Υλοποίηση Locality Sensitivity Hashing (LSH)

Ο αλγόριθμος Locality-Sensitive Hashing (LSH) είναι ένας αλγόριθμος που χρησιμοποιείται κυρίως στον τομέα της αναζήτησης και της αναγνώρισης πλησιέστερων γειτόνων σε δεδομένα πολυδιάστατους χώρους. Ο στόχος του αλγορίθμου LSH είναι να μετατρέψει τα δεδομένα ώστε να είναι ευαίσθητα στη γεωμετρική ομοιότητα, επιτρέποντας την αποδοτική αναζήτηση πλησιέστερων γειτόνων χωρίς την ανάγκη εξέτασης όλων των δυνατών συνδυασμών. Στην παρούσα εργασία ο αλγόριθμος υλοποιείται από τις κλάσεις MinHash και LSH στο αρχείο lsh.py.

Τα βασικά «συστατικά» στοιχεία του LSH είναι τα παρακάτω

**Συνάρτηση Hashing:** Ο LSH χρησιμοποιεί μια συνάρτηση κατακερματισμού (hash function) για να μετατρέψει τα δεδομένα από τον αρχικό χώρο σε έναν χώρο μικρότερων διαστάσεων. Αυτή η συνάρτηση hash προσπαθεί να είναι ευαίσθητη στη γεωμετρική ομοιότητα, δηλαδή να "κατακερματίζει" παρόμοια δεδομένα κοντά στην ίδια θέση. Αυτή η λειτουργία υλοποιείται στην κλάση MinHash από τις μεθόδους \_hash, build\_functions και \_signature\_matrix.

**Σύγκριση Buckets:** Οι μετατροπές των δεδομένων δημιουργούν "κάδους" ή "buckets" στον νέο χώρο. Δύο δεδομένα που καταλήγουν στον ίδιο κάδο θεωρούνται ότι είναι πιθανά πολύ κοντά γεωμετρικά στον αρχικό χώρο. Αυτή η λειτουργία υλοποιείται στη μέθοδο \_get\_candidates της κλάσης LSH.

**Αναζήτηση Γειτόνων:** Αφού τα δεδομένα αντιστοιχίστηκαν σε κάδους, μπορούμε να αναζητήσουμε γρήγορα πλησιέστερους γείτονες εξετάζοντας μόνο τους κάδους που περιέχουν τα δεδομένα που μας ενδιαφέρουν. Αυτή η λειτουργία υλοποιείται στη μέθοδο neighbors της κλάσης LSH.

Μετρικές ομοιότητας

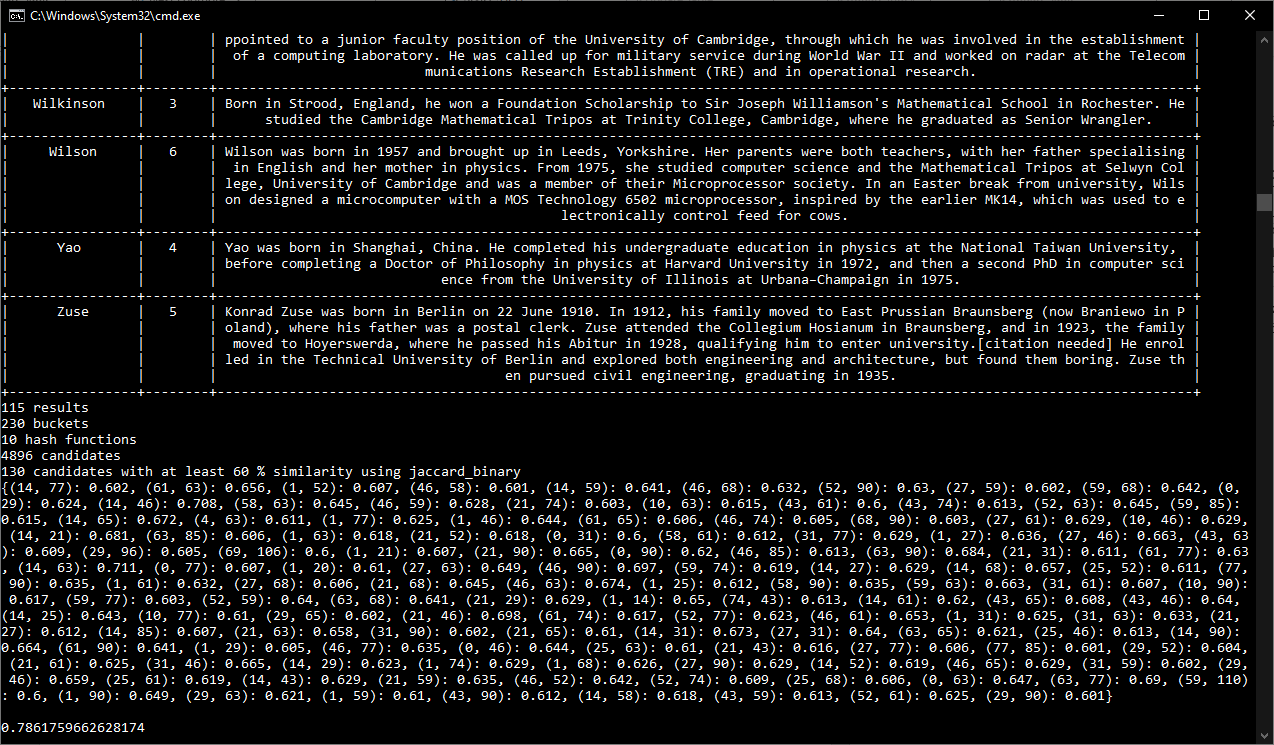
Στο αρχείο tools.py έχουμε υλοποιήσει τις συναρτήσεις που χρησιμοποιούνται για την εύρεση της ομοιότητας μεταξύ των διανυσμάτων στον πολυδιάστατο χώρο.

Η μετρική ομοιότητας cosine, γνωστή και ως "cosine similarity," είναι ένα μέτρο που χρησιμοποιείται για να αξιολογήσει τον βαθμό της ομοιότητας ανάμεσα σε δύο διανύσματα σε έναν πολυδιάστατο χώρο. Συνήθως, αυτός ο πολυδιάστατος χώρος είναι χώρος χαρακτηριστικών όπου κάθε διάσταση αντιστοιχεί σε ένα χαρακτηριστικό ή μια μετρική. Η cosine similarity είναι ιδιαίτερα χρήσιμη στον τομέα της ανάκτησης πληροφοριών, της αναζήτησης κειμένου και της συστατικής φιλτράρισης. Συνήθως, χρησιμοποιείται για τη σύγκριση της ομοιότητας μεταξύ κειμένων ή σετ δεδομένων όπου κάθε στοιχείο αναπαρίσταται ως ένα διάνυσμα χαρακτηριστικών. Η cosine similarity επιτρέπει να αξιολογηθεί πώς δύο διανύσματα σχετίζονται μεταξύ τους, ανεξάρτητα από το μέγεθος των διανυσμάτων, και μπορεί να βοηθήσει στην ανίχνευση παρόμοιων στοιχείων σε μεγάλα σύνολα δεδομένων. Αυτή η μετρική ομοιότητας υλοποιείται στη μέθοδο cosine\_similarity.

Η μετρική ομοιότητας Jaccard είναι ένα μέτρο που χρησιμοποιείται για να αξιολογήσει τον βαθμό της ομοιότητας ανάμεσα σε δύο σύνολα δεδομένων. Συνήθως, χρησιμοποιείται όταν τα δεδομένα προτιμούνται να αναπαριστούν ως δυαδικές (binary) μεταβλητές, που μπορούν να είναι είτε παρόντα (1) είτε απόντα (0). Η Jaccard similarity υπολογίζει την ομοιότητα ανάμεσα σε δύο σύνολα A και B ως τον λόγο του αριθμού των στοιχείων που υπάρχουν και στα δύο σύνολα προς τον αριθμό των στοιχείων που υπάρχουν σε οποιοδήποτε από τα δύο σύνολα. Ουσιαστικά, αυτή η μετρική συγκρίνει τον αριθμό των κοινών δυαδικών στοιχείων στα δύο σύνολα με τον συνολικό αριθμό των διαφορετικών δυαδικών στοιχείων που υπάρχουν σε αυτά τα σύνολα. Ο κύριος περιορισμός της Jaccard similarity είναι ότι δεν λαμβάνει υπόψη τη συχνότητα των διαφορετικών στοιχείων, αλλά μόνο το γεγονός ότι αυτά τα στοιχεία υπάρχουν ή όχι στα σύνολα. Αυτή η μετρική ομοιότητας υλοποιείται στη μέθοδο jaccard\_binary.

Εφαρμογή του αλγόριθμου στα αποτελέσματα αναζήτησης

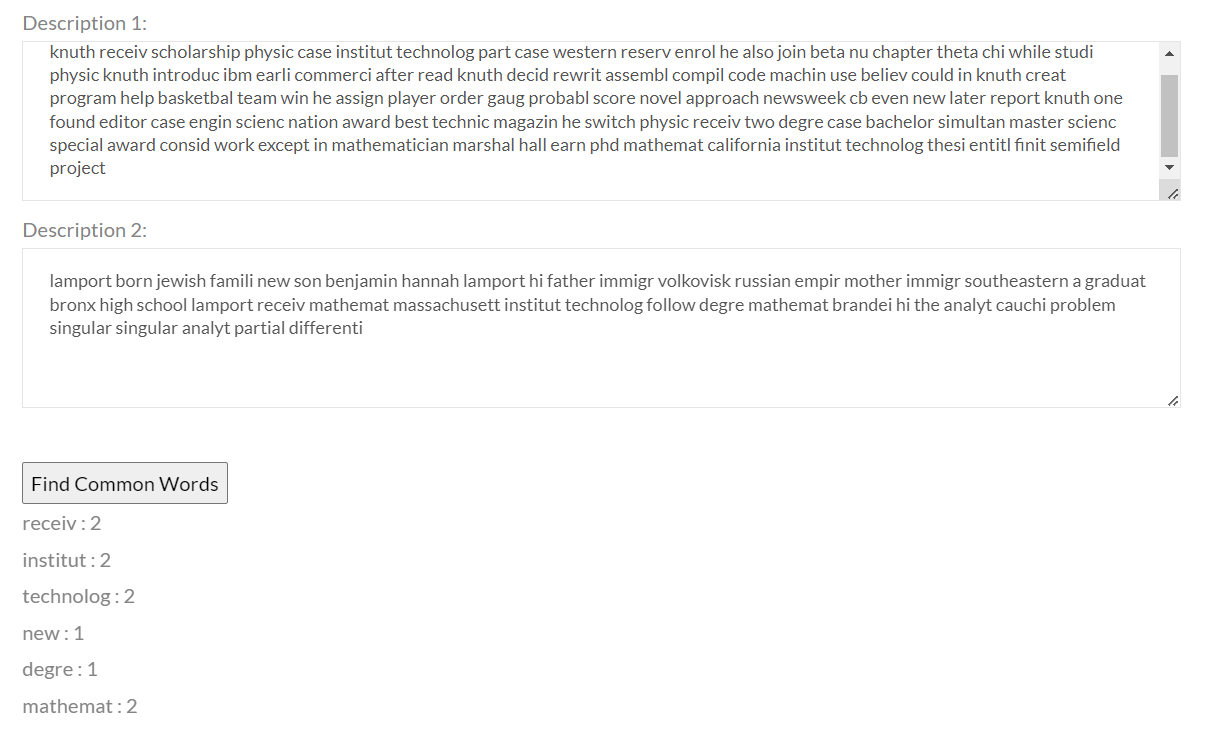
Στο αρχείο main.py έχουμε υλοποιήσει τη μέθοδο lsh\_test η οποία δέχεται ως όρισμα μια λίστα αποτελεσμάτων από την αναζήτηση, ένα κατώτατο όριο ομοιότητας και των αριθμό των buckets (ανάλογα με τα αποτελέσματα που θα έχουμε). Ακόμη, έχουμε θέσει ως default μετρική ομοιότητας την Jaccard. Η επιλογή αυτή έχει προτιμηθεί εξαιτίας του ότι μετατρέπουμε τα αποτελέσματα σε one hot αντί για tf-idf (όπου και θα ήταν προτιμότερο να χρησιμοποιούμε τη μετρική συνημιτόνου). Έχουμε επιλέξει να μεταβάλλουμε τον αριθμό των buckets ανάλογα με το πλήθος των αποτελεσμάτων ώστε να έχουμε όσο γίνεται πιο σταθερό πλήθος candidate pairs (π.χ. όταν έχουμε λίγα results και χρησιμοποιούμε πάρα πολλά buckets είναι απίθανο να παρατηρήσουμε όμοιες εξόδους από το lsh\_test). Παρακάτω βλέπουμε πως εφαρμόζεται ο αλγόριθμος LSH στα αποτελέσματα της αναζήτησης του R-Tree όπου έχουμε θέσει threshold = 0.6, letters = Α-Ζ, awards = 3.



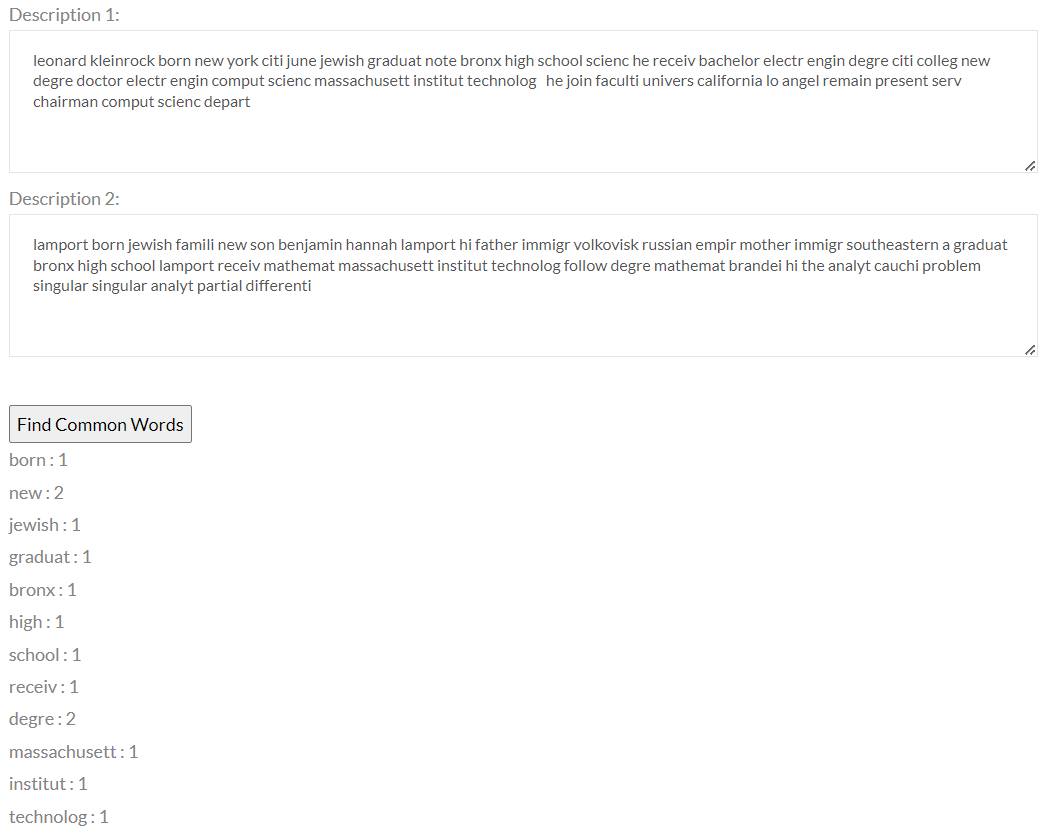
Εικόνα 1: Εφαρμογή του αλγόριθμου LSH σε R-Tree search results με similarity threshold.

Από τα 115 αποτελέσματα, έχουμε συνολικά 4896 πιθανά ζεύγη και 130 από αυτά παρουσιάζουν υψηλή μετρική ομοιότητας ( > 0.6).

Κάνοντας μία διαισθητική επαλήθευση πάνω σε τυχαία ζεύγη που βρέθηκαν παρατηρούμε τα παρακάτω

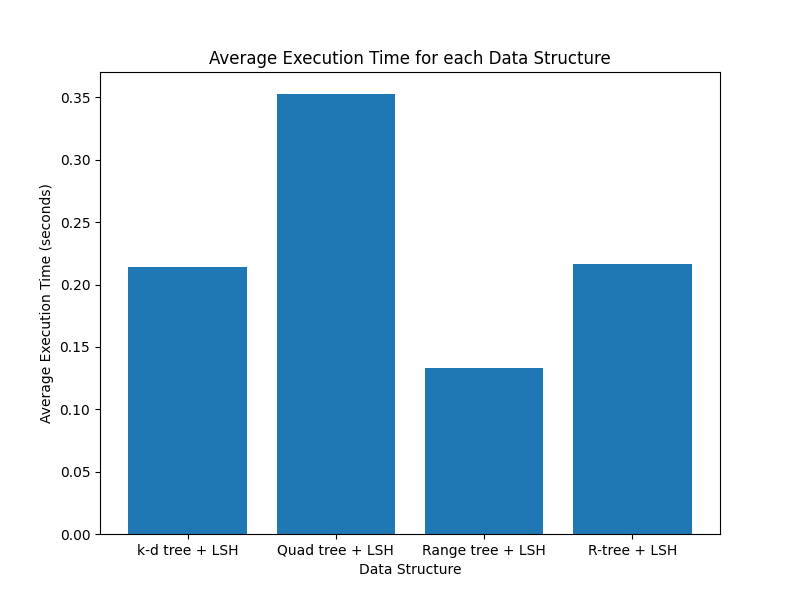


Εικόνα 2: Ένα ζεύγος που παρουσιάζει μετρική ομοιότητας jaccard 0.662



Εικόνα 3: Ένα ζεύγος που παρουσιάζει μετρική ομοιότητας jaccard 0.727

Πειραματική σύγκριση των πολυδιάστατων δομών μέσα από τον αλγόριθμο

Στο αρχείο experiments.py έχουμε υλοποιήσει ένα script που τρέχει τον αλγόριθμο για n φορές σε κάθε μια από τις 4 πολυδιάστατες δομές και επιστρέφει τους μέσους χρόνους εκτέλεσης του όπως παρακάτω

Εικόνα 4: Διάγραμμα μέσου χρόνου εκτέλεσης αλγόριθμου LSH για κάθε μια από τις πολυδιάστατες δομές δεδομένων.