ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ
ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

Ακαδημαϊκό έτος 2022-2023

Επεξεργασία Φωνής και Φυσικής Γλώσσας

Αναφορά της 1ης εργαστηριακής άσκησης

Γκοτζιάς Γεώργιος – 03119047
Δαμιανός Δημήτριος - 03119825

Προπαρασκευή:

Βήμα 1:

Για τις ανάγκες της εργασίας, κληθήκαμε να κατεβάσουμε το gutenberg corpus της βιβλιοθήκης NLTK.
Κατά την διαδικασία του preprocessing, κάνουμε τις εξής αλλαγές στα αρχεία που μας δόθηκαν:
α) Μετατρέπουμε τους κεφαλαίους χαρακτήρες σε πεζούς
β) Αναλύουμε συντομεύσεις (π.χ. don’t γίνεται do not)
γ) Αφαιρούμε μεγάλα κενά
δ) Αγνοούμε σημεία στίξης
Με αυτό το “επιθετικό” preprocessing καταφέρνουμε να δημιουργήσουμε ένα απλοποιημένο και πιο εύχρηστο corpus.

Όμως υπάρχουν περιπτώσεις που θα θέλαμε να διατηρήσουμε κάποια από τα παραπάνω, π.χ. τα σημεία στίξης σε περίπτωση που προσπαθούμε να κατηγοριοποιήσουμε το συναίσθημα μιας πρότασης.

Βήμα 2-3:

Σε αυτό τα βήματα κληθήκαμε να κατασκευάσουμε λεξικό και αρχεία εισόδου/εξόδου για τα FSTs, βασισμένα στο corpus που αποκτήσαμε από το Βήμα 1.

Για να το πετύχουμε αυτό, δημιουργήσαμε το script create_dict_symbols.py. Στο script αυτό “σώζουμε” τις εμφανίσεις κάθε λέξης σε ένα dictionary, διαγράφουμε αυτές που εμφανίζονται λιγότερο από 5 φορές (ώστε να διατηρήσουμε ένα μικρότερο λεξικό με συχνότερα εμφανιζόμενες λέξεις και άρα με περισσότερη πληροφορία) και αποθηκεύουμε τα αποτελέσματα στο αρχείο words.vocab.txt .

Στην συνέχεια, αποθηκεύουμε τις λέξεις με αλφαβητική σειρά και αυξανόμενο index αρχείο words.syms (με πρώτη λέξη την κενή <epsilon>,index=0) και τα γράμματα του αγγλικού αλφαβήτου (μαζί με το <epsilon>) στο αρχείο chars.syms .

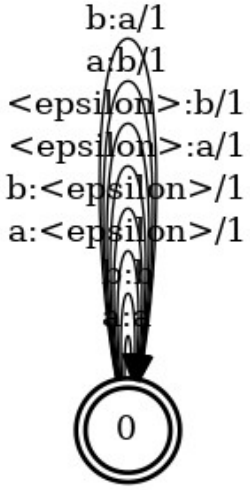
Βήμα 4:
Σε αυτό το βήμα καλούμαστε να υλοποιήσουμε το L transducer που βασίζεται στο Levenshtein Edit Distance, όπου γίνεται η παρακάτω αντιστοίχιση:
α) κάθε χαρακτήρας στον εαυτό του: κόστος 0 (no edit)
β) κάθε χαρακτήρας στο ε: κόστος 1 (deletion)
γ) ε σε κάθε χαρακτήρα: κόστος 1 (insertion)
δ) κάθε χαρακτήρας σε κάθε άλλο χαρακτήρα: κόστος 1

Αν εφαρμόσουμε τον L σε λέξη εισόδου και πάρουμε το shortest path, τότε προκύπτει η edited λέξη, όπου το edit έχει το ελάχιστο κόστος.

Κάποια άλλα edits που θα μπορούσαμε να έχουμε:

Αναφορικά με τα κόστη, θα μπορούσαμε να δώσουμε τιμές σύμφωνα με την πιθανότητα να συμβεί κάθε ένα από τα edits (π.χ. αν το cat εμφανίζεται περισσότερες από το at , τότε το βάρος χαρακτήρα σε χαρακτήρα από το deletion θα είναι μεγαλύτερο).

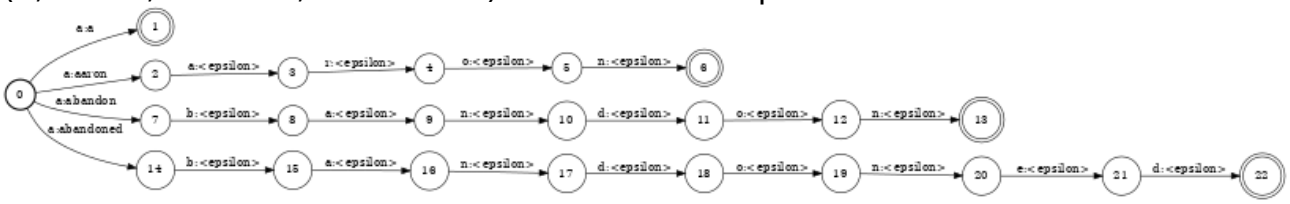
Για τις ανάγκες της οπτικοποίησης, δημιουργήσαμε το miniL.fst, που κάνει στην ίδια αντιστοίχιση στο αλφάβητο [α,β], που φαίνεται στη παρακάτω εικόνα.



Βήμα 5:

Σε αυτό το βήμα καλούμαστε να κατασκεύασουμε τον αποδοχέα V με μία αρχική κατάσταση, που αποδέχεται κάθε λέξη από το λεξικό που φτιάξαμε στο βήμα 2. Οι ακμές του V έχουν βάρος 0.

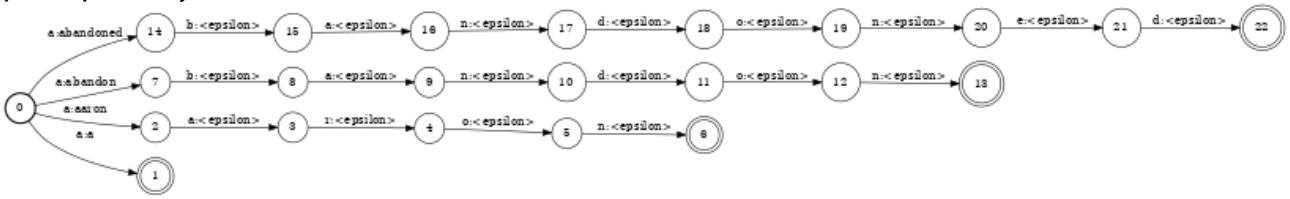
Για τις ανάγκες της οπτικοποίησης κατασκεύασαμε το miniV, που αποδέχεται 4 από τις λέξεις του λεξικού μας
(a, aaron,abandon,abandoned) και είναι το παρακάτω:



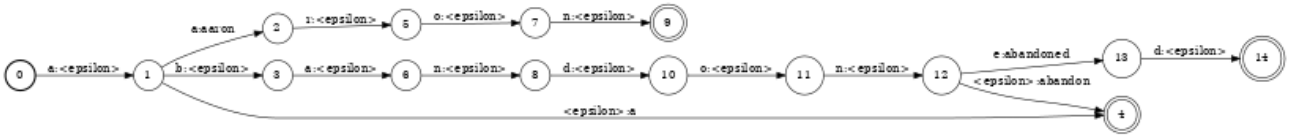
Παρατηρούμε ότι με την παρούσα μορφή του (κάθε λέξη έχει το δικό της μονοπάτι) το V θα είναι πολύ μεγάλο, συνεπώς η εύρεση του shortest path θα είναι πολύ χρονοβόρα.
Για αυτό το λόγο, μέσω των εντολών fstrmepsilon, fstdeterminize, fstminimize, το V “μικραίνει” σε μέγεθος.

Συγκεκριμένα η fstrmepsilon αφαιρεί ε-μεταβάσεις, η fstdeterminize κάνει ένα μη-ντετερμινιστικό FST ντετερμινικστικό, ενώ η fstminimize του μειώνει τον αριθμό των καταστάσεων ενός ντετερμινιστικού FST.

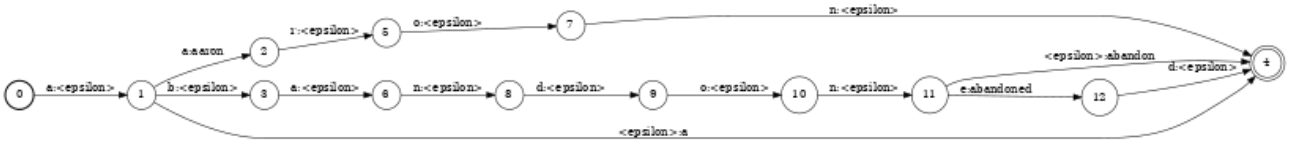
Παρακάτω βλέπουμε τα αποτελέσματα των παραπάνω εντολών στο miniV:
Μετά το fstrmepsilon (Δεν παρατηρείται κάποια αλλαγή, γιατί δεν είχαμε ε-μεταβάσεις):



Μετά το fstdeterminize:



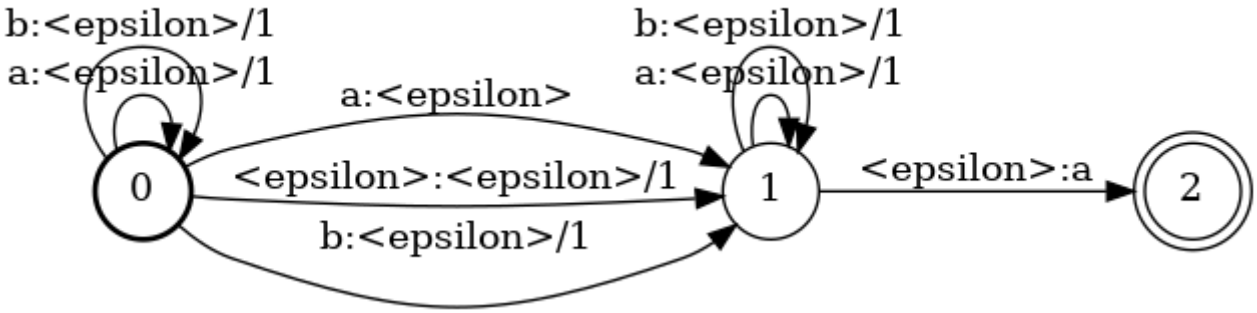
Μετά το fstminimize:



Παρατηρούμε ότι στο τέλος το miniV είναι αρκετά μικρότερο από το αρχικό, συνεπώς η εύρεση του shortest path υπολογιστικά πιο εύκολη.

Βήμα 6:

Σε αυτό το βήμα καλούμαστε να συνθέσουμε τον edit transducer L και acceptor V ώστε να δημιουργήσουμε τον spell checker S.
Μέσω της εντολής fstcompose κατασκευάζουμε τον S, όπου τα βάρη για κάθε spell correction προκύπτουν από τα βάρη του L.
Παρακάτω βλέπουμε την οπτικοποίηση του miniS, ο οποίος προκύπτει από την σύνθεση των miniL,miniV:

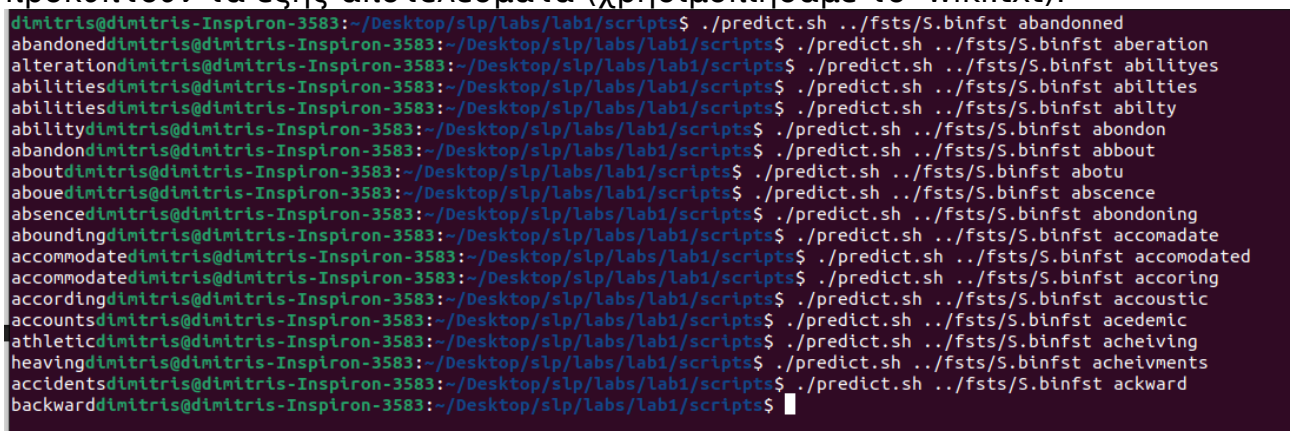


Αν τα βάρη ήταν ίσα μεταξύ τους (συνεπώς τα edits ήταν ισόβαρα) τότε η λέξη που θα επέστρεφε ο S θα ήταν “τυχαία”, αφού η επιλογή του shortest path δεν θα διέφερε με κανένα edit.
Αντίθετα, αν τα βάρη διέφεραν (όπως στην παρούσα μορφή του S) τότε η προκύπτουσα λέξη θα ήταν κυρίως αποτέλεσμα του edit με το μικρότερο βάρος (π.χ. όπως στην περίπτωση μας όπου το edit από χαρακτήρα στον εαυτό του έχει βάρος 0, τότε η λέξη της εξόδου έχει τις λιγότερο δυνατές αλλαγές για να την αποδεχτεί το V. Αν π.χ το deleterion είχε βάρος 0 και το insertion βάρος 1.5, τότε η λέξη θα είχε πολλά deletes και ελάχιστα insertions)

Για την παρούσα μορφή του S, για είσοδο cit μας δίνει city ή clit, ενώ για είσοδο cwt μας δίνει cut.

Βήμα 7:

Χρησιμοποιούμε τον spell checker που φτιάξαμε για τις 20 πρώτες λέξεις προκύπτουν τα εξής αποτελέσματα (χρησιμοπιήσαμε το wiki.txt).



Παρατηρούμε ότι ο spell checker S δίνει γενικά καλά αποτελέσματα, αλλά κάνει και προφανή λάθη (π.χ. το acedemic διορθώνεται σε athletic αντί για academic).

Η παραπάνω πρόβλεψη γίνεται με την βοήθεια του predict.sh,το οποίο εκτελεί το mkfstinput.py (που με την σειρά του δημιουργεί ένα fst που αποδέχεται την μια λέξη γράμμα-γράμμα) , και εκτυπώνει την λέξη που αποδέχεται ο S, μετά το shortest path.

Για την δημιουργία της διορθωμένης λέξης, βρίσκουμε το shortest path στο V, στο οποίο σε κάθε κόμβο/state το L επιβάλλει μια αλλαγή, ώστε η λέξη εισόδου τελικά γίνει αποδεκτή από το V.

Μέρος 1ο:

Βήμα 8:

Δημιουργήσαμε τον αποδοχέα M (που αποδέχεται την ανορθόγραφη λέξη) και τον αποδοχέα N (που αποδέχεται την σωστή λέξη).
Συνθέσαμε το MLN, οποίος βρίσκεις τι αλλαγές απαιτούνται για την ανορθογραφη ώστε να γίνει σωστή.
Συγκεκριμένα, τρέχοντας την εντολή που μας δόθηκε, προκύπτει:

```
10      9       a       a       0
0       0
1       0       d       d       0
2       1       e       e       0
3       2       n       n       0
4       3       n       <epsilon>       1
5       4       o       o       0
6       5       d       d       0
7       6       n       n       0
8       7       a       a       0
9       8       b       b       0
```

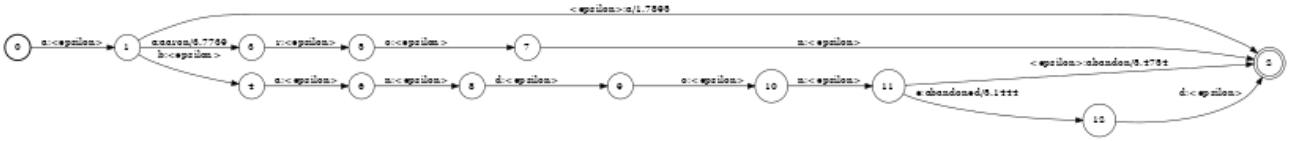
όπου βλέπουμε ότι αρκεί να αφαιρέσουμε ένα n (με κόστος 1) ώστε να διορθωθεί η λέξη.
Έτσι, με την βοήθεια του word_edits.sh, βρίσκουμε όλες τις αναγκαίες αλλαγές για το αρχείο wiki.txt, και δημιουργούμε τον transducer E.
Εξετάζοντας τον EV σε μερικά παραδείγματα έχουμε:

```
dimitris@dimitris-Inspiron-3583:~/Desktop/slp/labs/lab1/scripts$ ./predict.sh ../fsts/EV.binfst acedemic
demidimitris@dimitris-Inspiron-3583:~/Desktop/slp/labs/lab1/scripts$ ./predict.sh ../fsts/EV.binfst absolutly
absolutelydimitris@dimitris-Inspiron-3583:~/Desktop/slp/labs/lab1/scripts$ ./predict.sh ../fsts/EV.binfst absorbsion
absorbingdimitris@dimitris-Inspiron-3583:~/Desktop/slp/labs/lab1/scripts$ ./predict.sh ../fsts/EV.binfst absense
absencedimitris@dimitris-Inspiron-3583:~/Desktop/slp/labs/lab1/scripts$ ./predict.sh ../fsts/EV.binfst acident
accidentdimitris@dimitris-Inspiron-3583:~/Desktop/slp/labs/lab1/scripts$
```

Το EV, λόγω των διαφορετικών βαρών ανά μετάβαση, δίνει μεγάλη προτεραιότητα στα edits που βρέθηκαν στο wiki.txt, ενώ αποφεύγει τα υπόλοιπα λόγω του “άπειρου” κόστους τους.

Βήμα 9:

Με την βοήθεια του λεξιλογίου που φτιάξαμε στο βήμα 1, δημιουργήσαμε τον αποδέκτη λέξεων W, και στην συνέχεια τα EVW, LVW και VW.
Η οπτικοποίηση του VW είναι η παρακάτω (χρησιμοποίησαμε το miniV αντί για V):

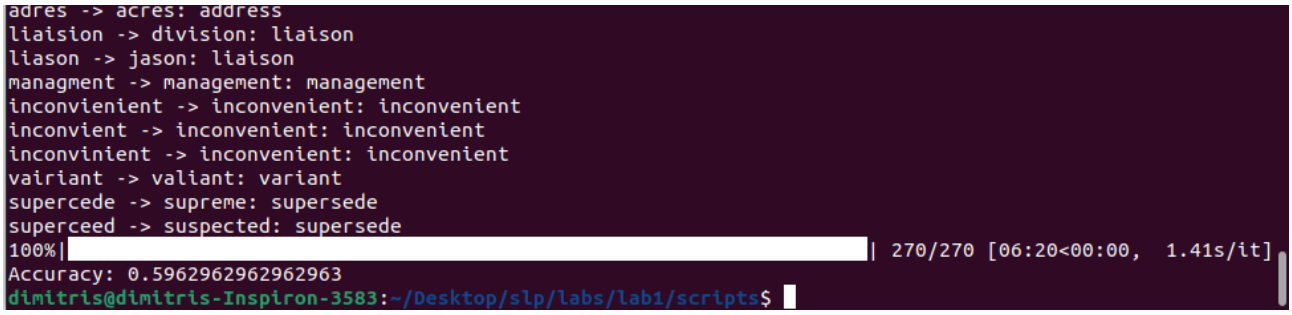


Το LVW, για είσοδο cit και cwt, μας δίνει it και στις 2 εξόδους, ενώ το LV μας δίνει cat, city. Ο λόγος αυτής της διαφοράς βρίσκεται στο W, το οποίο προσθέτει βάρος σε κάθε λέξη που αποδέχεται. Συνεπώς, αφού τα βάρη του W δεν είναι ίσα αλλά προκύπτουν από την συχνότητα της κάθε λέξης,θα υπάρχει bias ώστε το shortest path να μας δίνει λέξεις που εμφανίζονται συχνότερα, κάτι που δεν συμβαίνει με τον LV (δοκιμάσαμε το LVW για ίσα βάρη στο W, και είχε την ίδια συμπεριφορά με τον LV).

Βήμα 10:

Παρακάτω βλέπουμε τα αποτελέσματα των spell checkers LV, LVW, EV, EVW πάνω στο spell_test.txt.

LV:



LVW:

EV:

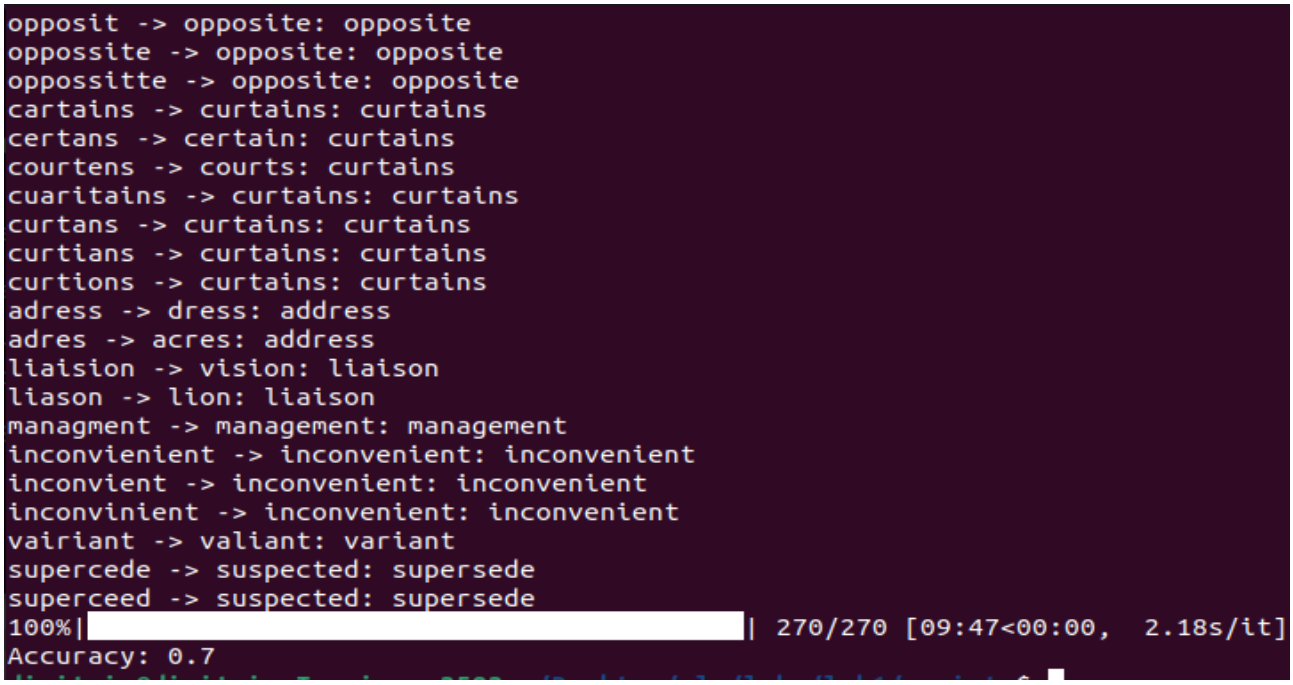
EVW:

Παρατηρούμε ότι τα EVW/EV έχουν καλύτερα αποτελέσματα από τα LVW/LV, καθώς εξαιτιάς του E έχουν την δυνατότητα να διορθώνουν λάθη που συμβαίνουν συχνά (όπως προκύπτει από το wiki.txt) και όχι να έχουν “τυχαίο” τρόπο διόρθωσης (όπως στο L, λόγω των ισοβαρών edits).
Παράλληλα, βλέπουμε ότι η προσθήκη του W μειώνει την επιτυχία, καθώς το W εισάγει μια μεροληψία στο αυτόματο να διορθώνει προς τις πιο συχνά εμφανιζόμενες λέξεις, αντί να διορθώνει καθαρά ορθογραφικά λάθη.

Βήμα 11:

Υλοποιήσαμε τον transducer E_laplace, ο οποίος δεν αποδίδει μηδενική πιθανότητα (και άρα άπειρο κόστος) σε edits που δεν βρήκαμε στο word_edits.txt, αλλά υλοποιεί add-1 smoothing και αποδίδει σε αυτά τα edits την πιθανότητα 1/(N+V), όπου N: σύνολο όλων των edits, V: όλα τα διαφορετικά edits.

Τα αποτελέσματα που προκύπτουν για την δοκιμή στο αρχείο spell_test.txt είναι τα εξής:



Παρατηρούμε ότι τα αποτελέσματα είναι λίγο καλύτερα από το προηγούμενο EV, και θα μπορούσαν να βελτιωθούν κι άλλο αν είχαμε μεγαλύτερο corpus.

Μέρος 2:

Στο αρχείο slp-lab1-part2.ipynb βρίσκονται οι κώδικες για το Μέρος 2.

Βήμα 12:

Δημιουργήσαμε την λίστα sentence, η οποία περιέχει “προτάσεις”-tokens από το αρχείο gutenberg.txt, με σημείο διαχωρισμού την αλλαγή γραμμής.
Με την βοήθεια της λίστας αυτής, εκπαιδεύσαμε το model1 με παράθυρο μήκους 5.

Μετά την εκπαίδευση, προσπαθήσαμε να βρούμε τις 10 σημασιολογικά κοντινότερες λέξεις στις λέξεις bible, book, bank, water.
Τα αποτελέσματα είναι τα παρακάτω:

```
Bible Similar:
('seriousness', 0.3718793988227844)
('dreamer', 0.36886540055274963)
('cob', 0.3591678738594055)
('propitious', 0.34457987546920776)
('syrian', 0.3390393853187561)
('counts', 0.3369024693965912)
('poetry', 0.3312138319015503)
('verie', 0.33115240931510925)
('pulpit', 0.3305303156375885)
('beads', 0.31800422072410583)

Book Similar:
('letter', 0.4842962622642517)
('written', 0.4792977273464203)
('temple', 0.46236783266067505)
('note', 0.447600781917572)
('pen', 0.4464194178581238)
('word', 0.42215245962142944)
('chronicles', 0.4212881922721863)
('chapter', 0.42086586356163025)
('mouth', 0.412833571434021)
('seal', 0.4087667167186737)
```

```
Bank Similar:
('top', 0.5457448959350586)
('table', 0.5084173083305359)
('floor', 0.48206016421318054)
('river', 0.468691885471344)
('side', 0.4670913517475128)
('hill', 0.46269726753234863)
('ridge', 0.4617703855037689)
('ground', 0.453009694814682)
('wall', 0.4523957669734955)
('trudged', 0.44693323969841003)

Water Similar:
('waters', 0.613674521446228)
('wine', 0.53905189037323)
('fire', 0.48961469531059265)
('wood', 0.48945316672325134)
('ground', 0.48344963788986206)
('oil', 0.48240533471107483)
('blood', 0.46017879247665405)
('smoke', 0.4569704830646515)
('hole', 0.44915759563446045)
('fowls', 0.44907060265541077)
```

Παρατηρούμε ότι ενώ κάποια αποτελέσματα είναι καλά (π.χ. water-fire, book-letter) άλλα δεν βγάζουν τόσο νόημα (π.χ. bank-river).

Στην συνέχεια δοκιμάζουμε δύο νέα μοντέλα, για 100 και 10 εποχές αντίστοιχα.
Τα αποτελέσματα για 100 εποχές είναι τα παρακάτω:

```
Bible Similar:
('dreamer', 0.4342961609363556)
('propitious', 0.40272602438926697)
('answerer', 0.4013298451900482)
('pliny', 0.3979036211967468)
('title', 0.3977380692958832)
('poetry', 0.3628053367137909)
('pub', 0.3592997193336487)
('falsely', 0.3583223223686218)
('jonas', 0.3550158143043518)
('seer', 0.3513900935649872)

Book Similar:
('chronicles', 0.559377133846283)
('written', 0.5343720316886902)
('letter', 0.49462002515792847)
('volume', 0.4642413556575775)
('covenant', 0.46199753880500793)
('chapter', 0.4580816924571991)
('epistle', 0.45133349299430847)
('mouth', 0.4467475414276123)
('law', 0.44571366906166077)
('headlines', 0.43450728058815)
```

```
Bank Similar:
('top', 0.5511630177497864)
('floor', 0.525230884552002)
('side', 0.5161129832267761)
('wall', 0.5136275291442871)
('ground', 0.5116709470748901)
('table', 0.5025489330291748)
('edge', 0.49797216057777405)
('river', 0.4974571764469147)
('hill', 0.4959597587585449)
('tower', 0.4829375743865967)

Water Similar:
('waters', 0.6162407994270325)
('river', 0.5352722406387329)
('streams', 0.5252775549888611)
('wine', 0.5239294171333313)
('camels', 0.5048832893371582)
('buckets', 0.4904658794403076)
('fowls', 0.4903753399848938)
('ground', 0.4889426827430725)
('sea', 0.4685840904712677)
('blood', 0.4673968255519867)
```

Τα αποτελέσματα για 10 εποχές είναι τα παρακάτω:

```
Bible Similar:
('eidolons', 0.73551344871521)
('gittite', 0.7245368957519531)
('thanes', 0.722448468208313)
('literatures', 0.7192274332046509)
('dwells', 0.7189253568649292)
('hashum', 0.7139861583709717)
('robin', 0.7126010060310364)
('remoue', 0.7103672623634338)
('massy', 0.708891749382019)
('natives', 0.7080622911453247)

Book Similar:
('chapter', 0.6683662533760071)
('chronicles', 0.6407068371772766)
('letter', 0.6085094213485718)
('volume', 0.6024599671363831)
('note', 0.5997639894485474)
('written', 0.5634008049964905)
('epistle', 0.549001932144165)
('prophet', 0.5489466786384583)
('folio', 0.5303294062614441)
('temple', 0.5270305871963501)
```

```
Bank Similar:
('lawn', 0.8556665182113647)
('floor', 0.8353729248046875)
('wall', 0.8078148365020752)
('top', 0.805381178855896)
('beach', 0.7994846701622009)
('hill', 0.79826819896698)
('road', 0.7869361639022827)
('threshold', 0.7685083746910095)
('shore', 0.7645455598831177)
('east', 0.7627404928207397)

Water Similar:
('wood', 0.6936861872673035)
('smoke', 0.6536219716072083)
('waters', 0.6452018618583679)
('ashes', 0.6295132637023926)
('streams', 0.6179492473602295)
('steel', 0.6118414998054504)
('corn', 0.6080697774887085)
('rivers', 0.5992569923400879)
('wine', 0.5987640619277954)
('fire', 0.5986717939376831)
```

Τα αποτελέσματα για παράθυρο μήκους 10 είναι τα παρακάτω:

```
Bible Similar:
('republic', 0.7240028381347656)
('spain', 0.7129253149032593)
('climes', 0.6993950605392456)
('persecutions', 0.6960647106170654)
('dwells', 0.688989520072937)
('displays', 0.6880658268928528)
('misapprehension', 0.687912881374359
('hauing', 0.6866286993026733)
('museum', 0.6856420040130615)
('exhorting', 0.6823752522468567)

Book Similar:
('chapter', 0.6891915202140808)
('chronicles', 0.6169758439064026)
('written', 0.6071396470069885)
('prophecy', 0.5652978420257568)
('prophet', 0.565168023109436)
('letter', 0.5636975765228271)
('note', 0.5513817667961121)
('vision', 0.5460059642791748)
('epistle', 0.5458642840385437)
('folio', 0.5403078198432922)
```

```
Bank Similar:
('floor', 0.8299848437309265)
('top', 0.8209477066993713)
('bay', 0.803145170211792)
('road', 0.8017860651016235)
('beach', 0.8013285994529724)
('hill', 0.7976561188697815)
('lawn', 0.7947807312011719)
('hedge', 0.7822613716125488)
('path', 0.7777345776557922)
('wall', 0.7771165370941162)

Water Similar:
('wood', 0.6504098773002625)
('waters', 0.6160298585891724)
('ashes', 0.6140490174293518)
('root', 0.6133626103401184)
('smoke', 0.6048316955566406)
('streams', 0.5931802988052368)
('rivers', 0.5920335054397583)
('brook', 0.5886960029602051)
('trees', 0.5876567363739014)
('pit', 0.5868188738822937)
```

Τα αποτελέσματα για παράθυρο μήκους 15 είναι τα παρακάτω:

```
Bible Similar:
('slavery', 0.701246440410614)
('patriot', 0.693037748336792)
('royalty', 0.6879929900169373)
('eidolons', 0.6872194409370422)
('amariah', 0.6867314577102661)
('baron', 0.6855996251106262)
('achilles', 0.684070348739624)
('tragedie', 0.6839808821678162)
('xi', 0.681718647480011)
('aristarchus', 0.6763778328895569)

Book Similar:
('chapter', 0.655816376209259)
('chronicles', 0.6072105169296265)
('note', 0.6068582534790039)
('letter', 0.6044731736183167)
('written', 0.6022891998291016)
('volume', 0.560756504535675)
('law', 0.5313780903816223)
('vision', 0.5311765670776367)
('writing', 0.5204434394836426)
('temple', 0.5164452791213989)
```

```
Bank Similar:
('floor', 0.8228034973144531)
('beach', 0.802124559879303)
('lawn', 0.7908689975738525)
('top', 0.7808291912078857)
('hedge', 0.7632803916931152)
('bay', 0.7524194717407227)
('pavement', 0.7513832449913025)
('road', 0.7498458623886108)
('crossing', 0.7413783073425293)
('hill', 0.7399481534957886)

Water Similar:
('wood', 0.6651054620742798)
('waters', 0.6112533807754517)
('ashes', 0.6062209606170654)
('streams', 0.6033791899681091)
('pit', 0.5971135497093201)
('smoke', 0.5882624983787537)
('trees', 0.5843954086303711)
('furnace', 0.580277144908905)
('brooks', 0.5800072550773621)
('field', 0.578273355960846)
```

Σχετικά με τον αριθμό των εποχών βλέπουμε ότι όσο μειώνεται ο αριθμός τους τα αποτελέσματα είναι χειρότερα, καθώς παρατηρείται κάποιες λέξεις να έχουν αρκετά μεγάλο cosine similarity χωρίς να συσχετίζονται, όπως η λέξη Bank με τη λέξη lawn (0.86), ενώ για περισσότερες εποχές δεν παρατηρούνται τόσο μεγάλες τιμές για το similarity.

Αυξάνοντας το παράθυρο σε 10 (οι εποχές έμειναν 10 οπότε συγκρίνονται με το προηγούμενο μοντέλο και όχι το αρχικό, για να μην είναι μεγάλος ο χρόνος εκπαίδευσης του μοντέλου) βλέπουμε ότι κάποια αποτελέσματα παραμένουν καλά και υπάρχει βελτίωση στα 10 κοντινότερα, προκύπτουν όμως και κακά αποτελέσματα (π.χ. bible-climes).

Παρατηρούμε ότι αν αυξήσουμε το παράθυρο σε 15 έχουμε ακόμη καλύτερα αποτελέσματα στα 10 κοντινότερα, αφού οι επιλεγμένες λέξεις έχουν μεγαλύτερη συνάφεια από τις προηγούμενες.

Επομένως, η αύξηση του παράθυρου δίνει καλύτερα αποτελέσματα, καθώς για την κάθε λέξη επιλέγονται περισσότερες λέξεις για να καθοριστεί το διάνυσμά του.

Για το κομμάτι των σημασιολογικών αναλογιών, για το μοντέλο μας είχαμε τα εξής αποτελέσματα:

```
Result for (girls,queens,kings): boys ,distance: 0.6931698322296143
Result for (good,taller,tall): good ,distance: 0.6434131860733032
Result for (france,paris,london): london ,distance: 0.754153847694397
```

Στη συνέχεια, φορτώνουμε τα προεκπαιδευμένα GoogleNews vectors. Για τις προηγούμενες λέξεις, προκύπτουν με βάση αυτό το μοντέλο οι παρακάτω λέξεις ως οι σημασιολογικά κοντινότερες:

```
Bible Similar:
('Bible', 0.736778199672699)
('bibles', 0.6052598357200623)
('Holy_Bible', 0.5989601612091064)
('scriptures', 0.574568510055542)
('scripture', 0.5697901844978333)
('New_Testament', 0.5638793110847473)
('Scripture', 0.5502957701683044)
('Scriptures', 0.5411645770072937)
('NRSV', 0.5341106057167053)
('Leviticus_##:##-##', 0.5247005224227905)

Book Similar:
('tome', 0.7485830783843994)
('books', 0.7379177808761597)
('memoir', 0.7302926778793335)
('paperback_edition', 0.6868364214897156)
('autobiography', 0.6741527318954468)
('memoirs', 0.6505153179168701)
('Book', 0.6479282975196838)
('paperback', 0.6471226811408997)
('novels', 0.6341459155082703)
('hardback', 0.6283079981803894)
```

```
Bank Similar:
('banks', 0.7440759539604187)
('banking', 0.690161406993866)
('Bank', 0.6698698401451111)
('lender', 0.6342284679412842)
('banker', 0.6092953085899353)
('depositors', 0.6031531691551208)
('mortgage_lender', 0.5797975659370422)
('depositor', 0.5716427564620972)
('BofA', 0.5714625120162964)
('Citibank', 0.5589520335197449)

Water Similar:
('potable_water', 0.6799106001853943)
('Water', 0.6706871390342712)
('sewage', 0.6619377732276917)
('groundwater', 0.6588346362113953)
('Floridan_aquifer', 0.6422534584999084)
('freshwater', 0.6307883262634277)
('potable', 0.6251927614212036)
('wastewater', 0.6212229132652283)
('brackish_groundwater', 0.6206730604171753)
('aquifer', 0.6143589615821838)
```

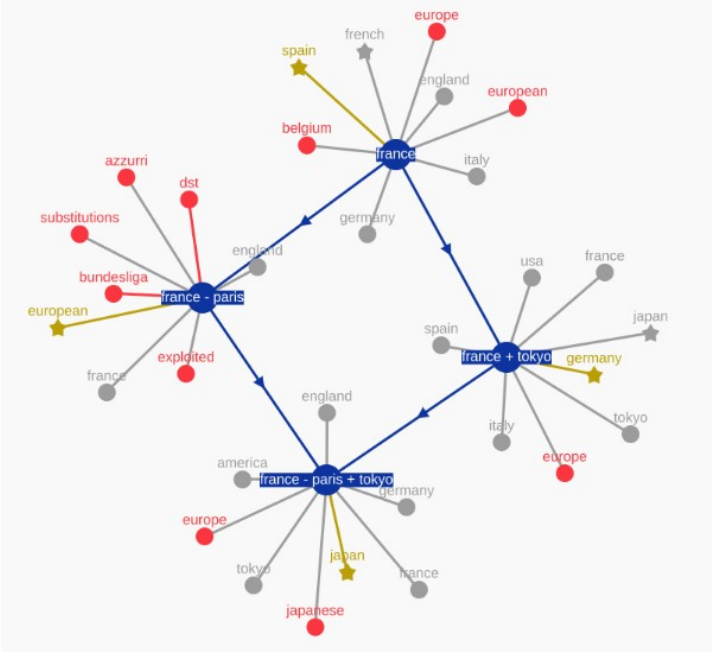
Παρατηρούμε ότι τα αποτελέσματα είναι σημαντικά καλύτερα από αυτά που έδιναν τα μοντέλα που βασίζονταν στο gutenberg.txt. Αυτό συμβαίνει, γιατί το corpus είναι σημαντικά μεγαλύτερο και χρησιμοποιεί για διανύσματα 300 διαστάσεων.

Σχετικά με τις σημασιολογικές αναλογίες, το μοντέλο των GoogleNews δίνει τα παρακάτω αποτελέσματα:

```
Result for (girls,queens,kings): boys ,distance: 0.6931698322296143
Result for (good,taller,tall): good ,distance: 0.6434131860733032
Result for (france,paris,london): london ,distance: 0.754153847694397
```

Βήμα 13:

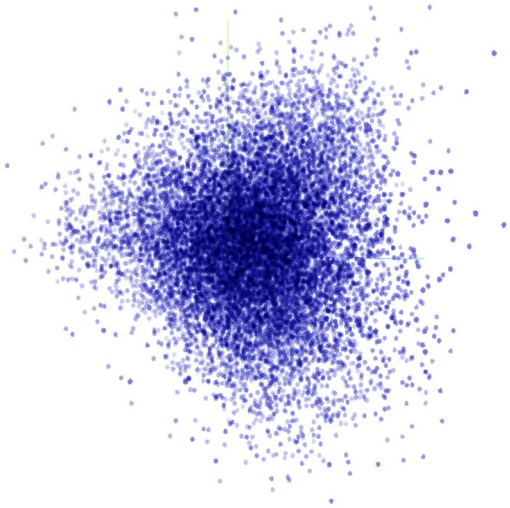
Για την είσοδο “france – paris + tokyo” έχουμε τα εξής αποτελέσματα:



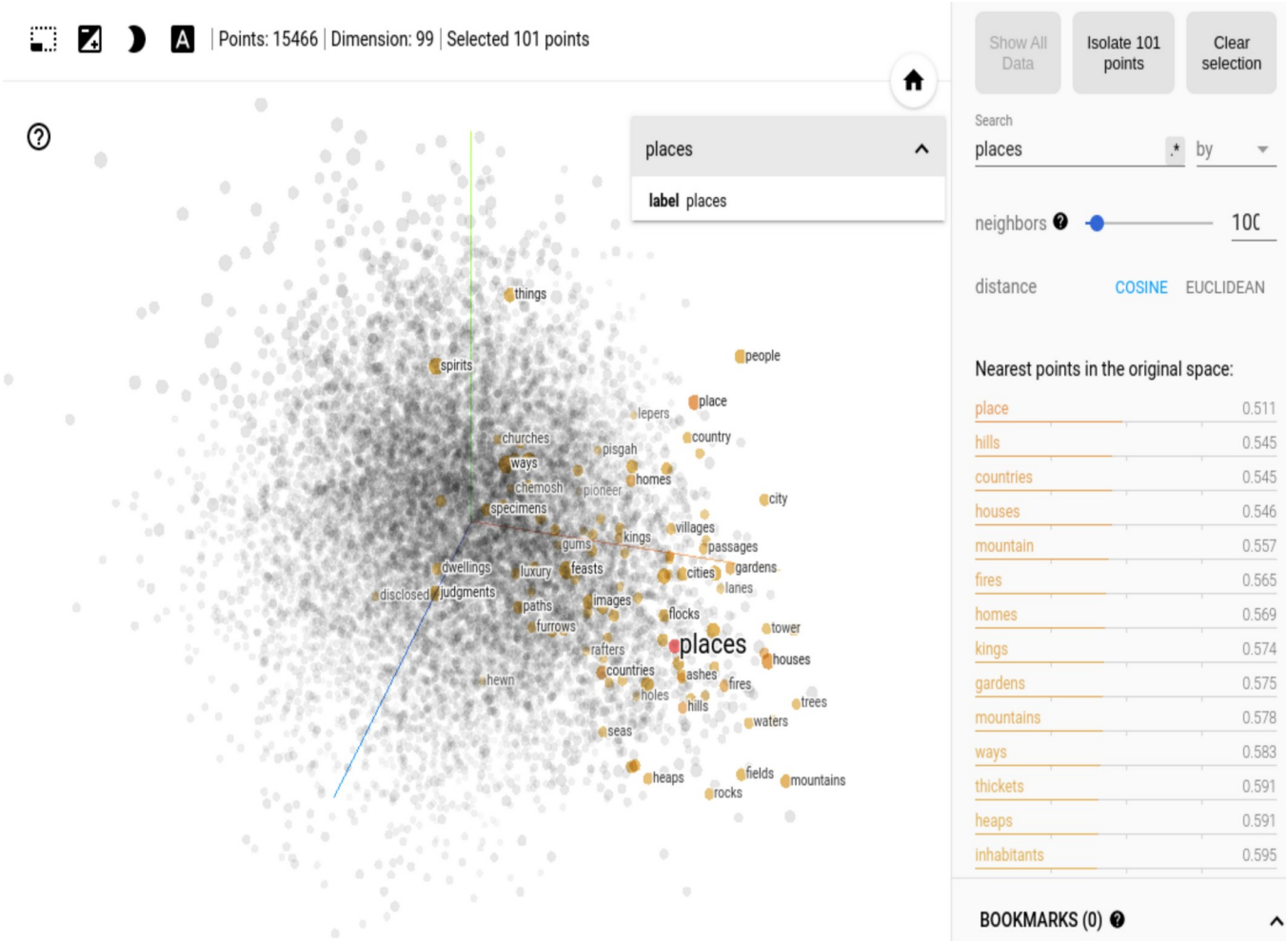
Παρατηρούμε ότι για το “france“ η κοντινότερη λέξη είναι η “spain”, ενώ ακολουθούν οι “belgium”,”europe”,”european”.
Αντίστοιχα το “france-paris” έχει ως κοντινότερη λέξη την “france”, με απόσταση 0.5, ενώ τα υπόλοιπα έχουν απόσταση 0.2.

Στην συνέχεια δημιουργήσαμε τα αρχεία embeddings.tsv και metadata.tsv. και τα δοκιμάσαμε στον emdedding projector. Για την μείωση της διαστατικότητας χρησιμοποιήσαμε το PCA, κάθως το t-SNE δυσκολεύοταν να τρέξει.

Αρχικά, χρησιμοποιήσαμε τη μέθοδο PCA, για την οποία προκύπτει ο παρακάτω τρισδιάστατος χώρος:

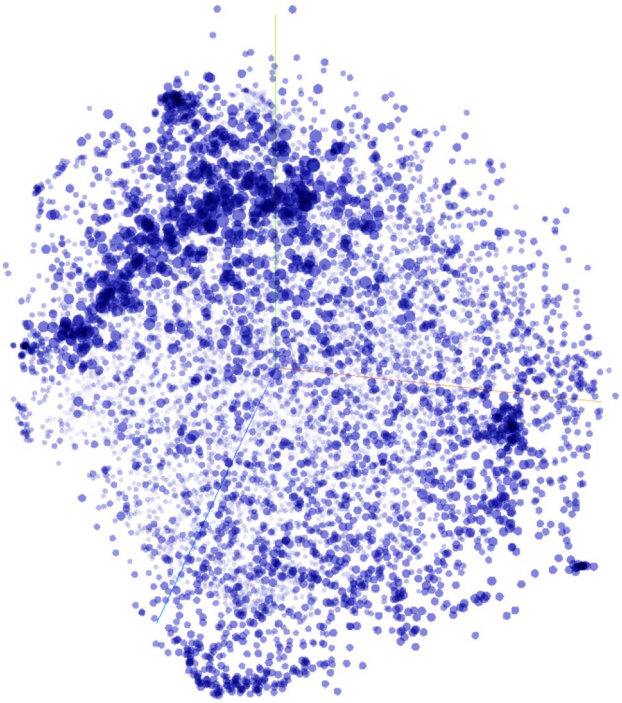


Επιλέγοντας κάποια σημεία/λέξεις του χώρου, παρατηρούμε τις κοντινότερες λέξεις τους:

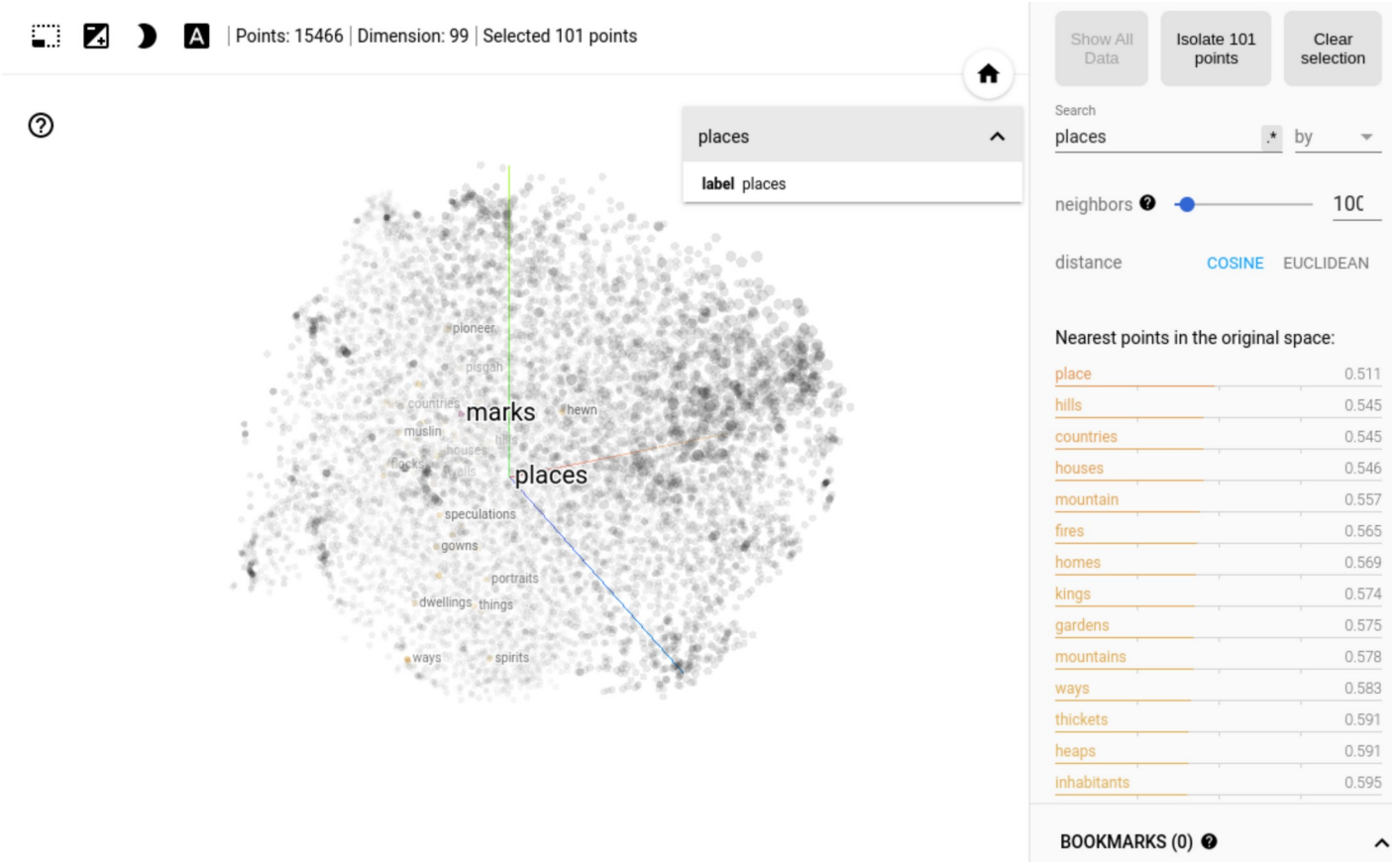


Παρατηρούμε ότι το μοντέλο μας δίνει για το διάνυσμα “places” τις λέξεις “place”,”hills”,”countries”, τα οποία έχουν μεταξύ τους συσχέτιση, όμως έχουμε και διανύσματα όπως “kings” που φαίνονται άσχετα με την αρχική μας λέξη.

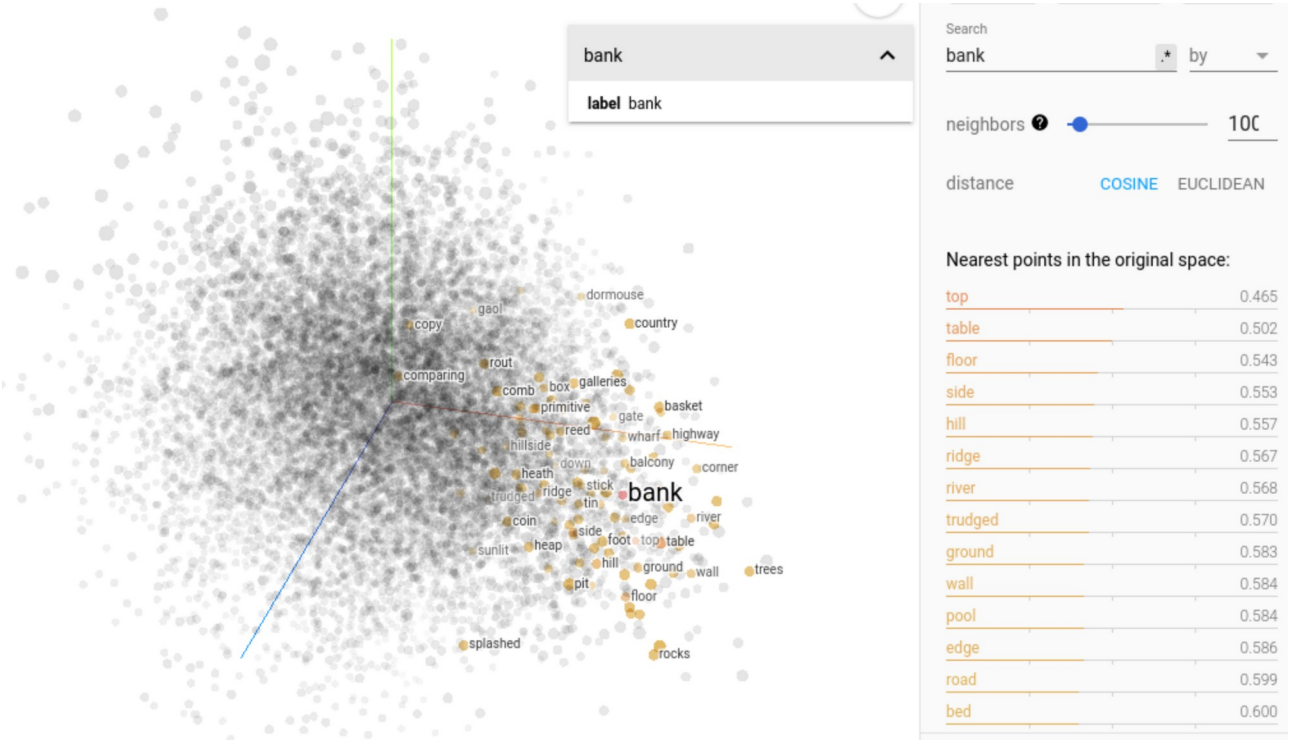
Στη συνέχεια, ελέγξαμε τη μέθοδο T-SNE. Με αυτή έχουμε τον παρακάτω τρισδιάστατο χώρο:



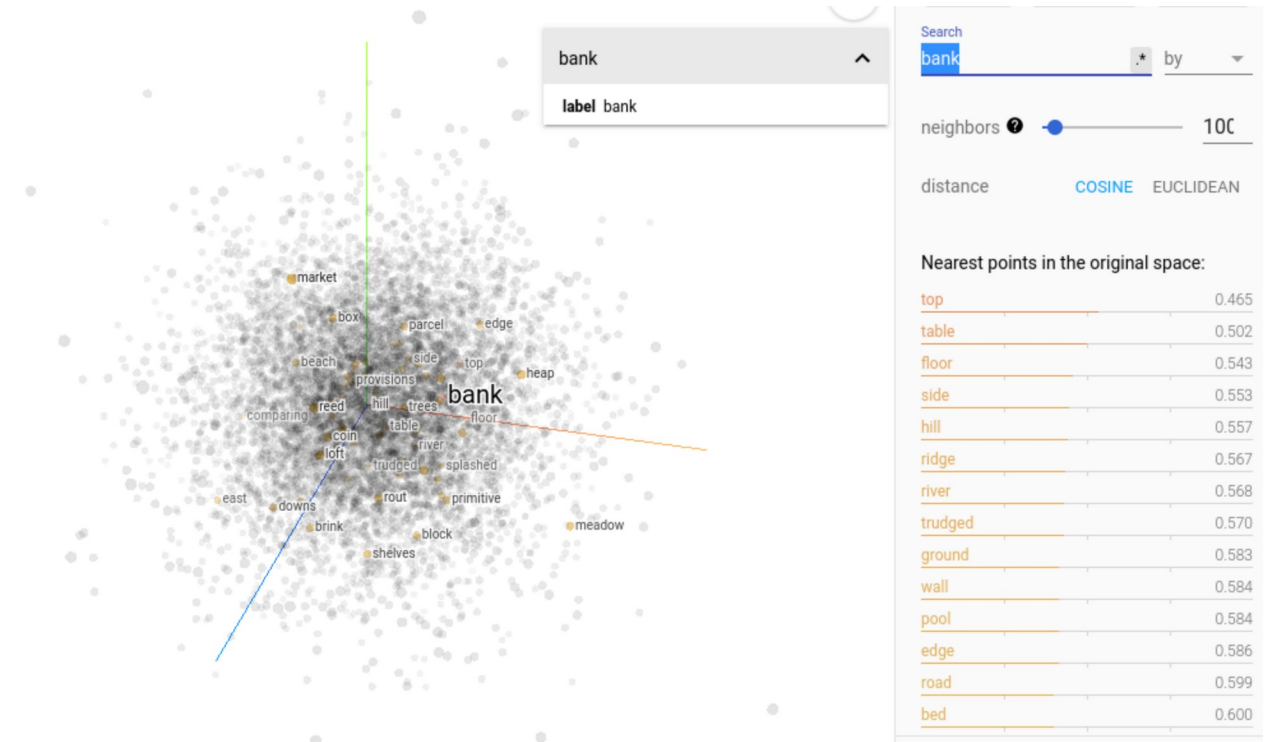
Οι κοντινότερες λέξεις για το “places” παραμένουν ίδιες:



Αντίστοιχα, για τη λέξη “Bank” παίρνουμε για PCA τα παρακάτω:



Ενώ με τη μέθοδο T-SNE παίρνουμε:



Τα κοντινότερα διανύσματα παραμένουν τα ίδια, ενώ είναι σχεδόν τα ίδια με αυτά που υπολογίσαμε ως κοντινότερα στο βήμα 12.

Βήμα 14:

Για την προεπεξεργασία μετατρέπουμε κάθε κεφαλαίο χαρακτήρα σε πεζό και αφαιρούμε τα σημεία στίξης και τέλος χωρίζουμε τις προτάσεις σε tokens.
Για την κατασκευή του Neural Bag of Words για κάθε πρόταση υπολογίζουμε το άθροισμα των διανυσμάτων των λέξεων της (για λέξεις εκτός λεξιλογίου δεν προσθέτουμε κάτι, το οποίο είναι ισοδύναμο με το να προσθέτουμε το μηδενικό διάνυσμα) και τέλος διαιρούμε με το πλήθος των λέξεων, για να πάρουμε τον μέσο όρο.
Χρησιμοποιούμε αρχικά το μοντέλο που κατασκευάσαμε αρχικά στο βήμα 12 και παίρνουμε ακρίβεια 0.7353. Η επίδοση του Logistic Regression μοντέλου θα μπορούσε να βελτιωθεί αν χρησιμοποιούσαμε κάποιο μοντέλο με μεγαλύτερο corpus.
Πράγματι, αν χρησιμοποιήσουμε το GoogleNews για την εκπαίδευση του μοντέλου η ακρίβεια αυξάνεται σε 0.8323.