 2 | 2 | 1 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 7 | 0 | 6 | 3 | 3 | 1 | 0 | 0 | 0 | 0 | 0 |
| x | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| y | 0 | 0 | 2 | 0 | 15 | 0 | 1 | 7 | 15 | 0 | 0 | 0 | 2 | 0 | 6 | 1 | 0 | 7 | 36 | 8 | 5 | 0 | 0 | 1 | 0 | 0 |
| z | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 5 | 0 | 0 | 0 | 0 | 2 | 21 | 3 | 0 | 0 | 0 | 0 | 3 | 0 |

Αυτό που μας δείχνει ουσιαστικά ο παραπάνω πίνακας είναι ότι είναι πιθανότερο κάποιος που πληκτρολογεί όταν θέλει να τυπώσει τον χαρακτήρα "a" να μπερδευτεί και να πατήσει τους χαρακτήρες "e", "i", και "o" αλλά είναι μάλλον απίθανο να μπερδέψει τον χαρακτήρα "a" με τον χαρακτήρα "b".

### Παραλλαγές της απόστασης επεξεργασίας

Υπάρχουν κάποιες παραλλαγές στον αλγόριθμο της απόστασης επεξεργασίας οι οποίες είναι πολύ σημαντικές στην Υπολογιστική Βιολογία (Computational Biology). Για παράδειγμα όταν θέλουμε να ευθυγραμμίσουμε νουκλεοτίδια ή πρωτεΐνες η δουλειά μας παίρνοντας τις δύο παρακάτω συμβολοσειρές:

| | | | | | | |
|---|---|---|---|---|---|---|
| A | G | G | C | T | A | ... |
| T | A | G | C | T | A | ... |

είναι να καταλήξουμε έτσι:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| - | A | G | G | C | T | A | ... |
| T | A | G | - | C | T | A | ... |

Στην Βιολογία αυτό είναι χρήσιμο για διάφορους λόγους, όπως για παράδειγμα για την σύγκριση γονιδίων ή περιοχών γονιδίων για την σύγκριση γονιδίων και την αναζήτηση μεταλλάξεων κλπ. Σε γενικές γραμμές μπορούμε να πούμε ότι στην επεξεργασία του φυσικού λόγου μιλάμε για απόσταση επεξεργασίας και θέλουμε αυτή να είναι όσο το δυνατόν μικρότερη, ενώ στην Βιολογία θέλουμε να είναι όσο το δυνατόν μεγαλύτερη οπότε ο αλγόριθμος μας πλέον γίνεται:

Αρχικοποίηση:

$$D(i,0) = -i * d$$

$$D(0,j) = -j * d$$

και

$$D(i,j) = max \begin{cases} D(i-1,j) + d \\ D(i,j-1) + d \\ D(i-1,j-1) + s[x(i), y(j)] \end{cases}$$

Όπου το d είναι το κόστος εισαγωγής ή διαγραφής χαρακτήρα και το s είναι το κόστος αντικατάστασης χαρακτήρα. Ο παραπάνω αλγόριθμος, είναι γνωστός ως αλγόριθμος Needleman-Wunch.

### 2.1.2 Ευθυγράμμιση συμβολοσειρών

Με τον όρο ευθυγράμμιση συμβολοσειρών (string alignment) εννοούμε την αντιστοιχία μεταξύ δύο υπο-συμβολοσειρών των αρχικών συμβολοσειρών. Έτσι, έχοντας αρχικά τις συμβολοσειρές intention και execution, το "I" αντιστοιχεί με την άδεια συμβολοσειρά (empty string) όπως φαίνεται στην παρακάτω εικόνα:

| I | N | T | E | * | N | T | I | O | N |
|---|---|---|---|---|---|---|---|---|---|
| * | E | X | E | C | U | T | I | O | N |
| d | s | s | | i | s | | | | |

Αυτό που φαίνεται στην παραπάνω εικόνα, είναι ότι για να μετατρέψουμε την συμβολοσειρά "INTENTION" στην συμβολοσειρά "EXECUTION", πρέπει να ακολοθήσουμε τα εξής βήματα:

- Σβήσιμο του χαρακτήρα "I" (συμβολίζεται με "d").
- Αντικατάσταση του χαρακτήρα "n" με "e" και συμβολίζεται με "s"
- Αντικατάσταση χαρακτήρα (το "t" με "x")
- Εισαγωγή του χαρακτήρα "u" (συμβολίζεται με "i")
- Αντικατάσταση χαρακτήρα (το "n" με "c")

Σύμφωνα με τον αλγόριθμο Levenshtein, η απόσταση μεταξύ "INTENTION" και "EXECUTION" είναι 8. Το να γνωρίζουμε την ελάχιστη απόσταση επεξεργασίας μεταξύ δύο συμβολοσειρών είναι χρήσιμο σε πολλές περιπτώσεις όπως για παράδειγμα σε λογισμικά διόρθωσης ορθογραφίας (spell-correction). Η ευθυγράμμιση συμβολοσειρών είναι πολύ σημαντική στην αυτόματη μετάφραση (machine translation) κατά την οποία κείμενο είναι γραμμένο σε δύο γλώσσες (parallel corpus) και πρέπει να αντιστοιχηθούν ανά πρόταση. Αν διευρύνουμε τον αλγόριθμο της απόστασης επεξεργασίας για να παράξουμε την ευθυγράμμιση, μπορούμε να το δούμε σαν εικόνα μέσα από τον παρακάτω πίνακα σαν μονοπάτι που διακρίνεται με έντονο χρώμα:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| n | 9 | ↓8 | ↙←↓9 | ↙←↓10 | ↙←↓11 | ↙←↓12 | ↓11 | ↓10 | ↙**8** |
| o | 8 | ↓7 | ↙←↓8 | ↙←↓9 | ↙←↓10 | ↙←↓11 | ↓10 | ↓9 | ↙**8** | ←9 |
| i | 7 | ↓6 | ↙←↓7 | ↙←↓8 | ↙←↓9 | ↙←↓10 | ↓9 | ↙**8** | ←9 | ←10 |
| t | 6 | ↓5 | ↙←↓6 | ↙←↓7 | ↙←↓8 | ↙←↓9 | ↙**8** | ←9 | ←10 | ←11 |
| n | 5 | ↓4 | ↙←↓5 | ↙←↓6 | ↙←↓7 | ↙←↓**8** | ↙←↓9 | ↙←↓10 | ↙←↓11 | ↙←↓10 |
| e | 4 | ↙3 | ←4 | ↙← **5** | ←**6** | ←7 | ←↓8 | ↙←↓9 | ↙←↓10 | ↓9 |
| t | 3 | ↙←↓4 | ↙←↓**5** | ↙←↓6 | ↙←↓7 | ↙←↓8 | ↙7 | ←↓8 | ↙←↓9 | ↓8 |
| n | 2 | ↙←↓**3** | ↙←↓4 | ↙←↓5 | ↙←↓6 | ↙←↓7 | ↙←↓8 | ↓7 | ↙←↓8 | ↙7 |
| i | **1** | ↙←↓2 | ↙←↓3 | ↙←↓4 | ↙←↓5 | ↙←↓6 | ↙←↓7 | ↙6 | ←7 | ←8 |
| # | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | # | **e** | x | e | c | u | t | i | o | n |

Κάθε κελί που περιέχει αριθμούς με έντονο χρώμα, αναπαριστά την ευθυγράμμιση ενός ζεύγους γραμμάτων των δύο συμβολοσειρών. Αν υπάρχουν 2 κελιά με έντονο χρώμα στην ίδια σειρά, τότε έχει πραγματοποιηθεί εισαγωγή χαρακτήρα από την πηγή στον στόχο, ενώ δύο κελιά με έντονους χαρακτήρες στην ίδια στήλη, σηματοδοτούν την διαγραφή ενός χαρακτήρα. Αυτό που κάνουμε στην πραγματικότητα με τον πίνακα αυτό, είναι να πάμε τον αλγόριθμο ένα βήμα μπροστά, ώστε να έχουμε δείκτες (backpointers) για κάθε κελί. Κάθε κελί μας δείχνει με τα βελάκια από που προήλθε (backtrace) Όπως είναι προφανές από τον ορισμό του αλγορίθμου σε ένα κελί μπορούμε να καταλήξουμε από ένα έως τρία κελιά. Έτσι ο αλγόριθμος τώρα μπορεί να γραφεί έτσι:

$$D(i,j) = min \begin{cases} D(i-1,j)+1 \\ D(i,j-1)+1 \\ D(i-1,j-1)+k \end{cases}$$

Όπου

$$k = \begin{cases} 0 & : x(i) = y(i) \\ 2 & : x(i) \neq y(i) \end{cases}$$

και όσον αφορά τον δείκτη ανάλογα με το που δείχνει, πρόκειται για διαφορετική διεργασία, όπως φαίνεται παρακάτω:

$$pointer(i,j) = \begin{cases} \text{ΑΡΙΣΤΕΡΑ (εισαγωγή χαρακτήρα)} \\ \text{ΚΑΤΩ (σβήσιμο χαρακτήρα)} \\ \text{ΔΙΑΓΩΝΙΑ (αντικατάσταση χαρακτήρα)} \end{cases}$$

Έτσι, ξεκινάμε από την πάνω δεξιά γωνία όπου βρίσκεται το κόστος και κάθε φορά πάμε ένα κελί πίσω ακολουθώντας τους δείκτες. Κάθε ολοκληρωμένο μονοπάτι ανάμεσα στο τελικό και το αρχικό κελί είναι ένα μονοπάτι που μας δείχνει την ελάχιστη απόσταση ευθυγράμμισης. Υπάρχουν διάφορα εργαλεία που υπολογίζουν την απόσταση επεξεργασίας όπως για παράδειγμα το UNIX diff [3]

τεστ

### 2.1.3 N-grams

Με τον όρο n-grams στον τομέα της Υπολογιστικής Γλωσσολογίας (Computational Linguistics) και των Πιθανοτήτων, είναι μία ακολουθία n αντικειμένων από μία δοθείσα ακολουθία κειμένου ή φωνής. Τα αντικείμενα αυτά μπορεί να είναι φωνήματα (phonems), συλλαβών (syllables), γραμμάτων (letters), λέξεων (words), ή ζεύγους βάσεων (base pair), όπως είναι γνωστά τα συμπληρωματικά ζεύγη βάσεων σε ένα δίκλωνο μόριο νουκλεϊκού οξέος.

Τα n-grams συλλέγονται από δείγματα κειμένου ή φωνής. Τα n-grams λέγονται και shingles. Τα n-grams μεγέθους 1, είναι γνωστά ως unigrams, τα n-grams μεγέθους 2 είναι γνωστά ως bigrams ή digrams, και τα n-grams μεγέθους 3, λέγονται tri-grams.

### 2.1.4 Εφαρμογές

Τα μοντέλα n-gram είναι γλωσσικά μοντέλα των οποίων ο σκοπός είναι να προβλέψουν το επόμενο αντικείμενο σε μία ακολουθία τάξης (n - 1) μοντέλων Markov.[4] Υπάρχουν διάφορα μοντέλα Markov, ανάλογα με την περίπτωση που τα χρησιμοποιούμε. Στην συγκεκριμένη περίπτωση, μιλάμε για αλυσίδες Markov (Markov chains), οι οποίες πήραν το όνομά τους από τον Andrey Markov, σύμφωνα με τον οποίο, μία αλυσίδα Markov, είναι μία τυχαία διαδικασία, η οποία υφίσταται μεταβάσεις από μία κατάσταση σε μία άλλη, σε ένα χώρο καταστάσεων (state space).

Θα πρέπει οι αλυσίδες αυτές να χαρακτηρίζονται από την απουσία μνήμης (memoryless). Αυτό πρακτικά σημαίνει ότι η κατανομή πιθανοτήτων στην επόμενη κατάσταση, εξαρτάται μόνο από την τρέχουσα κατάσταση και όχι από την αλληλουχία των γεγονότων που προηγήθηκαν. Τα μοντέλα των n-grams χρησιμοποιούνται ευρέως στις πιθανότητες, στην Επιστήμη της Επικοινωνίας(communication theory), στην Υπολογιστική Γλωσσολογίας(όπως για

[3] An introduction to natural language processing, Daniel Jurafsky & James H. Martin.

[4] Ρώσος μαθηματικός, γνωστός για το έργο του στις στοχαστικές διαδικασίες. (1856 - 1922)

παράδειγμα την στατιστική επεξεργασία του φυσικού λόγου), στην Υπολογιστική Βιολογία(όπως για παράδειγμα στην βιολογική ανάλυση ακολουθιών) αλλά και στην συμπίεση δεδομένων (data compression). Τα δύο βασικά πλεονεκτήματα των μοντέλων n-gram(και των αλγορίθμων που τα χρησιμοποιούν) είναι η σχετική τους απλότητα και η δυνατότητα να αναβαθμιστούν με απλή αύξηση του n. Τα μοντέλα αυτά, μπορούν να χρησιμοποιηθουν για την αποθήκευση περισσότερων συμφραζομένων με το κόστος της αργότερης εκτέλεσης, αλλά με χρήση λιγότερης μνήμης, δίνοντας την δυνατότητα σε μικρά πειράματα να είναι αποτελεσματικά.

## 2.2 Θεματοποίηση

Με τον όρο stemming στην γλωσσική μορφολογία και στην ανάκτηση πληροφοριών (information retrieval) εννοούμε την διαδικασία κατά την οποία δίνοντας μία λέξη, παίρνουμε την βάση της ή στο θέμα της ή όπως είναι αλλιώς γνωστό, την ρίζα της (stem ή base ή word stem). Δεν είναι απαραίτητο η λέξη που θα πάρουμε από το stemming να είναι ταυτόσημη με την ρίζα της λέξης. Συνήθως είναι αρκετό συσχετιζόμενες λέξεις να επιστρέφουν στο ίδιο θέμα, ακόμα και αν αυτό το θέμα δεν είναι η ρίζα της αρχικής λέξης.

Αλγόριθμοι για το stemming μελετώνται από την επιστήμη των υπολογιστών από την δεκαετία του 1960. Πολλές μηχανές αναζήτησης αντιμετωπίζουν τις λέξεις με το ίδιο θέμα σαν συνώνυμες σαν ένα είδος επέκτασης της αναζήτησης (query expansion) μία διαδικασία η οποία ονομάζεται ταύτιση.

### 2.2.1 Αλγόριθμοι θεματοποίησης

Ένας αλγόριθμος θεματοποίησης είναι ένας αλγόριθμος ο οποίος περιστέλλει τις λέξεις με την ίδια ρίζα (ή αν τα προθέματα παραμείνουν ανέπαφα λέξεις με το ίδιο θέμα) σε μία κοινή μορφή μέσω της απογύμνωσης κάθε λέξης από τα παραγωγικά και κλιτικά επιθήματά της.

Ερευνητές που ασχολούνται με την Υπολογιστική Γλωσσολογία και την ανάκτηση πληροφοριών βρίσκουν την θεματοποίηση των λέξεων ιδιαίτερα χρήσιμη για διάφορους λόγους. Στην περίπτωση της αυτοματοποιημένης μορφολογικής ανάλυσης, η ρίζα μίας λέξης μπορεί να είναι λιγότερο ενδιαφέρουσα από τα επιθήματά της, τα οποία μπορούν να χρησιμοποιηθούν ως ενδείξεις για την γραμματική δομή[5].

Από την άλλη, το πως εντοπίζονται τα επιθήματα, μπορεί να είναι δευτερεύουσας σημασίας στο πρόβλημα της απομάκρυνσής τους (το οποίο είναι και

[5] Earl, Lois L. "Part-of-Speech Implications of Affixes." Mechanical Translation and Computational Linguistics, vol. 9, no. 2 (Ιούνιος 1966).

το ζητούμενο) ώστε να αποκτήσουμε ταυτιζόμενα θέματα. Για παράδειγμα, η καταμέτρηση εμφάνισης της συχνότητας λέξεων ή η μαθηματική ανάλυση ενός γλωσσικού ¨σώματος' χρειάζεται ταυτιζόμενα θέματα. Όμως, συγκεκριμένα γλωσσολογικά προβλήματα είναι κοινά σε κάθε αλγόριθμο θεματοποίησης άσχετα με την τελική του χρήση.

### 2.2.2 Θεματοποίηση, μορφή και σημασία

Από την υπολογιστική του φύση, ένας αλγόριθμος θεματοποίησης, έχει εγγενείς περιορισμούς. Η ρουτίνα χειρίζεται επιμέρους λέξεις: δεν έχει πρόσβαση σε πληροφορίες σχετικά με τη γραμματική και τη σημασιολογική μεταξύ τους σχέση. Στην πραγματικότητα αυτό είναι βασισμένο στην παραδοχή μιας κλειστής συμφωνίας της σημασίας ανάμεσα στις λέξεις με την ίδια ρίζα. Αυτή η υπόθεση ενώ μπορεί να εφαρμοστεί στις περισσότερες περιπτώσεις, στα Αγγλικά στην καλύτερη περίπτωση αντιπροσωπεύει μια προσέγγιση. Είναι μια καλύτερη ή χειρότερη προσέγγιση ανάλογα με την σκοπούμενη χρήση του θέματος, οι σημασιολογικές ιδιαιτερότητες των επιμέρους ριζών και η δύναμη του αλγορίθμου (πόσο δραστικά μετατρέπει τις λέξεις). Ένας αλγόριθμος θεματοποίησης είναι αρκετά ισχυρός έτσι ώστε να ομαδοποιεί όλες τις λέξεις με την ίδια ρίζα, αλλά μπορεί να ειναι ακατάλληλος για καταμέτρηση συχνότητας λέξεων.

Για τέτοιες εφαρμογές θα χρειαζόταν μία λίστα πολύ περιορισμένων επιθημάτων όπου τα θέματα χρησιμοποιούνται ως μέσο σύνδεσης των σχετικών στοιχείων πληροφορίας. Σε οποιαδήποτε περίπτωση τα θέματα χρησιμοποιούνται σαν μέσα σύνδεσης μεταξύ των σχετιζόμενων αντικειμένων φαίνεται ότι είναι καλύτερο να χρησιμοποιείται ένας δυνατός αλγόριθμος ο οποίος θα συνδιάζει περισσότερες λέξεις στην ίδια ομαδοποίηση, παρέχοντας έτσι πιο πολλές παραπομπές του εγγράφου.

Αυτός, έχει την ευχέρεια να αποφασίσει την απόρριψη κάποιων από αυτές τις μορφές, οι οποίες δεν έχουν σχέση με το αντικείμενο της αναζήτησής του. Περιστασιακά, η έξοδος της ρουτίνας θεματοποίησης, μπορεί να είναι όχι μόνο αμφιλεγόμενη, αλλά ακόμη και να "μην είναι Αγγλικά". Αυτό συμβάινει όταν ένα επίθημα είναι πανομοιότυπο με το τέλος κάποιας ρίζας.

Για παράδειγμα, το -ate, είναι ένα επίθημα ουσιαστικών στην λέξη directorate, αλλά είναι μέρος της ρηματικής ρίζας στις λέξεις create και appreciate. Στα Αγγλικά, τέτοιου είδους καταστάσεις περιορίζουν τη χρήση των επιθημάτων ως ενδείξεις για τα μέρη του λόγου. Μερικές φορές οι πληροφορίες για τη γραμματική που απαιτούνται για τη θεματοποίηση, δεν προβλέπονται από αυτή.

Ωστόσο, δεν αποτελεί σοβαρό πρόβλημα η δημιουργία αυτών των μη-γλωσσικών θεμάτων, όπως είναι το cre- και το appreci-. Εάν ο σκοπός της θεματοποίησης είναι μόνο να επιτρέπει σε σχετικές λέξεις την αντιστοίχιση, τότε τα θέματα

που έχουν παραχθεί από ένα αλγόριθμο θεματοποίησης, δεν χρειάζεται να συμπίπτουν με αυτά που θα διαπίστωνε ένας γλωσσολόγος. Η ακριβής μορφή του θέματος δεν είναι αποφασιστικής σημασίας εάν είναι το ίδιο, χωρίς έτσι να επηρεάζεται από τα επιθήματα που έχουν αφαιρεθεί μετά από αυτό, ενώ σε τυχόν λανθασμένη θεματοποιήση, δεν δημιουργείται ασάφεια.

Παρομοίως, η κατάληξη που πρέπει να αφαιρεθεί προκειμένου να επιτευχθεί ένας συνεπής αλγόριθμος καθορίζεται σε συνάρτηση με το σύστημα θεματοποίησης ως σύνολο. Η κατάληξη μπορεί ή δεν μπορεί να είναι ακριβώς ισοδύναμη με κάποια άλλη καταχώρηση της Αγγλικής μορφολογίας, και μπορεί να είναι αποδεκτή η ύπαρξη ενός υπολογιστικού προγράμματος για την αφαίρεσή της. Από την άλλη ενώ ένας γλωσσολόγος δεν θα μπορούσε να πράξει το ίδιο, χωρίς να θιχτούν τα τελικά αποτελέσματα.

### 2.2.3 Τύποι αλγορίθμων θεματοποίησης

Οι δύο βασικές αρχές που χρησιμοποιούνται για την κατασκευή ενός αλγόριθμου θεματοποίησης είναι οι εξής: η επανάληψη και η μέγιστη αντιστοίχιση. Ένας αλγόριθμος που βασίζεται αποκλειστικά σε μία από αυτές τις μεθόδους, συχνά παρουσιάζει μειονεκτήματα τα οποία όμως μπορούν να αντισταθμιστούν με τον μερικό συνδυασμό των δύο αρχών.

Η επανάληψη συνήθως βασίζεται στο γεγονός πως οι καταλήξεις είναι συνδεδεμένες με θέματα σε μία συγκεκριμένη σειρά, υπάρχουν δηλαδή τάξεις σειρών των επιθημάτων [6]. Κάθε τάξη σειράς μπορεί ή δεν μπορεί να εκπροσωπείται σε κάθε δοσμένη λέξη. Η τελευταία τάξη σειράς -η τάξη που εμφανίζεται στο τέλος μιας λέξης- περιέχει κλιτά επιθήματα όπως είναι τα: -s, -es, -ed . Οι προγούμενες τάξεις σειρών είναι παραγωγικές.

Υπάρχουν διάφορες περιπτώσεις που είναι γνωστό το πότε ένα παραγωγικό επίθημα (-ness) ακολουθεί ένα κλιτό (-ed or -ing). Αυτό συμβαίνει με ορισμένα ονομαστικά επίθετα που προέρχονται από ρήματα, με τη χρήση ενός εκ των δύο κλιτών καταλήξεων, για παράδειγμα relatedness, disinterestedness, willingness.) Ένα παράδειγμα της χαμηλότερης τάξης σειράς σε μία λέξη μπορεί να είναι πως τεχνικά αποτελεί τμήμα της ρίζας (βλ. το παράδειγμα -ate παραπάνω), αλλά για τους σκοπούς του υπολογισμού, θεωρείται μέρος της κατάληξης.

Ένας αλγόριθμος θεματοποίησης επανάληψης, είναι απλά μία αναδρομική διαδικασία, όπως υποδηλώνει και το όνομα του, ο οποίος αφαιρεί ακολουθίες σε κάθε τάξη σειράς, μία κάθε φορά, ξεκινώντας από το τέλος μίας λέξης και κατευθυνόμενος προς την αρχή της.

[6] Lejnieks Valdis. "The System of English Suffixes"

Εξ ορισμού, δεν επιτρέπονται περισσότερες από μία αντιστοιχίσεις στο εσωτερικό μίας μονής τάξης σειράς. Θα πρέπει να αποφασίζεται πόσες τάξεις σειρών θα υπάρχουν, ποιές καταλήξεις θα υπάρχουν στην κάθε μία, και κατά πόσον ή όχι τα μέλη κάθε κατηγορίας θα πρέπει να καλούνται για σάρωση ονομαστικά.

Η αρχή της μέγιστης αντιστοίχισης δηλώνει πως στο εσωτερικό οποιασδήποτε δοσμένης τάξης καταλήξεων, εάν υπάρχουν πάνω από μία καταλήξεις προβλέπεται και μία αντιστοίχιση, ενώ εκείνη που είναι η μεγαλύτερη θα πρέπει να αφαιρεθεί. Η αρχή αυτή υλοποιείται με τη σάρωση των καταλήξεων σε οποιαδήποτε τάξη, κατά φθίνουσα σειρά μήκους. Για παράδειγμα, εάν το -ion αφαιρεθεί όταν επίσης υπάρχει μια αντιστοίχιση στο -ation, η διάταξη θα πρέπει να γίνει για την αφαίρεση του -at, δηλαδή για μία άλλη τάξη σειράς. Για την αποφυγή αυτής της επιπλέον τάξης σειράς, το -ation πρέπει να προηγείται του -ion στη λίστα.

Ένας αλγόριθμος που βασίζεται αποκλειστικά στην αρχή της μέγιστης αντιστοίχισης, χρησιμοποιεί μόνο μια μονή τάξη σειράς. Όλοι οι πιθανοί συνδυασμοί των επιθημάτων μεταγλωττίζονται, και στη συνέχεια τοποθετούνται κατά σειρά μήκους. Εάν δεν βρεθεί αντιστοίχιση στις μεγαλύτερες καταλήξεις, τότε σαρώνονται αυτές που είναι μικρότερου μήκους. Το προφανές μειονέκτημα αυτής της μεθόδου, έιναι ότι απαιτείται η παραγωγή όλων των πιθανών συνδιασμών των επιθημάτων. Το δεύτερο μειονέκτημα είναι το μέγεθος του αποθηκευτικού χώρου που απαιτείται για τις καταλήξεις.

Το πρώτο μειονέκτημα μπορεί επίσης να παρουσιαστεί σε μεγάλο βαθμό, όταν ρυθμίζεται έναν αλγόριθμος επανάληψης μέ όσες πιθανές τάξεις σειράς είναι δυνατόν. Για την ρύθμιση των τάξεων σειράς, πρέπει να εξετατάζονται πάρα πολλές καταλήξεις. Επιπλέον, δεν είναι πάντα προφανές το σε ποια τάξη μία ακολουθία, θα έχει τη μέγιστη αποδοτικότητα. Είναι επίσης απολύτως πιθανό πως η παρουσία των μελών κάποιων τάξεων σειράς, εξαρτάται από το αντίστοιχο πλαίσιο (βλ. παρακάτω). Εν συντομία, ενώ ένας αλγόριθμος επανάληψης απαιτεί μία μικρότερη λίστα καταλήξεων, προκύπτει μια σειρά επιπλοκών κατά την προετοιμασία της λίστας, και κατά τον προγραμματισμό της ρουτίνας.

Κάποια ιδέα για το εύρος αυτών των επιπλοκών, μπορεί να δωθεί από την εξέταση ενός άλλου βασικού χαρακτηριστικού ενός αλγόριθμου θεματοποίησης: αυτό είναι το να είναι ελεύθερος από συμφραζόμενα ο αλγόριθμος ή όχι. Από τη στιγμή που το συμφραζόμενα χρησιμοποιούνται εδώ για την κατανόηση οποιουδήποτε χαρακτηριστικού του εναπομείναντος θέματος, ένας αγλόριθμος ελεύθερος από συμφραζόμενα δεν συνεπάγεται με ποιοτικούς ή ποσοτικούς περιορισμούς σε σχέση με την αφαίρεση των καταλήξεων.

Σε έναν αλγόριθμο ελεύθεροο από συμφραζόμενα, γίνεται αποδεκτή η πρώτη κατάληξη στην οποία επιτυγχάνεται μία αντιστοίχιση. Πιθανώς όμως, να πρέπει να υπάρχουν τουλάχιστον κάποιοι ποσοτικοί περιορισμοί, με την έννοια ότι το εναπομείνον θέμα δεν πρέπει να είναι μηδενικού μεγέθους. Ένα παράδειγμα

αυτής της ακραίας περίπτωσης είναι η αντιστοίχιση του -ability σε ability , όπως και το computability.

Στην πραγματικότητα, κάθε χρήσιμο θέμα συνήθως αποτελείται από τουλάχιστον δύο γράμματα, και συχνά για το ελάχιστο αναγκαίο συνιστούνται δύο ή τρια. Ο περιορισμός του μήκους των θεμάτων ποικίλλει ανάλογα με την κατάληξη, και το πως μεταβάλλεται μπορεί να προσδιοριστεί μόνο σε σχέση με το συνολικό σύστημα. Υπάρχουν περιπτώσεις αλγορίθμων όπου συσχετίζουν ένα κατώτερο όριο με κάθε κατάληξη. Μερικά από τα όρια του είναι αρκετά υψηλά (πχ. επτά γράμματα). Υπάρχουν βέβαια, και λιγότερο συντηρητικές προτάσεις, που αναφέρονται σε δύο γράμματα σαν ελάχιστο μήκος του θέματος. Ωστόσο, ορισμένες καταλήξεις έχουν τον πρόσθετο περιορισμό πως το ελάχιστο μήκος του θέματός τους είναι τρία, τέσσερα ή πεντε γράμματα.

Το είδος των ποιοτικών περιορισμών για τα συμφραζόμενα που πρέπει να υπάρχουν είναι ένα κάπως ανοικτό ερώτημα. Για την επίτευξη καλύτερων αποτελεσμάτων, ορισμένες καταλήξεις δεν πρέπει να απομακρύνονται με την παρουσία ορισμένων γραμμάτων στο προκύπτον θέμα. Συνήθως αυτά είναι τα γράμματα που προηγούνται της κατάληξης. Η πιο επιθυμητή μορφή για τον κανόνα συμφραζομένων είναι μια γενική μορφή η οποία μπορεί να εφαρμοστεί σε έναν αριθμό καταλήξεων, αν και αυτοί οι κανόνες είναι λίγοι. Ένα παράδειγμα έιναι η, ¨μη αφαίρεση μιας κατάληξης που ξεκινάει με -en μετά από -e". Η παραβίαση αυτού του κανόνα θα αλλάξει το seen σε se-, που ενδεχομένως είναι ένα αμφίσημο θέμα (πχ, sea χωρίς το -a, seize χωρίς το -ize κλπ.). Ωστόσο, για την αποφυγή ορισμένων ειδικών περιπτώσεων που προσιδιάζουν σε αυτές τις καταλήξεις, πρέπει να δημιουργούνται μια σειρά από κανόνες για μεμονωμένες καταλήξεις. Προς αυτή την κατεύθυνση, θα μπορούσαν να χρησιμοποιηθούν τα μεγάλα μήκη, με όλο και μικρότερες επιστροφές. Προτιμάται όμως η χρήση ενός αριθμού των περισσότερων προφανών εξαιρέσεων, με την ελπίδα οτί το ποσοστό των λέξεων που δεν αντιπροσωπεύεται θα είναι αρκετά μικρό, αποκλείοντας έτσι την ανάγκη για πολλούς επιπλέον κανόνες.

Ένας αλγόριθμος θεματοποίησης επανάληψης, ο οποίος είναι αυτός που περιέχει περισσότερες από μία τάξεις σειρών καταλήξεων, προφανώς δεν είναι λιγότερο περίπλοκος από τους κανόνες συμφραζομένων σε έναν αλγόριθμο μονής τάξης. Οι εξαιρέσεις που σχετίζονται με τα μέλη της κάθε τάξης, εξαρτώνται από ένα πολύπλοκο πλαίσιο.

Για παράδειγμα, ας υποθέσουμε πως υπάρχει ένας κανόνας (σε έναν μη επαναλητικό αλγόριθμο) που δηλώνει πως το ελάχιστο μήκος του θέματος είναι πέντε, πριν το ionate. Οι καταλήξεις -ion και -ate, επίσης προκύπτουν ξεχωριστά με διαφορετικούς περιορισμούς. Σε μια επαναληπτική ρουτίνα, τα -ion και -ate, θα προκύψουν μόνο σε ξεχωριστές καταλήξεις σε διαφορετικές τάξεις σειρών, ενώ το -ion θα περιορίζεται από τον κανόνα που προηγείται του πλαισίου για το μήκος των 5 γραμμάτων, αν τυχόν το -ate έχει βρεθεί κατά την προη-

γούμενη επανάληψη. Με άλλα λόγια, οι καταλήξεις που αφαιρούνται μπορούν να επηρεάσουν τις καταλήξεις χαμηλότερης τάξης που μπορούν να αφαιρεθούν στη συνέχεια.

Οι επιπτώσεις στην απλότητα του προγραμματισμού είναι αυτονόητες. Σε έναν καθαρό αλγόριθμο μέγιστης αντιστοίχισης το μοναδικό πλαίσιο που πρέπει να ληφθεί υπόψη είναι το ίδιο το μελλοντικό θέμα. Από τότε που ο χώρος αποθήκευσης των καταλήξεων του υπολογιστή έπαψε να είναι άμεσο πρόβλημα, αποφασίστηκε η δοκιμή ενός μη επαναληπτικού αλγόριθμου θεματοποίησης, ο οποίος βασίζεται σε μια λίστα καταλήξεων μονής τάξης. Ακολουθείται δηλαδή, η διαισθητική αναποτελεσματική διαδικασία καταχώρισης και στον ενικό και στον πληθυντικό, προκειμένου να ελαχιστοποιηθεί ο αριθμός των κανόνων συμφραζομένων που είναι απαραίτητοι.

Πέρα από τις τρεις προηγούμενων μεγάλες προσπάθειες για την κατασκευή αλγόριθμων θεματοποίησης, ο καθηγητής Tukey από το Πανεπιστήμιο του Princeton προτείνει έναν κανόνα συμφραζομένων [7], δηλαδή έναν αλγόριθμο θεματοποίησης επανάληψης του οποίου οι καταλήξεις χωρίζονται σε τέσσερις τάξεις σειρών. Η πρώτη (υψηλότερης σειράς) τάξη περιέχει μόνο το τελικό s, το οποίο δεν απομακρύνεται μετά τα i,s,u . Η δεύτερη τάξη είναι αναδρομική, και η τρίτη είναι μη αναδρομική και στοιχισμένη βάση του μήκους. Η τέταρτη τάξη αποτελείται από τα υπόλοιπα τελικά σύμφωνα. Οι τρεις τελευταίες τάξεις έχουν επίσης μερικά μέλη το καθένα με απλούς περιορισμούς πλαισίου, ενώ όλες οι τάξεις έχουν όρια στο ελάχιστο μήκος του θέματος. (Η βασική δομή του αλγορίθμου "άφαίρεσης ουράς" δεν επηρεάζεται από τον πολύγλωσσο προσανατολισμό του, αν και οι καταλήξεις που χρησιμοποιούνται προφανώς θα διαφέρουν από αυτές που είναι στα Αγγλικά).

Ένα από τα πιο ενδιαφέροντα χαρακτηριστικά του συστήματος Turkey είναι η πολυπλοκότητα της δομής του. Μία τάξη χρησιμοποιεί την αρχή της μέγιστης αντιστοίχισης μόνο όταν μία άλλη είναι επαναληπτική (άρα δεν είναι μια σωστή τάξη σειράς). Προφανώς το αντικείμενο αυτής της ετερογενούς δομής είναι να αποφευχθεί με τον πιο συνοπτικό τρόπο, η επαναληπτικότητα μιας λίστας καταλήξεων μονής τάξης. Ωστόσο, όπως αναφέρθηκε και προηγουμένως, υπάρχει ένας συμβιβασμός μεταξύ της περιεκτικότητας των κανόνων και της απλότητας του προγραμματισμού.

Αντίθετα, ο αλγόριθμος που αναπτύχθηκε στο Πανεπιστήμιο του Χάρβαρντ από τον Michael Lesk[8] υπό την επίβλεψη του καθηγητή Gerard Salton,[9] βασίζεται σε μία επαναληπτική αναζήτηση για εύρεση της κατάληξης μέγιστης αντιστοίχισης. Όταν τελειώσει η εύρεση αντιστοιχίσεων, αφαιρούνται τα τε-

[7] Development of a Stemming Algorithm, Julie Beth Lovins, Ιούνιος 1968

[8] Development of a Stemming Algorithm, Julie Beth Lovins, Ιούνιος 1968

[9] The SMART Automatic Document Retrieval System, Gerard Salton

λικά i,a,e και ενδεχομένως στη συνέχεια τα τελικά σύμφωνα. Προφανώς δεν υπάρχουν κάθε είδους περιορισμοί πλαισίου. (Παρακάτω υπάρχει μια σύντομη περιγραφή του αλγορίθμου, συμπεριλαμβανομένης μια χρήσιμης λίστας με 194 καταλήξεις. Ένα δείγμα αυτών των καταλήξεων και περαιτέρω πληροφορίες σχετικά με τον αλγόριθμο, αναφέρονται στον Salton. [10]

Ένας τρίτος αλγόριθμος έχει αναπτυχθεί από τους James L.Dolby of R and D Consultants, Los Altos, California.[11] Αυτός ο αλγόριθμος λειτουργεί σε τρία επίπεδα, εκ των οποίων το πρώτο περιλαμβάνει ένα σύνολο μετασχηματισμών που εξαρτούνται από το πλαίσιο. Το μεγαλύτερο μέρος της αφαίρεσης γίνεται στο δεύτερο επίπεδο το οποίο είναι ελεύθερο από συμφραζόμενα. Πρόκειται για μία μέγιστης αντιστοίχισης αναδρομική διαδικασία, η οποία αφαιρεί τις καταλήξεις σε οποιαδήποτε σειρα, αλλά υπόκειται στον περιορισμό του ελάχιστου μήκους θέματος των δύο συλλαβών.

Στο τελικό επίπεδο, υπάρχει η πτώση των κλιτών μορφών που εξαρτούνται από το πλαίσιο. Οι καταλήξεις που χρησιμοποιήθηκαν προήλθαν από έναν αλγόριθμο από λίστες λέξεων στη βάση του ορθογραφικού πλαισίου, και τα ελάχιστα τμήματά του σε μήκος, είναι από ένα εώς τέσσερα γράμματα.

### 2.2.4 Σύνταξη λίστας καταλήξεων

Μία λίστα καταλήξεων μίας τάξης (συνδέσεις επιθημάτων) συντάχθηκε με τον ακόλουθο τρόπο: Μία προκαταρκτική λίστα βασίστηκε στις καταλήξεις που βρίσκονται σε ένα μικρό τμήμα του επαυξημένου καταλόγου ο οποίος αναπτύχθηκε από το Project Intrex,[12] όπως και στη λίστα καταλήξεων που χρησιμοποιείται στο Χάρβαρντ. Η προκαταρκτική λίστα αξιολογήθηκε εφαρμόζοντας τις καταλήξεις αυτής της λίστας σε ένα τμήμα της εξόδου της ρουτίνας αφαίρεσης ουράς, επιπέδου 1-3 και έντασης 5-7 απο τη "Κανονική και αντίστροφη Αγγλική λίστα λέξεων" [13](η ένταση 5-7 περιλαμβάνει σε αλφαβητική σειρά μη διασπασμένες λέξεις που γράφονται απο τα δεξιά προς τα αριστερά). Δεδομένου ότι κάθε μία από αυτές τις λίστες είναι δομημένη σύμφωνα με τις καταλήξεις των λέξεων, ήταν δυνατόν να παρατηρήσουμε κατά πόσον η απομάκρυνση μιας δωσμενης κατάληξης θα είχε σαν αποτέσμα:

1. δύο διαφορετικά θέματα που ταιριάζουν

[10] Automatic Information Organization and Retrieval, Gerard Salton, 1968

[11] Development of a Stemming Algorithm, Julie Beth Lovins, Ιούνιος 1968

[12] Το Project Intrex, είναι είναι πρόγραμμα πειραμάτων μεταφοράς πληροφοριών τα οποία απευθύνονται σε υπηρεσίες που αναζητούν πληροφορίες σε μεγάλες βιβλιοθήκες. Το Project Intrex, δημιουργήθηκε στο Τεχνολογικό Ινστιτούτο της Μασαχουσέτης.

[13] A. F. Brown, Normal and Reverse Word List, Πανεπιστήμιο Philadelphia - Air Force Office, 1963

2. ένα θέμα το οποίο δεν ταιριάζει και ένα θέμα το οποίο θα πρέπει να ταιριάζει.

Οποιαδήποτε από αυτές τις συνθήκες, (εκτός αν προκλήθηκε από ορθογραφική εξαίρεση ή από μη κατάλληλη αντιστοίχιση μόνο σε μερικές σπάνιες περιπτώσεις) κατέστησε αναγκαία την προσθήκη νέων καταλήξεων, την απόρριψη των παλαιών, καθώς και την προσθήκη κανόνων συμφραζομένων, εώς ότου το σύστημα να καταστεί το ίδιο σταθερό σε επαρκή βαθμό. Η θεματοποιημένη πειραματική λίστα περιείχε περίπου 260 καταλήξεις, διαιρεμένες σε 11 υποσύνολα. Τα υποσύνολα αυτά βρίσκονταν έναν αποθηκευτικό δίσκο, και προκειμένου να είναι εύκολος ο χειρισμός ήταν ταξινομημένα, εσωτερικά αλφαβητικά και σύμφωνα με το φθήνων μήκος των καταλήξεων. Η εσωτερική σειρά δεν επηρεάζει το τελικό αποτέλεσμα του αλγορίθμου.

Πριν από κάθε υποσύνολο προηγείται ένας ειδικός τίτλος που δίνει το μήκος της κατάληξης. Επίσης κάθε κατάληξη ακολουθείται από έναν κώδικα κατάστασης και επαναφοράς ως διαχωρηστικό. Ο κώδικας κατάστασης αποτελείται από ένα γράμμα της αλφαβήτου που περιέχει πληροφορίες για τις περιορισμούς στα συμφραζόμενα όσον αφορά το στέλεχος που προηγείται της κατάληξης. Η παρούσα λίστα καταλήξεων, η οποία είναι μια ελαφρώς τροποποιημένη εκδοχή του πρωτοτύπου δίνεται στις εικόνες στο παραρτήματα Α, ενώ οι δοκιμές για τους κανόνες συμφραζομένων που σχετίζονται με τις καταλήξεις δίνονται στο παράρτημα Β.

### 2.2.5 Μερικές λύσεις για τις ορθογραφικές εξαιρέσεις

Ο όρος "ὀρθογραφικές εξαιρέσεις" είναι ένας όρος για την γενική κάλυψη όλων των περιπτώσεων στις οποίες ένα θέμα είναι πιθανόν να είναι γραμμένο με έναν ή περισσότερους τρόπους. Η πλειοψηφία αυτών των περιπτώσεων στα Αγγλικά προκύπτουν στα Λατινογενή παράγωγα. Τα παραδείγματα που δίνονται παρακάτω δείχνουν ένα μέρος του εύρους και του είδους αυτών των περιπτώσεων. Τα προβληματικά σημεία είναι με πλάγια και έντονα γράμματα, ενώ το θέμα διαχωρίζεται από την κατάληξη με μια κατακόρυφη γραμμή:

- produc|er : produc***t***|ion
- inver***t***|ed : inver***s***|ion
- induc|ed : induc***t***|ion
- adhe***r***|e : adhe***s***|ion

- induct|ed : induc***t***|ion
- regist***er***|ing : regist***r***|ation
- consum|ed : consum***pt***|ion
- resolv|ed : resol***ut***|ion
- absor***b***|ing : absor***pt***|ion
- admi***tt***|ed : admi***ss***|ion
- atten***d***|ing : atten***t***|ion
- circ***l***|e : circ***ul***|ar
- expan***d***|ing : expan***s***|ion
- matri***x***| : matri***c***|es
- respon***d***| : respon***s***|ive
- lattic|e : lattic|es
- exclu***d***|e : exclu***s***|ion
- index| : indic|es
- colli***d***|ing : colli***s***|ion
- hypothe***s***|ized : hypothe***t***|ical
- analy***s***|is : analy***t***|ic

Πολλά άλλα είδη ορθογραφικών εξαιρέσεων μπορούν να προκύψουν επίσης, όπως ο διπλασιασμός συγκεκριμένων συμφώνων πριν την κατάληξη όπως για παράδειγμα: input:inputting , αλλά και η κατά περιπτώσεις διαφορετική ορθογραφία μεταξύ Αμερικάνικανικής και της Βρετανικής Αγγλικής γλώσσας για παράδειγμα: analysed:analyzed . Παρόλο που οι παραγωγικές ορθογραφικές αλλαγές συμβαίνουν μόνο πριν από συγκεκριμένες καταλήξεις, αυτό το σύνολο των καταλήξεων συνήθως είναι αρκετά μεγάλο.

Έτσι, δεν είναι πρακτικό να εξετάσει κανείς την εξαίρεση συμφώνων τελικού θέματος ως ένα μέρος των καταλήξεων σε έναν αλγόριθμο μίας τάξης. Ενώ και ο αριθμός των επιπλέον καταλήξεων που πρέπει να συμπεριληφθούν είναι απαγορευτικός. Ωστόσο, υπάρχουν δύο κύρια είδη διαδικασιών εκ των υστέρων, που έχουν τη δυνατότητα να εξαλείφουν αυτές τις εξαιρέσεις. Αυτές οι διαδικασίες ονομάζονται:

1. εκ νέου κωδικοποίηση

2. μερική αντιστοίχιση

(Ο Salton περιγράφει μια ρουτίνα η οποία περιλαμβάνει ορισμένα απο τα χαρακτηριστικά για κάθε διαδικασία που περιγράφεται στη συνέχεια. Ενώ γενικότερα υπάρχει ασχολία για τέτοιου είδους προβλήματα, όπως ο διπλασιασμός συγκεκριμένων συμφώνων, μέχρις στιγμής δεν έχει διατυπωθεί κάποια γενική λύση για πιο περίπλοκες ορθογραφικές εξαιρέσεις.) Η διαδικασία της εκ νέου κωδικοποίησης είναι κατάλληλη για ένα μέρος της ρουτίνας θεματοποίσης, παρόλο που σε αυτό εισάγει ένα στοιχείο επανάληψης. Η εκ νέου κωδικοποίηση προκύπτει αμέσως μετά την αφαίρεση μιας κατάληξης και δημιουργεί τις αλλαγές στο τέλος του προκύπτοντος θέματος που καθιστούν αναγκαία την απόλυτη αντιστοίχιση των μεταβλητών θεμάτων.

Αυτές οι αλλαγές είναι πιθανό να περιλαμβάνουν, την μετατροπή ενός θέματος σε ένα άλλο (π.χ. ο κανόνας rpt → rb αλλάζει το absorpt σε absorp), την αλλαγή και των δύο εμπλεκομένων θεμάτων με την εκ νέου κωδικοποίηση των τελικών συμφώνων τους σε κάποιο ουδέτερο στοιχείο (absorb >absor$\beta$ [14], absorpt → absor$\beta$), και την εξ ολοκλήρου αφαίρεση κάποιων γραμμάτων (absorb → absor, absorpt → absor). Προτείνοντας μια διαδικασία εκ νέου κωδικοποίησης, υπάρχει η παραδοχή πως οι περισσότερες από τις ορθογραφικές αλλαγές που προκύπτουν, μπορούν να καλυφθούν επαρκώς από ένα μικρό σύνολο μετασχηματιστικών κανόνων συμφραζομένων, των οποίων οι εξαιρέσεις είναι αρκετά προβλέψιμες, έτσι ώστε ο αριθμός των "τυχαίων" μετασχηματισμών να είναι τόσο μικρός που να μην μπορεί να επηρεάσει ολόκληρο το σύστημα θεμάτων. Ένα παράδειγμα ενός τέτοιου τυχαίου μετασχηματισμού είναι send → sens, που προκύπτει από τον κανόνα end → ens.

Αυτός ο κανόνας επρόκειτο αρχικά να καλύψει ζευγάρια όπως extend:extensive, αλλά αντ΄ αυτού τελικά έχει μετατρέψει με διφορούμενο τρόπο το θέμα sens (το οποίο πλέον αντιπροσωπεύει και το send και το sense).

Ευτυχώς η ασάφεια μπορεί να επιλυθεί με την αλλαγή του κανόνα σε end → ens εκτός αν ακολουθείται απο s , αν και λύσεις τέτοιου είδους ίσως να μην είναι εφικτές σε όλες τις περιπτώσεις. Αυτή η παραδοχή μιας μεγάλης ποσότητας κανονικότητας στις ορθογραφικές αλλαγές, είναι μια καλή απόφαση. Ωστόσο, οι εξαιρέσεις δεν είναι απόλυτα προβλέψιμες (δηλαδή, δεν είναι πάντα σχετιζόμενες με άμεσο ορθογραφικό πλαίσιο), ως εκ τούτου σαν αποτέλεσμα παράγεται ένας ορισμένος αριθμός λαθών, ο οποίος θα πρέπει να σταθμίζεται σε σχέση με τα πλεονεκτήματα αυτής της μεθόδου, όπως είναι η ταχύτητά του.

Είναι σημαντικό να σημειωθεί πως οι κανόνες που χρησιμοποιούνται στην εκ νέου κωδικοποίηση θα πρέπει να μην αφορούν μόνο τα συμφραζόμενα, αλλά

[14] Το $\beta$ συμβολίζει τον κενό χαρακτήρα

και τις εντολές. Ας υποθέσουμε πως έχουμε δύο κανόνες:

1. Αφαίρεση του ενός από τα διπλά b, d, g, m, n, p, r, s, t.

2. Αλλαγή του τελικού d,r,t,z σε s.

Ο δεύτερος κανόνας προορίζεται για να καλύψει περιπτώσεις όπως: collide: collision κτλ. Στη συνέχεια ας υποθέσουμε πως έχουμε τις λέξεις admittance και admission. Η πρώτη προέρχεται από το admitt και η δεύτερη από το admiss. Εάν οι κανόνες που εφαρμόζονται με τη δωσμένη σειρά, τότε: admit → admis και admiss → admis . Εάν είχαν επαναδιατυπωθεί όμως, τα αποτελέσματα θα είχαν ως εξής: admitt → admits, admiss → admis, τα οποία είναι και λανθασμένα.

Ένα πιο ολοκληρωμένο σύνολο κανόνων για την εκ νέου κωδικοποίηση παρόμοιο με το παραπάνω, παρατίθεται στο Παράρτημα Γ. Αυτοί οι κανόνες όπως είναι λογικό υπόκεινται σε αναθεωρήσεις, ενώ επιθυμητή θα ήταν και η αναπαραβολή των αποτελεσμάτων τους σε σχέση με τα αντίστοιχα που προκύπτουν είτε απο την εξουδετέρωση, είτε από την ακύρωση των μετασχηματισμών.

Το δεύτερο είδος της επιμέλειας για των ορθογραφικών εξαιρέσεων, είναι η μερική αντιστοίχιση, η οποία σαν μεθοδολογία είναι αρκετά διαφορετική από την εκ νέου κωδικοποίηση. Ωστόσο, οι βασικές παραδοχές όπως και τα αποτελέσματα, μπορεί να είναι παρόμοια. Η πρώτη παραδοχή είναι πως οι ορθογραφικές αλλαγές στα Αγγλικά προορίζονται για συγκεκριμένους τύπους στους οποίους μπορεί να συμβεί, αλλα αυτό δεν συμβαίνει πάντα. Η δεύτερη παραδοχή είναι πως αυτές οι αλλαγές δεν συμπεριλαμβάνουν περισσότερα από δύο γράμματα στο τέλος ενός θέματος. Αυτό βέβαια προς το παρόν, αποτελεί ένα εμπειρικό αποτέλεσμα το οποίο δεν έχει αντικρουστεί ακόμα. Έπίσης έχει παρατηρηθεί πως οι ακολουθίες γραμμάτων που προκαλούν δυσκολίες, συχνά είναι κοινές σε περισσότερες από μία κατηγορία εξαιρέσεων. Στην εκ νέου κωδικοποίηση, αυτό σημαίνει ότι κάποιοι κανόνες είναι δυνατόν να καλύπτουν περισσότερους από έναν τύπο εξαίρεσης, αν και αυτό δεν είναι κάτι το σύνηθες.

Η ειδοποιός διαφορά μεταξύ της εκ νέου κωδικοποίησης και της μερικής αντιστοίχισης είναι η εξής: η διαδικασία της εκ νέου κωδικοποίησης είναι μέρος του θεματοποιημένου αλγόριθμου, ενώ η μερική αντιστοίχιση δεν είναι. Η μερική αντιστοίχιση λειτουργεί στην έξοδο της παραγόμενης ρουτίνας στο σημείο όπου τα θέματα προέρχονται από τους όρους του καταλόγου στους οποίους ερευνήθηκαν για αντιστοιχίσεις, σύμφωνα με το ερώτημα του χρήστη. Ανακτώνται όλες οι μερικές αντιστοιχίσεις εντος ορισμένων ορίων, και όχι μόνο οι τέλειες αντιστοιχίσεις.

Οι διαφορές επιλύονται μετά την ανάκτηση, και όχι στην προηγούμενη θεματοποιημένη διαδικασία. Αυτό έχει τα πλεονεκτήματα της μείωσης της θεματοποίησης σε μια διαδιδικασία ενός βήματος, την απομάκρυνση μιας κατάληξης,

και την εξάλειψη των προδιαγραφών των συμφραζομένων κάτι που απαιτείται κάποιες φορές στην εκ νέου κωδικοποίηση. Τα μειονεκτήματα τα οποία δεν είναι τόσο προφανή, μπορούν να αναλυθούν μόνο μετά από μία πιο ολοκληρωμένη περιγραφή της διαδικασίας της μερικής αντιστοίχισης.

Η διαδικασία αυτή ξεκινάει με ένα μη τροποποιημένο θέμα $S_1$, και πάλι το absort είναι ένα καλό παράδειγμα. Το πρώτο βήμα είναι να γίνει αναζήτηση στη λίστα στους όρους του παραγόμενου καταλόγου για αυτά που αρχίζουν με $S_2$ χωρίς τα δύο τελευταία γράμματά του. Σε αυτή την περίπτωση, όλα τα θέματα οποιουδήποτε μήκους ξεκινούν με absor, το οποίο ονομάζουμε $S_2$.

Προφανώς, πρέπει να υπάρχουν ειδικές διατάξεις για τις περιπτώσεις στις οποίες το $S_1$ έχει μήκος μόνο δύο ή τρία γράμματα.Μεταξύ των θεμάτων που επιστρέφονται θα είναι το absorpt το absorp, όπως επίσης το absorbefaci και το absorbefacient. Αυτό το τελευταίο στοιχείο θα εξαλειφθεί από το επόμενο βήμα της διαδικασίας, η οποία απορρίπτει όλα τα θέματα με περισσότερους από δύο χαρακτήρες σε σχέση με το $S1$ (στην προκειμένη περίπτωση το μήκος είναι περισσότερο από εννέα γράμματα). Στη συνέχεια, γίνεται η συγκέντρωση όλων των θεμάτων τα οποία αντιστοιχίζονται με το absorpt εντός δύο γραμμάτων προς κάθε κατεύθυνση. Δεδομένου οποιασδήποτε κατάστασης, το $S_j$ είναι η τελική επιτρεπτή αντιστοίχιση μεταξύ το $S_j$ και του $S_1$, εάν και μόνο εάν, είτε $S_j$=$S_1$, είτε πληρούνται οι ακόλουθες προϋποθέσεις:

1. Τα θέματα Sj και $S_1$ πρέπει να αντιστοιχίζονται τουλάχιστον με δύο γράμματα, πριν από το τέλος του μακρύτερου εξ αυτών.

2. Εάν το $S_j$ και το $S_1$ έχουν το ίδιο μήκος και διαφέρουν κατά ένα γράμμα, τότε για κάθε θέμα πρέπει σε μία κλειστή λίστα (βλ. Παράρτημα $\Delta$) να προκύπτει αυτό το γράμμα και ένα κενό.

3. Εάν το $S_j$ και το $S_1$ έχουν το ίδιο μήκος και διαφέρουν κατά δύο γράμματα, τότε στη λίστα πρέπει να προκύψει κάθε ακολουθία αυτών των δύο γραμμάτων.

4. Εάν το $S_j$ και το $S_1$ διαφέρουν σε μήκος κατά ένα γράμμα, τότε στη λίστα πρέπει να προκύψουν τα δύο τελευταία γράμαμτα του μεγαλύτερου, το τελευταίο του μικρότερου και ένα κενό.

5. Εάν το $S_j$ και το $S_1$ διαφέρουν σε μήκος κατά δύο γράμματα, τότε στη λίστα πρέπει να προκύψουν τα δύο τελευταία γράμματα του μεγαλύτερου.

Οι παραπάνω κανόνες αφορούν ουσιαστικά την εξέταση των δύο τελευταίων γραμμάτων των θεμάτων που αντιστοιχίζονται μέχρι εκείνο το σημείο. Εάν τα

θέματα έχουν διαφορετικά μήκη, τότε όλα τα "γράμματα που λείπουν" αντιπροσωπεύονται απο κενά. Η "κλειστή λίστα" που απαιτείται για αυτή τη ρουτίνα, δίνεται στο παράρτημα Δ.

Μπορεί να φαίνεται πως ένας "μη αποδεκτός" αριθμός λανθασμένων αντιστοιχίσεων, θα είχε συνέπειες στη διαδικασία αυτή, δεδομένου ότι δεν υπάρχουν περιορισμοί στο ποιά ζευγάρια των στοιχείων στη λίστα, μπορούν να χρησιμοποιηθούν για την παραγωγή μιας αντιστοίχισης. Υπάρχουν δύο υπερασπίσεις εναντίον αυτής της άποψης:

Πρώτον, μια τέτοια κλειστή λίστα δεν υπάρχει. Δεν θα επιτρέπονται πολλές μερικές αντιστοιχίσεις. Από αυτές που επιτρέπονται λανθασμένα, πολλές θα έχουν παραχθεί επίσης από τη διαδικασία της εκ νέου κωδικοποίησης, για τους ίδιους λόγους.

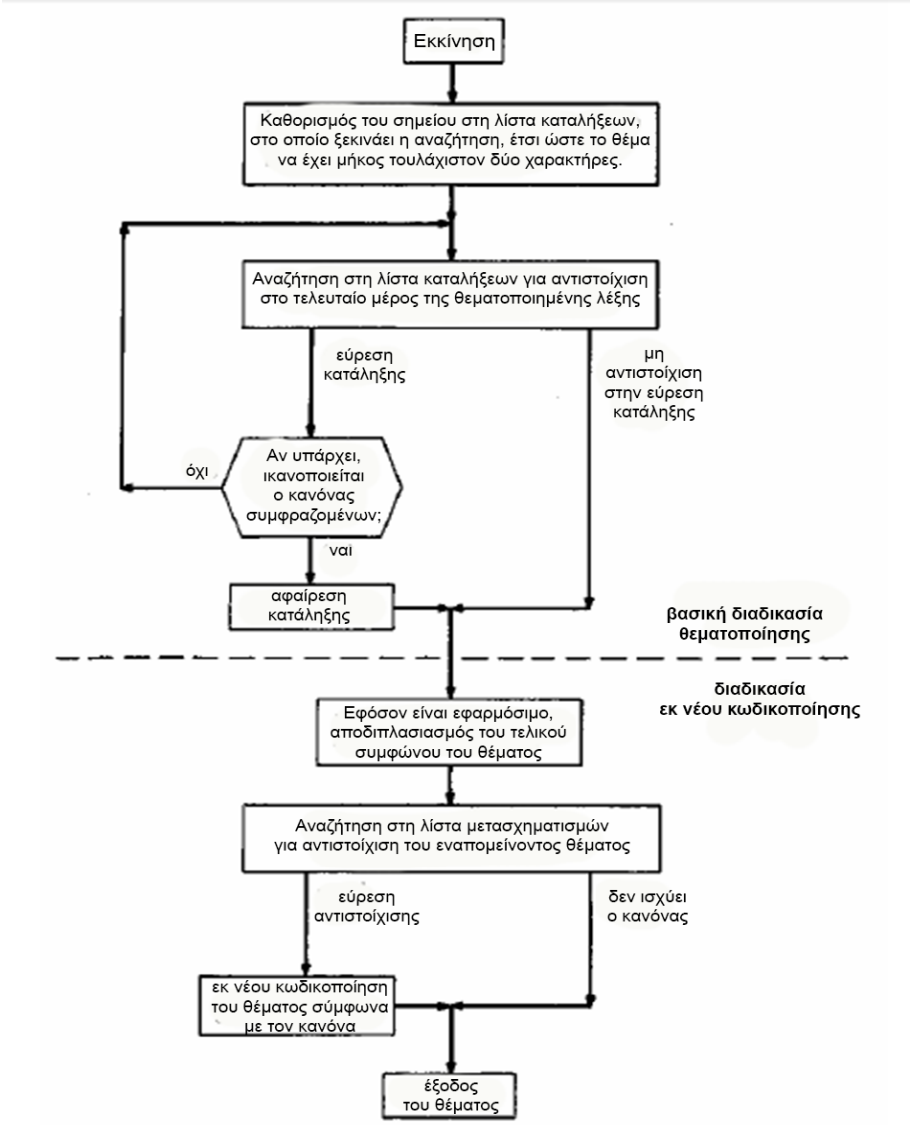
Δεύτερον, μπορούμε να κάνουμε ένα πιθανολογικό επιχείρημα. Τα περισσότερα από τα θέματα που χρησιμοποιούνται κατά πάσα πιθανότητα είναι αρκετά μεγάλα σε μήκος, και έτσι είναι απίθανο να υπάρξουν πολλές αντιστοιχίσεις μεταξύ δύο γραμμάτων. Κάθε $S_j$ που βρίσκεται, ψάχνοντας με το $S_2$, έχει μια καλή πιθανότητα να συνδέεται με το $S_2$, άρα και με το $S_1$.

Με λίγα λόγια, ενώ η διαδικασία της μερικής αντιστοίχισης δεν μπορεί να παράξει λιγότερες λανθασμένες αντιστοιχίσεις, είναι πιθανό να παράξει περισσότερες σωστές. Είναι εγγενώς πιο ευέλικτη από τους κανόνες της εκ νέου κωδικοποίησης. Όλες οι τάξεις των εξαιρέσεων δεν πρέπει να καθορίζονται εκ των προτέρων. Μέρος αυτών των ευέλικτων αποτελεσμάτων είναι η δυνατότητα για το $S_1$ και το $S_j$ να διαφέρουν προς οποιαδήποτε κατεύθυνση, δύο γράμματα σε μήκος. Ωστόσο, αυτή η κατάσταση παρέχει επίσης, ένα ενσωματωμένο φραγμό εναντίον ορισμένων τύπων λάνθασμένων αντιστοιχίσεων, όπως εξηγεί και το ακόλουθο παράδειγμα: Το convex κωδικοποιείται εκ νέου σε convic απο τον κανόνα ex → ic, convict, το στέλεχος του conviction κωδικοποιείται εκ νέου σε convic από τον κανόνα ct → c. Αυτή η λανθασμένη αντιστοίχιση δεν επιτρέπεται στη μερική αντιστοίχιση, δεδομένου αν ισχύει η κατάσταση (4), δεν ισχύει η κατάσταση (1). Η μερική αντιστοίχιση είναι ένα είδος ελεγχόμενης εκ νέου κωδικοποίησης. Η εκ νέου κωδικοποίηση πραγματοποιείται, μόνο αν βρεθεί μία μερική αλλά όχι πλήρης αντιστοίχιση. Το αρχικό θέμα διατηρείται ακόμη, παρέχοντας ωστόσο διαρκή έλεγχο για τυχόν παραβίαση της κατάστασης (1).

Χρησιμοποιώντας την μερική αντιστοίχιση σαν υποκατάστατο της εκ νέου κωδικοποίησης, υπάρχει το σημαντικό μειονέκτημα του συστηματος που χρησιμοποιείται για την αποθήκευση στο δίσκο, όπως κάνει το Intrex και είναι δυνητικά σοβαρό. Σε ορισμένες περιπτώσεις, είναι απαραίτητη η χρονοβόρα ανάκτηση από το δίσκο, μεγάλου αριθμού των μερικών αντιστοιχίσεων που ξεκινούν με το $S_2$. Αυτές οι περιπτώσεις είναι πιο πιθανό να συμβούν με πολύ μικρά θέματα. Το ερώτημα είναι αν σε τέτοιες περιπτώσεις, το $S_2$ μπορεί να επιμηκυνθεί (να

γίνει κοντινό με το $S_2$ ) αρκετά έτσι ώστε να αποφευχθεί αυτό το πρόβλημα και να συνεχίζεται η ανάκτηση όλων των αποδεκτών αντιστοιχίσεων. Απαιτούνται εμπειρικά δεδομένα για την απάντηση αυτού του ερωτήματος, όπως και για τον προσδιορισμό του αν ο αριθμός των μικρών θεμάτων που χρησιμοποιούνται, είναι αρκετά μεγάλος για προκαλέσει ανησυχία. Οποιαδήποτε χρονική στιγμή, ο προγραμματισμός ή και άλλες περίπλοκες διαδικασίες που η μερική αντιστοίχιση εισάγει, πρέπει να είναι αρκετά μικρές προκειμένου να υπάρξει εξισορρόπηση σε σχέση με άλλα πλεονεκτήματα που μπορεί να προσφέρει.

Ακολουθεί ένα διάγραμμα ροής που δείχνει την διαδικασία που ακολουθείται για την θεματοποίηση:



# Κεφάλαιο 3

# Επεξεργασία κειμένου

## 3.1 Τμηματοποίηση κειμένου

### 3.1.1 Η τέχνη της τμηματοποίησης

Με τον ορο τμηματοποίηση(tokenization) εννοείται η διαδικασία κατάτμησης κινούμενου κειμένου σε λέξεις και φράσεις. Το ηλεκτρονικό κείμενο είναι μια γραμμική ακολουθία συμβόλων (χαρακτήρες, λέξεις ή φράσεις). Φυσικά πριν να γίνει κάθε επεξεργασία πραγματικού κειμένου, το κείμενο πρέπει να κατατμηθεί σε γλωσσικές ενότητες όπως είναι οι λέξεις, τα σημεία στίξης, οι αριθμοί, τα αλφαριθμητικά, κλπ. Αυτή η διαδικασία ονομάζεται τμηματοποίηση.

Στα Αγγλικά, συχνά οι λέξεις χωρίζονται μεταξύ τους με κενά (λευκό κενό) αλλά δεν είναι όλο το λευκό διάστημα ίσο. Και το "Los Angeles" και το "rock 'n' roll" είναι προσωπικές σκέψεις, παρά το γεγονός ότι περιέχουν πολλαπλές λέξεις και διαστήματα. Μπορεί επίσης να χρειάζεται να γίνει διαχωρισμός σε απλές μεμονωμένες λέξεις, όπως το "I'm" στις ξεχωριστές λέξεις "I" και "am".

Η τμηματοποίηση κατά μία έννοια είναι ένα είδος προ-επεξεργασίας, δηλαδή προσδιορίζει τις βασικές μονάδες που θα επεξεργαστούν. Είναι συνηθισμένη η επικέντρωση στην καθαρή ανάλυση ή στην παραγωγή, την ώρα που οι βασικές ενότητες λαμβάνονται σαν δεδομένες. Όμως χωρίς να είναι σαφώς διαχωρισμένες αυτές οι βασικές ενότητες, είναι αδύνατο να πραγματοποιηθεί οποιαδήποτε ανάλυση ή παραγωγή.

Η ταυτοποίηση των ενοτήτων που δεν χρειάζονται περαιτέρω απο αποσύνθεση για μετέπειτα επεξεργασία είναι εξαιρετικά σημαντική. Σε αυτό το στάδιο τα λάθη που γίνονται είναι πολύ πιθανό να προκαλέσουν επιπλέον λάθη σε μεταγενέστερα στάδια της επεξεργασίας κειμένου, και έτσι είναι πολύ επικίνδυνο.

### 3.1.2 Τι θεωρειται τμήμα

Τι είναι αυτό που μετράει ως τμήμα στο NLP·

Η έννοια του τμήματος πρέπει να προσδιοριστεί πρώτα, πριν να μπορεί να προχωρήσει η υπολογιστική διαδικασία. Υπάρχουν περισσότερα στο ζήτημα, από την απλή ταυτοποίηση σε ακολουθίες που οριοθετούνται και στις δύο πλευρές με κενά ή σημεία στίξης.

Οι διαφορετικές αντιλήψεις εξαρτώνται από τα διαφορετικα αντικείμενα και συχνά και από τα διαφορετικά γλωσσικά υπόβαθρα. Ένα τμήμα είναι:

1. Σημαντικό γλωσσικά.
2. Χρήσιμο μεθοδολογικά.

Οι Webster[1] και Kit προτείνουν ότι η εξεύρεση σημαντικών τμημάτων εξαρτάται από την ικανότητα της αναγνώρισης μοτίβων εμφανίζοντας σημαντικό συνδιασμό λέξεων. Αντί απλά να στηρίζονται στο αν μια ακολουθία οριοθετείται από διαχωριστικά και στις δύο πλευρές, η κατάτμηση βασίζεται σε ένα είδος αναγνώρισης προτύπων. Ας σκεφτούμε αυτή την υποθετική ομιλία: "where is meadows dr who asked." Ο συνδιασμός λέξεων του μοτίβου να μπορούσε να βοηθήσει στον προσδιορισμό για το αν πρόκειται για λειβάδια dr (Οδήγηση) ή για dr (Γιατρό).

#### Πρότυπη τμηματοποίηση

Πρόκειται για την λευκού διασήματος(whitespace) τμηματοποίηση. Η τμηματοποίηση λέξεων, μπορεί να φαίνεται απλή σε μία γλώσσα που οι λέξεις χωρίζονται από έναν ειδικό χαρακτήρα "διαστήματος"(space).Ωστόσο δεν γίνεται σε κάθε γλώσσα αυτό (Κινέζικα, Ιαπωνικά κ.α.), ενώ μια προσεκτική εξέταση θα καταστήσει σαφές ότι το λευκό διάστημα από μόνο του, δεν είναι αρκετό ακόμα και για τα Αγγλικά.

#### Αντιπετώπιση ειδικών ζητημάτων

Η τμηματοποίηση γενικά θεωρείται ως εύκολη, και είναι από τα πιο αδιάφορα σε σχέση με άλλα ζητήματα στο NLP (για τα Αγγλικά και για άλλες κατατμημένες γλώσσες). Ωστόσο, τα σφάλματα που προκύπτουν σε αυτή τη φάση μεταδίδονται και σε μεταγενέστερες φάσεις και προκαλούν προβλήματα. Για την αντιμετώπιση αυτού του προβλήματος, έχουν αναπτυχθεί ένας αριθμός προηγμένων μεθόδων, οι οποίες ασχολούνται με συγκεκριμένα ζητήματα στη τμηματοποίηση, προκειμένου να συμπληρωθούν οι πρότυποι τμηματοποιητές.

[1]Webster Jonathan, Kit Chunyu, - "Tokenization as the initial phase in NLP", 1992

Ο Bob Carpenter[2] αναφέρει ότι η τμηματοποίηση είναι ιδιαίτερα προβληματική σε κείμενα της βιοϊατρικής, όπου υπάρχουν πάρα πολλές λέξεις (ή τουλάχιστον φραστικές και λεξιλογικές καταχωρήσεις) οι οποίες περιέχουν παρνθέσεις, παύλες κ.ο.κ.

Ένα άλλο ζήτημα για την τμηματοποίηση είναι το "βρώμικο κείμενο". Αυτό συμβαίνει όταν δεν έχει περάσει όλο το κείμενο από τη διαδικασία της επεξεργασίας και του ορθογραφικού ελέγχου. Το κείμενο που εξάγεται αυτόματα από αρχεία PDF, από βάσεις δεδομένων, ή από άλλες πηγές μπορεί να περιέχουν ανακριβή συνδιασμένα τμήματα, ορθογραφικά λάθη και απροσδόκητους χαρακτήρες. Σε ορισμένες περιπτώσεις, όταν το κείμενο είναι αποθηκευμένο σε μία βάση δεδομένων σε σταθερά πεδία, με πολλαπλές γραμμές ανά αντικείμενο, κάποιες φορές τα πεδία πρέπει να αθροίζονται ξανά, αλλά τα κενά έχουν (αντιφατικά) αφαιρεθεί.

Δεν είναι ασφαλές το να γίνει η παραδοχή πως η πηγή του κειμένου θα είναι τέλεια. Ένας τμηματοποιητής συχνά πρέπει να προσαρμόζεται στα εν λόγω δεδομένα.

### 3.1.3 Χαμηλού επιπέδου εναντίον υψηλού επιπέδου τμηματοποίηση

Χαμηλού επιπέδου εναντίον υψηλού επιπέδου Τμηματοποίηση

Ο προσδιορισμός για το αν δύο ή περισσότερες λέξεις, πρέπει να βρίσκονται μαζί για τον σχηματισμό ενός μονού τμήματος όπως το "Rational Software Architect" ), είναι ένα ζήτημα υψηλού επιπέδου τμηματοποίησης. Η κατάτμηση υψηλού επιπέδου (high-level tokenization), είναι πολύ πιο δομημένη γλωσσολογικά, από την κατάτμηση χαμηλού επιπέδου (low-level tokenization), και απαιτεί (τουλάχιστον) μια σχετικά επιφανειακή γλωσσική επεξεργασία.

### 3.1.4 Βήματα στην χαμηλού επιπέδου τμηματοποίηση

Βήματα στη χαμηλού επιπέδου τμηματοποίηση

#### Βήμα 1ο: Κατάτμηση κειμένου σε λέξεις

Το πρώτο βήμα στην πλειονότητα των εφαρμογών επεξεργασίας κειμένου, είναι η κατάτμηση του κειμένου σε λέξεις.

Σε όλες τις σύγχρονες γλώσσες που χρησιμοποιούν το Λατινικο, το Κυριλλικο, ή το Ελληνικό σύστημα γραφής, όπως είναι τα Αγγλικά και άλλες Ευ-

[2] έγραψε την Java κλάση (class) με το όνομα "Tokenizer"

ρωπαϊκές γλώσσες, τα λεκτικά τμήματα οριοθετούνται από ένα κενό διάστημα. Έτσι, για αυτές τις γλώσσες, οι οποίες ονομάζονται κατατμημένες γλώσσες, το όριο αναγνώρισης του τμήματος είναι ένα ζήτημα ασημαντου ρίσκου, δεδομένου ότι η πλειοψηφία των τμημάτων δεσμεύονται από σαφή διαχωριστικά, όπως είναι τα διαστήματα και τα σημεία στίξης. Ένα απλό πρόγραμμα που αντικαθιστά τα λευκά διαστήματα με τα όρια των λέξεων και αφαιρεί τα αρχικά και τα τελικά εισαγωγικά, τις παρενθέσεις και τα σημεία στίξης, παράγει ήδη μια λογική απόδοση.

Η πλειοψηφία των υφιστάμενων τμηματοποιητών απλού τμήματος οριοθετεί βάση των λευκών διαστημάτων. Έτσι, ένας τέτοιος τμηματοποιητής εισάγει ένα λευκό διάστημα, εάν βρει δύο τμήματα κοντινά μεταξύ τους, όπως για παράδειγμα, όταν μία λέξη ακολουθείται από ένα κομμα.

Στο παράδειγμα που δίνεται στην επόμενη ενότητα, φαίνεται πως ένας τμηματοποιητής πρότυπου λευκού διαστήματος αναποκρίνεται σε ένα πιο σύνθετο παράδειγμα.

### Βημα 2ο: Χειρισμός συντομογραφιών

Στα Αγγλικά και σε άλλες Ινδο-Ευρωπαϊκές γλώσσες, αν και η περίοδος είναι άμεσα συνδεδεμένη με την προηγούμενη λέξη, είναι σύνηθες ένα ξεχωριστό τμήμα το οποίο σηματοδοτεί το τέλος της πρότασης. Ωστόσο, όταν μία τελεία ακολουθεί μια σύντομογραφία, αποτελεί αναπόσπαστο μέρος αυτής της σύντομογραφίας και θα πρέπει να τμηματοποιηθεί μαζί με αυτή. π.χ. "the dr. lives in a blue box".

Χωρίς την αντιμετώπιση του ζητήματος που θέτει η συντομογραφία, η γραμμή αυτή θα πρέπει να οριοθετηθεί ως εξής: the dr. lives in a blue box. Δυστυχώς δεν υπάρχουν διεθνώς αποδεκτά πρότυπα για πολλές συντομογραφίες και ακρωνύμια.

Η πιο ευρέως αποδεκτή προσέγγιση για την αναγνώριση των συντομογραφιών είναι η διατήρηση μίας λίστας με τις γνωστές συντομογραφίες. Έτσι κατά τη διάρκεια της τμηματοποίησης μίας λέξης που έχει τελεία στο τέλος μπορεί να ανευρεθεί σε μία τέτοια λίστα. Εάν βρεθεί εκεί τμηματοποιείται σαν ένα μονό τμήμα, διαφορετικά η τελεία τμηματοποιείται σαν ξεχωριστό τμήμα. Φυσικά, η ακρίβεια αυτής της προσέγγισης εξαρτάται από το πόσο καλά η λίστα των συντομογραφιών είναι προσαρμοσμένη στο υπο επεξεργασία κείμενο. Είναι σχεδόν βέβαιο,ότι θα υπάρχουν συντομογραφίες στο κείμενο οι οποίες και δεν θα περιλαμβάνονται στη λίστα. Επίσης κάποιες συντομογραφίες στη λίστα μπορεί να συμπίπτουν με κοινές λέξεις και να προκαλέσουν λανθασμένη τμηματοποίηση. Για παράδειγμα, το "in" μπορεί να είναι μία συντομογραφία για το "inches", το "no" μπορεί να είναι μια συντομογραφία του "number", το "bus" μπορεί να είναι μια συντομογραφία του "business", το "sun" μπορεί να είναι μία

συντομογραφία του "Sunday", κλπ.

Οι παρακάτω λίστες δεν είναι με κανένα τρόπο πλήρεις.

Συνηθισμένα ακρωνύμια με σημεία στίξης

1. I.O.U.
2. .M.D.
3. N.B.
4. P.O.
5. U.K.
6. U.S.
7. U.S.A.
8. P.S.

Συνηθισμένες λέξεις που περιέχουν τελείες

1. .c
2. mr.
3. mrs.
4. .com
5. dr.
6. .sh
7. .java
8. st.

**Βήμα 3ο: Χειρισμός λέξεων που ενώνονται με παύλα**

Η κατάτμηση των λέξεων που ενώνονται με παύλα, απαντάει στην ερώτηση "Μία λέξη ή δύο;". Τα τμήματα που είναι ενωμένα με παύλες παρουσιάζουν, μια περίπτωση αμφιβολίας για έναν τμηματοποιητή. Κάποιες φορές η παύλα είναι μέρος του θέματος. Πχ. self-assessment, F-15, forty-two και κάποιες φορές δεν είναι, πχ. Los Angeles-based.

Η κατάτμηση των λέξεων που ενώνονται με παύλα είναι ένα εξαρτημένο ζήτημα. Για παράδειγμα, τα pos(part of speech) taggers, συνήθως αντιλαμβάνονται τις λέξεις που είναι ενωμένες με παύλα σαν μονή συντακτική μονάδα κι έτσι προτιμούν να τις τμηματοποιούν ως μονά θέματα. Από την άλλη πλευρά, τα συστήματα αναγνώρισης ονομαστικής καταχώρησης (NER, named entity recognition) προσπαθουν να διαχωρίσουν μια ονομαστική καταχώρηση από το υπόλοιπο του κομματιού που είναι ενωμένο με παύλα. Π.χ. στην ανάλυση του κομματιού 'Moscow-based' ένα τέτοιο σύστημα χρειάζεται να τμηματοποιήσει το 'Moscow' ξεχωριστά από το 'based', προκειμένου να μπορεί να το επισημάνει ως τοποθεσία.

### 3.1.5 Τύποι παυλών

Τύποι παυλών

1. Παύλες στο τέλος της γραμμής
2. Πραγματικές παύλες
   (α′) Λεξιλογικές παύλες
   (β′) Προτασιακές καθοριστικές παύλες

### 3.1.6 Παύλες στο τέλος της γραμμής

Οι παύλες στο τέλος της γραμμής χρησιμοποιούνται για τον διαχωρισμό ολόκληρων λέξεων σε μέρη, προκειμένου να πραγματοποιηθεί η στοίχιση του κειμένου κατά τη διάρκεια της στοιχιοθεσίας.

### 3.1.7 Πραγματικές παύλες

Πραγματικές παύλες

Από την άλλη, οι πραγματικές παύλες αποτελούν αναπόσπαστο μέρος των σύνθετων τμημάτων, όπως είναι πχ. το forty-seven και δεν πρέπει να αφαιρούνται. Μερικές φορές είναι δύσκολο να διακριθεί μία πραγματική παύλα από μία

που βρίσκεται στο τέλος της γραμμής, όταν η παύλα εμφανίζεται στο τέλος της γραμμής.

### 3.1.8 Λεξιλογικές παύλες

Οι παύλες ενώνουν λέξεις, οι οποίες έχουν πάρει το δρόμο τους σε ένα τυπικό λεξιλόγιο μιας γλώσσας. Για παράδειγμα, ορισμένα προθέματα (και λιγότερο συχνά επιθήματα) έιναι συχνά γραμμένα με παύλες, πχ. co-, pre-, meta-, multi-, κλπ.

### 3.1.9 Προτασιακές καθοριστικές παύλες

Προτασιακές καθοριστικές παύλες

Εδώ οι μορφές παυλών δημιουργούνται δυναμικά ως μηχανισμός για την πρόληψη λανθασμένης ανάλυσης για τη φράση στην οποία εμφανίζονται οι λέξεις. Υπάρχουν διάφοροι τύποι παυλών σε αυτή την κατηγορία. Μία δημιουργείται όταν ένα ουσιαστικό τροποποιείται από "ed"-ρήμα για δημιουργήσει δυναμικά ένα επίθετο. πχ. case-based, computer-linked, hand-delivered. Μία άλλη περίπτωση αφορά μία ολόκληρη έκφραση όταν χρησιμοποιείται ως μετατροπέας σε μια ομάδοποίηση ουσιαστικών, όπως το a three-to-five-year direct marketing plan. Για την αντιμετώπιση αυτών των περιπτώσεων, μια λεξιλογική στρατηγική look-up, δεν βοηθάει ιδιαίτερα και κανονικά οι εκφράσεις αυτές αντιμετωπίζονται σαν ένα μονό τμήμα, εκτός αν υπάρχει ανάγκη να αναγνωριστούν σαν συγκεκριμένο τμήμα, όπως είναι οι ημερομηνίες, τα ονόματα, τα μεγέθη, οπότε διαχειρίζονται από εξειδικεύμενες υπογραμματικές. Αυτή η υποθετική φράση αντιμετωπίζει αρκετά ζητήματα: "the New York-based co-operative was fine- tuning forty-two K-9-like models."

| Τμήμα | Τύπος |
|---|---|
| New York-based | Προτασιακός |
| co-operative | Λεξιλογικός |
| fine-tuning | Τέλους Γραμμής, αλλά θα μπορούσε επίσης να θεωρηθεί σαν λεξιλογική παύλα βάση των τεχνοτοπικών προτιμήσεων του συγγραφέα |
| Forty-two | Λεξιλογικός |
| K-9-like | Λεξιλογικός και Προτασιακός |

### 3.1.10 Βήμα 4ο: Αριθμητικές και ειδικές εκφράσεις

Παραδείγματα:

1. Διευθύνσεις e-mail
2. URLs
3. Σύνθετη καταμέτρηση στοιχείων
4. Τηλεφωνικοί αριθμοί
5. Ημερομηνίες
6. Ώρα
7. Μεγέθη
8. Αριθμοί διπλώματος οδήγησης
9. Παραπομπές σε βιβλία και εργασίες
10. κ.α.

Αυτά μπορούν να παράξουν μεγάλη σύγχυση στον τμηματοποιητή επειδή συνήθως περιλαμβάνουν σύνθετη σύνταξη με αλφαριθμητικά και με σημεία στίξης. Οι τηλεφωνικοί αριθμοί για παράδειγμα έχουν μια ποικιλία μορφών:

1. 123-456-7890
2. (123)-456-7890
3. 123.456.7890
4. (123) 456-7890
5. κ.α.

Ένας προεπεξεργαστής θα πρέπει να έχει σχεδιαστεί για να αναγνωρίζει τους τηλεφώνικούς αριθμούς και να εκτελεί την κανονικοποίηση. Στη συνέχεια όλοι οι τηλεφωνικοί αριθμοί θα είναι σε μονή μορφή, κάνοντας έτσι την εργασία του τμηματοποιητή πιο εύκολη. Μορφές ημερομηνίας:

1. 8th-Feb
2. 8-Feb-2013
3. 02/08/13
4. February 8th, 2013
5. Feb 8th
6. κ.α.

Ένας προεπεξεργαστής θα μπορούσε να αναγνωρίσει όλες αυτές τις ξεχωριστές παραλλαγές και να τις κανονικοποιήσει σε μια μονή έκφραση. Παράδειγμα τμηματοποίησης "I said, 'what're you? Crazy?'" said Sandowsky. "I can't afford to do that."

Ο απλός αναλυτής λευκού διαστήματος (Naive Whitespace Parser) )(βλ. επόμενη σελίδα) εδώ φαίνεται να έχει χαμηλή απόδοση. Ο τμηματοποιητής του Stanford, αποδίδει κάπως καλύτερα από τον τμηματοποιητή του OpenNLP, κάτι που είναι και αναμενόμενο. Ο προσαρμοσμένος (custom ) αναλυτής στην τέταρτη στήλη, έχει μια σχεδόν τέλεια απόδοση, χωρίς όμως την εγκλιτική επέκταση που φαίνεται στο πρώτο υποθετικό πέρασμα. Η πιο ακριβής (και σύνθετη) διαδικασία κατάτμησης στην τέταρτη και στην πέμπτη στήλη, απαιτεί μια διαδικασία μορφολογικής ανάλυσης.

Στα τρία πρώτα παραδείγματα ορισμένα από αυτά τα ζητήματα μπορούν να αντιμετωπιστούν με την επεξεργασία των σημείων στίξης, και επιπλέον χρησιμοποιώντας το λευκό διάστημα σαν όριο των λέξεων. Ωστόσο, τα σημεία στίξης εμφανίζονται συχνά στο εσωτερικό, σε παραδείγματα όπως u.s.a., Ph.D., AT&T, ma'am, cap'n, 01/02/06, stanford.edu. Παρομοίως στην περίπτωση που θέλουμε το 7.1 ή το 82.4 ως λέξη, δεν μπορούμε να κατατμήσουμε σε κάθε τελεία, δεδομένου ότι αυτό θα τα κατατμήσει σε 7 και 1 και σε 82 και 4. Η βάση δεδομένων θα πρέπει να θεωρείται ως δύο ξεωριστά τμήματα ή ως ένα μονό τμήματα; Ο αριθμός $2,023.74 θα πρέπει να θεωρείται ως ένα μονό τμήμα, αλλά σε αυτή την περίπτωση το κόμμα και η τελέια δεν αντιπροσωπεύουν διαχωριστικά, ενώ σε άλλες περιπώσεις θα γινόταν. Θα πρέπει επίσης, το $ να θεωρείται μέρος του εν λόγω θέματος, ή ένα ξεχωριστό τμήμα από μόνο του;

Η κλάση (class) τμηματοποίησης java.util.SimpleTokenizer της Java είναι ένα παράδειγμα τμηματοποιητή λευκού διαστήματος, οπού μπορεί να οριστεί το σύνολο των χαρακτήρων που σηματοδοτούν τα όρια των τμημάτων. Μία άλλη κλάση της Java, η java.text.BreakIterator, μπορεί να αναγνωρίσει τα όρια μιάς λέξης ή μίας πρότασης, αλλά ακόμα δεν έχει τη δυνατότητα να χειριστεί ασάφειες.

| | Naïve Whitespace Parser | Apache Open NLP 1.5.2 (using en-token.bin) | Stanford 2.0.3 | Custom | Hypothetical Tokenizer (Ideal Tokenization) |
|---|---|---|---|---|---|
| 1 | | " | " | " | " |
| 2 | "i | i | i | i | i |
| 3 | said, | said | said | said | said |
| 4 | | , | , | , | , |
| 5 | 'what're | 'what | ` | ' | ' |
| 6 | | | what | what're | what |
| 7 | | 're | 're | | are |
| 8 | you? | you | you | you | you |
| 9 | | ? | ? | ? | ? |
| 10 | crazy?'" | crazy | crazy | crazy | crazy |
| 11 | | ? | ? | ? | ? |
| 12 | | ' | ' | ' | ' |
| 13 | said | said | said | said | said |
| 14 | sandowsky. | sandowsky | sandowsky | sandowsky | sandowsky |
| 15 | | . | . | . | . |
| 16 | | ' | ' | ' | " |
| 17 | 'i | i | i | i | i |
| 18 | can't | ca | ca | can't | can |
| 19 | | n't | n't | | not |
| 20 | afford | afford | afford | afford | afford |
| 21 | to | to | to | to | to |
| 22 | do | do | do | do | do |
| 23 | that.' | that | that | that | that |
| 24 | | . | . | . | . |
| 25 | | ' | ' | ' | ' |

### 3.1.11 Εξαγωγή ονομαστικής καταχώρησης

Είναι σχεδόν αδύνατο να διαχωριστεί η τμηματοποίηση από την εξαγωγή ονομαστικής καταχώρησης. Πραγματικά δεν είναι δυνατόν να υπάρχει κατάληξη

σε ένα γενικό σύνολο κανόνων το οποίο θα χειρίζεται όλες τις αμφιλεγόμενες περιπτώσεις στα Αγγλικα. Η πιο απλή προσέγγιση, συνήθως είναι η ύπαρξη λεξικών που περιέχουν εκφράσεις πολλών λέξεων. Για παράδειγμα, στην φράση "Install Rational Software Architect on AIX 5.3":

| | Naïve Whitespace Parser | Hypothetical Tokenizer (Ideal Tokenization) |
|---|---|---|
| 1 | install | install |
| 2 | rational | rational software architect for websphere |
| 3 | software | |
| 4 | architect | |
| 5 | for | |
| 6 | websphere | |
| 7 | on | on |
| 8 | aix | aix 5.3 |
| 9 | 5.3 | |

Τα λεξικά πρέπει να υπάρχουν στη διαδικασία της τμηματοποίησης για να εκφράζουν πως το "Rational Software Architect for WebSphere" είναι ένα μονό τμημα (ένα προϊόν), ενώ και το "AIX 5.3", είναι επίσης ένα μονό προϊόν.

Ο αντίκτυπος της τμηματοποίησης κατά το υπόλοιπο της διαδικασίας δεν μπορεί να υποτιμηθεί. Ένα τυπικό επόμενο βήμα, μετά την τμηματοποίηση, είναι να σταλθεί το κατατμημένο κείμενο σε έναν ισχυρό αναλυτή. Στην πρώτη στήλη, το ορθολογικό προϊόν θα αναλυθεί σε βάθος σε μία δομή σαν την εξής:

```
OpenNLP 1.5.2 (en-parser-chunking.bin):
  <node prob="0.99" span="Rational Software Architect for WebSphere" type="NP">
    <node prob="1.0" span="Rational Software Architect" type="NP">
      <node prob="0.86" span="Rational" type="NNP"/>
      <node prob="0.94" span="Software" type="NNP"/>
      <node prob="0.93" span="Architect" type="NNP"/>
    </node>
    <node prob="0.99" span="for WebSphere" type="PP">
      <node prob="0.93" span="for" type="IN"/>
        <node prob="1.0" span="WebSphere" type="NP">
          <node prob="0.24" span="WebSphere" type="NNP"/>
      </node>
    </node>
  </node>
```

(Η έξοδος του Stanford 2.0.3 είναι ίδια). Να σημειωθεί ο σχηματισμός της εμπρόσθετης φράσης (PP, prepositional phrase) γύρω από το "for WebSphere" και η τρίγραμμη φράση ουσιαστικού "Rational Software Architect". Εάν η πρόταση κατατμηθεί σημασιολογικά με τη βοήθεια ενός λεξικού με πολλές λέξεις, η έξοδος από τον ισχυρό αναλυτή θα είναι κάπως έτσι:

```
  <node span="Rational Software Architect for WebSphere" type="NP">
    <node span="Rational Software Architect for WebSphere" type="NNP"/>
  </node>
```

Υπάρχει μια φράση μονού ουσιαστικού που περιέχει ένα ουσιαστικό (NNP = μοναδικό ουσιαστικό).

### 3.1.12 Αγγλικά εγκλιτικά

Ένα κλιτικό είναι μία μονάδα της οποίας η κατάσταση βρίσκεται μεταξύ ενός προσφύματος και μίας λέξης. Η φωνολογική συμπεριφορά ενός κλιτικού είναι σαν του προσφύματος, τείνει να είναι σύντομο και άτονο. Η συντακτική

τους συμπεριφορά μοιάζει περισσότερο με λέξεις, ενεργώντας συχνά ως αντωνυμίες, άρθρα, σύνδεσμοι ή ρήματα. Τα κλιτικά που προηγούνται μιας λέξης ονομάζονται προκλιτικά, ενώ αυτά που ακολουθούν ονομάζονται εγκλιτικά. Τα εγκλιτικά στα Αγγλικά περιλαμβάνουν:

Τη συντομευμένη μορφή του be:

1. ‘m in σε I‘m
2. ‘re σε you‘re
3. ‘s σε she‘s

Τη συντομευμένη μορφή βοηθητικών ρημάτων:

1. ‘ll σε they‘ll
2. ‘ve σε they‘ve
3. ‘d σε you‘d

Αξίζει να σημειωθεί πως τα κλιτικά στα Αγγλικά συχνά είναι ασαφή. Η λέξη ”she’s” μπορεί να σημαίνει ”she has” ή ”she is”. Ένας τμηματοποιητής μπορεί επίσης να χρησιμοποιηθεί για την επέκταση των κλιτικών συνηρημένων λέξεων που έχουν σημανθεί με αποστρόφους, για παράδειγμα:

- what’re → what are
- we’re → we are

Αυτό απαιτεί μια ασαφή ανάλυση, δεδομένου ότι οι απόστροφοι χρησιμοποιούνται επίσης όπως στο ”the book’s over in the containers’ above”ή ως αποσπασματικοί δείκτες. Αν και αυτές οι συνηρημένες λέξεις τείνουν να είναι κλιτικά, δεν δεν σημειώνονται όλα τα κλιτικά με αυτόν τον τρόπο. Σε γενικές γραμμές, η συνέχεια της κατάτμησης και της επέκτασης των κλιτικών, είναι το να ενταχθούν στο πλαίσιο της διαδικασίας μορφολογικής ανάλυσης.

## 3.2 Γράφοντας ένα τμηματοποιητή

Για να μπορέσουμε να καταλάβουμε πως λειτουργεί τμηματοποίηση κειμένου, θα προσπαθήσουμε να γράψουμε ένα πρωτότυπο(prototype) tonkenizer[3] στην γλώσσα προγραμματισμού Python. Πριν το κάνουμε αυτό ας δούμε πως θα μπορούσε να υλοποιηθεί ένας parser όπως δείξαμε νωρίτερα στο κεφάλαιο αυτό:

```
1  #!/usr/bin/env python
2
3  # whitespace tokenizer
4  def whitespace_tokenizer(fname):
5      # use with in order to explicitly close the file after finishing
6      with open (fname, "r") as myfile:
7          data = myfile.read().replace('\n', '') # whole text of file as a string
8          # convert string to lowercase and return it as a list of strings
9          return data.lower().split()
10
11 # check if character chr exists in string str
12 def char_exists(str, chr):
13     if chr in str:
14         return True
15     else:
16         return False
```

Αυτό που κάνουμε ουσιαστικά, είναι να ορίσουμε μία συνάρτηση (function) στην γραμμή 4 η οποία δέχεται σαν όρισμα ένα αρχείο κειμένου. Στην συνέχεια, στην γραμμή 5, στην μεταβλητή (variable) με το όνομα data, αποθηκεύουμε όλο το κείμενο στο αρχείο σαν μία συμβολοσειρά (string), αφαιρώντας τον χαρακτήρα της δημιουργίας νέας γραμμής "\n".

Αξίζει να σημειωθεί ότι διαφορετικά λειτουργικά συστήματα χρησιμοποιούν άλλο χαρακτήρα για την αλλαγή γραμμής. Για παράδειγμα σε περιβάλλον Windows τα αρχεία κειμένου ξεχωρίζουν μία γραμμή από την επόμενη με την ακολουθία χαρακτήρων: "\r\n".

Από την άλλη, τα περισσότερα λειτουργικά συστήματα που είναι βασισμένα στο UNIX, (GNU/Linux, OpenBSD, FreeBSD κλπ) χρησιμοποιούν την ακολουθία "\n". Οι συγγραφείς της παρούσας εργασίας έγραψαν όλο τον κώδικα

[3]Software prototyping ονομάζεται η διαδικασία κατά την οποία δημιουργούνται πρωτότυπα εφαρμογών, δηλαδή ατελείς μορφές του προγράμματος το οποίο αναπτύσσεται

σε περιβάλλον GNU/Linux, γι᾽ αυτό και χρησιμοποιείται η δεύτερη ακολουθία. Τέλος, στην γραμμή 7, η συνάρτηση μας επιστρέφει όλο το κείμενο σαν μία λίστα λέξεων χωρισμένες από το λευκό κενό(whitespace) κι αφού μετατρέψει όλους τους κεφαλαίους χαρακτήρες σε μικρούς.

Ας δοκιμάσουμε τώρα τον απλοϊκό τμηματοποιητή στην πράξη. Πρώτα τρέχουμε στο τερματικό μας τον interpreter της Python:

```
$ python
Python 3.4.3 (default, Mar 25 2015, 17:13:50)
[GCC 4.9.2 20150304 (prerelease)] on linux
Type ‘‘help’’, ‘‘copyright’’, ‘‘credits’’ or ‘‘license’’ for more information.
```

Στην συνέχεια φορτώνουμε τον κώδικα μας στον Python interpreter, καλούμε την συνάρτηση whitespace_tokenizer για το αρχείο κειμένου με το όνομα "test.txt", κι αποθηκεύουμε το αποτέλεσμα στην μεταβλητή "tokens", η οποία είναι μία λίστα συμβοσειρών. Κάθε στοιχείο της λίστας ειναι και ένα token. Στη συνέχεια τυπώνουμε κάθε token σε μία γραμμή για να είνα ξεκάθαρο πως χώρισε ο απλοϊκός τμηματοποιητής το κείμενο μας.

```
>>> import naive
>>> tokens = naive.whitespace_tokenizer('test.txt')
>>> for t in tokens:
...     print(t)
...
"i
said,
'what're
you?
crazy?'"
said
sandowfksy.
"i
can't
afford
to
```

```
do
that"
```

Το αρχείο κειμένου με το όνομα "test.txt" Έχει το εξής κείμενο:

```
"i said, 'what're you? crazy?'" said sandowfksy. "i can't afford to do that"
```

Όπως είναι προφανές, ο τμηματοποιητής αυτός απέχει κατά πολύ από κάτι χρήσιμο. Ας προσπαθήσουμε να κάνουμε κάτι πιο αποτελεσματικό:

```
#!/usr/bin/env python
# demo.py

import naive

tokens = naive.whitespace_tokenizer("test.txt")
newtokens = []

for t in tokens:
    if naive.char_exists(t, "'"):
        toks = t.split("'", 1) # split on first occurence of charachter '
        if t.startswith("'"):
            newtokens.append("'")#, toks[1])
            newtokens.append(toks[1])
        elif t.endswith("'"):
            newtokens.append(toks[0])
            newtokens.append("'")
        else:
            newtokens.append(toks[0])
            newtokens.append("'"+toks[1])
    elif naive.char_exists(t, "\""):
        toks = t.split("\"")
        if t.startswith("\""):
```

```
24              newtokens.append("\"")
25              newtokens.append(toks[1])
26          else:
27              newtokens.append(toks[0])
28              newtokens.append("\"")
29      elif naive.char_exists(t, ","):
30          toks = t.split(",")
31          newtokens.append(toks[0])
32          newtokens.append(",")
33      elif naive.char_exists(t, "?"):
34          toks = t.split("?")
35          newtokens.append(toks[0])
36          newtokens.append("?")
37      elif naive.char_exists(t, "."):
38          toks = t.split(".")
39          newtokens.append(toks[0])
40          newtokens.append(".")
41      else:
42          newtokens.append(t)
43
44  for t in newtokens:
45      print(t)
```

Ο παραπάνω κώδικας βρίσκεται αποθηκευμένος στο αρχείο με το όνομα demo.py. Ας δοκιμάσουμε να το τρέξουμε στο τερματικό να δούμε το αποτέλεσμα:

```
$ ./demo.py
"
i
said
,
’
what’re
you
?
crazy?
’"
said
sandowfksy
.
"
i
can
’t
afford
to
do
that
"
```

Πράγματι, ο τμηματοποιητής μας αυτή την φορά φαίνεται πιο αποτελεσματικός. Πλησιάζει αρκετά στον υποθετικό ιδανικό τμηματοποιητή όπως τον ειχαμε περιγράψει σε προηγούμενες σελίδες.

### 3.2.1 Πως λειτουργεί ο κώδικας

- **γραμμή 4:** Ουσιασικά συμπεριλαμβάνουμε στο πρόγραμμα μας τον κώδικα του απλοϊκού τμηματοποιητή λευκού διαστήματος.
- **γραμμές 6-7:** Δημιουργούμε μία λίστα με το όνομα "tokens" στην οποία βάζουμε την έξοδο του τμηματοποιητή μας. Στην συνέχεια, αρχικοποιούμε μία άδεια λίστα με το όνομα "newtokens".
- **γραμμές 9-42:** Ξεκινάμε ένα "for loop" για κάθε στοιχείο στην λίστα "tokens". Πιο συγκεκριμένα:

- **γραμμές 10-20:** Ελέγχουμε κάθε token αν περιέχει τον χαρακτήρα ' . Αν αυτό είναι αληθές, βρίσκουμε σε ποιο σημείο του token βρίσκεται(αρχή, τέλος ή ενδιάμεσα) και στην συνέχεια "σπάμε" εκ νέου το token σε επιμέρους tokens. Τα αποτελέσματα αποθηκεύονται στην λίστα "newtokens".
- **γραμμές 21-28, 29-32, 33-36, 37-40:** Επαναλαμβάνουμε την ίδια διαδικασία για τους χαρακτήρες " , , , ? , και . , αντίστοιχα.

- **γραμμές 44-45:** Ένα απλό for loop το οποίο τυπώνει κάθε token της λίστας "newtokens" σε μία γραμμή.

### 3.2.2 Συμπεράσματα - Επίλογος

Παρόλο που φαίνεται ότι το prototype του τμηματοποιητή μας να πλησιάζει αρκετά το αποτέλεσμα ενός ιδανικού τμηματοποιητή, στην πραγματικότητα αυτό δεν είναι αληθές. Θα πρέπει να συμπεριληφθούν πολλοί περισσότεροι κανόνες για να μπορέσει να φτάσει σε ένα ικανοποιητικό επίπεδο για να συναγωνιστεί με τους διάσημους τμηματοποιητές όπως τα **NLTK, OpenNLP, Stanford Tokenizer** κλπ αλλά και ακόμα περισσότερος χρόνος μελέτης.

Αδιαμφισβήτητα, ως συγγραφείς της παρούσας πτυχιακής αποκόμισαμε πάρα πολλές γνώσεις σχετικά με την επεξεργασία του φυσικού λόγου, το εύρος των εφαρμογών της στο παρόν αλλά ακόμα περισσότερο στο μέλλον. Επιπλέον είδαμε με λεπτομέρειες πως κατασκευάζεται ένας τμηματοποιητής κειμένου. Ελπίζουμε και ο αναγνώστης να απολαύσει το ίδιο με εμάς το αποτέλεσμα της εκπόνησης μίας τέτοιας εργασίας.

# Παράρτημα Α΄

# Λίστα με καταλήξεις

Ακολουθεί η λίστα με τις καταλήξεις όπως αναφέραμε στην σελίδα 33:

LIST OF ENDINGS*

| .11. | .09. | .09.—*Cont.* | .08. |
|---|---|---|---|
| alistically B | allically C | entiality A | ableness A |
| arizability A | antaneous A | entialize A | arizable A |
| izationally B | antiality A | entiation A | entation A |
| | arisation A | ionalness A | entially A |
| .10. | arization A | istically A | eousness A |
| antialness A | ationally B | itousness A | ibleness A |
| arisations A | ativeness A | izability A | icalness A |
| arizations A | eableness E | izational A | ionalism A |
| entialness A | entations A | | ionality A |

| .08.—*Cont.* | .06.—*Cont.* | .05.—*Cont.* | .03. |
|---|---|---|---|
| ionalize A | ialist A | inate A | acy A |
| iousness A | iality A | iness A | age B |
| izations A | ialize A | ingly B | aic A |
| lessness A | ically A | inism J | als BB |

.07.
ability A
aically A
alistic B
alities A
ariness E
aristic A
arizing A
ateness A
atingly A
ational B
atively A
ativism A
elihood E
encible A
entally A
entials A
entiate A
entness A
fulness A
ibility A
icalism A
icalist A
icality A
icalize A
ication G
icianry A
ination A
ingness A
ionally A
isation A
ishness A
istical A
iteness A
iveness A
ivistic A
ivities A
ization F
izement A
oidally A
ousness A

.06.
aceous A
acious B
action G
alness A
ancial A
ancies A
ancing B
ariser A
arized A
arizer A
atable A
ations B
atives A
eature Z
efully A
encies A
encing A
ential A
enting C
entist A
eously A
icians A
icists A
ifully A
ionals A
ionate D
ioning A
ionist A
iously A
istics A
izable E
lessly A
nesses A
oidism A

.05.
acies A
acity A
aging B
aical A
alist A
alism B
ality A
alize A
allic BB
anced B
ances B
antic C
arial A
aries A
arily A
arity B
arize A
aroid A
ately A
ating I
ation B
ative A
ators A
atory A
ature E
early Y
ehood A
eless A
elily A
ement A
enced A
ences A
eness E
ening E
ental A
ented C
ently A
fully A
ially A
icant A
ician A
icide A
icism A
icist A
icity A
idine I
iedly A
ihood A
ional A
ioned A
ished A
istic A
ities A
itous A
ively A
ivity A
izers F
izing F
oidal A
oides A
otide A
ously A

.04.
able A
ably A
ages B
ally B
ance B
ancy B
ants B
aric A
arly K
ated I
ates A
atic B
ator A
ealy Y
edly E
eful A
eity A
ence A
ency A
ened E
enly E
eous A
hood A
ials A
ians A
ible A
ibly A
ical A
ides L
iers A
iful A
ines M
ings N
ions B
ious A
isms B
ists A
itic H
ized F
izer F
less A
lily A
ness A
ogen A
ward A
wise A
ying B
yish A
ars O
ary F
ata A
ate A
eal Y
ear Y
ely E
ene E
ent C
ery E
ese A
ful A
ial A
ian A
ics A
ide L
ied A
ier A
ies P
ily A
ine M
ing N
ion Q
ish C
ism B
ist A
ite AA
ity A
ium A
ive A
ize F
oid A
one R
ous A

.02.
ae A
al BB
ar X
as B
ed E
en F
es E
ia A
ic A
is A
ly B
on S
or T
um U
us V
yl R
s' A
's A

.01.
a A
e A
i A
o A
s W
y B

# Παράρτημα Β΄

# Δοκιμές κανόνων συμφραζομένων

Ακολουθούν οι κώδικες κατάστασης για τους κανόνες που βασίζονται στα συμφραζόμενα και σε συγκεκριμένες καταλήξεις όπως αναφέραμε στην σελίδα 33:

**A** . . . No restrictions on stem
**B** . . . Minimum stem length = 3
**C** . . . Minimum stem length = 4
**D** . . . Minimum stem length = 5
**E** . . . Do not remove ending after *e*
**F** . . . Minimum stem length = 3 and do not remove ending after *e*
**G** . . . Minimum stem length = 3 and remove ending only after *f*
**H** . . . Remove stem ending only after *t* or *ll*
**I** . . . Do not remove ending after *o* or *e*
**J** . . . Do not remove ending after *a* or *e*
**K** . . . Minimum stem length = 3 and remove ending only after *l*, *i*, or *uae* (where *α* stands for any letter)
**L** . . . Do not remove ending after *u*, *x*, or *s*, unless *s* follows *o*
**M** . . . Do not remove ending after *a*, *c*, *e*, or *m*
**N** . . . Minimum stem length = 4 after *sα*, elsewhere = 3
**O** . . . Remove ending only after *l* or *i*
**P** . . . Do not remove ending after *c*
**Q** . . . Minimum stem length = 3 and do not remove ending after *l* or *n*
**R** . . . Remove ending only after *n* or *r*
**S** . . . Remove ending only after *dr* or *t*, unless *t* follows *t*
**T** . . . Remove ending only after *s* or *t*, unless *t* follows *o*
**U** . . . Remove ending only after *l*, *m*, *n*, or *r*
**V** . . . Remove ending only after *c*
**W** . . . Do not remove ending after *s* or *u*
**X** . . . Remove ending only after *l*, *i*, or *uae*
**Y** . . . Remove ending only after *in*
**Z** . . . Do not remove ending after *f*
**AA** . . . Remove ending only after *d*, *f*, *ph*, *th*, *l*, *er*, *or*, *es*, or *t*
**BB** . . . Minimum stem length = 3 and do not remove ending after *met* or *ryst*
**CC** . . . Remove ending only after *l*

# Παράρτημα Γ′

# Κανόνες κωδικοποίησης

1 . . . Remove one of **double** *b, d, g, l, m, n, p, r, s, t*
2 . . . iev → **ief**
3 . . . **uct** → **uc**
4 . . . umpt → **um**
5 . . . rpt → **rb**
6 . . . urs → **ur**
7 . . . istr → **ister**
7*a* . . **metr** → **meter**
8 . . . olv → **olut**
9 . . . **ul** → **l** except following *a, i, o*
10 . . . bex → **bic**
11 . . . dex → **dic**
12 . . . pex → **pic**
13 . . . tex → **tic**
14 . . . ax → **ac**
15 . . . ex → **ec**
16 . . . ix → **ic**
17 . . . lux → **luc**
18 . . . uad → **uas**
19 . . . vad → **vas**
20 . . . **cid** → **cis**
21 . . . lid → **lis**
22 . . . erid → **eris**
23 . . . pand → **pans**
24 . . . end → **ens** except following *s*
25 . . . ond → **ons**
26 . . . lud → **lus**
27 . . . rud → **rus**
28 . . . her → **hes** except following *p, t*
29 . . . mit → **mis**
30 . . . end → **ens** except following *m*
31 . . . ert → **ers**
32 . . . et → **es** except following *n*
33 . . . yt → **ys**
34 . . . yz → **ys**

# Παράρτημα Δ΄

# Τερματισμός θεματοποίησης

Με $\beta$ συμβολίζονται τα κενά.

| | |
|---|---|
| ββ | er |
| bβ | es |
| cβ | ex |
| dβ | gg |
| gβ | ic |
| lβ | il |
| nβ | is |
| pβ | nn |
| rβ | or |
| sβ | os |
| tβ | ot |
| vβ | pp |
| xβ | pt |
| zβ | ss |
| bb | tt |
| ct | ul |
| dd | ut |

# Παράρτημα Ε΄

# Βιβλιογραφία


1. Taming Text, Grant S. Ingersoll, Thomas S. Morton, Andrew S. Farris
2. An introduction to natural language processing, Daniel Jurafsky & James H. Martin
3. Earl, Lois L. "Part-of-Speech Implications of Affixes." Mechanical Translation and Computational Linguistics, vol. 9 Ιούνιος 1966
4. Lejnieks Valdis. "The System of English Suffixes"
5. Development of a Stemming Algorithm, Julie Beth Lovins, Ιούνιος 1968
6. The SMART Automatic Document Retrieval System, Gerard Salton
7. Automatic Information Organization and Retrieval, Gerard Salton, 1968
8. A. F. Brown, Normal and Reverse Word List
9. Webster Jonathan, Kit Chunyu, - "Tokenization as the initial phase in NLP", 1992