Τμήμα Μηχανικών Η/Υ και Πληροφορικής Πανεπιστημίου Ιωαννίνων
MYE047: *Αλγόριθμοι για Δεδομένα Ευρείας Κλίμακας*
Ακαδημαϊκό Έτος 2021-22
Διδάσκων: Σπύρος Κοντογιάννης

1ο Σετ Ασκήσεων: Ανίχνευση Κοντινότερων Γειτόνων σε Συλλογή Εγγράφων

Ανακοίνωση: Πέμπτη, 31 Μαρτίου 2022 **Παράδοση:** Πέμπτη, 21 Απριλίου 2022
Τελευταία Ενημέρωση: Τρίτη, 19 Απριλίου 2022

1. ΠΕΡΙΓΡΑΦΗ ΕΡΓΑΣΙΑΣ

Στόχος της παρούσας εργασίας είναι να αξιολογήσουμε την ποιότητα των μεθόδων MINHASH (για παραγωγή υπογραφών) και LSH (για φιλτράρισμα πριν τον υπολογισμό ομοιότητας, είτε μέσω Jaccard είτε μέσω υπογραφών) όταν ζητούμενο είναι ο υπολογισμός των πιο κοντινών γειτόνων ενός εγγράφου X που περιλαμβάνεται σε μια συλλογή εγγράφων. Η αξιολόγηση της εγγύτητας (όσο πιο μικρή γίνεται) ή, ισοδύναμα, της ομοιότητας (όσο πιο μεγάλη γίνεται) των άλλων εγγράφων από το έγγραφο X θα γίνει με δυο βασικούς τρόπους:

(ι) Μέσω της μετρικής της Jaccard-ομοιότητας $JacSim(X,Y)$ του εγγράφου X με καθένα από τα άλλα έγγραφα Y που ανήκουν στη συλλογή. Ως ποιότητα λύσης, θα αποτελέσει τη βάση μας για σύγκριση με τις άλλες μετρικές.

(ιι) Μέσω της ομοιότητας-υπογραφών $SigSim(X,Y)$ του εγγράφου X με καθένα από τα άλλα έγγραφα Y που ανήκουν στη συλλογή.

Η επιλογή των εγγύτερων εγγράφων σε συγκεκριμένο έγγραφο X θα γίνει επίσης με δυο διαφορετικούς τρόπους:

(ιιι) Επιλογή των K εγγράφων από τη συλλογή (φυσικά, εκτός του ίδιου του X) που έχουν τους μεγαλύτερους βαθμούς ομοιότητας με το X, σύμφωνα με τη μετρική της Jaccard ομοιότητας.

(ιν) Επιλογή των K εγγράφων από τη συλλογή (φυσικά, εκτός του ίδιου του X) που έχουν τους μεγαλύτερους βαθμούς ομοιότητας με το X, σύμφωνα με τη μετρική της ομοιότητας-υπογραφών.

Τέλος, όταν επιθυμούμε να βρούμε τους K-πλησιέστερους γείτονες για όλα τα έγγραφα, αντί να υπολογίσουμε όλους τους βαθμούς ομοιότητας (για τα $\frac{D\cdot(D-1)}{2}$ ζεύγη μεταξύ D διαφορετικών εγγράφων), μπορούμε να φιλτράρουμε τα έγγραφα (ως προς τα διανύσματα υπογραφών τους) με χρήση του LSH, ώστε τελικά να μετρήσουμε, είτε τη Jaccard ομοιότητα είτε την ομοιότητα υπογραφών, μόνο μεταξύ των υποψηφίων ζευγών εγγράφων που θα προκύψουν.

Ζητούμενο είναι να αποτιμηθεί η **ταχύτητα υπολογισμού** και η **ποιότητα** της παραγόμενης λύσης, για:

- Τη **BASELINE** λύση, που κάνει $\frac{D\cdot(D-1)}{2}$ υπολογισμούς ομοιότητας σύμφωνα με τη μετρική Jaccard, και υπολογίζει, ανά έγγραφο X, των K-κοντινότερων στο X εγγράφων.
- Τη **MINHASH** λύση, που υπολογίζει το μητρώο υπογραφών, κάνει $\frac{D\cdot(D-1)}{2}$ υπολογισμούς ομοιότητας σύμφωνα με τη μετρική υπογραφών, και υπολογίζει, ανά έγγραφο X, των K-κοντινότερων στο X εγγράφων.
- Την **LSH+JAC** λύση, που υπολογίζει το μητρώο υπογραφών, εφαρμόζει την LSH τεχνική ως προς το μητρώο υπογραφών (για να προκύψουν τα υποψήφια ζεύγη εγγράφων), υπολογίζει Jaccard-ομοιότητα μόνο για τα υποψήφια ζεύγη, και τέλος επιλέγει, ανά έγγραφο X, τα K-πλησιέστερα έγγραφα στο X σύμφωνα με τη Jaccard μετρική.

- Την **LSH+MINHASH** λύση, που υπολογίζει το μητρώο υπογραφών, εφαρμόζει την LSH τεχνική ως προς το μητρώο υπογραφών (για να προκύψουν τα υποψήφια ζεύγη εγγράφων), υπολογίζει ομοιότητα-υπογραφών μόνο για τα υποψήφια ζεύγη, και τέλος επιλέγει, ανά έγγραφο X, τα Κ-πλησιέστερα έγγραφα στο X σύμφωνα με τη μετρική ομοιότητας υπογραφών.

(1α) Ανάγνωση και Προεπεξεργασία Συνόλου Δεδομένων

Τα δύο σύνολα δεδομένων που θα χρησιμοποιήσετε είναι της μορφής «Bag-of-Words», δηλαδή, περιλαμβάνουν μετρητές εμφάνισης λέξεων που περιέχονται σε διάφορα κείμενα της συλλογής. Το πρώτο σύνολο δεδομένων αφορά την ανταλλαγή μηνυμάτων σχετικά με την κατάρρευση του ενεργειακού κολοσσού ENRON. Μπορείτε αν θέλετε να αναζητήσετε στο διαδίκτυο πληροφορίες σχετικά με το συγκεκριμένο σκάνδαλο. Το συγκεκριμένο σύνολο δεδομένων περιλαμβάνει περίπου 40,000 ενδοεταιρικά μηνύματα επικοινωνίας στην ENRON, τα οποία έχουν επίσημα δημοσιοποιηθεί μετά το ξέσπασμα του σκανδάλου. Το δεύτερο σύνολο δεδομένων αφορά μια συλλογή επιστημονικών άρθρων από τη συλλογή NIPS. Αφορά 1,500 διαφορετικά επιστημονικά άρθρα.

Θα βρείτε όλα τα αρχεία για κάθε σύνολο δεδομένων (για την πρώτη συλλογή, με το πρόθεμα «DATA_1-» στο όνομά τους, και για τη δεύτερη συλλογή, με πρόθεμα «DATA_2-» στο όνομά τους) στην ετικέτα FILES του καναλιού LAB-1 στην MS-TEAMS ομάδα του μαθήματος (2021-22_CSE.UOI_MYE047 -- Algorithms for Big Data).

Μελετήστε προσεκτικά, κατ' αρχάς, το αρχείο README.TXT, προκειμένου να κατανοήσετε τη δομή της πληροφορίας που παρέχεται στα αρχεία DATA_1-docword.enron.txt και DATA_2-docword.nips.txt, καθώς πρόκειται για τα πραγματικά δεδομένα που καλείστε να επεξεργαστείτε. Τα συγκεκριμένα αρχεία δεν περιέχουν ούτε ονόματα εγγράφων ούτε λέξεις, παρά μόνο ακεραίους αριθμούς που λειτουργούν ως αναγνωριστικά (εγγράφων και λέξεων), και μετρητές εμφάνισης (συγκεκριμένης λέξης σε συγκεκριμένο έγγραφο). Οι ίδιες οι συμβολοσειρές που αφορούν τις λέξεις δεν είναι απαραίτητες για τις ανάγκες της άσκησης, αλλά αν ενδιαφέρεστε γι' αυτές, μπορείτε να τις βρείτε στο αρχείο DATA_1-vocab.enron.txt και DATA_2-vocab.nips.txt. Όσο για τα ονόματα των αρχείων, αυτά δεν είναι διαθέσιμα για λόγους προστασίας της ιδιωτικότητας.

Κατεβάστε στον τοπικό σας δίσκο τα αρχεία DATA_1-docword.enron.txt και DATA_2-docword.nips.txt, προκειμένου να τα χρησιμοποιήσετε στην παρούσα εργασία. Κάθε αρχείο έχει τον εξής μορφότυπο:

```
D

W

NNZ

docID wordID wordCountInDocument

docID wordID wordCountInDocument

docID wordID wordCountInDocument

docID wordID wordCountInDocument

...

docID wordID wordCountInDocument

docID wordID wordCountInDocument
```

docID wordID wordCountInDocument

όπου:

- D είναι το πλήθος των εγγράφων.
- W είναι το πλήθος των διαφορετικών λέξεων που περιλαμβάνονται στο λεξιλόγιο των εγγράφων.
- N είναι το συνολικό πλήθος εμφανίσεων λέξεων από το λεξιλόγιο στα έγγραφα (δεν είναι τμήμα του αρχείου εισόδου, η τιμή του συγκεκριμενοποιείται στο README.TXT).
- NNZ είναι το πλήθος των μη μηδενικών μετρητών συνολικά στο αρχείο.

Σε πρώτη φάση θα αγνοήσετε τους μετρητές (πεδίο $wordCountInDocument$ σε κάθε γραμμή), επιχειρώντας να αποτυπώσετε τα χαρακτηριστικά διανύσματα εμφάνισης λέξεων, όπως συζητήθηκε και στο μάθημα. Άλλωστε, η ομοιότητα Jaccard παρακολουθεί μόνο εάν μια λέξη εμφανίζεται ή όχι σε ένα έγγραφο, κι όχι πόσες φορές ακριβώς εμφανίζεται.

Θα πρέπει να διαβάσετε το αρχείο εισόδου, πχ, DATA_1-vocab.enron.txt, και να «φορτώσετε» από αυτό το αρχείο ολόκληρη την πληροφορία που αφορά ένα συγκεκριμένο πλήθος $numDocuments$ εγγράφων που προσδιορίζεται κατά την εκτέλεση από τον χρήστη, όπου $1 \leq numDocuments \leq D$.

(1β) ΒΗΜΑ 1: Ανάγνωση & Επεξεργασία Δεδομένων Εισόδου

Θα πρέπει να δημιουργήσετε κάποια δομή δεδομένων στη μνήμη σας που να αποθηκεύει, με ΣΥΜΠΑΓΗ τρόπο, τα χαρακτηριστικά διανύσματα των πρώτων $numDocuments$ εγγράφων στη συλλογή δεδομένων. ΠΡΟΣΟΧΗ: Από τη στιγμή που το DATA_1 περιλαμβάνει 39861 έγγραφα και 28102 διαφορετικές λέξεις, αν επιχειρήσετε να κατασκευάσετε ένα μητρώο όπου στήλες του είναι τα χαρακτηριστικά διανύσματα των εγγράφων, όπως αναφέρει το βιβλίο, τότε προφανώς θα χρειαστείτε χώρο για $39{,}861 \times 28{,}102 = 1{,}120{,}173{,}822$ δίτιμα κελιά (δηλαδή, τουλάχιστον 140 Mbytes) αν χρησιμοποιήσετε κάποια συμπαγή αναπαράσταση σε επίπεδο bits. Αν χρησιμοποιήσετε διδιάστατο μητρώο, όπως στο βιβλίο, απαιτώντας στην καλύτερη περίπτωση ένα byte ανά κελί, τότε η κατάσταση είναι μάλλον απαγορευτική για τον υπολογιστή σας.

Αντίθετα, αν φυλάξετε ΜΟΝΟ τα πραγματικά σύνολα λέξεων (ως συλλογές από διαφορετικές $wordID$ τιμές) που εμφανίζονται στο έγγραφο, τότε συνολικά απαιτούνται (περίπου) $N = 6{,}400{,}000$ ακέραιοι, δηλαδή, $6{,}400{,}000 \times 32 = 204{,}800{,}000$ bits (ισοδύναμα, ~25.6 MBytes).

Η γλώσσα PYTHON έχει τη δυνατότητα της αποδοτικής διαχείρισης συνόλων, μέσω των δομών δεδομένων **set** και **frozenset**. Η διαφορά του set από το frozenset είναι ότι το πρώτο είναι mutable και δε μπορεί να δοθεί ως κλειδί σε έναν πίνακα κατακερματισμού ή σε ένα λεξικό (δομή τύπου dict στην PYTHON), ενώ το δεύτερο είναι immutable και γι' αυτό μπορεί να χρησιμοποιηθεί ως κλειδί για έναν πίνακα κατακερματισμού ή ένα λεξικό (αφού μετά τη δημιουργία του δε μεταβάλλεται). Προτείνεται η χρήση των frozensets για την αναπαράσταση των συνόλων λέξεων που εμφανίζονται σε κάθε έγγραφο.

PROGRAMMING TASK PT1: Δημιουργήστε μια ρουτίνα $MyReadDataRoutine()$ που δέχεται ως είσοδο (πχ, από το πληκτρολόγιο, ή από μενού πολλαπλής επιλογής) το όνομα του αρχείου δεδομένων, καθώς και έναν ακέραιο αριθμό $numDocuments$, και κατασκευάζει τα frozensets των πρώτων $numDocuments$ εγγράφων στη συλλογή (ξεκινώντας από το έγγραφο #1 της συλλογής). Όλα τα frozensets θα πρέπει να οργανωθούν σε μια λίστα από frozensets που δεικτοδοτείται από τις τιμές $d-1$ για κάθε έγγραφο με αναγνωριστικό $d \in \{1, \ldots, numDocuments\}$.

Παρατήρηση: Η επιλογή να επεξεργαζόμαστε, όχι όλα τα έγγραφα, αλλά μόνο ένα συγκεκριμένο πλήθος εγγράφων από τη συλλογή, είναι πολύ χρήσιμη καθώς κάνουμε debugging στον κώδικά μας. Μας επιτρέπει, για το ίδιο

σύνολο δεδομένων, να δημιουργούμε υποσύνολα με τα πρώτα $numDocuments$ έγγραφα, επιτρέποντας τον ταχύτερο έλεγχο ορθότητας του κώδικά μας. Να συμπεριλάβετε οπωσδήποτε αυτή τη δυνατότητα στο πρόγραμμά σας, διευκολύνει και εσάς (κατά την υλοποίηση) αλλά και εμάς (κατά τον έλεγχο).

(1γ) ΒΗΜΑ 2: Συναρτήσεις για Jaccard Ομοιότητα

Στη συνέχεια θα πρέπει να κατασκευάσετε τη δική σας ρουτίνα $MyJacSim(docID1, docID2)$ που να υπολογίζει τη Jaccard ομοιότητα για δυο συγκεκριμένα αναγνωριστικά εγγράφων (για την ακρίβεια, συνόλων) $docID1$ και $docID2$ από τη συλλογή σας (αρκεί να ισχύει ότι $1 \leq docID1$, $docID2 \leq numDocuments$). Η υλοποίησή σας θα πρέπει να συγκρίνει τα frozensets των $docID1$ και $docID2$, και από αυτή τη σύγκριση να υπολογίζει τη ζητούμενη Jaccard ομοιότητα. Θυμηθείτε ότι:

$$JacSim(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Ουσιαστικά λοιπόν, το μόνο που πρέπει να κάνουμε για να βρούμε την τιμή της Jaccard ομοιότητας δυο συνόλων A και B, είναι να υπολογίσουμε τον πληθάριθμο $|A \cap B|$ της τομής τους. Αυτό μπορεί να γίνει πολύ εύκολα, με ένα διπλό **for-loop** στα στοιχεία των A και *B*, πληρώνοντας όμως ένα πλήθος συγκρίσεων ανάλογο του γινομένου $|A| \cdot |B|$. Ο ίδιος υπολογισμός, βέβαια, μπορεί να γίνει πολύ πιο γρήγορα, σε πλήθος συγκρίσεων ανάλογο του αθροίσματος $|A| + |B|$, αν εκμεταλλευτούμε το γεγονός ότι μπορούμε να σαρώσουμε τα σύνολα ακεραίων στην PYTHON σε <u>αύξουσα σειρά</u> των στοιχείων τους, αν τα μετατρέψουμε πρώτα σε διατεταγμένες λίστες. Αυτό, μπορεί να γίνει όπως φαίνεται στο ακόλουθο παράδειγμα:

```
>>> myfrozenset1 = {1,2,3,-1,-2,-3}

>>> myfrozenset1

frozenset({1, 2, 3, -2, -3, -1})

>>> mylist1 = sorted(myfrozenset1)

>>> mylist1

[-3, -2, -1, 1, 2, 3]
```

<u>Απλός τρόπος υπολογισμού της τομής:</u> Καθώς τα σύνολα λέξεων που θα φτιάξετε για τα έγγραφα θα περιέχουν αναγνωριστικά λέξεων (δηλαδή, ακεραίους) που εμφανίζονται στο έγγραφο, οπότε μπορούμε (στα σύνολα) να κάνουμε μια συστηματική σάρωση των στοιχείων που περιλαμβάνουν, πχ, με την εντολή **for** $wordID$ **in** $wordsets[docID]$. Ο ψευδοκώδικας για τον υπολογισμό της τομής, λοιπόν, μπορεί να γίνει, πολύ απλά αλλά αρκετά χρονοβόρα (εκτελώντας τετραγωνικό πλήθος επαναλήψεων), με ένα διπλό FOR-LOOP που σαρώνει κατά ζεύγη ($wordID1$, $wordID2$) τα στοιχεία των συνόλων *A* και *B*, και για κάθε ζεύγος που απαρτίζεται από τα ίδια στοιχεία (δηλαδή, $wordID1 == wordID2$) αυξάνει τον μετρητή της τομής κατά 1.

<u>Αποδοτικός τρόπος υπολογισμού της τομής:</u> Μπορούμε να δημιουργήσουμε (προσωρινά, μόνο για τη σύγκριση) ταξινομημένες λίστες από τα σύνολα, με την εντολή sorted(myset) που αναφέραμε νωρίτερα. Ο χρόνος για τη δημιουργία των συγκεκριμένων ταξινομημένων λιστών από τα σύνολα είναι σχεδόν γραμμικός στο μέγεθος των συνόλων, και σίγουρα πολύ μικρότερος από τον τετραγωνικό χρόνο που απαιτεί ο προηγούμενος υπολογισμός.

Στη συνέχεια, συγκρίνουμε τα στοιχεία των δύο λιστών ως εξής: Συντηρούμε δείκτες σε συγκεκριμένα στοιχεία των δύο λιστών, τους οποίους εμείς θα καθορίζουμε, Δεδομένου ότι οι λίστες έχουν τα στοιχεία τους ταξινομημένα σε αύξουσα σειρά, ξεκινάμε από την αρχή τους δυο δείκτες, συγκρίνοντας κάθε φορά τα στοιχεία των δύο λιστών στις

αντίστοιχες θέσεις. Αν τα στοιχεία συμπίπτουν, αυξάνουμε τον μετρητή της τομής κατά 1 και αυξάνουμε επίσης κατά ένα και καθέναν από τους δυο δείκτες. Διαφορετικά, απλά αυξάνουμε κατά 1 τον δείκτη που υποδεικνύει το μικρότερο από τα δύο στοιχεία. Αυτό συνεχίζεται μέχρι τη στιγμή που κάποιος από τους δύο δείκτες να φτάσει στο τέλος της δικής του λίστας. Ο ψευδοκώδικας για τον συγκεκριμένο υπολογισμό της τομής είναι ο εξής (για τις ταξινομημένες λίστες L1 και L2):

```
…

pos1 =0; pos2 = 0; intersectionCounter = 0

len1 = len(L1)

len2 = len(L2)

while pos1 < len1 and pos2 < len2:

        if L1[pos1] == L2[pos2]:

                intersectionCounter += 1; pos1 += 1; pos2 += 1

        else:

                if L1[pos1] < L2[pos2]: pos1 += 1

                else: pos2 += 1

…
```

Στο ακόλουθο παράδειγμα εκτέλεσης, σε δύο τυχαία σύνολα ακεραίων από το σύμπαν $\{0,1,\ldots,99999\}$, με 20000 στοιχεία το καθένα, φαίνεται η διαφορά σε πλήθος επαναλήψεων στους βρόχους, αλλά και σε απόλυτους χρόνους εκτέλεσης, για τις δύο μεθόδους υπολογισμού της Jaccard ομοιότητας:

```
====================================
CREATION OF A PAIR OF RANDOM SETS
WITH  20000  ITEMS FROM {0,1,... 99999 }.
------------------------------------
Time for creation of random sets = 0.032124775776672363
====================================
COMPUTATION OF JSIM(A,B) BEGINS
====================================
FIRST METHOD: ORDER AND THEN COMPARE LISTS
------------------------------------
#comparisons     =   33042
Execution time   =   0.015772342681884766
Jaccard Sim.     =   0.09720078680587078
====================================
SECOND METHOD: COMPARE UNORDERED LISTS (SETS) VIA DOUBLE FOR-LOOP
------------------------------------
#comparisons     =   328639206
Execution time   =   35.346688747406006
Jaccard Sim.     =   0.09720078680587078
====================================
JSIM2_TIME / JSIM1_TIME  =   2241.055083592829
====================================
>>>
```

Όπως φαίνεται και στο παράδειγμα, η υλοποίηση με τις ταξινομημένες λίστες υπερτερεί κατά 3 τάξεις μεγέθους (στους χρόνους εκτέλεσης) της αφελούς μεθόδου υπολογισμού. Οι χρόνοι είναι σε δευτερόλεπτα, κι έχουν καταμετρηθεί με τη ρουτίνα βιβλιοθήκης time.time().

PROGRAMMING TASK PT2: Δημιουργήστε δυο ρουτίνες για τον υπολογισμό της Jaccard Similarity των συνόλων-λέξεων για δυο έγγραφα.

PT2a $MyJacSimWithSets(docID1, docID2)$, που συγκρίνει απευθείας δυο δομές τύπου frozenset (όπως δηλαδή δημιουργήθηκαν στο PROGRAMMING TASK 1) για τα έγγραφα αυτά, υπολογίζοντας την τομή με διπλό FOR-LOOP, κι επιστρέφει την τιμή $JacSim(docID1, docID2)$.

PT2b $MyJacSimWithOrderedLists(docID1, docID2)$, που χρησιμοποιεί δύο δείκτες για να συγκρίνει δυο ταξινομημένες λίστες για τα έγγραφα αυτά, προκειμένου να υπολογίσει την τομή και στη συνέχεια επιστρέφει την τιμή $JacSim(docID1, docID2)$.

(1δ) ΒΗΜΑ 3: Συναρτήσεις για Ομοιότητα Υπογραφών

Αυτή τη φορά θα πρέπει να φτιάξετε μια ρουτίνα που να υλοποιεί τη συνάρτηση $MinHash()$ για την κατασκευή του μητρώου (διανυσμάτων) υπογραφών, για όλα τα έγγραφα της συλλογής, χρησιμοποιώντας ένα συγκεκριμένο πλήθος $numPermutations$ από τυχαίες μεταθέσεις (δηλαδή, αναδιατάξεις) των W γραμμών στο μητρώο χαρακτηριστικών διανυσμάτων (που δε φτιάξατε). Θυμηθείτε ότι η τιμή W ισούται με το πλήθος διαφορετικών λέξεων που εμφανίζονται στα έγγραφα της συλλογής που επεξεργάζεστε, και είναι γνωστή (δείτε το README.TXT). Το ερώτημα, βέβαια, είναι πώς κανείς μπορεί να δημιουργήσει τυχαίες μεταθέσεις για το σύνολο $\{1,2,\dots,W\}$. Το σκεπτικό που προτείνεται να ακολουθήσετε είναι το εξής:

- Επιλέξτε εκ νέου μια τυχαία συνάρτηση κατακερματισμού, που αναθέτει για οποιονδήποτε ακέραιο αριθμό X την τιμή $\big((\alpha * X + \beta)\%P\big)\%M$ όπου P είναι ένας πολύ μεγάλος πρώτος αριθμός, (α, β) είναι ένα ζεύγος τυχαία επιλεγμένων αριθμών από το $\{1,2,\dots,P-1\}$, και M είναι ένας αριθμός που εξασφαλίζει ότι δεν παραβιάζετε την ακρίβεια του υπολογιστή σας στις αναπαραστάσεις ακεραίων αριθμών. Για παράδειγμα, μπορείτε να κάνετε χρήση του ακόλουθου κώδικα (δίνεται, για άλλον λόγο, και στη σελίδα 47 των διαφανειών για ομοιότητα εγγράφων):

```
import random

def create_random_hash_function(p=2**33-355, m=2**32-1):
      a = random.randint(1,p-1)
      b = random.randint(0, p-1)
      return lambda x: 1 + (((a * x + b) % p) % m)
```

  Η κλήση $h = create\_random\_hash\_function()$ δημιουργεί αυτή τη νέα (τυχαία) συνάρτηση κατακερματισμού που χρειαζόμαστε.
- Δημιουργήστε ένα λεξικό $randomHash$, με ακριβώς W ζεύγη κλειδιού-τιμής, της μορφής $X: h(X)$. Για παράδειγμα: $randomHash = \{ x: h(x)$ **for** x **in** $range(W) \}$. Το αποτέλεσμα αυτής της ενέργειας θα έχει την εξής μορφή (πχ, για $W = 5$ γραμμές):

```
>>> h = create_random_hash_function()
>>> randomHash = { x:h(x) for x in range(5) }
>>> randomHash
{0: 77533261, 1: 24441289, 2: 4266316612, 3: 4213224640, 4: 4160132668}
```

- Ταξινομήστε τα ζεύγη από το λεξικό $randomHash$ σε ένα νέο λεξικό (πχ, $myHash$) με βάση τις τιμές (αντί για τα κλειδιά) των ζευγών, δίνοντας όμως τώρα νέες τιμές στα ζεύγη, ανάλογα με τη θέση του κάθε ζεύγους κλειδιού-τιμής μετά την ταξινόμηση του $randomHash$. Για παράδειγμα, το ζεύγος 0:77533261 στο $randomHash$ θα πρέπει πλέον να γίνει 0:1 στο $myHash$, γιατί το κλειδί 0 έχει τη δεύτερη μικρότερη τιμή στο $randomHash$. Αντίστοιχα, το ζεύγος 1:24441289 του $randomHash$ θα πρέπει τώρα να γίνει 1:0 στο $myHash$, αφού το κλειδί 1 έχει τη μικρότερη τιμή στο $randomHash$. Επίσης το ζεύγος 2:4266316612 στο

$randomHash$ θα γίνει 2:4 στο $myHash$ αφού το κλειδί 2 έχει τη μέγιστη τιμή εντός του $randomHash$. Μπορείτε να το πετύχετε αυτό όπως δείχνει ο ακόλουθος κώδικας:

```
>>> myHashKeysOrderedByValues = sorted(randomHash, key=randomHash.get)
>>> myHash = { myHashKeysOrderedByValues[x]:x for x in range(5) }
>>> myHash
{1: 0, 0: 1, 4: 2, 3: 3, 2: 4}
```

Έχοντας πλέον τρόπο να δημιουργήσετε λεξικά που αντιστοιχούν σε τυχαίες μεταθέσεις του συνόλου $\{1,2,\dots,W\}$, είστε σε θέση να υλοποιήσετε τη συνάρτηση $MyMinHash()$ που δημιουργεί το μητρώο υπογραφών για όλα τα έγγραφα (για την ακρίβεια, για τα έγγραφα από το 1 μέχρι το $numDocuments$) στη συλλογή σας. Ο συγκεκριμένος αλγόριθμος έχει περιγραφεί με λεπτομέρεια στις διαλέξεις του μαθήματος, μελετήστε προσεκτικά τις διαφάνειες 39-40 στη διάλεξη για «Ομοιότητα Εγγράφων», καθώς και τις ενότητες 3.3.5-3.3.7 του βιβλίου.

ΠΡΟΣΟΧΗ: Για τη συγκεκριμένη υλοποίηση του αλγορίθμου MinHash, απαιτείται η σάρωση του μητρώου με τα χαρακτηριστικά διανύσματα (συλλογή από 0/1-στήλες για όλα τα έγγραφα), που δέχεται ως είσοδο ο αλγόριθμος MinHash, κατά γραμμές. Η λίστα με τα frozensets, που έχετε ήδη αποθηκευμένα κατά την ανάγνωση των δεδομένων στο ΒΗΜΑ 1, σας δίνει τη δυνατότητα της σάρωσης του συγκεκριμένου μητρώου κατά στήλες. Σαφώς είναι πάρα πολύ αραιό το μητρώο αυτό, οπότε δε συμφέρει να το δημιουργήσει κανείς ώστε να μπορεί να το σαρώνει και κατά γραμμές, γιατί θα σπαταλήσει δίχως λόγο πάρα πολύ χώρο. Προτείνεται να κάνετε χρήση μιας λίστας από λίστες-ακεραίων (list of lists) για την αναπαράσταση του 0/1-μητρώου χαρακτηριστικών διανυσμάτων κατά γραμμές, ώστε να είναι εύκολη η σάρωσή του και κατά γραμμές (δηλαδή, ανά λέξη), όπως ακριβώς απαιτεί ο αλγόριθμος MinHash:

- Η λίστα-λιστών $wordsList$ θα αποθηκεύει, για κάθε αναγνωριστικό λέξης $wordID$, τη λίστα ακεραίων $wordsList[wordID-1]$ που διατηρεί όλα τα αναγνωριστικά $docID$ εγγράφων τα οποία περιέχουν τη λέξη $wordID$.

ΠΡΟΣΟΧΗ: Τα αναγνωριστικά λέξεων (όπως και τα αναγνωριστικά εγγράφων) είναι θετικοί ακέραιοι αριθμοί, από το 1 μέχρι το W. Οι αντίστοιχες λίστες λιστών ξεκινούν από τη λίστα 0 μέχρι τη λίστα $W-1$.

PROGRAMMING TASK PT3: Δημιουργήστε ρουτίνα $MyMinHash()$ που δέχεται ως είσοδο τη λίστα με τα σύνολα λέξεων για όλα τα έγγραφα (δεικτοδοτούμενη με $docID-1$ τιμές, για όλα τα έγγραφα $docID \in \{1,\dots,numDocuments\}$), και έναν ακέραιο αριθμό K (το μέγιστο πλήθος τυχαίων μεταθέσεων που θα χρησιμοποιήσουμε). Η ρουτίνα αυτή αρχικά δημιουργεί μια λίστα-λιστών $wordsList$, με τη λίστα $wordlist[wordID-1]$ να περιλαμβάνει όλα τα αναγνωριστικά $docID$ των εγγράφων που εμφανίζουν τη λέξη $wordID$. Στη συνέχεια, παράγει το $K \times D$ μητρώο υπογραφών SIG, όπου για κάθε $d \in \{0,1,\dots,numDocuments-1\}$, η στήλη d του SIG δίνει το διάνυσμα υπογραφών του εγγράφου με αναγνωριστικό $docID = d+1$.

ΣΗΜΕΙΩΣΗ: Προτείνεται οι λίστες που θα δημιουργήσετε να εμφανίζουν, επίσης, τα εμπλεκόμενα αναγνωριστικά εγγράφων (docIDs) σε αύξουσα σειρά. Αυτό επιτυγχάνεται εύκολα αν δημιουργήσετε όλες τις λίστες $wordLists[wordID-1]$ παράλληλα, σαρώνοντας τη λίστα με όλα τα frozensets εγγράφων, $documentsList[docID-1]$ για το έγγραφο $docID$, κατ' αύξουσα σειρά, και κάνοντας τις κατάλληλες προσθήκες του εκάστοτε $docID$ σε όλες τις εμπλεκόμενες λίστες $\mathrm{wordLists}[\mathrm{wordID}-1]$, ανάλογα με το ποια αναγνωριστικά-λέξεων ($wordID$ τιμές) περιλαμβάνονται στο $documentsList[docID-1]$. Ο απαιτούμενος χώρος, συνολικά για τα frozensets των εγγράφων και τις λίστες των λέξεων, είναι διπλάσιος του πλήθους των άσσων στο 0/1-μητρώο χαρακτηριστικών διανυσμάτων που θα έδινε η φάση της σφυρηλάτησης ως είσοδο για το $MinHash$.

(1ε) ΒΗΜΑ 4: Ομοιότητα Υπογραφών

Έχοντας ήδη δημιουργήσει το μητρώο υπογραφών μέσω της $MyMinHash()$, θα πρέπει να γράψετε κώδικα για τον υπολογισμό της ομοιότητας-υπογραφών μεταξύ δυο συγκεκριμένων (αναγνωριστικών) εγγράφων. Η ρουτίνα σας θα δέχεται ως είσοδο και έναν θετικό ακέραιο αριθμό $numPermutations$, που ορίζει ότι θα πρέπει να ληφθούν υπόψη κατά τον υπολογισμό ΜΟΝΟ οι υπογραφές που αντιστοιχούν τις πρώτες $numPermutations$ μεταθέσεις (δηλαδή, γραμμές) από το μητρώο υπογραφών, για κάποιο $1 \leq numPermutations \leq K$. Αν η υλοποίηση του $MyMinHash()$ και του $MySigSim(docID1, docID2, numPermutations)$ είναι σωστές, τότε θα πρέπει, καθώς μεγαλώνει η τιμή του $numPermutations$, η τιμή του $MySigSim(docID1, docID2)$ να συγκλίνει στην τιμή του $MyJacSim(docID1, docID2)$.

PROGRAMMING TASK PT4: Δημιουργήστε ρουτίνα $MySigSim(docID1, docID2, numPermutations)$ που δέχεται ως τα αναγνωριστικά για δύο έγγραφα (αντιστοιχούν στις στήλες $docID1 - 1$ και $docID2 - 1$ των μητρώων υπογραφών) και να επιστρέφει την ομοιότητα υπογραφών όπως αυτή προσδιορίζεται ΜΟΝΟ από τις πρώτες $numPermutations$ γραμμές του μητρώου SIG.

(1ζ) ΒΗΜΑ 5: Υπολογισμός Πλησιέστερων Γειτόνων ανά Έγγραφο

Στόχος πλέον είναι ο υπολογισμός των $NumNeighbors$ πλησιέστερων γειτόνων, για συγκεκριμένο έγγραφο $docID \in \{1,2,\dots,numDocuments\}$ (που περιγράφεται είτε από το σύνολο στη θέση $docID - 1$ στη λίστα συνόλων που δημιουργήσατε, είτε από τη στήλη $docID - 1$ του μητρώου υπογραφών SIG που κατασκευάσατε με τη $MyMinHash()$. Θα υλοποιήσετε δυο διαφορετικές μεθόδους, τη μέθοδο της ωμής βίας και τη μέθοδο του LSH.

(1ζι) Μέθοδος ωμής βίας

Για κάθε έγγραφο $docID$, θα πρέπει να κάνετε τα εξής:

Υπολογίστε την ομοιότητα του συγκεκριμένου $docID$ με κάθε άλλο έγγραφο στη συλλογή. Θεωρήστε δυο διαφορετικές εκδοχές, ανάλογα με το ποια μετρική ομοιότητας χρησιμοποιείτε ($JacSim$ ή $SigSim$).

- Καταγράψτε τους χρόνους υπολογισμού ομοιότητας και αποθηκεύστε όλες τις αποστάσεις άλλων εγγράφων από το $docID$ σε ένα λεξικό που δεικτοδοτείται με κλειδιά τα αναγνωριστικά εγγράφων και τιμές τις αποστάσεις αυτών των εγγράφων από το $docID$. Θυμηθείτε ότι $Dist(d, docID) = 1 - Sim(d, docID)$.
- Ταξινομήστε το λεξικό αυτό ως προς τις τιμές απόστασης.
- Για το έγγραφο $docID$, αποθηκεύστε τα πρώτα $NumNeighbors$ (άλλα) έγγραφα του (ταξινομημένου πλέον) λεξικού αποστάσεων, σε ένα λεξικό $myNeighborsDict[docID]$ με ζεύγη κλειδιού τιμής της μορφής $d: Sim(docID, d)$.

Προκειμένου να υπολογίσουμε ένα απλό μέτρο εγγύτητας του συγκεκριμένου εγγράφου $docID$ με τους $numNeighbors$ γείτονές του, υπολογίζουμε τον **μέσο όρο ομοιότητας** (ας το ονομάσουμε $AvgSim[docID]$) των γειτόνων αυτών από το $docID$.

Τελικά, για να προκύψει ένας συγκεκριμένος «μέσος βαθμός ομοιότητας με τους κοντινότερους γείτονες» για ολόκληρο το σύνολο δεδομένων που επεξεργάζεστε, υπολογίστε τον μέσο όρο των μέσων όρων εγγύτητας:

$$AvgSim = \frac{1}{numDocuments} \cdot \sum_{d=1}^{numDocuments} AvgSim[d]$$

ΠΡΟΣΟΧΗ: Είναι να κάνετε τους υπολογισμούς σας, όχι σε ολόκληρο το σύνολο δεδομένων, αλλά μόνο στα $numDocuments$ πρώτα έγγραφα, καθώς οι συγκεκριμένοι υπολογισμοί είναι ιδιαίτερα χρονοβόροι και πιθανότητα ο υπολογιστής σας θα αργήσει να τους ολοκληρώσει.

PROGRAMMING TASK PT5: Δημιουργήστε τη **μέθοδο της ωμής βίας**, για τον υπολογισμό του **μέσου όρου ομοιότητας** $AvgSim$ ενός τυπικού εγγράφου από τους κοντινότερους γείτονές του, για τα πρώτα $numDocuments$ έγγραφα από τη συλλογή που επεξεργάζεστε.

(1ζιι) Μέθοδος LSH

Σε αυτό το τελικό στάδιο, θα πρέπει να υλοποιήσετε τη μέθοδο LSH, η οποία περιγράφεται στην ενότητα 3.4.1 του βιβλίου. Η ρουτίνα σας θα πρέπει να δέχεται ως είσοδο το (ήδη κατασκευασμένο από τη Min-Hashing τεχνική) μητρώο υπογραφών SIG, καθώς και τον αριθμό $rowsPerBands$ που δηλώνει το πλήθος r των γραμμών που περιλαμβάνονται σε κάθε μπάντα (με εξαίρεση ίσως την τελευταία μπάντα, που περιλαμβάνει τουλάχιστον r και το πολύ $2r - 1$ γραμμές). Το πλήθος $numBands$ των μπαντών που επιθυμούμε να δημιουργήσουμε είναι εύκολο να υπολογιστεί: Θεωρήστε ότι έχετε $numBands = \left\lfloor \frac{K}{rowsPerBands} \right\rfloor$.

Προτείνεται για την επεξεργασία κάθε μπάντας $currentBand$ να κάνετε χρήση ενός λεξικού, έστω $LSHdicts[currentBand]$, όπου θα τοποθετείτε ζεύγη της μορφής $docID : hashLSH(SIG[bandRows : docID])$. Δηλαδή, θα πρέπει ως κλειδί να δίνεται το αναγνωριστικό του εγγράφου, και ως τιμή να δίνεται ο κάδος που καθορίζει η (τυχαία, αλλά κοινή για όλες τις μπάντες) συνάρτηση κατακερματισμού $hashLSH$ που θα δημιουργήσετε καλώντας (μια φορά, για όλες τις μπάντες, δε χρειάζεται να αλλάζει) τη ρουτίνα `create_random_hash_function` που περιγράφηκε νωρίτερα.

Στη συνέχεια, για τη συγκεκριμένη μπάντα $currentBand$, θα πρέπει να ταξινομήσετε το λεξικό $LSHdicts[currentBand]$ με βάση τις τιμές. Για τη λειτουργία αυτή μπορείτε να κάνετε χρήση του κώδικα που φαίνεται στο ακόλουθο παράδειγμα:

```
>>> myDict
{0: 1, 1: 0, 2: 4, 3: 3, 4: 2}
>>> newDict = {k: v for k, v in sorted(myDict.items(), key=lambda item: item[1])}
>>> newDict
{1: 0, 0: 1, 4: 2, 3: 3, 2: 4}
```

Τέλος, με ένα σάρωμα του λεξικού $LSHdicts[currentBand]$ μπορείτε να εντοπίσετε όλα τα ζεύγη (διαδοχικών πλέον στο λεξικό, μετά την ταξινόμηση) αναγνωριστικών εγγράφων που έχουν κοινή τιμή, δηλαδή ανατέθηκαν στον ίδιο κάδο. Τέτοιου είδους ζεύγη θα είναι υποψήφια για μέτρηση ομοιότητας, στην επόμενη φάση του αλγορίθμου.

Προτείνεται να δημιουργήσετε λίστα με τα κλειδιά του λεξικού ταξινομημένα σύμφωνα με τις τιμές τους, και στη συνέχεια να κάνετε σε αυτή τη σάρωση στη λίστα, ώστε να μπορείτε να δεικτοδοτείτε τόσο την τρέχουσα θέση όσο και την επόμενή της στη λίστα (αυτό δεν είναι εφικτό στο λεξικό, οπότε θα ήσασταν υποχρεωμένοι να κάνετε χρήση διπλού for-loop για την εύρεση υποψηφίων ζευγών στη συγκεκριμένη μπάντα, ενώ ένα απλό for-loop στη λίστα μπορεί να πετύχει ακριβώς το ίδιο αποτέλεσμα).

ΣΗΜΕΙΩΣΗ: Προκειμένου να κατακερματίσετε ένα διάνυσμα από ακεραίους σε έναν ακέραιο αριθμό (τον οποίο στη συνέχεια θα αναθέσετε σε κάδο σύμφωνα με την $hashLSH$), μπορείτε να χρησιμοποιήσετε τη συνάρτηση βιβλιοθήκης $hash$ της PYTHON, που δέχεται ως είσοδο **tuples** (όχι όμως λίστες) κι επιστρέφει ακεραίους αριθμούς. Δείτε για παράδειγμα τον ακόλουθο κώδικα:

```
>>> t1 = (4,0,1,2,3)
>>> t2 = (0,1,2,3,4)
```

```
>>> t3 = (0,1,3,2,4)
>>> hash(t1)
-7816828038797482742
>>> hash(t2)
891674017800185211
>>> hash(t3)
7393255412568654894
>>> hash(t1) == hash(t2)
False
>>> hash(t1) == hash(t3)
False
>>> hash(t2) == hash(t3)
False
```

Σαρώνοντας τα λεξικά όλων των μπαντών, εντοπίζουμε όλα τα υποψήφια ζεύγη για τα οποία πρέπει να ελέγξουμε ομοιότητα (όσα είναι μη-υποψήφια ζεύγη, θα έχουν πολύ μικρή ομοιότητα μεταξύ τους και γι' αυτό δεν ασχολούμαστε).

ΠΡΟΣΟΧΗ: Η κατάλληλη επιλογή του κατωφλίου s που θα επιλέξετε για την LSH επηρεάζει σαφώς τους κοντινότερους γείτονες που μπορούμε να εντοπίσουμε. Πχ, για $s = 0$ θα θεωρήσουμε ως υποψήφια όλα τα ζεύγη εγγράφων, όπως ακριβώς και στη μέθοδο της ωμής βίας, ενώ για πολύ μεγάλο s πρακτικά θα απορριφθούν τα περισσότερα ζεύγη και κινδυνεύουμε να μην υπάρχουν εντός των υποψηφίων ζευγών επαρκές πλήθος υποψηφίων ζευγών για κάθε έγγραφο (θα έπρεπε να έχουμε τουλάχιστον $numNeighbors$ υποψήφια ζεύγη που περιλαμβάνουν ένα συγκεκριμένο έγγραφο, για καθένα από τα πρώτα $numDocuments$ έγγραφα). Βέβαια, η τιμή του s αυτή τη φορά καθορίζεται από το $numBands$ και από το $rowsPerBand$ (θυμηθείτε: $s \sim (1/b)^{(1/r)}$ για να μη δημιουργεί πολλά false-postives ή false-negatives ο LSH. Άρα, ουσιαστικά θα πρέπει να παίξετε με τη μοναδική παράμετρο που έχετε στη διάθεσή σας, το $rowsPerBand$, και να ορίσετε τις τιμές των $numBands$ και s όπως προαναφέρθηκε. Προτείνεται να θεωρήσετε την επιλογή του K (γραμμές του SIG) ως δύναμης του 2, να ξεκινήσετε με $rowsPerBand = \frac{K}{4}$, $numBands = 4$, και άρα $s = (1/4)^{(4/K)}$. Αν για κάθε έγγραφο $docID$ προκύπτουν τουλάχιστον $numNeighbors$ υποψήφια ζεύγη που το περιλαμβάνουν, τότε είστε εντάξει. Διαφορετικά, υποδιπλασιάζετε το $rowsPerBand$ κι επαναλαμβάνετε.

Τέλος, (μόνο) για τα υποψήφια ζεύγη που ανέδειξε η LSH, μετρήστε την ομοιότητα (κατά Jaccard-Similarity, ή κατά Signature-Similarity) μεταξύ τους και υπολογίστε τους $numNeighbors$ πλησιέστερους γείτονες για κάθε έγγραφο $docID$, όπως ακριβώς κάνατε και με τη μέθοδο της ωμής βίας.

PROGRAMMING TASK PT6: Δημιουργήστε τη **μέθοδο LSH**, για την ανάδειξη (επαρκούς πλήθους) υποψηφίων ζευγών εγγράφων. Υπολογίστε τις ομοιότητες (και αποστάσεις) μόνο για τα υποψήφια ζεύγη εγγράφων που ανέδειξε η LSH. Υπολογίστε τον **μέσο όρο ομοιότητας** $AvgSim$ ενός τυπικού εγγράφου από τους κοντινότερους γείτονές του, για τα πρώτα $numDocuments$ έγγραφα από τη συλλογή που επεξεργάζεστε, όπως ακριβώς κάνατε και στη μέθοδο της ωμής βίας.

2. ΠΕΙΡΑΜΑΤΙΚΗ ΑΞΙΟΛΟΓΗΣΗ

Για να είναι εφικτή η αξιολόγηση της εργασίας σας, στο πρόγραμμα σε PYTHON (έκδοση >= 3.9.4) που θα παραδώσετε θα πρέπει να κάνετε τα εξής:

1. Χρησιμοποιήστε (στην αρχή του κώδικα) σχόλια για να προσδιορίστε επακριβώς τον τρόπο εκτέλεσης του προγράμματός σας, και τη θέση στην οποία (θεωρείτε ότι) βρίσκονται τα δεδομένα. Ιδανικά, καλό θα ήταν να χρησιμοποιείτε τον ίδιο κατάλογο όπου βρίσκεται ο κώδικας και για την αποθήκευση του αρχείου δεδομένων προς επεξεργασία.

2. Ζητήστε από τον χρήστη:
    a. Να εισάγει το όνομα του αρχείου δεδομένων (ή, να επιλέξει από ένα μενού), καθώς και το πλήθος $numDocuments$ των εγγράφων που θα λάβετε υπόψη σας και θα διαβάσετε από το συγκεκριμένο αρχείο (τα υπόλοιπα έγγραφα του αρχείου απλά αγνοούνται στη συγκεκριμένη εκτέλεση).
    b. Να εισάγει το επιθυμητό πλήθος $numNeighbors$ από κοντινότερους γείτονες που θα πρέπει να εντοπιστούν, ανά έγγραφο. Η συγκεκριμένη τιμή θα πρέπει να είναι κάποιος μικρός ακέραιος αριθμός (πχ, 2,3,4 ή 5).
    c. Να προσδιορίσει το πλήθος K των τυχαίων μεταθέσεων που θα λάβετε υπόψη σας, για την κατασκευή του μητρώου υπογραφών (η κατασκευή αυτή γίνεται μία φορά, δε χρειάζεται να επαναληφθεί).
    d. Να προσδιορίσει την επιθυμητή μετρική ομοιότητας προς χρήση ($JacSim$, ή $SigSim$).
    e. Να προσδιορίσει αν επιθυμεί να γίνει χρήση της μεθόδου ωμής βίας, ή της μεθόδου LSH προκειμένου να υπολογίσουμε ομοιότητες μεταξύ (όλων για την ωμή βία, μόνο των υποψηφίων για την LSH) των ζευγών εγγράφων.
    f. Εφόσον επιλεγεί η μέθοδος LSH, να προσδιοριστεί και το επιθυμητό πλήθος $rowsPerBand$ των γραμμών ανά μπάντα, για να χρησιμοποιηθεί κατά τον LSH (θυμηθείτε: ενδεχομένως η συγκεκριμένη τιμή να μην εξασφαλίζει τον ελάχιστο αριθμό υποψηφίων ζευγών ανά έγγραφο, οπότε ειδοποιείτε τον χρήστη. Δώστε την επιλογή να αφήνεται στον LSH να διαλέξει το κατάλληλο πλήθος γραμμών ανά μπάντα, όπως εξηγήθηκε στην ενότητα (1ζιι).
3. Δώστε στον χρήστη την επιλογή, αν θέλει, να εισάγει τα αναγνωριστικά $docID1$ και $docID2$ δύο εγγράφων (και του $numPermutations$), προκειμένου να γίνει υπολογισμός της $JacSim(docID1, docID2)$ (και της εκτίμησης $SigSim(docID1, docID2)$ αξιοποιώντας μόνο τις πρώτες $numPermutations$ γραμμές του μητρώου SIG), για λόγους ελέγχου ορθότητας. Θυμηθείτε ότι τα έγγραφα αριθμούνται στο αρχείο δεδομένων με αναγνωριστικά από το $\{1,2,\dots,D\}$. Σε περίπτωση που επιλεγεί η χρήση της Jaccard ομοιότητας, θα πρέπει να γίνει (για τα ίδια έγγραφα) υπολογισμός της Jaccard ομοιότητας και με τους δύο τρόπους υπολογισμού, καταγράφοντας για καθέναν από τους δύο τρόπους και τον χρόνο εκτέλεσης, για λόγους σύγκρισης των δύο μεθόδων.
4. Υπολογίστε τους πλησιέστερους γείτονες για κάθε έγγραφο, καταγράφοντας τον χρόνο υπολογισμού $ExecTime$ (προτείνεται να κάνετε χρήση της βιβλιοθήκης time της PYTHON, που μετρά χρόνους σε δευτερόλεπτα):
    a. Υπολογίστε όλες τις αποστάσεις εγγράφων (προς όλα τα άλλα έγγραφα, ή προς υποψήφιους κοντινούς γείτονες, ανάλογα με την επιλογή που έγινε στο 2e), και σύμφωνα με τη μετρική ομοιότητας που επιλέχθηκε στο 2d.
    b. Προσδιορίστε τους $numNeighbors$ κοντινότερους γείτονες ανά έγγραφο, και υπολογίστε τη μετρική $AvgSim$.
    c. Δώστε την επιλογή στον χρήστη να ζητά τους κοντινότερους γείτονες για συγκεκριμένα έγγραφα (βάσει των docID αναγνωριστικών τους).
5. Τυπώστε τις τιμές των $AvgSim$ και $ExecTime$, αναφέροντας και τις συγκεκριμένες επιλογές και παραμετροποιήσεις που έγιναν από τον χρήστη, καθώς και την «ιδανική» επιλογή του $rowsPerBand$ που χρησιμοποιήθηκε από τον κώδικά σας στον LSH (εφόσον αυτή καθορίστηκε από τον αλγόριθμο κι όχι από τον χρήστη).

3. ΠΑΡΑΔΟΣΗ ΕΡΓΑΣΙΑΣ

Δημιουργήστε τα εξής αρχεία:

(α) ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ: Ένα συμπιεσμένο (ZIP) αρχείο, που να περιέχει όλα τα αρχεία του πηγαίου κώδικα που δημιουργήσατε για τις ανάγκες της εργασίας σας. Χρησιμοποιήστε την εξής ονοματολογία για το ZIP αρχείο που θα παραδώσετε: **MYE047_LAB1_<YOUR AM>_SOURCE-FILES.zip**.

Επειδή μια MS FORM δεν επιτρέπει τη μεταφόρτωση (upload) ενός ZIP αρχείου, μετονομάστε το αρχείο ZIP που δημιουργήσατε ως εξής (ώστε να είναι εφικτή η μεταφόρτωσή του στη φόρμα υποβολής): **MYE047_LAB1_<YOUR AM>_SOURCE-FILES_zip.pptx**

ΠΡΟΣΟΧΗ: Δε χρειάζεται να συμπεριλάβετε τα (ούτως ή άλλως πολύ μεγάλα για να χωρέσουν) αρχεία δεδομένων της εργασίας. Ζητείται ΜΟΝΟ ο δικός σας κώδικας.

(β) ΠΕΙΡΑΜΑΤΙΚΕΣ ΜΕΤΡΗΣΕΙΣ: Ένα αρχείο XLSX όπου θα περιλαμβάνονται (κατ' ελάχιστον) τα πειραματικά δεδομένα που πήρατε από τις μετρήσεις σας, για 8 διαφορετικά πειράματα (ανάλογα με τις επιλογές που δώσατε). Χρησιμοποιήστε την εξής ονοματολογία, για το αρχείο πειραματικών μετρήσεων που θα παραδώσετε: **MYE047_LAB1_<YOUR AM>_EXPERIMENTAL-EVALUATION.xlsx**

(γ) ΓΡΑΠΤΗ ΑΝΑΦΟΡΑ: Τη γραπτή αναφορά σας (σε DOCX ή/και σε PDF μορφή) για την εργαστηριακής άσκηση MYE047/LAB-1. Χρησιμοποιήστε την εξής ονοματολογία, για το αρχείο πειραματικών μετρήσεων που θα παραδώσετε: **MYE047_LAB1_<YOUR AM>_REPORT.docx** ή **MYE047_LAB1_<YOUR AM>_REPORT.pdf**

Στη γραπτή αναφορά σας θα περιγράφετε αναλυτικά τις μεθόδους που υλοποιήσατε και, κυρίως, θα παραθέτετε όλα τα πειραματικά αποτελέσματα συνοδευόμενα με τον κατάλληλο σχολιασμό. Θα αξιολογηθεί, πέρα από την πληρότητα και την ορθότητα των πειραματικών μετρήσεων, και η επαρκής εξήγηση των μετρήσεων που πήρατε. Π.χ., είναι τα πράγματα όπως τα περιμένατε? Εμφανίστηκε κάποια απροσδόκητη μέτρηση, κι αν ναι, πού μπορεί να οφείλεται αυτό?

Θα πρέπει ρητά να σχολιάσετε τις επιδόσεις του LSH σε σχέση με την ωμή βία, ως προς την ευρεθείσα μέση ομοιότητα από γείτονες που βρήκατε. Προσέξτε ότι υπάρχουν 4 διαφορετικοί συνδυασμοί (LSH ή brute force? $JacSim$ ή $SigSim$?) που πρέπει να λάβετε υπόψη σας και να σχολιάσετε. Είναι ιδιαίτερα χρήσιμη και η παράθεση και διαγραμμάτων για την επίδοση (ως σημεία ($AvgSim$, $ExecTime$) για διαφορετικές τεχνικές και για διαφορετικές τιμές των παραμέτρων, προκειμένου να αναδειχθεί η ευαισθησία κάθε συνδυασμού τεχνικών στην επιλογή τιμών για συγκεκριμένες παραμέτρους. Και πάλι κρίνεται ιδιαίτερα σημαντικός και ο σχολιασμός των διαγραμμάτων, κι όχι η απλή παράθεσή τους.

Η παράδοση της εργασίας θα γίνει μέσα από τη φόρμα υποβολής με το όνομα

2021-22_CSE/MYE047 : : LAB-1 SUBMISSION FORM…

που θα βρείτε αναρτημένη στο κανάλι LAB-1 στην MS-TEAMS ομάδα του μαθήματος.

Υπενθυμίζεται ότι όλα τα αρχεία για κάθε σύνολο δεδομένων (για την πρώτη συλλογή, με το πρόθεμα «DATA_1-» στο όνομά τους, και για τη δεύτερη συλλογή, με πρόθεμα «DATA_2-» στο όνομά τους), καθώς και ένα πρότυπο αρχείο XLSX για την καταγραφή πειραματικών αποτελεσμάτων, θα τα βρείτε στην ετικέτα FILES του καναλιού LAB-1 στην MS-TEAMS ομάδα του μαθήματος (2021-22_CSE.UOI_MYE047 -- Algorithms for Big Data).