Κ23γ: Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Χειμερινό εξάμηνο 2018-19
3η Προγραμματιστική Εργασία στη γλώσσα C/C++

Η εργασία έχει 3 σκέλη. Πρέπει να υλοποιηθεί σε σύστημα Linux και να υποβληθεί στις Εργασίες του e-class το αργότερο την Κυριακή 13/01/2019 στις 23.59.

Υλοποίηση αλγορίθμων υπόδειξης κρυπτονομισμάτων (Recommendation)

Έστω σύνολο τιτιβισμάτων (tweets) T, σύνολο χρηστών U = {u1, u2, …, uN}. Κάθε τιτίβισμα δίδεται μετά από προεπεξεργασία (tokenization) ως ένα σύνολο από όρους (συμβολοσειρές) που ανήκουν σε λεξικό Λ. Χρησιμοποιώντας λεξικό Α θα γίνει ανάλυση «συναισθήματος» για την εξαγωγή της θετικής, αρνητικής ή ουδέτερης βαθμολογίας Si κάθε τιτιβίσματος ως πραγματικού αριθμού στο [-1,1]. Το Α περιέχει λέξεις με τον αντίστοιχο βαθμό «συναισθήματος». Για κάθε τιτίβισμα, έστω totalscore το άθροισμα των βαθμών των λέξεων που ανήκουν στο Α, που κανονικοποιούνται χρησιμοποιώντας τον τύπο:

$$Si = \frac{totalscore}{\sqrt{totalscore^{2} + alpha}}, \text{ όπου } alpha = 15$$

Αν καμία λέξη από το λεξικό Α δεν υπάρχει στο τιτίβισμα τότε θεωρούμε ότι είναι ουδέτερο (totalscore Si=0).

Δίδεται επιπλέον λεξικό Κ που περιλαμβάνει κ όρους που αντιστοιχούν στα κρυπτονομίσματα, το οποίο αποτελεί υποσύνολο του Λ. Βάσει της συσχέτισης χρηστών-τιτιβισμάτων δημιουργείται για κάθε χρήστη ένα διάνυσμα uj (1 <= j <= N) με κ συντεταγμένες που αντιστοιχούν σε κάθε κρυπτονόμισμα. Η τιμή της κάθε συντεταγμένης προκύπτει ως το άθροισμα του «συναισθήματος» των αναφορών σε κάθε κρυπτονόμισμα εντός των τιτιβισμάτων τα οποία αντιστοιχούν στον συγκεκριμένο χρήστη και στα οποία αναφέρεται το αντίστοιχο κρυπτονόμισμα.

Τα τιτιβίσματα ομαδοποιήθηκαν σε συστάδες (clusters) στην 2η εργασία. Το dataset που χρησιμοποιήθηκε στην εργασία 2 είναι διανύσματα xi, αποτέλεσμα επεξεργασίας των τιτιβισμάτων αυτών. Χρησιμοποιώντας τους δείκτες των xi μπορούμε να ανακτήσουμε τα αντίστοιχα τιτιβίσματα ti καθώς έχουν ίδιο index (και σειρά) στα αρχεία.

Χρησιμοποιώντας αποτελέσματα από την εργασία 2 θα κατασκευαστεί για κάθε συστάδα ένα διάνυσμα cj (1 <= j <= #clusters) με κ διαστάσεις που αντιστοιχούν στα κρυπτονομίσματα. Η τιμή της κάθε συντεταγμένης προκύπτει ως άθροισμα του «συναισθήματος» των αναφορών σε κάθε κρυπτονόμισμα εντός των τιτιβισμάτων της συστάδας στα οποία αναφέρεται το κρυπτονόμισμα. Δηλαδή το διάνυσμα cj είναι παρόμοιο με το uj αλλά βάσει της συσχέτισης συστάδας-τιτιβισμάτων.

Θα υλοποιήσετε αλγορίθμους Recommendation για τα εξής προβλήματα υπόδειξης:
Α. 5 «βέλτιστων» κρυπτονομισμάτων από το σύνολο Κ σε κάθε χρήστη στο U, στα οποία ο χρήστης δεν έχει αναφερθεί πρωτύτερα συγκρίνοντας με τα διανύσματα uj.
Β. 2 «βέλτιστων» κρυπτονομισμάτων από το σύνολο Κ σε κάθε χρήστη στο U, στα οποία ο χρήστης δεν έχει αναφερθεί πρωτύτερα, συγκρίνοντας με τα διανύσματα cj που προκύπτουν από τις συστάδες.

1. Cosine LSH Recommendation

1. Εύρεση P πλησιέστερων γειτόνων κάθε χρήστη / κάθε συστάδα με χρήση του range Cosine-LSH search. Το «συναίσθημα» για κάθε κρυπτονόμισμα κανονικοποιείται βάσει του μέσου «συναισθήματος» κάθε χρήστη / συστάδας. Πρόβλεψη του «συναισθήματος» των μη αξιολογημένων κρυπτονομισμάτων R'(u,i) με χρήση του σταθμισμένου αθροίσματος R'(u,i)=z*Σsim(u,υ)*(R'(υ,i)), όπου υ οι P γείτονες του u.
2. Επίλυση προβλημάτων Α και Β.

2. Clustering Recommendation

1. Συσταδοποίηση των χρηστών σε k συστάδες με αλγόριθμο της επιλογής σας. Κάθε χρήστης αναπαρίσταται από το διάνυσμα uj των «συναισθημάτων» για κάθε κρυπτονόμισμα. Τα «συναισθήματα» που αντιστοιχούν σε κάθε χρήστη κανονικοποιούνται βάσει του μέσου «συναισθήματος». Η απόσταση των χρηστών δίνεται από την Ευκλείδεια μετρική. Βελτιστοποίηση του k βάσει της αξιολόγησης Silhouette, αλλά και χρήση του k = N/P.
2. Επίλυση προβλημάτων Α και Β.

Και οι 2 αλγόριθμοι θα αξιολογηθούν βάσει του μέσου απόλυτου σφάλματος με τη μέθοδο 10-fold cross-validation στην περίπτωση Α. Στο Β το validation set θα επιλεγεί με τυχαίο τρόπο από τα γνωστά ζεύγη uj,si και η διαδικασία τυχαίας επιλογής θα επαναληφθεί 10 φορές ώστε να υπολογιστεί το μέσο απόλυτο σφάλμα.

ΕΙΣΟΔΟΣ

Αρχείο κειμένου `input.dat` διαχωρισμένο με στηλοθέτες (tab-separated), το οποίο θα έχει την ακόλουθη γραμμογράφηση:

```
P: [Integer]
1    915  bitcoin  profitable    …
1    952  ethereum losses        …
1    964  stellar  bull      …
2    93   tether   bear      …
2    127  ethereum profitable    …
```

Στην πρώτη γραμμή ορίζεται ο αριθμός P των κοντινότερων γειτόνων που θα χρησιμοποιηθούν (default 20). Η πρώτη στήλη αντιστοιχεί στο userId, η δεύτερη στήλη στο tweetId (αμφότερες σε αύξουσα ακολουθία) και οι επόμενες στήλες στο σύνολο των όρων που αντιστοιχεί σε κάθε τιτίβισμα.

Το αρχείο `input.dat` δίνεται μέσω παραμέτρου στη γραμμή εντολών. Η εκτέλεση γίνεται μέσω της εντολής:

```
$./recommendation –d <input file> -o <output file>
```

Η αξιολόγηση των μεθόδων πραγματοποιείται μέσω της εντολής:

```
$./recommendation –d <input file> -o <output file> -validate
```

ΕΞΟΔΟΣ

Η εκτέλεση του προγράμματος παράγει αρχείο κειμένου, το οποίο θα περιλαμβάνει τα κρυπτονομίσματα που θα υποδειχθούν σε κάθε χρήστη / συστάδα για κάθε αλγόριθμο. [Τα αποτελέσματα είναι ενδεικτικά]

```
Cosine LSH
<u1> bitcoin  ethereum stellar  Cardano  XRP
<u2> ethereum monero   stellar  XRP  IOTA
.............................................................
<uN> ethereum XRP      monero   IOTA     bitcoin
Execution Time: <milliseconds>

Clustering
<u1> bitcoin  ethereum stellar  Cardano  XRP
<u2> ethereum monero   stellar  XRP  IOTA
.............................................................
<uN> ethereum XRP  monero   IOTA bitcoin
Execution Time: <milliseconds>
```

Η εκτέλεση του προγράμματος για την αξιολόγηση των αλγόριθμων εκτυπώνει:

```
Cosine LSH Recommendation MAE: <Double>
Clustering Recommendation MAE: <Double>
```

Επιπρόσθετες απαιτήσεις

1. Το πρόγραμμα πρέπει να είναι καλά οργανωμένο με χωρισμό των δηλώσεων / ορισμών των συναρτήσεων, των δομών και των τύπων δεδομένων σε λογικές ομάδες που αντιστοιχούν σε ξεχωριστά αρχεία επικεφαλίδων και πηγαίου κώδικα. Η μεταγλώττιση του προγράμματος πρέπει να γίνεται με την χρήση του εργαλείου make και την ύπαρξη κατάλληλου Makefile. Βαθμολογείται και η ποιότητα του κώδικα (π.χ. αποφυγή memory leaks).

2. Το παραδοτέο πρέπει να είναι επαρκώς τεκμηριωμένο με πλήρη σχολιασμό του κώδικα και την ύπαρξη αρχείου readme το οποίο περιλαμβάνει κατ' ελάχιστο: α) τίτλο και περιγραφή του προγράμματος, β) κατάλογο των αρχείων κώδικα / επικεφαλίδων και περιγραφή τους, γ) οδηγίες μεταγλώττισης του προγράμματος, δ) οδηγίες χρήσης του προγράμματος, ε) πλήρη στοιχεία του φοιτητή.

3. Η υλοποίηση του προγράμματος θα πρέπει να γίνει με την χρήση συστήματος διαχείρισης εκδόσεων λογισμικού και συνεργασίας (Git ή SVN).

4. Χρήση κατάλληλης βιβλιοθήκης και εκτέλεση ελέγχων μονάδων λογισμικού (unit testing).