ΜΑΥΡΟΦΟΡΑΚΗ ΔΗΜΗΤΡΑ
1115201400104

# 3η ΕΡΓΑΣΙΑ

# Υλοποίηση αλγορίθμων υπόδειξης κρυπτονομισμάτων (Recommendation)

Το πρόγραμμά μου υλοποιεί τους recommendation αλγορίθμους τόσο σε έτοιμο dataset όσο και σε dataset στο οποίο καλούμαι να βρω το sentiment .

- **Διάρθρωση αρχείων κώδικα**

*recommendation.cc* : Η main του προγράμματος μου. Σ' αυτό το αρχείο δημιουργώ τις δομές που χρειάζομαι. Στη συνέχεια ξεκινάω την εκτέλεση των ζητημάτων. Τέλος διαγράφω τις δομές που δέσμευσα δυναμικά.

*datastructs.h :* Header που περιλαμβάνει τις κλάσεις και τις συναρτήσεις του προγράμματος.

*datstructs.cc :* Υλοποιεί τις συναρτήσεις των κλάσεων DataVector, Euclidean, Cosine, Cluster και Twitter.

*general_functions.cc* : Περιλαμβάνει συναρτήσεις γενικής χρήσης που βασικός τους σκοπός είναι να απομονώσει εργασίες και να συμβάλλει στη διατήρηση ενός διαχειρίσιμου μεγέθους του recommendation.cc. Τέτοιες συναρτήσεις είναι για παράδειγμα η εκτύπωση των αποτελεσμάτων, η ανάγνωση του configuration file αλλά και του dataset και η αποδέσμευση των δομών.

*twitter_analysis.cc:* Περιλαμβάνει τους αλγορίθμους που αφορούν την εύρεση των σκορ, την δημιουργία των cj και uj και την κανονικοποίησή τους.

*algorithms.h:* Header με τους ορισμούς των συναρτήσεων συσταδοποίησης

*algorithms.cc* : Περιλαμβάνει την υλοποίηση των αλγορίθμων συσταδοποίησης.

*Makefile*: Αρχείο για την μεταγλώττιση του προγράμματος.

*files.conf: C*onfiguration file της παρακάτω μορφής που αφορά το path των αρχείων.
coins:sample_datasets/Ergasia_3/coins_queries.csv
lexicon:sample_datasets/Ergasia_3/vader_lexicon.csv
ready_tweets:sample_datasets/Ergasia_2/twitter_dataset_small.csv

*clustering.conf*: Configuration file της παρακάτω μορφής που αφορά δεδομένα του clustering.

number_of_clusters:100
number_of_hashfunctions:4
number_of_hashtables:1 //απαιτείται υποχρεωτικά να είναι 1
w:2
initialization:1 //default τιμή για την μέθοδο αρχικοποίησης του cluster
assignment:2 //default τιμή για την μέθοδο ανάθεσης του cluster
update:1 //default τιμή για την μέθοδο ενημέρωσης του cluster

- **Σχόλια για την επιλογή των δομών και των κλάσεων**

Τα ονόματα των κρυπτονομισμάτων αποθηκεύονται σε έναν vector<string>. Σε κάθε θέση του vector αποθηκεύω μια σειρά από το coins_queries.csv.

Το λεξικό αποθηκεύεται στη δομή map<string, double> lexicon όπου αντιστοιχίζω κάθε λέξη με το σκορ της.

Σε μια δομή τύπου map <int,vector<int>> αντιστοιχίζω τον χρήστη με τα tweets που έχει ποστάρει. Ενώ σε μια δομή τύπου map <int, vector<int>> αποθηκεύω τα κρυπτονομίσματα για τα οποία ο χρήστης δεν έχει αναφερθεί στα tweets του (θέση στον πίνακα cryptocurrencies).

Η βασική κλάση των διανυσμάτων είναι η DataVector. Ανάλογα με το αν έχει επιλεγεί να χρησιμοποιηθεί ευκλείδια μετρική ή μετρική συνημιτόνου δημιουργούνται δυναμικά objects τυπου Cosine ή Euclidean ( κλάσεις παιδιά της DataVector). Ωστόσο όλες οι συναρτήσεις έχουν όρισμα τύπου DataVector * για να είναι generic. Οι δομές ενός hashtable και ενός cluster παίρνουν σαν ορίσματα data τύπου DataVector, οπότε εκτός από την uj και cj δομή τύπου map<int,vector<double>> αρχικοποιώ δύο vectors<DataVector*> με διανύσματα από το uj και το cj. Επειδή κατα τη διάρκεια της κανονικοποίησης αλλάζω τα περιεχόμενα του DataVector κάθε φορά αρχικοποιώ εκ νέου τους δύο vectors ώστε να μην επηρεάζεται η προηγούμενη μέθοδος από τα δεδομένα της επόμενης
( clear_regulated_datapoints_vector()).

Στην κλάση twitter αποθηκεύεται το twitter id, το user id και το score του twitter αυτού. Επίσης αποθηκεύω και μία λίστα με τα κρυπτονομίσματα τα οποία αναφέρει το tweet (θέση στον πίνακα cryptocurrencies)

- **Μεταγλώττιση**

make

- **Εκτέλεση**

./recommendation -d sample_datasets/Ergasia_3/tweets_dataset_small.csv -conf clustering.conf -files files.conf -o output.dat

- **Παρατηρήσεις μεθόδων**

1A

Αφού έχω δημιουργήσει το uj με τα score των κρυπτονομισμάτων και με τιμή 10000 στα άγνωστα κρυπτονομίσματα, βάζω των μέσο όρο του score του χρήστη στα άγνωστα κρυπτονομίσματα. Στη συνέχεια βάζω το uj σε μία δομή hash table. Για κάθε διάνυσμα του uj βρίσκω του P πιο κοντινούς γείτονες στο bucket του hashtable που ανήκει και συμπληρώνω βάσει αυτων, τις τιμές στα άγνωστα κρυπτονομίσματα.

Ενδεικτικός χρόνος: 60.1403 ms
Σε κάθε bucket υπάρχουν πολλοί γείτονες γι αυτό και η αναζήτηση των P καλύτερων θέλει παραπάνω ώρα.

1B

Έχω βάλει σε clusters το dataset της προηγούμενης άσκησης και βάσει αυτών, δημιούργησα τον cj πίνακα που είναι στην πραγματικότητα οι εικονικοί χρήστες. Για τα άγνωστα κρυπτονομίσματα στον cj βάζω και πάλι τον μέσο όρο προκειμένου να μπορέσω να hashάρω το cj. Για κάθε διάνυσμα uj, βρίσκω τα P πιο κοντινότερα διανύσματα ανάλογα με το bucket που βρίσκεται στο cj_hashtable και βάσει αυτών των γειτόνων συμπληρώνω τις τιμές για τα άγνωστα κρυπτονομίσματα.

Ενδεικτικός χρόνος: 30.9419 ms
Οι virtual χρήστες είναι προκαθορισμένοι (~100) . Επομένως σε κάθε bucket υπάρχουν λιγότερα στοιχεία που πρέπει να διατρέξουμε αυτή τη φορά.

2A

Στο 2A τα διανύσματα του uj μπαίνουν σε clusters. Αντίστοιχα με το 1A για κάθε διάνυσμα του uj, βρίσκω σε ποιό cluster ανήκει και βρίσκω του P κοντινότερους γείτονες μέσω τον οποίων συμπληρώνω τις άγνωστες τιμές.

Ενδεικτικός χρόνος: 13.423 ms
Τα clusters που δημιουργούμε είναι προκαθορισμένα και αναλογικά μεγάλος ο αριθμός. Επομένως σε κάθε cluster αντιστοιχεί μικρός αριθμός στοιχείων, που κάνει την αναζήτηση γρηγορότερη.

2B

Τα διανύσματα cj μπαίνουν σε clusters. Για κάθε διάνυσμα από το uj , ο αλγόριθμος στο 2Β βρίσκει σε ποιό cluster των cj ανήκει ώστε από αυτό το cluster να επιλεγούν οι P που ζητούνται για να εκτιμηθεί η τιμή των άγνωστων κρυπτονομισμάτων.

Ενδεικτικός χρόνος: 0.027332 ms
Ο αριθμός των cj είναι μικρός και επομένως τα σύνολα που κατανέμονται στα clusters περιέχουν ακόμα λιγότερα στοιχεία από τα clusters uj.

Ο χρόνος αναζήτησης όμως είναι διαφορετικός από τον χρόνο κατασκευής των δομών. Το hashing είναι πιο γρήγορη διαδικασία από το clustering.

Οι αλγόριθμοι οι οποίοι έχουν δοκιμαστεί για τη δημιουργία των clusters είναι ο random initialization, ο lsh assignment και ο lloyds update.

Ο χρόνος έχει μετρηθεί μόνο από το σημείο που για κάθε uj αναζητούμε τους P κοντινότερους γείτονες σε οποιαδήποτε δομή, χωρίς να συμπεριληφθεί ο χρόνος πού κάνουν οι δομές για να δημιουργηθούν.