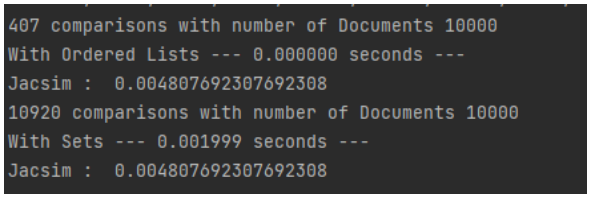
ΑΝΑΦΟΡΑ ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ

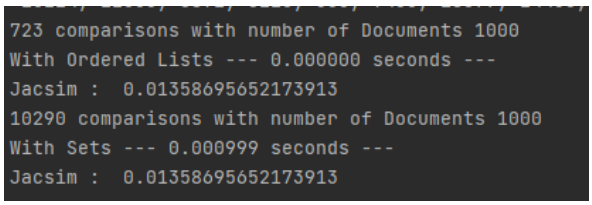
ΑΛΓΟΡΙΘΜΟΙ ΓΙΑ ΔΕΔΟΜΕΝΑ ΕΥΡΕΙΑΣ ΚΛΙΜΑΚΑΣ

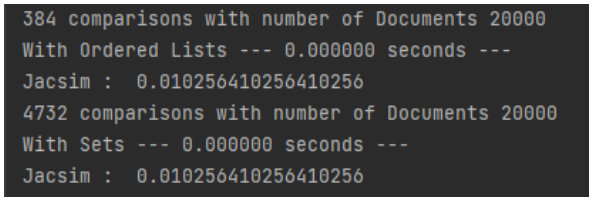
Ντόντης Βασίλειος : 3300

Ταλαμάγκας Ζήσης-Προκόπιος : 3340

Η πρώτη μέθοδος που θα χρησιμοποιήσουμε είναι η JaccardSimularity η οποία παίρνει 2 δομές, στην περίπτωση μας 2 frozenset, όπως είναι και τα συγκρίνει για να βρει τις ομοιότητες τους. Στην πραγματικότητα αυτό που κάνει είναι ότι εντοπίζει από 2 σύνολα την τομή ($|A \cap B|$) τους και διαιρώντας την με την ένωση τους ($|A|+|B|-|A \cap B|$) . Αυτός ο τρόπος όμως δεν είναι όσο αποδοτικός θα θέλαμε. Αυτό μπορούμε εύκολα να διορθώσουμε αν αντί για 2 τυχαία frozenset δημιουργώντας ταξινομημένες λίστες (μικρό κόστος σε χρόνο μιας και ο χρόνος δημιουργίας τους είναι σχεδόν γραμμικός). Συγκρίνοντας έτσι τα στοιχεία των λιστών στις αντίστοιχες θέσεις αυξάνοντας τον μετρητή για τις ομοιότητες όταν βρίσκονται κοινά στοιχεία. Όταν ο μετρητής μιας εκ των δυο λιστών τελειώσει υπολογίζουμε την ομοιότητα διαιρώντας τον μετρητή των κοινών σημείων με το άθροισμα των μετρητών των λιστών μείον τον μετρητή κοινών σημείων (Jacksim = intersectionCounter / L1 + L2- intersectionCounter). Στην πρώτη περίπτωση βρίσκαμε την jaccardsimularity με frozenset ενώ στην δεύτερη με ταξινομημένες λίστες (ordered lists). Από κάτω βλέπετε τις διαφορές στους χρόνους τους σε μερικά πειράματα καθώς και το πλήθος των συγκρίσεων που έκανε η κάθε μια. Το αποτέλεσμα είναι φανερά αυτό που περιμέναμε καθώς οι συγκρίσεις που γίνονται με ταξινομημένες λίστες είναι κατά πολύ μικρότερες από αυτές που γίνονται με sets. Μπορεί να είναι λίγο περίεργο το αποτέλεσμα των συγκρίσεων στα 1000 αρχεία σε σχέση με την σύγκριση στα 10000 αλλά αυτό οφείλεται καθαρά στην τυχαιότητα των αρχείων.







Η επόμενη μέθοδος που θα υλοποιήσουμε είναι η MinHash. Ο συγκεκριμένος αλγόριθμος χρησιμοποιεί μια τυχαία συνάρτηση κατακερματισμού που περιέχει όμως αριθμούς (τυχαίους) που τείνουν στο άπειρο. Αρχικοποιώντας τώρα την συνάρτηση κατακερματισμού σε ένα λεξικό όχι με βάση την τιμή τους αλλα την θέση τους ήμαστε έτοιμοι να δημιουργήσουμε το μητρώο υπογραφών για όλα τα μοναδικά έγγραφα. Αρχικά θα δημιουργήσουμε το μητρώο εγγραφών (λίστα λιστών), με στήλες τον αριθμό των μοναδικών εγγράγων και γραμμές τον αριθμό των μοναδικών λέξεων. Αρχικοποιούμε κάθε θέση ως ‘0’ και αφού ελέγξουμε ποια λέξη υπάρχει και σε ποιο έγγραφο, κάνουμε την θέση αυτή ‘1’. Στην συνέχεια θα δημιουργήσουμε και το μητρώο υπογραφών με πλήθος γραμμών τις μεταθέσεις όπου θα μας τις δώσει ο χρήστης και πλήθος στηλών οσα και τα μοναδικά έγγραφα στο οποίο οι τιμές του θα είναι αρχικοποιημένες στο «άπειρο»(numpy.inf). Τρέχοντας τώρα το μητρώο εγγράφων, σε οποιοδήποτε σημείο εντοπίζουμε ‘1’ πάμε και τσεκάρουμε τις(ην) αντίστοιχες(η) θέσεις (ίδια γραμμή με τη θέση που έχει ‘1’) στις μεταθέσεις του και αν η τιμή στην θέση των μεταθέσεων είναι μικρότερη από αυτήν στον πίνακα υπογραφών (με γραμμή τον αριθμό της μετάθεσης και στήλη τον αριθμό που είχε και το ‘1’ στο μητρώι εγγραφών) τότε ανανεώνουμε τον πίνακα υπογραφών, στην αντίστοιχη θέση, με την τιμή του πίνακα μεταθέσεων. Αυτή η διαδικασία θα γίνει για κάθε μοναδίκο έγγραφο και για όλα τα έγγραφα. Αυτός είναι ο

αλγόριθμος της MinHash ο οποίος θα μας επιστρέψει σύντομες υπογραφές αντί για μακροσκελοί διανύσματα οπού η ομοιότητα τους σχετίζεται με αυτήν Jaccard.

Βασισμένοι τώρα στην MinHash θα δημιουργήσουμε μία ρουτίνα MySigSim η οποία έχει ως σκοπό η ομοιότητα, δύο συγκρεκριμένων εγγραφών, η οποία βρίσκουμε με την MinHash για μέγαλο αριθμό μεταθέσεων να συγκλίνει με την ομοιότητα της jaccard το οποιό και ισχύει οπώς βλέπετε από κάτω.

```
number of permutation 1
With Ordered Lists --- 0.026316 seconds ---
Jaccard Similarity(Ordered Lists) of documents with number 1 and 2 is :  0.02631578947368421

comparisons 2610
With Sets --- 0.000967 seconds ---
Jaccard Similarity(Sets) of documents with number 1 and 2 is :  0.02631578947368421

Signature Similarity of documents with number 1 and 2 is :  0.0
```

```
number of permutation 2
With Ordered Lists --- 0.055556 seconds ---
Jaccard Similarity(Ordered Lists) of documents with number 6 and 5 is :  0.05555555555555555

comparisons 2124
With Sets --- 0.001024 seconds ---
Jaccard Similarity(Sets) of documents with number 6 and 5 is :  0.05555555555555555

Signature Similarity of documents with number 6 and 5 is :  0.5
```

```
number of permutation 10
With Ordered Lists --- 0.009009 seconds ---
Jaccard Similarity(Ordered Lists) of documents with number 8 and 2 is :  0.009009009009009009

comparisons 2175
With Sets --- 0.000000 seconds ---
Jaccard Similarity(Sets) of documents with number 8 and 2 is :  0.009009009009009009

Signature Similarity of documents with number 8 and 2 is :  0.0
```

Έχοντας ως στόχο πλέον να βρούμε τους πλησιέστερους γείτωνες για ένα συγκεκριμένο αρχείο θα υλοποιήσουμε την μέθοδο της ομής βίας. Αρχικά θα υπολογίσουμε την ομοιότητα ενός συγκεκριμένου αρχείου (είτε με την Jaccard είτε με την SigSim). Έχοντας βρει τις ομοιότητες μπορούμε εύκολα να βρούμε και τις αποστάσεις ($Dist(d,docID)$=1–$Sim(d,docID)$.) και θα τις αποθηκεύσουμε σε ένα λεξικό με κλειδί τα αναγνωριστικά των εγγράφων και θα τα ταξινομήσουμε ως προς τις τιμές των αποστάσεων. Αποθηκεύουμε μετά τους πρώτους ‘χ’ γείτωνες (ανάλογα με τον αριθμό που μας δίνει ο χρήστης) με βάση την απόσταση από το ταξινομημένο λεξικό στους πίνακες docsidlist και mindistances για να γνωρίζουμε ποιοι είναι οι κοντινότεροι γείτονες

κάθε docid. Στην θέση 2 του docsidlist βρίσκονται τα ids των κοντινότερων γειτόνων του εγγράφου με αναγνωριστικό 3.

Για να βρούμε τον μέσο όρο ομοιότητας υπολογίζουμε τον μέσο όρο ομοιότητας των γειτώνων με το έγγραφο. Όσο για τον μεσο βαθμό ομοιότητας θα υπολογίσουμε τον μέσο όρο των μέσων όρων εγγύτητας για όλο το σύνολο δεδομένων.

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

Τέλος μας έχει μείνει να φτιάξουμε την μέθοδο LSH, η οποία αρχικά παίρνει σαν είσοδο το μητρώο υπογραφών από τον minhash (στην συνάρτηση μας δεν το παίρνει σαν όρισμα καθώς την έχουμε διαθέσιμη σε όλο το κομμάτι κώδικα που διαχειριζόμαστε και δεν χρειάζεται να περαστεί ως παράμετρος της συνάρτησης) και ένα αριθμό δοσμένο από τον χρήστη για το μέγεθος των γραμμών της κάθε μπάντας. Κάθε μπάντα θα έχει την μορφή ενός λεξικού με κλειδί το αναγνωριστικό εγγράφου και ως τιμή να έχει τον κάδο ο οποίος καθορίζεται μια φορά τυχαία για όλες τις μπάντες από την συνάρτηση κατακερματισμού. Έπειτα ταξινομώντας το λεξικό με βάση τις τιμές μπορούμε με ένα σάρωμα να εντοπίσουμε όλα τα ζεύγη αναγνωριστικών εγγράφων(για την συγκεκριμένη μπάντα) που ανατέθηκαν στον ίδιο κάδο(έχουν κοινή τιμή), δηλαδή τα έγγραφα τα οποία θα είναι υποψήφια για μέτρηση ομοιότητας. Σαρώνοντας τα λεξικά όλων των μπαντών, εντοπίζουμε όλα τα υποψήφια ζεύγη για τα οποία πρέπει να ελέγξουμε ομοιότητα. Τώρα μένει να δούμε πως θα μετρήσουμε την ομοιότητα των αρχείων. Αρχικά θα χρειαστεί να ορίσουμε ένα s (s~$(1/b)(1/r)$ όπου b numbands και r rowperbands) σαν κατώφλι για την επιλογή των πλησιέστερων γειτόνων. Η επιλογή δεν θα είναι τυχαία καθώς θα πρέπει να υπάρχουν τουλάχιστον πλήθος γειτόνων υποψήφια ζευγάρια που περιλαμβάνουν ένα συγκεκριμένο έγγραφο. Αν για κάθε έγγραφο δεν προκύπτουν τουλάχιστον πλήθος γειτόνων υποψήφια ζευγάρια τότε ο LSH υποδιπλασιάζει τις γραμμές ανά μπάντα και επαναλαμβάνουμε. Έχοντας πλέον τα υποψήφια ζεύγη μετράμε την jaccard similarity

μεταξύ τους και υπολογίζουμε τους κοντινότερους γείτονες για κάθε αναγνωριστικό έγγραφο όπως στην ωμή βία.

Περιμένουμε ο LSH να είναι πολύ πιο γρήγορος από τον Brude Force αλγόριθμο, επειδή χρησιμοποιεί τις ομοιότητες ΜΟΝΟ για τα υποψήφια ζεύγη εγγραφών.