Big Data Computing **A.Y. 2022/2023**
Homework 4 **Deadline: December 22nd, 2022**

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 the assignment, 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 results of the testing experiment. It is perfectly fine (and preferred) 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:
To: `becchetti@diag.uniroma1.it`
Subject: `HW4 <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 assignment is to use dimensionality reduction to perform clustering and automatic keyword extraction from a document collection. In this case, you are requested to write a Colab notebook that at least applies the following baselines to perform this task: i) standard k-means and ii) SVD. While these are the basic tools I expect you to use, you are encouraged, if necessary or appropriate, to explore further techniques or combinations of them that can make computation feasible and/or improve quality of the results. These include random projections onto a few hundred dimensions (e.g., 200 - 500) followed by k-means, big data compatible versions

of k-means such as mini-batch k-means, refined feature engineering techniques, truncated SVD followed by k-means applied to embedded vectors. As a test case, we will use the relatively large Amazon Amazon Books Reviews dataset, available at `https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews`. Please refer to the Colab notebook on exploratory analysis of the Amazon Fine Food Reviews dataset and the Colab notebook on dimensionality reduction and clustering applied to the 20 newsgroups dataset for dataset retrieval, text feature extraction and sklearn implementation of the SVD. The general question(s) we want to answer (and your task) is:

> What do people read? What do they like most? Which are people's preferred topics? Which are their preferred genres?

To achieve this task, you should use the review texts.

How to proceed.

- Implement all necessary text prepropressing required to transform each review into a tf-idf vector (or a simple count vectorizer if this provides better results). To this purpose, you can reuse (and improve!) the code available in the above mentioned notebooks.
- To perform clustering, you should decide the value of k, the number of clusters. In this case, you do not know how many topics you are supposed to find, so in part (but only in part) you should proceed by trial-and-error. Principled ways to find a reasonable tentative value for k are the following: i) if you are using k-means, you can plot inertia of the clustering you compute for increasing values of k and apply the elbow method seen in class. Inertia of a clustering is maintained in the attribute `inertia_` of the `sklearn.cluster.KMeans` object; ii) if you are using SVD, you can use the `explained_variance_` or `explained_variance_ratio_` attribute of the `sklearn.decomposition.TruncatedSVD` object to access the explained variance values of the k components (singular vectors) you kept. In this second case, you can keep the k largest components that account for 80-90% of the total variance. In normal datasets, this corresponds to a relatively small number of components. If you think this is still too large a number of components, you can set k to some predefined value (e.g., 100 to begin with) and then plot explained variance against the number n of components for n ranging from 1 to the predefined value of k you chose. If you already notice an elbow in this interval you are done. These are simple heuristics and it should be clear that it is up to you to proceed in a sensible and feasible way.

What to return. For each of the methods you identified as effective for this task (k-means and standalone SVD should anyway appear as baselines) produce the following:

- Assign keywords (i.e., labels) to each of the topics you identified in the dataset, producing a word cloud for each topic. Note that the number of clusters that

produces best results might in general differ when using k-means vs truncated SVD, though the values might be similar.

- Identify, if you can do this in a principled way, the most important high-level topics (i.e., which genres people like the most) identified by the propose method, such as sci-fi, romance, history etc.

In general, do not focus on "how much" you have to do, but try to do a good job, finding out what this dataset is telling us as to people's reading habits, i.e., try to answer the general questions I asked you above!

Hints and suggestions. A couple of remarks:

- For k-means, you might have problems applying the standard method to the dataset you are supposed to use, especially running times might be very high. For this reason, you are encouraged to use `sklearn.cluster.MiniBatchKMeans`. This "is a variant of the k-means algorithm which uses mini-batches to reduce the computation time, while still attempting to optimise the same objective function". Please refer to the sklearn's user guide for this class, which also points you to the WWW conference paper that proposed the algorithm. It is available at `https://scikit-learn.org/stable/modules/clustering.html#mini-batch-kmeans`.
- It might be very sensible to first test your algorithms on a *random* sample of the dataset (e.g. 10% of the reviews) to test your algorithms first, before moving to the entire dataset; you might even test your algorithms on multiple, independent random samples of the same size, to see if you get consistent results even using a sample.
- Always remember that text preprocessing in general can affect effectiveness of downstream algorithms very much, e.g., by removing "noise" more or less aggressively.