Τμήμα Μηχανικών Η/Υ και Πληροφορικής Πανεπιστημίου Ιωαννίνων
MYE047: *Αλγόριθμοι για Δεδομένα Ευρείας Κλίμακας*
Ακαδημαϊκό Έτος 2019-20

Διδάσκων: Σπύρος Κοντογιάννης

1ο Σετ Ασκήσεων: ΟΜΟΙΟΤΗΤΑ ΑΝΤΙΚΕΙΜΕΝΩΝ

Ανακοίνωση: Παρασκευή, 6 Μαρτίου 2020 **Παράδοση:** Παρασκευή, 27 Μαρτίου 2020

ΠΕΡΙΓΡΑΦΗ ΕΡΓΑΣΙΑΣ

Σε ένα σύστημα αξιολόγησης ταινιών από τους θεατές, μελετάται η αναζήτηση ζευγών θεατών που έχουν δει αρκετές κοινές ταινίες. Θεωρούμε τη διμελή σχέση

watched(X,Y): «*ο χρήστης X έχει δει την ταινία Y*»,

αναπαριστώντας την (μόνο εννοιολογικά, δε θα το κατασκευάσετε) ως ένα 0/1-μητρώο όπου οι γραμμές αντιστοιχούν σε χρήστες και οι στήλες αντιστοιχούν σε ταινίες.

Θεωρούμε ότι οι K θεατές (χρήστες) είναι ήδη αριθμημένοι με διαδοχικούς αριθμούς-προσδιοριστικά (`userId`) στο διάστημα {1,2,…,K-1,K}, ενώ οι N ταινίες έχουν επίσης μοναδικούς αριθμούς-προσδιοριστικά (`movieId`), τα οποία όμως δεν είναι απαραίτητα διαδοχικοί αριθμοί στο {1,2,…,N-1,N}.

(1α) Προεπεξεργασία: Αναπαράσταση της σχέσης watched(X,Y) με λίστες γειτνίασης

Ως **είσοδος** δίνεται το όνομα ενός αρχείου αξιολογήσεων ταινιών από τους χρήστες (πχ, το αρχείο ratings_100users.csv για 100 χρήστες, ή το αρχείο ratings.csv για 605 χρήστες).

Κάθε γραμμή αυτού του αρχείου, μετά την πρώτη γραμμή που προσδιορίζει τη μορφή των επόμενων γραμμών, αναπαριστά (διαχωρισμένες με κόμματα) τις τιμές του προσδιοριστικού-χρήστη που παρέχει την αξιολόγηση, του προσδιοριστικού-ταινίας που αξιολογείται, και της χρονοσφραγίδας παροχής της αξιολόγησης. Ο μορφότυπος κάθε γραμμής είναι ο εξής (όπως εξηγείται και στην πρώτη γραμμή του αρχείου):

```
userId,movieId,rating,timestamp
```

Οι γραμμές (μετά την πρώτη) είναι ταξινομημένες κατ' αύξουσα τιμή των `userId` και, για τον ίδιο χρήστη, κατ' αύξουσα τιμή των `movieId`. Οι αξιολογήσεις (αν και δεν τις αξιοποιούμε στην παρούσα εργασία) είναι τιμές από το σύνολο {0.5, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5}. Οι χρονοσφραγίδες (πεδίο `timestamp`) αναπαριστούν τα δευτερόλεπτα από τα μεσάνυχτα της 1ης Ιανουαρίου 1970 (Coordinated Universal Time -- UTC).

Ως **έξοδοι** της προεπεξεργασίας κατασκευάζονται τα εξής (κάνοντας ένα μόνο πέρασμα στο αρχείο εισόδου):

- Ένα λεξικό (dictionary) λιστών, `userList`, όπου κάθε γραμμή `userList[userId]` αντιστοιχεί και σε έναν διαφορετικό χρήστη `userId`, ενώ η τιμή της είναι μια λίστα με τις όλες ταινίες (σε αύξουσα σειρά των `movieId` πεδίων) που έχει (αξιολογήσει, άρα και) δει ο χρήστης. Πρόκειται δηλαδή για ένα λεξικό λιστών. Εναλλακτικά, κάποιος θα μπορούσε να χρησιμοποιήσει μια λίστα λιστών, όπου για τον χρήστη `userId`, η λίστα των ταινιών του βρίσκεται στη θέση `userList[userId-1]`, γιατί η αρίθμηση των χρηστών είναι διαδοχική.
- Ένα λεξικό `movieMap` που αντιστοιχεί τις ταινίες σε μοναδικούς αριθμούς από το {1,…,N}. Προσέξτε ότι η τιμή του N δεν είναι εκ των προτέρων γνωστή (σε αντίθεση με την τιμή K των χρηστών, που είναι

ίση με τη μέγιστη τιμή `userId` στο αρχείο εισόδου). Κατά συνέπεια, καθώς σαρώνουμε το αρχείο εισόδου, για κάθε ταινία `movieId` που διαβάζουμε να πρέπει να εξετάζουμε αν ήδη βρίσκεται στο `movieMap` (οπότε απλά την αγνοούμε), ή αν απουσιάζει (στην οποία περίπτωση θα πρέπει να την προσθέσουμε με αύξοντα αριθμό κατά ένα μεγαλύτερο από το τρέχον πλήθος ταινιών στο `movieMap`.

- Ένα λεξικό λιστών `movieList`, όπου κάθε γραμμή `movieList[movieId]` αντιστοιχεί και σε μια διαφορετική ταινία `movieId`, ενώ η τιμή της είναι μια λίστα με τα `userId` όλων των χρηστών που την έχουν παρακολουθήσει. Κι εδώ μπορεί να γίνει χρήση λίστας λιστών: Η λίστα χρηστών που είδαν την ταινία `movieId` θα βρίσκεται στη θέση `movieList[movieMap[movieId]-1]`. Τότε όμως θα πρέπει να κατασκευαστεί πρώτα το λεξικό `movieMap` και στη συνέχεια η λίστα λιστών `movieList`.

Προσέξτε ότι η λίστα `userList[userId]` ουσιαστικά αναπαριστά την αντίστοιχη γραμμή `userId` του KxN 0/1-μητρώου **M**, όπου **M**`[userId-1,movieMap(movieId)-1]] == 1` αν και μόνο αν η ταινία `movieId` περιλαμβάνεται στη λίστα `userList[userId]` των ταινιών που έχει παρακολουθήσει ο χρήστης `userId`. Όταν λοιπόν χρειάζεται να επεξεργαστούμε τη γραμμή του μητρώου **M** που αντιστοιχεί στον χρήστη `userId`, δεν έχουμε παρά να «σαρώσουμε» τη λίστα `userList[userId]`: για κάθε `movieId` που συναντάμε, το αντίστοιχο κελί **M**`[userId-1,movieMap[movieId]-1]` θα έπρεπε να έχει τιμή 1. Αντιστοίχως, η λίστα `movieList[movieId]` αναπαριστά τη στήλη `movieMap[movieId]]-1` του **M** που αντιστοιχεί στην ταινία `movieId`.

(1β) Υπολογισμός Jaccard-Ομοιότητας μεταξύ των ταινιών

Για δυο συγκεκριμένες ταινίες, έστω `movieId1` και `movieId2`, η ρουτίνα

```
jaccardSimilarity(movieId1,movieId2)
```

θα επιστρέφει την Jaccard-ομοιότητα των στηλών του **M** που αντιστοιχούν στις δυο ταινίες. Σημειώνεται ότι έχοντας στη διάθεσή μας τις αντίστοιχες λίστες, `movieList[movieId1]` και `movieList[movieId2]`, μπορούμε να υπολογίσουμε τη Jaccard ομοιότητα ως εξής:

```
>> s1 = set( movieList[movieId1] )

>> s2 = set( movieList[movieId2] )

>> JACCARD = ( len(s1.intersection(s2)) / len(s1.union(s2)) )
```

(1γ) Δημιουργία Min-Hash Υπογραφών για τις ταινίες

Έχοντας κατά νου (αλλά ΟΧΙ υλοποιημένο) το μητρώο **M**, να υλοποιήσετε συνάρτηση `minHash`, η οποία κατασκευάζει ένα nxN μητρώο **SIG** με τα διανύσματα υπογραφών (κάθε διάνυσμα υπογραφών ταινίας είναι και μια διαφορετική στήλη στο μητρώο υπογραφών).

Ως **είσοδο** θεωρήστε το πλήθος n των διαφορετικών μεταθέσεων των χρηστών (δηλαδή των γραμμών του μητρώου **M**). Σαρώνοντας ακριβώς μία φορά όλες τις λίστες χρηστών (ισοδυναμεί με σάρωση κατά γραμμές του μητρώου **M**), να δημιουργήσετε το μητρώο **SIG** με τις υπογραφές όλων των ταινιών, όπως εξηγήθηκε στο μάθημα και παρουσιάζεται στην ενότητα 3.3.5 του βιβλίου.

Αντί για πραγματικά τυχαίες μεταθέσεις, προτείνεται να επιλέγετε κάθε φορά (για κάθε μετάθεση δηλαδή) και μια διαφορετική συνάρτηση κατακερματισμού $h_{a,b}: \{1, \dots, m\} \to \{1, \dots, m\}$ από μια καθολική οικογένεια $\mathcal{H}$ συναρτήσεων κατακερματισμού. Μια τέτοια οικογένεια συναρτήσεων υλοποιεί το αρχείο `universalHashFunctions.py` (το m θα πρέπει να είναι ίσο με το πλήθος K των χρηστών).

(1δ) Υπολογισμός Signature-Ομοιότητας μεταξύ των ταινιών

Ως είσοδος θα δίνονται δυο ταινίες, `movieId1` και `movieId2`, το nxN μητρώο υπογραφών **SIG**, καθώς και το πλήθος n' των πρώτων γραμμών του **SIG** που θέλουμε να λάβουμε υπόψη μας (προφανώς το n' θα πρέπει να είναι θετικός ακέραιος, που δε θα ξεπερνά το πλήθος n των γραμμών του **SIG**).

Η ρουτίνα `signatureSimilarity(movieId1,movieId2,n')` θα επιστρέφει την ομοιότητα των διανυσμάτων υπογραφών τους από τις πρώτες n' γραμμές του **SIG**, συγκρίνοντας τις στήλες `movieMap[movieId1]-1` και `movieMap[movieId2]-1` του **SIG**, που αντιστοιχούν στις ταινίες `movieId1` και `movieId2`.

(1ε) Δημιουργία Υποψηφίων Ζευγών με τη μέθοδο Locality-Sensitive Hashing

Θα πρέπει να υλοποιήσετε μια συνάρτηση `LSH`, η οποία τοποθετεί σε b κάδους καθεμιά από τις ταινίες, ανάλογα με την τιμή της υπογραφής της σε κάθε μπάντα υπογραφών, εφαρμόζοντας την τεχνική του *Locality-Sensitive Hashing*. Η συνάρτηση θα λαμβάνει στην είσοδο το μητρώο **SIG** των MinHash υπογραφών, και ένα κατώφλι $s \in (0,1)$ για τον ελάχιστο βαθμό ομοιότητας (υπογραφών) που απαιτούμε μεταξύ δύο ταινιών προκειμένου να τις θεωρήσουμε όμοιες.

Η συνάρτηση `LSH` χρησιμοποιεί μια συνάρτηση κατακερματισμού (χρησιμοποιήστε την ίδια συνάρτηση από το `universalHashFunctions.py` για όλες τις μπάντες, αλλά με διαφορετικούς κάδους ανά μπάντα) για τις ταινίες. Για μια μπάντα B, οι ταινίες `movieId1` και `movieId2` τοποθετούνται στον ίδιο κάδο, αν τα διανύσματα υπογραφών τους στη συγκεκριμένη μπάντα κατακερματίζονται στον ίδιο κάδο. Συνεπώς, για κάθε μπάντα ελέγχουμε τον κάδο κάθε ταινίας `movieId`, φροντίζοντας να δημιουργούμε (αν δεν υπάρχουν ήδη) υποψήφια ζεύγη της `movieId` με όλες τις ταινίες που ήδη βρίσκονται στον κάδο αυτό της μπάντας B. Μόλις ολοκληρώσουμε με όλες τις ταινίες για τη συγκεκριμένη μπάντα, καταστρέφουμε τους κάδους και συνεχίζουμε με την επόμενη μπάντα.

Ως **είσοδό** της η `LSH` θα δέχεται τις παραμέτρους n (πλήθος υπογραφών ανά ταινία), b (πλήθος μπαντών) και r (πλήθος υπογραφών ανά ταινία και μπάντα). Στην έξοδό της η `LSH` θα δημιουργεί μια λίστα από υποψήφια ζεύγη ταινιών για ομοιότητα.

(1ζ) Πειραματική Αξιολόγηση

Μπορείτε να χρησιμοποιήσετε για την πειραματική αξιολόγηση των προγραμμάτων σας τα εξής αρχεία εισόδου:

- Αρχείο ratings_100users.csv (http://www.cse.uoi.gr/~kontog/courses/Algorithms-For-Big-Data/locked-stuff/assignments/lab-1/datasets/ratings_100users.csv): Περιλαμβάνει τις αξιολογήσεις ταινιών από 100 διαφορετικούς χρήστες.
- Αρχείο ratings.csv (http://www.cse.uoi.gr/~kontog/courses/Algorithms-For-Big-Data/locked-stuff/assignments/lab-1/datasets/ratings.csv): Περιλαμβάνει τις αξιολογήσεις ταινιών από 605 διαφορετικούς χρήστες.

Η πειραματική σας αξιολόγηση θα γίνει σύμφωνα με το μικρότερο αρχείο (ratings_100users.csv). Εφόσον το επιθυμείτε, μπορείτε να δοκιμάσετε και το μεγαλύτερο αρχείο και να καταγράψετε τα αποτελέσματα της πειραματικής σας αξιολόγησης και για το αρχείο αυτό (θα δοθούν επιπρόσθετοι πόντοι γι' αυτό).

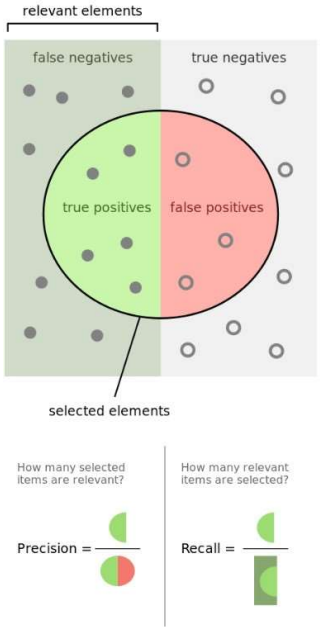
(1ζ1) Πειραματισμός για Min-Hashing

Υπολογίστε τη **Jaccard-ομοιότητα** που δίνει η ρουτίνα `jaccardSimilarity`, για όλα τα ζεύγη των 20 πρώτων στη σειρά ταινιών (δηλαδή, για αυτές με τα μικρότερα `movieId`-προσδιοριστικά). Πχ, στο σύνολο δεδομένων

ratings_100users.csv είναι οι ταινίες με προσδιοριστικά 1,2,3,4,5,6,7,8,10,11,12,13,14,15,16,17,18,19,20,21. Στη συνέχεια, μεταξύ αυτών βρείτε τα ζεύγη που έχουν Jaccard-ομοιότητα τουλάχιστον 0.5.

Κατόπιν, υπολογίστε το μητρώο υπογραφών **SIG**, για όλες τις ταινίες, για n=30 συναρτήσεις κατακερματισμού. Υπολογίστε την **ομοιότητα-υπογραφών** που δίνει η ρουτίνα `signatureSimilarity`, για τα ζεύγη μεταξύ των 20 πρώτων στη σειρά ταινιών που θεωρήσατε παραπάνω, λαμβάνοντας υπόψη τις n' = 5, 10, 15, 20, 25, 30 πρώτες γραμμές του **SIG**.

Για κάθε τιμή του n', εκτιμήστε την ακρίβεια της προσέγγισης υπολογίζοντας το πλήθος των **false positives** (ζεύγη με ομοιότητα υπογραφών τουλάχιστον 0.5, αλλά με Jaccard-ομοιότητα είναι μικρότερη του 0.5), και το πλήθος των **false negatives** (ζεύγη με Jaccard-ομοιότητα τουλάχιστον 0.5, αλλά με ομοιότητα-υπογραφών μικρότερη του 0.5). Επίσης, καταγράψτε για κάθε περίπτωση τις εξής μετρικές αξιολόγησης της προσεγγιστικής μεθόδου, όπως αυτές ορίζονται από το ακόλουθο σχήμα (δείτε επίσης https://en.wikipedia.org/wiki/F1_score):



- PRECISION = true_positives / (true_positives + false_positives)
- RECALL = true_positives / (true_positives + false_negatives)
- F1 = 2 * RECALL * PRECISION / (RECALL + PRECISION)

Σχεδιάστε τη γραφική παράσταση των τιμών για τα false-positives και false-negatives, PRECISION, RECALL και F1, ως συναρτήσεις της τιμής του n που δοκιμάσατε.

(1ζ2) Πειραματισμός για LSH

Για την περίπτωση των n'=30 υπογραφών, χωρίστε το μητρώο υπογραφών σε b μπάντες, με r συναρτήσεις κατακερματισμού σε κάθε μπάντα (προφανώς, n' = b·r**,**) και εκτελέστε τη ρουτίνα `LSH` που δημιουργήσατε στο **1ε**. Στόχος είναι να εντοπίσουμε υποψήφια ζεύγη ταινιών (μεταξύ όλων των ταινιών αυτή τη φορά) με Jaccard-ομοιότητα τουλάχιστον 0.5. Εστιάστε μόνο στις 20 πρώτες ταινίες που μελετήθηκαν και στο **1ζ1**.

Δοκιμάστε τις τιμές (r=3, b = 10), (r=5, b=6), (r=6, b=5), και (r = 10, b = 3). Αυτή τη φορά μετράμε, ανάμεσα στα υποψήφια ζεύγη που επιστρέφει η LSH, αυτά που όντως έχουν Jaccard-ομοιότητα 0.5 (**true positives**), αυτά που έχουν Jaccard-ομοιότητα < 0.5 παρότι είναι υποψήφια ζεύγη (**false positives**), καθώς και τα ζεύγη με Jaccard-ομοιότητα τουλάχιστον 0.5 τα οποία όμως δεν χαρακτηρίστηκαν υποψήφια ζεύγη από την `LSH` (**false negatives**).

Σχεδιάστε τη γραφική παράσταση των τιμών για τα false-positives, false-negatives, PRECISION, RECALL και F1, ως συναρτήσεις των 4 συνδυασμών τιμών που δοκιμάσατε. Αιτιολογήστε τη διαφορά στην ποιότητα των προσεγγίσεων για τις παραπάνω τρεις περιπτώσεις.

ΠΑΡΑΔΟΣΗ ΕΡΓΑΣΙΑΣ

Θα πρέπει να αναρτήσετε στο eCourse, το αργότερα μέχρι την **Παρασκευή 27/03/2020**, ένα ZIP αρχείο με όνομα της μορφής **2019-20_CSE-UOI_MYE047-ABD_< EPWNYMO >-< ONOMA >_< AM >_ASSIGNMENT-1.ZIP**, το οποίο περιλαμβάνει τα εξής:

(ι) Φάκελο **SOURCES** με όλα τα προγράμματά σας σε Python. **(ιι)** Φάκελο **EXPERIMENTS** με τα μητρώα και λεξικά που παράγετε, και snapshots από κάθε παράδειγμα εκτέλεσης. **(ιιι)** Αναλυτική αναφορά (σε μορφή MS WORD ή LaTeX), η οποία θα περιγράφει την υλοποίησή σας, και θα παρουσιάζει τα αποτελέσματα (και συνοπτική ερμηνεία τους) των παραδειγμάτων εκτέλεσης και της πειραματικής αξιολόγησης των προγραμμάτων σας.