Big data computing

Homework 1

You can discuss homeworks with other students of the course. However, you must understand well your solutions and the final writeup must be yours and written in isolation.
Please refer to course's Web page for detailed information about collaboration policy.

Make sure that the solutions are typewritten or clear to read.

Hand in your solutions and keep a copy for yourself. Solutions will be posted or presented after due date. In the final exam, you will be asked to explain your solutions and/or to go over your mistakes.

Due date: April 28th, 2017, 11.59PM.

Problem 1. This is a problem about statistical significance, false positives etc.

Consider a user-based collaborative filtering systems in which we have collected past users' purchases in the form of a set of m binary vectors of n components. The j-th user is represented by a vector $\mathbf{u}_j$, whose i-th component $\mathbf{u}_j(i)$ is 1 if the user bought the i-th item, it is 0 otherwise. Assume we adopt cosine similarity or, equivalently in this case, use the number of common purchases as a measure of pairwise user similarity. Assume further that in our dataset, a user purchases an average fraction p of available items[1].

You want to retrieve a set of candidate pairs for recommending items. To this purpose, you select all pairs $(\mathbf{u}_r, \mathbf{u}_s)$, such that $\mathbf{u}_r \cdot \mathbf{u}_s \geq qn$, for a suitable $q \leq 1$.

Assignment. Discuss a principled way to choose q, so that the (expected) fraction of false positive candidate pairs is at most a fraction $\alpha \leq 1$ of the total number of pairs, i.e., it is at most $\alpha\binom{m}{2}$. You should discuss the hypotheses you do and *quantitatively show* why your choice of q makes sense.

Hints: i) in the absence of any other reasonable criteria, you might define a pair $(\mathbf{u}_r, \mathbf{u}_s)$ to be a false positive if each of their entries was generated independently from each other with probability p; ii) you might need Chebyshev's inequality at some point.

Problem 2. You have a large, undirected graph $G = (V, E)$, stored on secondary storage. G represents friendship relations among users of a social networking platform.

G is stored as a list of edges. Each edge e is simply specified by its endpoint vertices, e.g., $e = (i, j)$. Without loss of generality you can think of i and j as integers.

Assume that, if $n = |V|$, you have kn RAM words to represent G in main memory, for some constant k (e.g., $k = 32$). Unfortunately, $|E| = \omega(n)$, hence you cannot store the whole adjacency list of G in main memory.

Assignment. Design an algorithm that: i) builds a suitable representation of G in main memory, executing a *single pass* of the data (i.e., the adjacency list) on secondary storage; ii) allows you to answer the following type of query without accessing secondary storage:

Query: Given $u, v \in V$, estimate $sim(u, v)$, where $sim(\cdot, \cdot)$ is a similarity measure that you think is suitable for this application.

[1]For a commercial system, p is likely going to be very small.

Requirements: You should i) justify your choice of $sim(\cdot, \cdot)$, possibly referring to existing literature on the topic; ii) you should be specific on the algorithmic tools/data structures yous use; iii) you should be specific on the algorithm, detailing it; iv) finally, you should explain (if this is the case) why it can accurately estimate pairwise similarity.

Problem 3. In the year 2045, robot swarms have finally become a reality. These are crowds of tiny robots, endowed with sensors, that crawl about our cities for the most various purposes, completely unattended. Each robot comes with a unique identifier (e.g., an extended MAC number) and, upon each encounter with one of its fellow robots, can exchange a limited amount of information with it (a few dozen bytes at most).

Assignment: Design an algorithm that will allow each robot to keep track of the number of distinct fellow robots it encountered since it was activated.

Requirements: i) you should explain why your solution will work, possibly referring to existing literature; ii) you should be specific on the algorithmic tools/data structures yous use; iii) you should be specific on the algorithm, detailing it; iv) finally, you should elaborate on accuracy of the estimation.

Problem 4. Nozama is one of the largest on line retailers in the world. As such, it uses a sophisticated recommender system. To this end, it has to maintain, in real time, statistical information about user profiles that are most likely to provide useful insights to the purpose of targeted advertising. The following is an example:

- The system receives real time information about user activity, in the form of a stream of *transactions*, where the t-th transaction $U(t)$ is a pair $U(t) = (i, j)$, where $i \in [n]$ denotes a user and $j \in [m]$ identifies an item she purchased. Note that the same pair (i, j) can be observed multiple times (e.g., I am likely to buy the same printer cartridge model several times). We denote by $S(t)$ the (multi)set of all transactions observed up to time t. We also denote by $m(t)$ the total number of *distinct* items purchased by all users up to time t.
- The *basket* of user i after the t-th transaction has been observed is the set $B(i, t) = \{(i, j) \in S(t)\}$. Note that $B(i, t)$ corresponds to the *set* of items user i purchased *at least once*. Let $f(i, t) = |B(i, t)|$.
- User i is a ϕ-heavy hitter at time t if $f(i, t) \geq \phi m(t)$.

Assignment: You should design a streaming algorith that, using space $o(\max\{n, m\})$ (possibly polylogarithmic at most), for every time t keeps an accurate estimate of all ϕ-heavy hitters at time t with high confidence, similar to [1, Theorem 6].

Requirements: i) you should explain why your solution will work, possibly referring to existing literature; ii) you should be specific on the algorithmic tools/data structures yous use; iii) you should be specific on the algorithm, detailing it; iv) finally, you should elaborate on accuracy of the estimation and efficiency (time and space) of your solution, providing a formal analysis as much as possible.

Problem 5. The homework requires implementing a map-reduce based code using Apache Spark.

Input. The input text is an ebook from The Project Gutenberg, titled "Plutarch: Lives of the noble Grecians and Romans by Plutarch" which is available at the URL:

`https://www.gutenberg.org/ebooks/674.txt.utf-8`

Output.

a) For each word in the input text, you must compute the occurrences of other words appear in every line that it appeared. For example, assume that we have the following line as an input text.

"Theseus seemed to me to resemble Romulus in many particulars."

With the word "Theseus": "to" occurs twice, "seemed" occurs once, "me" occurs once and so on.

With the word "to", "Theseus" occurs once, "to" occurs once (not zero times!), and so on.

For this assignment, you are going to compute the occurrences of every word for all lines, not just for a single line.

b) Produce a list of words, with the top-10 words that appear with this word for all lines.

c) For each word in the input text, you must compute the occurrences of other words that appear in every line that it appeared, right after the appearance of the word. For example, assume that we have the following line as an input text

"Theseus seemed to me to resemble Romulus in many particulars."

With the word "Theseus", "seemed" occurs once. With the word "seemed", "to" occurs once.

Produce a list of words, with the top-10 words that appear right after this word for all lines.

d) Generalize step (c) so that you include all words that appear up to distance k after a given word.

Guidelines :

1. Convert every string to lower-case
2. Remove any punctuation, i.e., any character other than [a...z]
3. For each sub-question submit 2 files: text file with the results (any format you believe is more suitable) and the python code.

What to submit. A .zip file containing the following:

1. Source code.
2. Output text file, named "hw1_out.txt", containing the outputs for the given input file.
3. Short description of the algorithm you used for this problem and how you implemented it in Spark.

References

[1] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.