1η Άσκηση Αλγορίθμων Δεδομένων Ευρείας Κλίμακας

Στυλιανός Σύρρος , ΑΜ : 4805

1. Εισαγωγή

Σκοπός της παρούσας εργασίας είναι η μελέτη και υλοποίηση αλγορίθμων υπολογισμού ομοιότητας εγγράφων μεγάλης κλίμακας.
Συγκεκριμένα υλοποιούνται:

- Jaccard Similarity (με σύνολα και ταξινομημένες λίστες)
- MinHash & Signature Similarity
- Brute Force υπολογισμός κοντινότερων γειτόνων (PT5)
- Locality Sensitive Hashing (LSH) για αποδοτικό εντοπισμό υποψήφιων ζευγών (PT6)

Η εργασία βασίζεται στο dataset **Enron emails** (DATA_1-docword.enron.txt).

2. Δομή Συστήματος

Η υλοποίηση χωρίζεται σε δύο βασικά μέρη:

- **Backend (Lab1_with_LSH.py)**
  Περιλαμβάνει όλους τους αλγορίθμους και τις βασικές ρουτίνες.
- **Frontend (FrontEndLab_with_LSH1.py)**
  Διαχειρίζεται το menu, την αλληλεπίδραση με τον χρήστη και την εκτύπωση αποτελεσμάτων.

Η επικοινωνία frontend–backend γίνεται μέσω **global μεταβλητών**, όπως ορίζει και η εκφώνηση.

3. Περιγραφή Backend Ρουτινών

3.1 MyReadDataRoutine(nameDoc, numDocuments)

Διαβάζει τα πρώτα numDocuments έγγραφα από το αρχείο DATA_1-docword.enron.txt και επιστρέφει μία λίστα από frozensets, όπου κάθε frozenset περιέχει τα wordIDs του αντίστοιχου εγγράφου.

Τα frozensets αποθηκεύονται στη λίστα:

listOfFrozensets[docID-1]

3.2 MyJacSimWithSets(docID1, docID2)

Υπολογίζει την Jaccard Similarity δύο εγγράφων συγκρίνοντας απευθείας δύο frozensets με διπλό for-loop.

Χρησιμοποιείται κυρίως για έλεγχο ορθότητας.

3.3 MyJacSimWithOrderedLists(docID1, docID2)

Βελτιστοποιημένη εκδοχή Jaccard Similarity, η οποία:

- μετατρέπει τα frozensets σε ταξινομημένες λίστες
- χρησιμοποιεί δύο δείκτες (two-pointer technique)
- υπολογίζει την τομή σε γραμμικό χρόνο

Αυτή είναι η κύρια μέθοδος Jaccard που χρησιμοποιείται τόσο στο Brute Force όσο και στο LSH.

3.4 create_random_hash_dictionary(maxNum)

Δημιουργεί τυχαία συνάρτηση κατακερματισμού για wordIDs, βασισμένη στη συνάρτηση που δόθηκε στην εκφώνηση.

Χρησιμοποιείται για τη δημιουργία των MinHash permutations.

3.5 MyMinHash(listOfFrozensets, k)

Υλοποιεί τον αλγόριθμο MinHash:

- Δημιουργεί το δυαδικό μητρώο λέξεων–εγγράφων
- Παράγει k τυχαίες hash συναρτήσεις
- Κατασκευάζει το signature matrix SIG
- Επιστρέφει signatures ανά έγγραφο (μορφή: SIG[docID][row])

3.6 MySigSim(docID1, docID2, numPermutations)

Υπολογίζει Signature Similarity συγκρίνοντας τις πρώτες numPermutations γραμμές των υπογραφών δύο εγγράφων.

Η SigSim προσεγγίζει την Jaccard Similarity όσο αυξάνεται ο αριθμός των permutations.

4. Brute Force – PT5

BruteForce(numDocuments, numNeighbors, numPermutations)

Η μέθοδος Brute Force:

1. Υπολογίζει **μία φορά** το signature matrix SIG
2. Υπολογίζει **όλες τις ζεύγη-ομοιότητες**:
    - Jaccard Similarity
    - Signature Similarity
3. Μετατρέπει τις ομοιότητες σε αποστάσεις
4. Βρίσκει τους numNeighbors κοντινότερους γείτονες για κάθε έγγραφο
5. Υπολογίζει:

- AvgSim ανά έγγραφο
- Συνολικό AvgSim

⏱ Παρατήρηση:

- Η Jaccard Brute Force είναι εξαιρετικά αργή ($O(n^2)$)
- Για 100 έγγραφα απαιτεί αρκετά λεπτά
- Η SigSim είναι σαφώς ταχύτερη

5. Locality Sensitive Hashing – PT6

MyLSH(SIG, bands, rows_per_band, numNeighbors)

Η υλοποίηση LSH περιλαμβάνει:

1. **Διάσπαση του signature matrix σε bands**
2. **Hashing κάθε band σε buckets**
3. **Εντοπισμό υποψήφιων ζευγών**
4. **Υπολογισμό ομοιότητας μόνο στα υποψήφια ζεύγη**
5. **Εύρεση Κ κοντινότερων γειτόνων**
6. **Υπολογισμό AvgSim όπως στο Brute Force**

Χρησιμοποιείται σταθερό hash space:

hash(band_tuple) % 5000

Η LSH δεν συγκρίνει όλα τα ζεύγη, αλλά μόνο όσα έγγραφα πέσουν στο ίδιο bucket.

6. Frontend & Menu

Το frontend καθοδηγεί τον χρήστη μέσω menu.

Βασικές επιλογές:

- Υπολογισμός Jaccard / SigSim
- MinHash
- Brute Force
- LSH
- Αλλαγή αριθμού εγγράφων

Menu LSH

Μετά την εκτέλεση LSH, ο χρήστης μπορεί να επιλέξει:

1. Candidate Neighbors (LSH buckets)

2. Final LSH Neighbors (K-NN)
3. AvgSim ανά έγγραφο
4. Total AvgSim
5. Exit

Οι συναρτήσεις LSH **δεν επιστρέφουν τιμές**, αλλά ενημερώνουν global μεταβλητές — όπως ακριβώς και στο Brute Force.

7. Πειραματικά Αποτελέσματα

Παρατηρήσεις

- Η Brute Force Jaccard είναι πρακτικά μη εφαρμόσιμη για μεγάλα N
- Η SigSim συγκλίνει στην Jaccard όσο αυξάνονται τα permutations
- Η LSH:
    - Μειώνει δραστικά τον χρόνο εκτέλεσης
    - Παράγει αποδεκτά AvgSim
    - Εξαρτάται έντονα από τα parameters (bands, rows)

Η LSH επιβεβαιώνει τη θεωρία ότι μπορούμε να πετύχουμε **καλή προσέγγιση** με **πολύ μικρότερο κόστος**.

8. Συμπεράσματα

Η εργασία παρουσιάζει πλήρη υλοποίηση:

- Exact μεθόδων (Brute Force)
- Προσεγγιστικών μεθόδων (MinHash, LSH)

και αναδεικνύει ξεκάθαρα γιατί τεχνικές όπως το LSH είναι απαραίτητες σε δεδομένα μεγάλης κλίμακας.