Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τριτη Εργαστηριακη Άσκηση
Επεξεργασια Φωνης και Φυσικης Γλωσσας

Αναστασία Χριστίνα Λίβα
03119029
Μυρτώ Ορφανάκου
03119173

Περιεχόμενα

1 Μέρος 1: Ανάπτυξη Ορθογραφικού Ελέγχου

Το αρχείο `script_part1.py` ενσωματώνει την υλοποίηση των πρώτων δέκα βημάτων του αρχικού μέρους της άσκησης. Επιπλέον, το αρχείο `script_step11.py` περιέχει το ενδέκατο, προαιρετικό βήμα της άσκησης. Κατά τη διαδικασία της υλοποίησης, γίνεται χρήση πληθώρας αρχείων από τον φάκελο `scripts`, ορισμένα από τα οποία έχουν επεξεργαστεί ή έχουν τροποποιηθεί ανάλογα.

Τα αρχεία `part1.py` και `step11.py` πρέπει να εκτελούνται από τον φάκελο `slp_lab`, ώστε να αποφευχθούν τυχόν σφάλματα κατά την αναζήτηση των σχετικών αρχείων που χρησιμοποιούνται στη διαδικασία.

1.1 Βήμα 1: Δημιουργία Corpus

Στο πρώτο στάδιο της άσκησης, εκτελούμε το παρεχόμενο script με στόχο την αποθήκευση του corpus στο αρχείο `corpus.txt`. Κατά την εκτέλεση του script, πραγματοποιείται η ακόλουθη προεπεξεργασία του corpus:

1. **Απομόνωση Γραμμών**: Το script απομονώνει κάθε γραμμή του αρχείου.
2. **Αφαίρεση Κενών Χαρακτήρων**: Αφαιρεί τους κενούς χαρακτήρες από την αρχή και το τέλος κάθε γραμμής.
3. **Μετατροπή σε Μικρά Γράμματα**: Μετατρέπει όλους τους χαρακτήρες σε πεζά γράμματα.
4. **Αποκατάσταση Συναιρέσεων**: Επιστρέφει την πλήρη μορφή των φράσεων που περιέχουν συναίρεση.
5. **Καθαρισμός Κενών Διαστημάτων**: Αφαιρεί πολλαπλά κενά διαστήματα και διατηρεί μόνο πεζούς χαρακτήρες και μονά κενά διαστήματα.
6. **Διαχωρισμός Tokens**: Διαχωρίζει το προεπεξεργασμένο κείμενο σε μεμονωμένα tokens χρησιμοποιώντας το κενό διάστημα ως διαχωριστικό, τα οποία αποθηκεύει σε λίστα.
7. **Δημιουργία Τελικής Έξοδου**: Από τη λίστα που δημιουργείται για κάθε γραμμή, παράγει την τελική έξοδο, αγνοώντας τις κενές συμβολοσειρές.

Τα συγκεκριμένα βήματα τυποποιούν το κείμενο με στόχο να αποφευχθούν σφάλματα κατά τη χρήση των εργαλείων στα επόμενα στάδια και να αφαιρεθούν στοιχεία που μπορεί να λειτουργήσουν ως θόρυβος. Παρόλα αυτά, υπάρχουν περιπτώσεις όπου η τόσο επιθετική προεπεξεργασία μπορεί να μην είναι επιθυμητή.

Για παράδειγμα, στην ανάλυση συναισθήματος, τα σημεία στίξης, όπως τα θαυμαστικά, μπορούν να παρέχουν σημαντικές ενδείξεις για το συναίσθημα που εκφράζεται. Επιπλέον, η διαχείριση κειμένων που περιέχουν πολλά αρκτικόλεξα μπορεί να επωφεληθεί από τη διατήρηση της αρχικής μορφής τους για να αποφεύγονται παρανοήσεις ή απώλεια σημαντικής πληροφορίας.

Η επέκταση του corpus με περισσότερα βιβλία ή κείμενα από ποικίλες πηγές προσφέρει σημαντικά πλεονεκτήματα πέρα από την απλή αύξηση του μεγέθους των δεδομένων. Αυτά τα πλεονεκτήματα περιλαμβάνουν:

1. **Μεγαλύτερη Ποικιλομορφία Δεδομένων**: Η ενσωμάτωση κειμένων με ιδιωματισμούς, εξειδικευμένη ορολογία, αργκό και άλλες γλωσσικές ποικιλομορφίες συμβάλλει στην καλύτερη γενίκευση του μοντέλου. Αυτό επιτρέπει στο μοντέλο να επεκτείνεται και να εφαρμόζεται σε ευρύτερο φάσμα κειμένων, βελτιώνοντας την απόδοσή του σε διαφορετικά γλωσσικά περιβάλλοντα.

2. **Μείωση Ανωμαλιών και Ακραίων Σημείων**: Η προσθήκη ποικιλίας κειμένων βοηθά στη μείωση των ανωμαλιών και των ακραίων σημείων, αποφεύγοντας το φαινόμενο του υπερπροσαρμογής (overfitting). Για παράδειγμα, σε ένα corpus όπως το Project Gutenberg, όπου περιλαμβάνεται το βιβλίο «Η Αλίκη στη Χώρα των Θαυμάτων», η λέξη "Αλίκη" μπορεί να εμφανίζεται με μεγάλη συχνότητα. Με την προσθήκη επιπλέον κειμένων, η σχετική συχνότητα αυτής της λέξης θα μειωθεί, εξασφαλίζοντας πιο ισορροπημένη κατανομή λέξεων και βελτιωμένη γενίκευση του μοντέλου.

1.2 Βήμα 2: Κατασκευή Λεξικού

Το λεξικό μας έχει τη μορφή {`token: key`}, όπου το κλειδί αντιπροσωπεύει τον αριθμό εμφανίσεων κάθε λέξης. Αυτή η δομή διευκολύνει το φιλτράρισμα στο επόμενο βήμα. Η αφαίρεση λέξεων που εμφανίζονται λιγότερες από πέντε φορές είναι σημαντική, καθώς μειώνει την πολυπλοκότητα (χρονική και χωρική) για τις διαδικασίες των επόμενων βημάτων, όπως η δημιουργία των FSTs (Finite State Transducers), η διόρθωση και η αξιολόγηση, χωρίς να επηρεάζεται σημαντικά η ακρίβεια του ορθογράφου. Αυτό συμβαίνει επειδή οι λέξεις που αφαιρούνται είναι σπάνιες και δεν συνεισφέρουν ουσιαστικά στη συνολική απόδοση του συστήματος.

Αποθηκεύουμε το λεξικό στο αρχείο `vocab/words.vocab.txt`, με κάθε γραμμή να έχει τη μορφή {`word`}\t{`frequency`}.

1.3 Βήμα 3: Δημιουργία Συμβόλων Εισόδου/Εξόδου

Σε αυτό το βήμα, δημιουργούμε τα αρχεία συμβόλων εισόδου/εξόδου για να διευκολύνουμε τις επόμενες διαδικασίες.

1. **Αρχείο Χαρακτήρων**: Δημιουργούμε το αρχείο `chars.syms` στον φάκελο `vocab`, το οποίο περιέχει δύο στήλες: τους χαρακτήρες και τους αντίστοιχους δείκτες . Η μορφή των δεδομένων στο αρχείο είναι διαχωρισμένη με tab.
    - Στην πρώτη γραμμή του αρχείου περιλαμβάνουμε το σύμβολο `<eps>` με δείκτη `0`.
    - Στις επόμενες γραμμές, εισάγουμε τους αγγλικούς πεζούς χαρακτήρες (lowercase) με τον ASCII κωδικό τους ως δείκτη.
2. **Αρχείο Λέξεων**: Δημιουργούμε το αρχείο `words.syms` με παρόμοιο τρόπο:
    - Στην πρώτη γραμμή περιλαμβάνεται το σύμβολο `<eps>` με δείκτη `0`.
    - Στις επόμενες γραμμές, περιλαμβάνονται οι λέξεις του λεξικού, με ακέραιους δείκτες που ξεκινούν από το `1` και αυξάνονται μέχρι το πλήθος των λέξεων.

1.4 Βήμα 4: Κατασκευή Μετατροπέα Edit Distance

Σε αυτό το βήμα, κατασκευάζουμε τον μετατροπέα L που υλοποιεί την απόσταση Levenshtein, αντιστοιχίζοντας τις ακόλουθες μετατροπές:

1. **Καμία Αλλαγή**: Κάθε χαρακτήρας αντιστοιχεί στον εαυτό του με βάρος 0.
2. **Διαγραφή**: Κάθε χαρακτήρας αντιστοιχεί στο κενό (ϵ) με βάρος 1.
3. **Εισαγωγή**: Το κενό (ϵ) αντιστοιχεί σε κάθε χαρακτήρα με βάρος 1.
4. **Αντικατάσταση**: Κάθε χαρακτήρας αντιστοιχεί σε κάθε άλλο χαρακτήρα με βάρος 1.

Για την κατασκευή των μετατροπέων, ορίσαμε τη συνάρτηση `format_arc`, η οποία επιστρέφει τα ορίσματά της στοιχισμένα κατάλληλα. Η συνάρτηση αυτή είναι κρίσιμη για τη σωστή μορφοποίηση των *arcs* που αποτελούν τις μεταβάσεις στο FST.

Ακολουθώντας το βέλτιστο μονοπάτι σε μία λέξη εισόδου, ο μετατροπέας θα παράγει τη λέξη στην οποία μετασχηματίζεται η είσοδος, επιλέγοντας το μονοπάτι μετασχηματισμών με το ελάχιστο κόστος.

Αποθηκεύουμε τον μετατροπέα L ως `L.fst` και με τη χρήση της εντολής `fstcompile` παράγουμε το δυαδικό αρχείο `L.binfst`.

1.4.1 Άλλα Πιθανά Edits

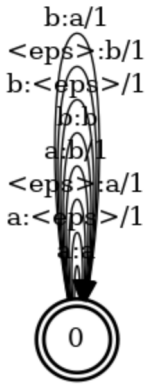
Εκτός από τις βασικές μετατροπές, μπορούμε να συμπεριλάβουμε και άλλες, όπως:

- **Αντιστροφή Χαρακτήρων (Transposition)**: Με μικρότερο βάρος από δύο αλλαγές χαρακτήρων, καθώς αυτή η αλλαγή μπορεί να συμβεί συχνά κατά την πληκτρολόγηση.
- **Συμπερίληψη Χαρακτήρων για Δυσλεξία**: Μπορούμε να συμπεριλάβουμε ειδικούς χαρακτήρες που επηρεάζουν άτομα με δυσλεξία, όπως η αντιστοίχιση του '3' με το 'ε', λόγω της κοντινής τους θέσης στο πληκτρολόγιο.

Η εισαγωγή πρότερης γνώσης μπορεί να βελτιώσει τον μετατροπέα και να βοηθήσει στην καλύτερη επιλογή βαρών για συγκεκριμένες τροποποιήσεις. Αυτή η διαδικασία περιλαμβάνει την αναπροσαρμογή των βαρών για τα edits με βάση συγκεκριμένα κριτήρια, όπως η εγγύτητα των χαρακτήρων στο πληκτρολόγιο και η φωνητική ή οπτική ομοιότητα των χαρακτήρων.

1.4.2 Προσαρμογές Βαρών για Edits

1. **Κοντινοί Χαρακτήρες στο Πληκτρολόγιο**:
   - Οι αλλαγές μεταξύ χαρακτήρων που βρίσκονται σε διπλανές θέσεις στο πληκτρολόγιο θα έχουν μικρότερο βάρος.
   - Παράδειγμα: Η αλλαγή από `i` σε `o` θα έχει μικρότερο βάρος από την αλλαγή από `i` σε `a`.
2. **Φωνητική ή Οπτική Ομοιότητα**:
   - Οι αλλαγές μεταξύ χαρακτήρων με φωνητική ή οπτική ομοιότητα θα έχουν μικρότερο βάρος.
   - Παράδειγμα: Οι αλλαγές `s` $\rightarrow$ `c` και `s` $\rightarrow$ `z` θα έχουν μικρότερο βάρος σε σύγκριση με άλλες τροποποιήσεις.



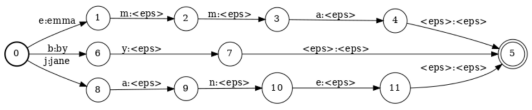
Σχήμα 1: Levenshtein

1.5 Βήμα 5: Κατασκευή Αποδοχέα Λεξικού

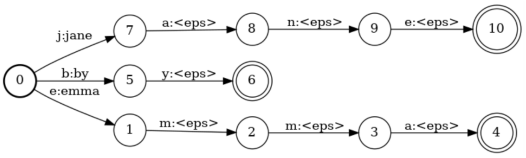
Σε αυτό το βήμα, κατασκευάζουμε τον αποδοχέα V, ο οποίος αποδέχεται τις λέξεις που ανήκουν στο λεξικό. Ο αποδοχέας V διαθέτει μία αρχική κατάσταση και μηδενικά βάρη. Η λειτουργία του περιλαμβάνει τα εξής:

1. **Αρχική Κατάσταση και Μεταβάσεις**:
   - Με την πρώτη μετάβαση, ο αποδοχέας αποδέχεται τη λέξη μέσω του πρώτου γράμματος.
   - Για τα υπόλοιπα γράμματα, πραγματοποιούνται μεταβάσεις $\{\texttt{char} \rightarrow \epsilon\}$ που καταλήγουν στην κατάσταση αποδοχής.
2. **Βελτιστοποίηση του Αποδοχέα** V: Για να βελτιστοποιήσουμε τον αποδοχέα, χρησιμοποιούμε τις παρακάτω συναρτήσεις:
   - `fstrmepsilon`: Αφαιρεί τις μεταβάσεις $\epsilon \rightarrow \epsilon$, καθιστώντας τον μετατροπέα πιο απλό και αποδοτικό.
   - `fstdeterminize`: Καθιστά τον μετατροπέα ντετερμινιστικό, δηλαδή εξασφαλίζει ότι δεν υπάρχει κατάσταση με περισσότερες από μία μεταβάσεις για την ίδια είσοδο.
   - `fstminimize`: Ελαχιστοποιεί τον μετατροπέα, δημιουργώντας έναν ισοδύναμο μετατροπέα με τον ελάχιστο αριθμό καταστάσεων.

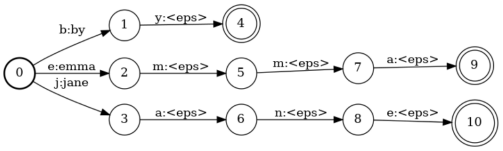
Με την ολοκλήρωση αυτού του βήματος, έχουμε κατασκευάσει έναν αποδοχέα V που είναι αποτελεσματικός και αποδοτικός για την αναγνώριση των λέξεων του λεξικού.



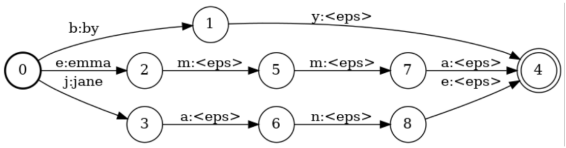
Σχήμα 2: Αποδοχέας V πριν από την εφαρμογή οποιασδήποτε βελτιστοποίησης



Σχήμα 3: χρήση της rmepsilon



Σχήμα 4: Χρησιμοποιείται η fstdeterminize: δεν παρατηρείται αλλαγή στις καταστάσεις, μόνο αλφαβητική ταξινόμηση στην απεικόνιση



Σχήμα 5: βελτιστοποιημένος V μετά την fstminimize

1.6 Βήμα 6: Κατασκευή Ορθογράφου

Για την κατασκευή του ορθογράφου S, συνθέτουμε τον μετατροπέα L με τον αποδοχέα V, χρησιμοποιώντας τη μέθοδο του ελάχιστου μονοπατιού. Αυτή η μέθοδος ελαχιστοποιεί το κόστος των απαραίτητων διορθώσεων για τη μετατροπή μιας λανθασμένης λέξης εισόδου σε μια ορθή λέξη, δηλαδή σε μια από τις λέξεις που ανήκουν στο λεξικό.

Πριν από τη σύνθεση, εφαρμόζουμε τη συνάρτηση `fstarcsort` στην έξοδο του L και στην είσοδο του V. Αυτή η συνάρτηση εξασφαλίζει τη σωστή ταξινόμηση των *arcs*, αποτρέποντας την εμφάνιση σφαλμάτων και μειώνοντας τον χρόνο εκτέλεσης κατά τη σύνθεση των δύο μετατροπέων.

Κατά την πρώτη περίπτωση, όπου οι διορθώσεις έχουν το ίδιο κόστος, ο ορθογράφος επιδιώκει να επιτύχει τις λιγότερες δυνατές μεταβολές μέχρι να φτάσει σε μια αποδεκτή λέξη. Ωστόσο, όταν τα *edits* έχουν διαφορετικά βάρη, ο μετατροπέας προσπαθεί να ελαχιστοποιήσει το συνολικό κόστος των διορθώσεων. Αυτό μας δίνει τη δυνατότητα να καθοδηγήσουμε τον ορθογράφο προς συγκεκριμένες διορθώσεις που εμφανίζονται με μεγαλύτερη συχνότητα, βελτιώνοντας έτσι την απόδοσή του.

1.6.1 Παράδειγμα Διορθώσεων

Για τις εισόδους `"cit"` και `"cwt"`, οι πιθανές προβλέψεις του ορθογράφου είναι οι εξής:

- `cit`:
  - `kit` (1 αλλαγή)

- `cut` (1 αλλαγή)
- `cat` (1 αλλαγή)
- `it` (1 διαγραφή)
- `city` (1 προσθήκη)

Σημείωση: Η λέξη `"cite"` με κόστος 1 δεν περιλαμβάνεται στο λεξικό.

- `cwt`:
  - `cut` (1 αλλαγή)
  - `cut` (επαναλαμβάνεται λόγω ίδιου κόστους και επιλογής)

1.7 Βήμα 7: Δοκιμή Ορθογράφου

Αφού κατεβάσαμε το σύνολο δεδομένων για την αξιολόγηση, χρησιμοποιήσαμε το script `predict.sh` (στο οποίο αλλάξαμε την εντολή `head -n -1` σε `sed '$d'` για να λειτουργεί στα μηχανήματά μας) για τις πρώτες 20 λέξεις που περιλαμβάνει. Το συγκεκριμένο script χρησιμοποιεί το `mkfstinput.py` για να δημιουργήσει έναν αποδοχέα για τη λανθασμένη λέξη και έπειτα τον συνθέτει με τον ορθογράφο, βρίσκοντας το συντομότερο μονοπάτι. Στη συνέχεια, αφαιρεί τις μεταβάσεις $\epsilon \to \epsilon$ και τυπώνει με την `fstprint` τις καταστάσεις που περιλαμβάνει το κοντινότερο μονοπάτι και τις εξόδους τους.

1.7.1 Λειτουργία του script

1. **Δημιουργία Αποδοχέα**: Το script `mkfstinput.py` δημιουργεί έναν αποδοχέα για τη λανθασμένη λέξη.
2. **Σύνθεση με τον Ορθογράφο**: Ο αποδοχέας συντίθεται με τον ορθογράφο για να βρεθεί το συντομότερο μονοπάτι.
3. **Αφαίρεση Μεταβάσεων** $\epsilon \to \epsilon$: Οι μεταβάσεις $\epsilon \to \epsilon$ αφαιρούνται από την έξοδο για να καθαριστεί το αποτέλεσμα.
4. **Εκτύπωση Καταστάσεων**: Η εντολή `fstprint` χρησιμοποιείται για την εκτύπωση των καταστάσεων που περιλαμβάνει το κοντινότερο μονοπάτι και των εξόδων τους.
5. **Διατήρηση Μόνο των Εξόδων**: Από την έξοδο του `fstprint`, διατηρούνται μόνο οι έξοδοι. Οι ϵ-έξοδοι και η γραμμή της τελικής κατάστασης αφαιρούνται, και προκύπτει η διορθωμένη λέξη εξόδου.

Wrong word	Corrected word	Correct word
contentped	contented	contented
contende	contended	contented
begining	beginning	beginning
problam	problem	problem
promblem	problem	problem
dirven	given	driven
exstacy	ecstasy	ecstasy
ecstacy	ecstasy	ecstasy
leval	legal	level
triangulaur	triangular	triangular
juise	juice	juice
poartry	party	poetry
poetry	poetry	poetry
unexspected	unexpected	unexpected
varable	parable	variable

1.8 Βήμα 8: Υπολογισμός Κόστους των Edits

Σε αυτό το βήμα, χρησιμοποιούμε το αρχείο `wiki.txt`, το οποίο περιέχει συχνά ορθογραφικά λάθη, με σκοπό την καλύτερη στάθμιση των βαρών για κάθε *edit*.

1.8.1 Διαδικασία Υπολογισμού Κόστους

1. **Αφαίρεση Μη Αγγλικών Χαρακτήρων**: Αρχικά, αφαιρούμε από το αρχείο `wiki.txt` τις λέξεις που περιέχουν χαρακτήρες που δεν είναι αγγλικοί (π.χ., á).
2. **Εκτέλεση του Script `word_edits.sh`**: Τρέχουμε το script `word_edits.sh` για το παράδειγμα που δίνεται (π.χ., `abandonned` → `abandoned`). Αυτό το script εκτελεί τα εξής βήματα:
   (α′) **Δημιουργία Αποδοχέα** M: Δημιουργεί τον αποδοχέα M για τη λανθασμένη λέξη.
   (β′) **Δημιουργία Μετατροπέα** N: Δημιουργεί τον μετατροπέα N.
   (γ′) **Σύνθεση** MLN: Παράγεται η σύνθεση MLN.
   (δ′) **Εύρεση Συντομότερου Μονοπατιού**: Καλείται η `fstshortestpath` για την εύρεση του συντομότερου μονοπατιού στο MLN.
   (ε′) **Εκτύπωση των Edits**: Χρησιμοποιείται η `fstprint` μαζί με τις εντολές `grep` και `cut` για να τυπωθούν τα *edits*, διατηρώντας μόνο αυτά με μη μηδενικό κόστος.

```
(base) anastasia@anastasia-Inspiron-5570:~/Desktop/slp_lab$ bash scripts/word_edits.sh octomber
    ↪ october
m    <eps>
(base) anastasia@anastasia-Inspiron-5570:~/Desktop/slp_lab$ bash scripts/word_edits.sh acheive
    ↪ achieve
i    e
e    i
(base) anastasia@anastasia-Inspiron-5570:~/Desktop/slp_lab$ bash scripts/word_edits.sh amentment
    ↪ ammendmend
t    d
t    d
<eps> m
```

```
(base) anastasia@anastasia-Inspiron-5570:~/Desktop/slp_lab$
```

Η παραπάνω διαδικασία εκτελείται για κάθε γραμμή του `wiki.txt` με σκοπό την αποθήκευση όλων των *edits* στο αρχείο `edits.txt`. Στη συνέχεια, δημιουργούμε ένα λεξικό που περιλαμβάνει κάθε *edit* καθώς και τη συχνότητα εμφάνισής του.

1.8.2 Δημιουργία του Transducer E

Ο transducer E κατασκευάζεται παρόμοια με τον transducer L, ωστόσο τα βάρη κάθε ακμής του είναι διαφορετικά. Συγκεκριμένα, από τη συχνότητα εμφάνισης κάθε *edit* εξάγουμε την πιθανότητα εμφάνισής του (δηλαδή, *freq. of edit / total number of edits*) και θέτουμε ως βάρος τον αρνητικό λογάριθμο της συγκεκριμένης πιθανότητας. Εάν κάποιο *edit* δεν περιέχεται στο λεξικό που σχηματίσαμε, θέτουμε ως βάρος τη σταθερά `infinity`. Με αυτή τη μέθοδο, ωθούμε τον transducer να προτιμά *edits* που εμφανίζονται πιο συχνά, όπως αναφέρθηκε και σε προηγούμενα στάδια.

Αφού κατασκευάσουμε τον transducer E, επαναλαμβάνουμε τα βήματα 6 και 7:

1. **Σύνθεση του Ορθογράφου** EL: Σύνθεση του transducer E με τον αποδοχέα L για να δημιουργήσουμε τον ορθογράφο EL.
2. **Αξιολόγηση**: Καλούμε εκ νέου τη συνάρτηση αξιολόγησης για να ελέγξουμε την απόδοση του ορθογράφου EL.

Wrong word	**Corrected word**	**Correct word**
contentped	contented	contented
contende	contend	contented
begining	beginning	beginning
problam	problem	problem
promblem	problem	problem
dirven	driven	driven
exstacy	ecstasy	ecstasy
ecstacy	ecstasy	ecstasy
leval	level	level
triangulaur	triangular	triangular
juise	juice	juice
poartry	poetry	poetry
poetry	poetry	poetry
unexspected	unexpected	unexpected
varable	invariable	variable

Παρατηρούμε, όπως αναμενόταν, ότι η έξοδος έχει βελτιωθεί, καθώς οι μεταβολές που επιβάλλει ο ορθογράφος είναι πιο στοχευμένες και ακριβείς. Οι διορθώσεις πλέον βασίζονται στις συχνότερα εμφανιζόμενες αλλαγές, οδηγώντας σε μια πιο αποτελεσματική και ακριβή διόρθωση των ορθογραφικών λαθών.

1.9 Βήμα 9: Εισαγωγή της Συχνότητας Εμφάνισης Λέξεων (Unigram Word Model)

Σε αυτό το βήμα, χρησιμοποιούμε τη συχνότητα εμφάνισης των λέξεων του *corpus* ώστε να προτείνονται από τον ορθογράφο οι λέξεις που εμφανίζονται περισσότερο. Χρησιμοποιώντας το λεξικό του

Βήματος 2, κατασκευάζουμε τον αποδοχέα μίας κατάστασης W, ο οποίος αντιστοιχίζει κάθε λέξη στον εαυτό της με βάρος τον αρνητικό λογάριθμο της πιθανότητας εμφάνισης της λέξης στο *corpus*.

1.9.1 Διαδικασία Κατασκευής και Βελτιστοποίησης του W

1. **Κατασκευή του Αποδοχέα W**:
   - Χρησιμοποιούμε το λεξικό για να δημιουργήσουμε τον αποδοχέα W.
   - Κάθε λέξη αντιστοιχεί στον εαυτό της με βάρος τον αρνητικό λογάριθμο της πιθανότητας εμφάνισης της λέξης στο *corpus*.
2. **Βελτιστοποίηση του W**:
   - Καλούμε τις κατάλληλες συναρτήσεις (`fstrmepsilon`, `fstdeterminize`, `fstminimize`) για να βελτιστοποιήσουμε τον W.

1.9.2 Κατασκευή Ορθογράφων LVW και EVW

Ακολουθώντας τα βήματα των προηγούμενων σταδίων, κατασκευάζουμε τους ορθογράφους LVW και EVW:

1. LVW: Συνθέτουμε τον transducer L με τον αποδοχέα W.
2. EVW: Συνθέτουμε τον transducer E με τον αποδοχέα W.

1.9.3 Αξιολόγηση

Τρέχουμε τα κατάλληλα scripts για την αξιολόγηση του LVW και παίρνουμε τα εξής αποτελέσματα:

Wrong word	**Corrected word**	**Correct word**
contentped	the	contented
contende	the	contented
begining	beginning	beginning
problam	of	problem
promblem	the	problem
dirven	the	driven
exstacy	the	ecstasy
ecstacy	the	ecstasy
leval	the	level
triangulaur	and	triangular
juise	the	juice
poartry	the	poetry
poetry	the	poetry
unexspected	and	unexpected
varable	the	variable

Παρατηρούμε ότι ο ορθογράφος έχει χάσει τη λειτουργικότητά του, καθώς σχεδόν όλες οι λέξεις μετατρέπονται σε "the" και "and". Οι συγκεκριμένες λέξεις παρουσιάζονται με πολύ μεγαλύτερη συχνότητα και επιλέγονται έναντι όλων των υπόλοιπων μονοπατιών.

Συγκρίνοντας τα αποτελέσματα των LVW και LV για τα tokens "cwt" και "cit", παρατηρούμε τις εξής εξόδους:

LVW: cwt → the cit → it

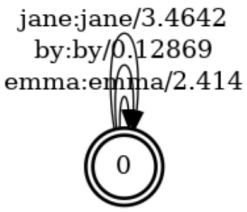
LV: cwt → cut cit → city

Όπως φαίνεται, ο ορθογράφος LVW εμφανίζει μεροληψία προς λέξεις που εμφανίζονται συχνότερα, ενώ ο ορθογράφος LV προσπαθεί να φτάσει σε μία αποδεκτή λέξη ελαχιστοποιώντας τον αριθμό των edits.

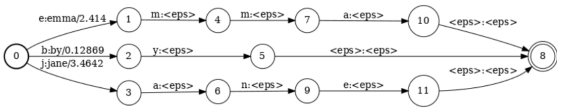
1.9.4 Συγκρίνοντας τα αποτελέσματα των *LVW* και *LV* για τα tokens `"cwt"` και `"cit"`

- *LVW*:
  - `cwt` → `the`
  - `cit` → `it`
- *LV*:
  - `cwt` → `cut`
  - `cit` → `city`

Όπως φαίνεται, ο ορθογράφος *LVW* εμφανίζει μεροληψία προς λέξεις που εμφανίζονται συχνότερα, ενώ ο ορθογράφος *LV* προσπαθεί να φτάσει σε μία αποδεκτή λέξη ελαχιστοποιώντας τον αριθμό των *edits*.



Σχήμα 6



Σχήμα 7

1.10 Βήμα 10: Αξιολόγηση των Ορθογράφων

Εκτελούμε το script `run_evaluation.py` για να αξιολογήσουμε τους τέσσερις ορθογράφους: `LV`, `LVW`, `EV`, και `EVW`. Το συγκεκριμένο script χρησιμοποιεί για τη σύγκριση όλο το εύρος του αρχείου `spell_test.txt`.

1.10.1 Αποτελέσματα Αξιολόγησης

Τα αποτελέσματα ως προς την ακρίβεια των ορθογράφων είναι τα εξής:

- **LV accuracy**: 59.629%
- **LVW accuracy**: 4.444%
- **EV accuracy**: 68.888%
- **EVW accuracy**: 64.074%

1.10.2 Ανάλυση Αποτελεσμάτων

Παρατηρούμε ότι η προσθήκη του W και η μεροληψία που δημιουργεί προς συχνότερα εμφανιζόμενες λέξεις προκαλεί πτώση στην ακρίβεια των ορθογράφων σε κάθε περίπτωση. Ωστόσο, υπάρχει διαφοροποίηση στις επιπτώσεις ανάλογα με τη μέθοδο:

- **LVW**: Ο ορθογράφος χάνει πρακτικά τη λειτουργικότητά του, καθώς οι λέξεις με μεγαλύτερη συχνότητα προτιμώνται σε πολύ μεγαλύτερο βαθμό, οδηγώντας σε δραματική πτώση της ακρίβειας.
- **EVW**: Οι επιπτώσεις του W αντισταθμίζονται από την καλύτερη στάθμιση των βαρών για κάθε *edit*, με αποτέλεσμα ο ορθογράφος να διατηρεί σε μεγάλο βαθμό την ακρίβειά του.

1.10.3 Συμπέρασμα

Η αξιολόγηση αποδεικνύει ότι η απλή εισαγωγή της συχνότητας εμφάνισης λέξεων μέσω του W δεν βελτιώνει την απόδοση του ορθογράφου και μπορεί να προκαλέσει σημαντική πτώση στην ακρίβεια, όπως φάνηκε στην περίπτωση του LVW. Αντίθετα, η μέθοδος EVW, που συνδυάζει τη στάθμιση των βαρών των *edits* με τη συχνότητα εμφάνισης των λέξεων, διατηρεί καλύτερη ισορροπία και αποδεικνύεται πιο αποτελεσματική.

1.11 Βήμα 11: (Bonus) Βελτιώσεις του Ορθογράφου

Σε αυτό το βήμα, υλοποιούμε το Add-1 (Laplacian) smoothing για τη βελτίωση του ορθογράφου. Προσθέτουμε 1 στον αριθμητή κατά τον υπολογισμό της πιθανότητας ενός *edit* και N στον παρονομαστή, όπου N είναι η πληθικότητα των *edits*.

1.11.1 Λειτουργία του Add-1 Smoothing

Η χρήση του Add-1 smoothing εξομαλύνει τις πιθανότητες, αποδίδοντας μία μικρή πιθανότητα στα *edits* που δεν έχουν παρατηρηθεί. Αυτό αποτρέπει τη χρήση της σταθεράς `INFINITY` κατά την εφαρμογή των βαρών και αυξάνει την ακρίβεια του ορθογράφου σε σπάνιες περιπτώσεις.

1.11.2 Δημιουργία Νέων Transducers

Δημιουργούμε τους νέους transducers E και EV, εφαρμόζοντας το Add-1 smoothing. Εκτελούμε το script `run_evaluation` για να αξιολογήσουμε την απόδοση του ορθογράφου με τα νέα δεδομένα.

1.11.3 Αποτελέσματα

Με την εφαρμογή του Add-1 smoothing, η ακρίβεια του ορθογράφου αυξάνεται στο 69.26%.

1.11.4 Συμπέρασμα

Η εφαρμογή του Add-1 smoothing βελτιώνει την απόδοση του ορθογράφου, εξομαλύνοντας τις πιθανότητες και αποδίδοντας μικρές πιθανότητες σε μη παρατηρηθέντα *edits*. Αυτό έχει ως αποτέλεσμα την αύξηση της ακρίβειας του ορθογράφου μας.

1.11.5 Περαιτέρω Δοκιμές και Βελτιστοποίηση του Ορθογράφου

Για περαιτέρω δοκιμές και βελτιστοποίηση των ορθογράφων, δοκιμάσαμε την κατασκευή του V με επεκταμένο *corpus*. Συγκεκριμένα, δημιουργήσαμε ένα script `fetch_url`, αντίστοιχο με το `fetch_gutenberg`, το οποίο κατεβάζει το λεξιλόγιο δοθέντων URLs και το υποβάλλει σε προεπεξεργασία. Στη συνέχεια, χρησιμοποιήσαμε την Python και την εντολή `sed` για την αφαίρεση μη αγγλικών χαρακτήρων, τη συνένωση των αρχείων με το `words.syms` και την παραγωγή νέων αρχείων συμβόλων.

1.11.6 Χρησιμοποιούμενα Corpora

Τα *corpora* που χρησιμοποιήσαμε προέρχονται από τις εξής πηγές:

- 40k πιο συχνά χρησιμοποιούμενες λέξεις της αγγλικής γλώσσας
- 50k πιο συχνά εμφανιζόμενες λέξεις σε αγγλικούς υπότιτλους το 2016
- 500k πιο συχνά εμφανιζόμενες λέξεις σε αγγλικούς υπότιτλους το 2016

Το πρώτο *corpus* περιέχει τις 40 χιλιάδες πιο συχνά χρησιμοποιούμενες λέξεις της αγγλικής γλώσσας, ενώ τα επόμενα δύο περιλαμβάνουν τις 50 χιλιάδες και 500 χιλιάδες πιο συχνά εμφανιζόμενες λέξεις (ή tokens) σε αγγλικούς υπότιτλους το έτος 2016. Για το τελευταίο *corpus*, έγινε μείωση στις πρώτες 250 χιλιάδες λέξεις, δηλαδή αυτές με συχνότητα μεγαλύτερη του 5. Παρόλο που το πρώτο *corpus* είναι μικρότερο, θεωρείται πιο ποιοτικό, καθώς τα επόμενα δύο περιέχουν tokens που δεν συνιστούν πραγματικές λέξεις (π.χ., επιφωνήματα όπως `"argh"`, `"aah"`). Συνεπώς, κατασκευάζουμε τρεις μεταδρομείς V και συνθέτουμε τρεις ορθογράφους EV (με Add-1 smoothing), οι οποίοι χρησιμοποιούν ως *corpus* το Gutenberg, εμπλουτισμένο με κάποιο από τα παραπάνω σύνολα δεδομένων. Εκτελώντας τα scripts αξιολόγησης, λαμβάνουμε τα εξής αποτελέσματα:

- **EV-40k πιο συχνές λέξεις**: Ακρίβεια 80%
- **EV-50k πιο συχνές λέξεις σε υπότιτλους**: Ακρίβεια 68.88%
- **EV-250k πιο συχνές λέξεις**: Ακρίβεια 47.4%

Από τα παραπάνω δεδομένα καθίσταται προφανής η σημασία της ποιότητας του *corpus*. Παρατηρούμε ότι το πρώτο εκτεταμένο λεξικό οδήγησε σε σημαντική αύξηση της επίδοσης. Αντιθέτως, τα επόμενα δύο, παρόλο που είχαν μεγαλύτερο μέγεθος, οδήγησαν σε πτώση της ακρίβειας, καθώς περιέχουν περιττά *tokens* που διαβρώνουν το λεξικό, οδηγώντας στην αποδοχή λανθασμένων διορθώσεων. Η πτώση αυτή είναι πιο έντονη στο λεξικό των 250 χιλιάδων λέξεων, λόγω της περιεκτικότητάς του σε ακόμα περισσότερη άχρηστη πληροφορία. Είναι επίσης σημαντικό να σημειωθεί ότι η χρήση αυτών των λεξικών επιφέρει σημαντική αύξηση και στους χρόνους εκτέλεσης του προγράμματος.

1.11.7 Πιθανές Βελτιώσεις για τον Ορθογράφο

Πιθανές βελτιώσεις για τον ορθογράφο περιλαμβάνουν:

1. **Επέκταση με Corpora από Διάφορους Τομείς**:

- Χρήση λεξικών από διαφορετικούς τομείς για την ανίχνευση λέξεων που μπορεί να μην εμφανίζονται συχνά σε γενικά *corpora*.

2. **Προσαρμογή ανάλογα με το Context**:
   - Αν και η προσαρμογή ανάλογα με το *context* μπορεί να προκαλέσει *overfitting* σε γενικές περιπτώσεις, η προσεκτική εφαρμογή της μπορεί να βελτιώσει την ακρίβεια σε εξειδικευμένα κείμενα.
3. **Εκμετάλλευση των n-grams**:
   - Χρήση n-grams και καλύτερων τεχνικών smoothing για τη βελτίωση της πρόβλεψης λέξεων και τη μείωση της πιθανότητας λανθασμένων διορθώσεων.
4. **Ενσωμάτωση Μοντέλων Μηχανικής Μάθησης**:
   - Χρήση μοντέλων μηχανικής μάθησης για τον καλύτερο υπολογισμό των βαρών των *edits*, προσφέροντας μεγαλύτερη ακρίβεια και ευελιξία.
5. **Χρήση Κατάλληλων Δομών Δεδομένων**:
   - Εφαρμογή δομών δεδομένων όπως *prefix trees* για την αποθήκευση του λεξικού και των παραγόμενων αρχείων, μειώνοντας την πολυπλοκότητα κατά τη σύνθεση του ορθογράφου.

2 Μέρος 2: Εξοικείωση με το Word2Vec (W2V)

Η υλοποίηση του μέρους 2 βρίσκεται στο script `part2.py`, το οποίο, όπως και το αρχείο `part1.py`, χρησιμοποιεί προγράμματα που βρίσκονται στον φάκελο `scripts`. Στόχος του δεύτερου μέρους είναι η εξοικείωση με τα μοντέλα Word2Vec, μέσω των εξής δραστηριοτήτων:

1. **Κατασκευή ενός Μοντέλου Word2Vec**:
   - Εκπαίδευση ενός μοντέλου Word2Vec από την αρχή χρησιμοποιώντας ένα δεδομένο *corpus*.
2. **Σύγκριση με Προεκπαιδευμένο Μοντέλο**:
   - Σύγκριση του κατασκευασμένου μοντέλου με ένα άλλο προεκπαιδευμένο μοντέλο για να αξιολογηθεί η απόδοσή του.
3. **Χρήση για Ανίχνευση Συναισθήματος**:
   - Εφαρμογή του μοντέλου Word2Vec για την ανίχνευση συναισθήματος σε κείμενα.

2.1 Βήμα 12: Εξαγωγή Αναπαραστάσεων Word2Vec

Για το συγκεκριμένο βήμα της άσκησης εργαζόμαστε με το *Gutenberg corpus*. Για τη διευκόλυνσή μας κατά την εκπαίδευση των μοντέλων, χρησιμοποιούμε τις συναρτήσεις `sent_tokenize` και `word_tokenize` του πακέτου `nltk`. Στη συνέχεια, εκπαιδεύουμε τέσσερα εκατοδιάστατα Word2Vec *embeddings* με παράθυρο 5 λέξεων για 1000, 2000, 100 και 10 εποχές αντίστοιχα, και ένα με παράθυρο 10 λέξεων για 100 εποχές. Αυτό γίνεται για να ελέγξουμε την επιρροή του μεγέθους των εποχών και του παραθύρου στα επόμενα βήματα.

Αρχικά, χρησιμοποιούμε τη συνάρτηση `most_similar`, η οποία βασίζεται στην ομοιότητα συνημιτόνου, για να βρούμε τις σημασιολογικά κοντινότερες λέξεις για τις λέξεις "bible", "book", "bank" και "water". Τα αποτελέσματα που λαμβάνουμε είναι τα εξής:

1000 Epoch Model

```
Similar words for: bible [('poetry', 0.383050799369812), ('blasphemous',
0.3548435866832733), ('reverend', 0.3501461744308471 7), ('preacher',
0.3461860716342926), ('propitious', 0.344009667634964)]
Similar words for: book [('chronicles', 0.5576196908950806), ('written', 0.553611159324646),
('letter', 0.475374698638916), ('volume', 0.45197001099586487), ('note',
0.4486658573150635)]
Similar words for: bank [('pool', 0.5286288857460022), ('ridge', 0.5067192316055298), ('top',
0.48990094661712646), ('table', 0.46766746044158936), ('hill', 0.46434348821640015)]
Similar words for: water [('waters', 0.6529315710067749), ('wine', 0.5079317092895508),
('liquid', 0.5070357918739319), ('lake', 0.5038613080978394), ('river', 0.5010083913803101)]
```

2000 Epoch Model

```
Similar words for: bible [('scarf', 0.3493695855140686), ('blasphemous',
0.3418530225753784), ('copyright', 0.3370893597602844), ('republic',
0.3366149961948395), ('poetry', 0.3350933790206909)]
Similar words for: book [('chronicles', 0.5492503643035889), ('written',
0.5475344657897949), ('temple', 0.4714668393135071), ('papers', 0.45232826471328735),
('letter', 0.4420188367366791)]
Similar words for: bank [('pool', 0.5393407344818115), ('hill', 0.5045983791351318), ('ridge',
0.48803168535232544), ('table', 0.47779619693756104), ('rocks', 0.4545815587043762)]
Similar words for: water [('waters', 0.6228045225143433), ('lake', 0.4930165410041809),
('river', 0.49141889810562134), ('sea', 0.48895180225372314), ('streams',
0.48705822229385376)]
```

100 Epoch Model

```
Similar words for: bible [('matthew', 0.40671759843826294), ('republic',
0.40476125478744507), ('lies', 0.4022599458694458), ('title', 0.3794362545013428), ('pliny',
0.35957080125808716)]
Similar words for: book [('chronicles', 0.6147681474685669), ('written',
0.6054839491844177), ('volume', 0.5395889282226562), ('letter', 0.5110397338867188),
('papers', 0.4572550654411316)]
Similar words for: bank [('ridge', 0.5609272718429565), ('pool', 0.5606678128242493),
('platform', 0.5254971385002136), ('lawn', 0.5201282501220703), ('hill',
0.5196443796157837)]
Similar words for: water [('waters', 0.6477859616279602), ('streams', 0.5492191314697266),
('river', 0.5402520298957825), ('camels', 0.5272562503814697), ('sea',
0.5269639492034912)]
```

100 Epoch Model, Larger Window

```
Similar words for: bible [('stands', 0.3699774742126465), ('matthew', 0.36912718415260315),
('sets', 0.3682869076728821), ('republic', 0.35443124175071716), ('discernment',
0.3514211177825928)]
Similar words for: book [('written', 0.6045722961425781), ('chronicles',
0.5423722863197327), ('volume', 0.5365967750549316), ('temple', 0.48301851749420166),
('letter', 0.4672126770019531)]
Similar words for: bank [('pool', 0.5196660757064819), ('table', 0.5143560171127319),
('beach', 0.5127063989639282), ('footpath', 0.4977200925350189), ('side',
0.4851554334163666)]
Similar words for: water [('waters', 0.6160950660705566), ('lake', 0.5543476939201355),
```

```
('streams', 0.5165333151817322), ('tumbler', 0.4947237968444824), ('pitcher',
0.4894286096096039)]
```

10 Epoch Model

```
Similar words for: bible [('deliverer', 0.5361034870147705), ('republic', 0.5348396301269531),
('yojo', 0.532321572303772), ('camerado', 0.520039975643158), ('salem',
0.5165165066719055)]
Similar words for: book [('chronicles', 0.6864470839500427), ('written',
0.6520118713378906), ('letter', 0.5922365188598633), ('volume', 0.572303056716919),
('note', 0.5709046125411987)]
Similar words for: bank [('floor', 0.8180829286575317), ('lawn', 0.7880439758300781),
('beach', 0.7847914099693298), ('shelf', 0.7794530391693115), ('road',
0.7497352361679077)]
Similar words for: water [('ashes', 0.618657648563385), ('waters', 0.6139789819717407),
('wood', 0.5876113772392273), ('brook', 0.5782594680786133), ('bushes',
0.5739002227783203)]
```

Παρατηρούμε ότι στις περισσότερες περιπτώσεις τα αποτελέσματα είναι ικανοποιητικά, καθώς υπάρχει νοηματική συνάφεια με τις λέξεις που προκύπτουν ως έξοδος. Τα αποτελέσματα βελτιώνονται καθώς αυξάνουμε τις εποχές από 10 στις 1000. Για τις 2000 εποχές, δεν παρατηρείται ιδιαίτερη βελτίωση και, στην περίπτωση της λέξης "bible", προκύπτουν λέξεις που δεν φαίνεται να έχουν μεγάλη συγγένεια. Επιπλέον, οι αλλαγές στο μέγεθος του παραθύρου δεν προκαλούν κάποια σημαντική διαφορά στα αποτελέσματα για αυτήν την περίπτωση. Η βέλτιστη επιλογή του αριθμού των εποχών και του παραθύρου εξαρτάται από την εφαρμογή. Ο μεγαλύτερος αριθμός εποχών τείνει να βελτιώνει τα αποτελέσματα, αλλά πολύ μεγάλες τιμές μπορεί να προκαλέσουν *overfitting*. Το μέγεθος του παραθύρου επιλέγεται ανάλογα με τη σημασία που πρέπει να δοθεί στο επιμέρους *context*. Επιπλέον, μπορούμε να πειραματιστούμε με άλλες υπερπαραμέτρους, όπως το *learning rate*, για τη βελτιστοποίηση του μοντέλου. Μία στρατηγική που μπορεί να βελτιώσει σημαντικά τα αποτελέσματα είναι η αύξηση των δεδομένων.

Ακολουθώντας την ίδια μεθοδολογία, για το μοντέλο των 1000 εποχών, υπολογίζουμε τις πιο κοντινές σημασιολογικά λέξεις για τα αποτελέσματα των πράξεων μεταξύ των διανυσμάτων.

1000 Epoch Model

Word trio: (girls, queen, kings)

Most similar words for *girls* - *queen* + *kings*:

```
[('parts', 0.48722755908966064), ('men', 0.48240259289741516), ('cities', 0.4541364908218384),
('creatures', 0.44944313168525696), ('nations', 0.443217933177948)]
```

Word trio: (good, tall, taller)

Most similar words for *good* - *tall* + *taller*:

```
[('dear', 0.4083581566810608), ('used', 0.39686906337738037), ('merry', 0.38100019097328186),
('better', 0.3686302900314331), ('please', 0.3660450875759125)]
```

Word trio: (france, paris, london)

Most similar words for *france* - *paris* + *london*:

```
[('england', 0.41968631744384766), ('town', 0.40732717514038086), ('court', 0.4003077745437622),
('sources', 0.3972579538822174), ('hotel', 0.3879925310611725)]
```

Παρατηρούμε ότι το μοντέλο μας αδυνατεί στην πρώτη περίπτωση να επιλέξει τη λέξη "boys", η οποία διαισθητικά φαίνεται να είναι η πιο ταιριαστή λόγω συμφραζομένων. Αντίθετα, παραθέτει τη λέξη "men", η οποία είναι αρκετά κοντά εννοιολογικά. Στις δύο επόμενες περιπτώσεις, παρουσιάζονται οι λέξεις που φαίνεται ότι θα ταίριαζαν, ωστόσο οι υπόλοιπες έξοδοι μπορούν να γίνουν πιο στοχευμένες με τη χρήση ενός βελτιωμένου μοντέλου.

Φορτώνοντας το μοντέλο της Google με `limit = 300,000`, επαναλαμβάνουμε τα ίδια βήματα για να αξιολογήσουμε τη βελτίωση στα αποτελέσματα.

Google Model

Similar words for: bible

```
[('Bible', 0.7367782592773438), ('bibles', 0.6052597761154175), ('Holy Bible', 0.5989601612091064),
('scriptures', 0.5745684504508972), ('scripture', 0.5697901844978333)]
```

Similar words for: book

```
[('tome', 0.7485830783843994), ('books', 0.7379177808761597), ('memoir', 0.730292797088623),
('paperback_edition', 0.6868364810943604), ('autobiography', 0.6741527318954468)]
```

Similar words for: bank

```
[('banks', 0.7440758943557739), ('banking', 0.690161406993866), ('Bank', 0.6698697805404663),
('lender', 0.6342284679412842), ('banker', 0.6092953681945801)]
```

Similar words for: water

```
[('potable water', 0.6799106597900391), ('Water', 0.6706870794296265), ('sewage',
    ↪ 0.6619377136230469),
('groundwater', 0.6588345766067505), ('freshwater', 0.6307883262634277)]
```

Word trio: (girls, queen, kings)

Most similar words for *girls* - *queen* + *kings*:

```
[('boys', 0.7183727622032166), ('men', 0.5000520348548889), ('teenagers', 0.4967767894268036),
('schoolboys', 0.46822962164878845), ('kids', 0.45648258924484253)]
```

Word trio: (good, tall, taller)

Most similar words for *good* - *tall* + *taller*:

```
[('better', 0.7245720624923706), ('nicer', 0.5474838018417358), ('great', 0.5431875586509705),
('quicker', 0.5431677103042603), ('bad', 0.5423746705055237)]
```

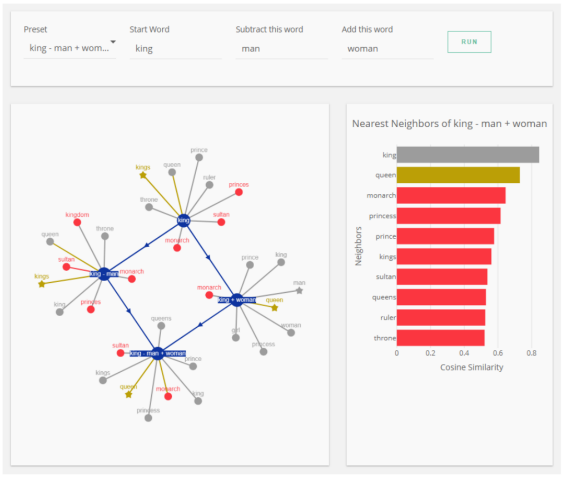
Word trio: (france, paris, london)

Most similar words for *france* - *paris* + *london*:

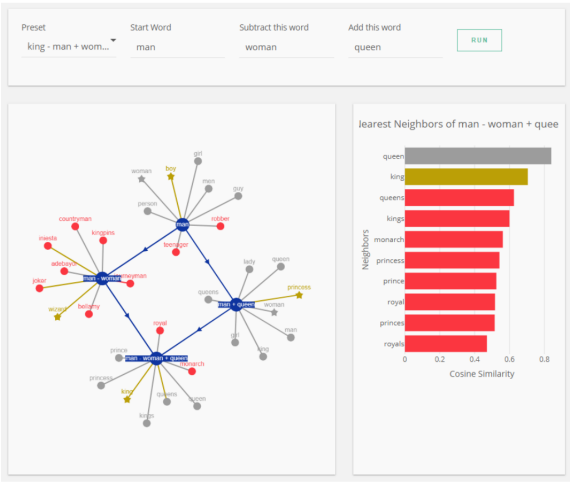
```
[('england', 0.5836853384971619), ('europe', 0.5529575347900391), ('european', 0.5125302076339722),
('africa', 0.4936657249927521), ('spain', 0.48787450790405273)]
```

Παρατηρούμε ότι τα αποτελέσματα παρουσιάζουν βελτίωση και στις δύο περιπτώσεις. Αυτό είναι λογικό, καθώς το συγκεκριμένο μοντέλο αποτελείται από διανύσματα 300 διαστάσεων και περιλαμβάνει ένα *corpus* με περισσότερες από 3.000.000 λέξεις, συγκριτικά με το *corpus* των περίπου 15 χιλιάδων λέξεων που χρησιμοποιούμε.

2.2 Βήμα 13: Οπτικοποίηση των Word Embeddings



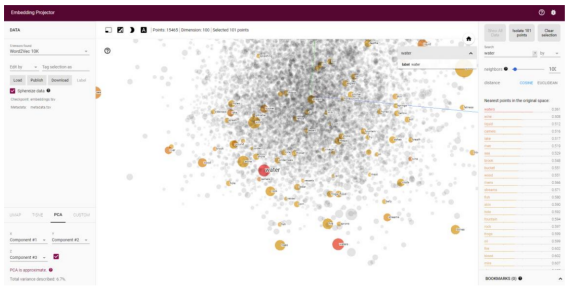
Σχήμα 8: king – man + woman



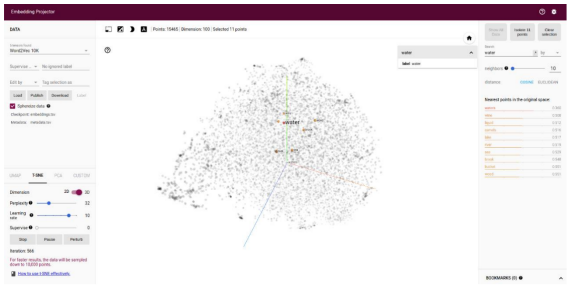
Σχήμα 9: man – woman + queen

Είναι αξιοσημείωτο ότι επιτελείται μια αντίστοιχη λειτουργία με αυτή που υλοποιεί το μοντέλο μας, επιτρέποντάς μας να οπτικοποιήσουμε τη διαδικασία και τη διάταξη των πλησιέστερων γειτόνων. Μέσω της αποθήκευσης των *embeddings* και της μεταφόρτωσής τους στο *embedding projector*, αποκτούμε μια βελτιωμένη κατανόηση της διάρθρωσης των λέξεων στον πολυδιάστατο χώρο, καθώς και των

μεθόδων μείωσης διαστατικότητας που διευκολύνουν την οπτικοποίηση. Ως αποτέλεσμα, για τη λέξη "water" προκύπτει η εξής απεικόνιση:



Σχήμα 10: Enter Caption

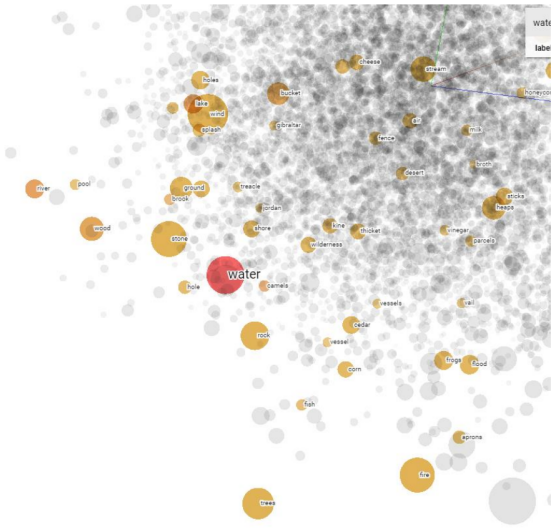


Σχήμα 11: Enter Caption

Η πρώτη εικόνα παρουσιάζει τον πολυδιάστατο χώρο μετά την εφαρμογή της μεθόδου PCA, ενώ η δεύτερη εικόνα απεικονίζει τον χώρο μετά την εφαρμογή της μεθόδου T-SNE. Το PCA, γνωστό ως ανάλυση κύριων συνιστωσών, είναι μια γραμμική μέθοδος που επιτυγχάνει μια αναπαράσταση χαμηλότερων διαστάσεων των δεδομένων, προβάλλοντάς τα σε ένα νέο σύνολο αξόνων οι οποίοι αποτυπώνουν τη μεγαλύτερη διακύμανση των δεδομένων. Αυτοί οι νέοι άξονες, γνωστοί ως κύριες συνιστώσες, είναι ορθογώνιοι μεταξύ τους και ταξινομούνται βάσει της ποσότητας της διακύμανσης που εξηγούν.

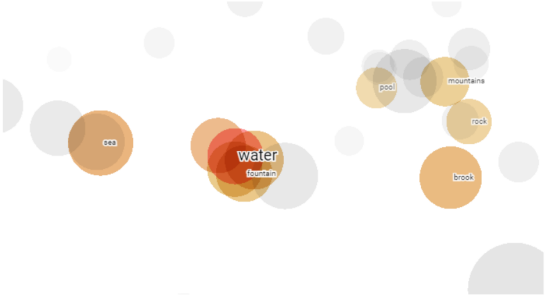
Αντίθετα, το t-SNE (t-distributed Stochastic Neighbor Embedding) είναι μια μη γραμμική τεχνική που διατηρεί τις τοπικές ομοιότητες μεταξύ των σημείων δεδομένων σε ένα χώρο χαμηλών διαστάσεων. Κατά την απεικόνιση των σημείων σε έναν δισδιάστατο ή τρισδιάστατο χώρο, το t-SNE διατηρεί τις σχετικές αποστάσεις μεταξύ των σημείων. Αυτή η τεχνική είναι ιδιαίτερα χρήσιμη για την οπτικοποίηση δεδομένων υψηλών διαστάσεων που παρουσιάζουν πολύπλοκες δομές ή συμπλέγματα, καθώς μπορεί να διατηρήσει με μεγαλύτερη ακρίβεια τη δομή και την τοπολογία των δεδομένων.

Τα ανωτέρω ευρήματα επιβεβαιώνονται και μέσω της επισκόπησης των διαμορφούμενων χώρων. Συγκεκριμένα, παρατηρείται ότι για τη λέξη "water", οι πλησιέστερες λέξεις στον αρχικό χώρο ήταν "wine", "liquid", "camels", "lake", "river", "sea" και "brook". Με τη χρήση της μεθόδου PCA, διαπιστώνεται ότι πολλές από τις πιο κοντινές σημασιολογικά λέξεις στον αρχικό χώρο (οι οποίες απεικονίζονται με κίτρινο χρώμα) έχουν απομακρυνθεί, ενώ λέξεις όπως "wind", "wood" και "flames" εμφανίζονται πλησιέστερα στη λέξη "water". Αυτό φαίνεται καθαρότερα στην παρακάτω απεικόνιση:



Σχήμα 12: Enter Caption

Αντίθετα, με τη χρήση της μεθόδου t-SNE, οι λέξεις που είχαν μικρή απόσταση στον αρχικό χώρο διατηρούν τις χαμηλές αποστάσεις τους. Έτσι, η τοπολογία των δεδομένων παραμένει πιο συνεκτική και αντιπροσωπευτική της αρχικής διάρθρωσης.



Σχήμα 13: Enter Caption

Η επιλογή της κατάλληλης μεθόδου εξαρτάται από την εκάστοτε εφαρμογή, καθώς κάθε μέθοδος διαθέτει συγκεκριμένα πλεονεκτήματα και μειονεκτήματα. Για παράδειγμα, στην επεξεργασία φυσικής γλώσσας, η χρήση του t-SNE κατά την εκπαίδευση ενός μοντέλου μπορεί να αποδώσει καλύτερα αποτελέσματα, καθώς επιδιώκουμε να διατηρήσουμε τις σημασιολογικές συνδέσεις μεταξύ των λέξεων. Αντίθετα, κατά την εκπαίδευση ενός διαφορετικού μοντέλου με ένα σύνολο δεδομένων που περιλαμβάνει πολλαπλά χαρακτηριστικά διαφορετικού τύπου, η μείωση διαστατικότητας και η διατήρηση μεγάλου ποσοστού της διακύμανσης μέσω PCA μπορεί να προσφέρει καλύτερες επιδόσεις.

2.3 Βήμα 14: Ανάλυση Συναισθήματος με Word2Vec Embeddings

Στο τελευταίο βήμα της άσκησης, χρησιμοποιούμε τα δεδομένα που παρέχονται και τα μοντέλα που δημιουργήθηκαν στα προηγούμενα ερωτήματα για την εκπαίδευση ενός ταξινομητή συναισθήματος με

Logistic Regression, ο οποίος θα είναι ικανός να διαχωρίζει θετικές από αρνητικές κριτικές του IMDb.

Αρχικά, χρησιμοποιώντας τη συνάρτηση `read_sample`, πραγματοποιούμε την κατάλληλη προεπεξεργασία και διαβάζουμε τα δεδομένα. Στη συνέχεια, με τη συνάρτηση `create_corpus`, κατασκευάζουμε τα *training* και *test sets*.

Με την τεχνική *Neural Bag of Words (NBoW)*, δημιουργούμε αναπαραστάσεις για κάθε σχόλιο. Αρχικά, κάνουμε *tokenize* κάθε σχόλιο και το μετατρέπουμε σε μια ακολουθία *word embeddings*, χρησιμοποιώντας τα μοντέλα *word embeddings* που έχουμε εκπαιδεύσει. Στη συνέχεια, υπολογίζουμε τον μέσο όρο των *word embeddings* σε όλη την ακολουθία, για να λάβουμε μια διανυσματική αναπαράσταση σταθερού μήκους για κάθε σχόλιο.

Έχοντας κατασκευάσει τις αναπαραστάσεις για κάθε σχόλιο, μπορούμε να προχωρήσουμε στην εκπαίδευση ενός μοντέλου *Logistic Regression* για την ταξινόμηση των σχολίων. Τέλος, αξιολογούμε την απόδοση του μοντέλου στο *test set* μας, ώστε να διαπιστώσουμε την αποτελεσματικότητά του στη διάκριση θετικών και αρνητικών κριτικών. Ακολουθώντας την παραπάνω διαδικασία, προκύπτουν τα εξής αποτελέσματα:

- **Accuracy: 74.4%** για το μοντέλο που εκπαιδεύσαμε για 1000 εποχές.
- **Accuracy: 84.1%** για το μοντέλο της Google.

Η αυξημένη επίδοση του μοντέλου της Google είναι αναμενόμενη, καθώς, όπως αναφέραμε προηγουμένως, πρόκειται για ένα μοντέλο μεγαλύτερων διαστάσεων που έχει εκπαιδευτεί σε πολύ μεγαλύτερο εύρος λέξεων. Συνεπώς, για τη βελτίωση των επιδόσεων, είναι σημαντική η εκπαίδευση των *word embeddings* σε μεγαλύτερα *corpora* και η χρήση μεγαλύτερου μεγέθους διανυσμάτων.

Επιπλέον, οι εξής βελτιώσεις μπορούν να συμβάλουν στην αύξηση της απόδοσης:

- Αύξηση του όγκου των δεδομένων εκπαίδευσης.
- Εισαγωγή *bias*.
- Εκπαίδευση των *embeddings* σε συγκεκριμένα είδη κειμένων ανάλογα με την εφαρμογή.
- Χρήση πιο πολύπλοκων μοντέλων ταξινομητών, όπως *Support Vector Machines (SVMs)*.
- Κατάλληλη επιλογή των παραμέτρων των μοντέλων ταξινομητών, όπως *kernel*, *alpha* και *learning rate* για νευρωνικά δίκτυα.

Με αυτές τις στρατηγικές, μπορούμε να βελτιώσουμε τις επιδόσεις του μοντέλου μας και να πετύχουμε πιο ακριβή και αξιόπιστα αποτελέσματα στην ανάλυση συναισθήματος.