Big Data Computing **A.Y. 2023/2024**
Homework 2 **Deadline: November 20th, 11.59pm**

Instructions. You should hand in your homeworks within the due date and time. Late deliveries will be penalized, please check homework delivery policies in the exam info.

Handing in. You should submit your work as *one zip file* containing: i) one folder with your code for **Assigment 1**, clearly organized; ii) a brief report (say 1-2 pages), clearly explaining 1) whatever is needed to understand your code and run it, 2) the implementation choices you made and 3) the *first 10 lines of the file you are supposed to produce* (see below); iii) *one pdf* containing your answers and solutions to the theoretical assignments. Answers should be either typed up or hand written **very clearly** and scanned. The first option is preferred.

It is perfectly fine if your code is contained in a *clearly commented* Colab notebook addressing points 1), 2) and 3) above. In this second case, you do not need to write a separate report. However, it is important that the code and your choices are adequately commented and explained and that I can run the code myself. Part of the score will depend on this.

Please deliver your homework by attaching the above zip file to an email addressed to me as follows (please format the subject as indicated below):
To: `becchetti@diag.uniroma1.it`
Subject: `HW2 <Your last name> <Your first name> <Your Sapienza ID number>`

Typesetting in Latex. Latex is very handy for typesetting (especially math), but you need to install it. If you do not want to install Latex, you can go for Overleaf, providing an integrated, Web interface, accessible for free in its basic version (which is enough for your needs). It allows you to both type in Latex using a Web interface, and compiling your code to produce a pdf document. Overleaf's documentation also contains a tutorial on Latex essentials.

Important note. Grading of your answers to the theoretical assignments will depend on i) correctness and solidity of the arguments, ii) clarity of the presentation and iii) mathematical rigour. E.g., ill-defined quantities, missing assumptions, undefined symbols etc. are going to penalize you. Rather than writing a lot, try to write what is needed to answer and write it well.

Assignment 1.
The goal of the first assignment is to compute (in adjacency list format) the co-review graph of the Amazon Fine Food Reviews dataset, available at `https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews` and already used in a previous notebook. Please refer to the Colab notebook on exploratory analysis of the above dataset for dataset retrieval and text feature extraction. In more detail you should:

1. Retrieve the dataset.

2. Use Apache pySpark to compute, for every user u, the list of all users who reviewed at least one item in common with u. For each such v, you should compute the number of items u and v co-reviewed.

3. In more detail, you should return and store on a file a list of tuples with the following format:

`[(<user>, <list>)].`

For a user u, `<list>` is itself a list of tuples of the form `(<user>, count)`, where <`user`> is a user v who reviewed at least one item in common with u, while `count` is the number of items that u and v have in common. <`list`> should be sorted in descending order of `count`.

So for example, assume user `A14FD1299PLTV2` has 8 items in common with user `A3JOYNYL458QHP` and 4 in common with user `A2RCZ8YKLE8B3O` (and no common reviewed items with any other user). Then, the list you return should contain, among others, the following tuple:

```
[... ("A14FD1299PLTV2", [("A14FD1299PLTV2", 8), ("A2RCZ8YKLE8B3O", 4)]) ...]
```

It should be noticed that extracting this information is essential for use in recommender systems. For example, in collaborative filtering, given a user, u we want to quickly access the subset of other users who have rewied items in common with u. This is a simple, yet tedious and computationally intensive task. You can go for approximate techniques such as LSH, or you can distribute the computation over a cluster.

Hints and suggestions. 1) An initial Colab notebook is accompanying this homework, relieving you from the task of implementing data retrieval; 2) I have checked that this computation can be performed on Google Colab for the entire dataset. During development, you may consider sampling a fraction of the dataset for quick testing. Sampling should be done carefully in order to be representative. In the stub notebook I make available, I added a cell containing a function (code to be completed) that allows you to sample a desired fraction of the users and store the relevant information in a new csv file; 3) you can reuse many ideas from the notebook on common friends, though you have to adapt them.

Finally, it should be noted that, other than writing the final list to a text file, all required computational steps can (and *should*) be performed in pySpark.

Assignment 2.

Consider vectors in $\mathbb{R}^d$. Assume that we use the technique seen in class (in particular, Section 3.7.2 of the Mining Massive Datasets book[1]) to estimate the angle (in radians) between any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. Answer the following questions:

[1] `http://www.mmds.org/`.

(a) Propose an estimator of the angle $\phi(\mathbf{x}, \mathbf{y})$ between any two vectors $\mathbf{x}$ and $\mathbf{y}$ (measured in radians) using a suitably defined family of hash functions.

(b) Denote by m the number of independent random vectors/hash functions you use to compute the signature of a vector. For any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, denote by $\hat{\phi}(\mathbf{x}, \mathbf{y})$ the value of your estimator for $\phi(\mathbf{x}, \mathbf{y})$ (notice that this should be a random variable if you are arguing correctly). Considered a minimum threshold angle θ that we want to be able to estimate accurately, compute the minimum value of m such that, if $\phi(\mathbf{x}, \mathbf{y}) > \theta$, then

$$\mathbb{P}\left(|\hat{\phi}(\mathbf{x}, \mathbf{y}) - \phi(\mathbf{x}, \mathbf{y})| > \epsilon\phi(\mathbf{x}, \mathbf{y})\right) \leq \delta,$$

where $0 < \varepsilon < 1$ and $0 < \delta < 1$ are constants that respectively measure accuracy and confidence of your estimator. Intuitively, you would like to choose m, so as to have an accurate estimate of any $\phi(\mathbf{x}, \mathbf{y})$ exceeding a *given* threshold θ.

(c) Assume next that we have a collection of n vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$. Find the minimum value m such that, for the same threshold θ and ε, δ defined as before we have:

$$\mathbb{P}\left(\exists i, j \in \{1, \ldots n\} : |\hat{\phi}(\mathbf{x}_i, \mathbf{x}_j) - \phi(\mathbf{x}_i, \mathbf{x}_j)| > \epsilon\phi(\mathbf{x}_i, \mathbf{x}_j)\right) \leq \delta.$$

I.e., this time, the condition should hold for *all pairs of vectors in the collection at the same time.*

Hints. You should be able to apply a Chernoff bound to address part (b). If you did things properly, the answer to part (c) should follow easily from your answer to part (b), without needing to do everything from scratch.

Important note. Please check the collaboration policy on the course web page.