Machine Learning Assignment 3
Clustering and Dimension Reduction
Submission deadline: March 29, 2020

In this assignment you will get familiar with some common techniques used in clustering of data and its analysis.

A. Dataset Preparation: [5 points]

Download the Religious Texts Dataset from here (also to be uploaded on Moodle). Use the **Labelled** dataset for this assignment. The dataset contains the Document Term Matrix (DTM) from 8 different Religious Texts.

1. The first column contains the names of the religious text and their corresponding chapters. Replace the labels with only the names of the religious text (remove the chapter numbers, for eg., "Buddhism_Ch1" should become "Buddhism"). These are the class labels, and there are 8 of them. The rest of the columns represent the term frequency (frequency of the corresponding terms in the documents).

2. Now we want to convert the DTM to a TF-IDF matrix. **Note: Do NOT use the text provided in the corpus for computation of TF-IDF.**

   To calculate the TF-IDF value of a term 't' in the document 'd', use

   $$tf\text{-}idf\,(d, t) = tf\,(t, d) \times idf\,(t),$$

   where $tf\,(t, d)$ represents the term frequency of the term in the document (as given in the DTM), and $idf\,(t)$ represents the inverse document frequency of the term.

   Use $idf\,(t) = \log\,[\,(1 + n) / (1 + df\,(t))\,]$, where n is the total number of documents, and df(t) represents the document frequency (number of documents in which the term is present).

   Finally we consider each document as a vector of TF-IDF scores for the different terms. Normalize each vector by dividing it with the length of the vector (L2 norm).

   You can use scikit-learn (or any equivalent library in other programming languages) to obtain this TF-IDF matrix.

We define the similarity between two vectors by their cosine similarity, which is equivalent to the dot product of the two unit normalized vectors. For this assignment, consider the **distance** between two vectors as the inverse of their cosine similarity (**NOT** the Euclidean distance).

B. Agglomerative Clustering: [35 points]

Implement a hierarchical agglomerative clustering algorithm, to obtain 8 clusters of documents. Use the single linkage strategy to join clusters. Note: Do NOT use any ML library for this part.

Print the clusters into a file named **'agglomerative.txt'** in the following format: Each line will represent a different cluster, and will contain a sorted comma separated list of the indices of the data points in that cluster. Sort the clusters by the minimum index of the data points present in that cluster.
Eg: if suppose you obtain clusters [1,3,5], [2], [4,0], then print out:
0,4
1,3,5
2
Here the numbers represent the index of the corresponding documents in the dataset (the line number, excluding the header).

C. KMeans Clustering: [35 points]

Implement the standard KMeans clustering algorithm, using the given notion of distance, to obtain K=8 clusters of documents. Initialize the cluster centers randomly. Note: Do NOT use any ML library for this part.
Print the clusters into a file named **'kmeans.txt'** in the same format as in Part B.

D. Attribute Reduction by Principal Component Analysis: [5 points]

Reduce the number of attributes of the dataset to 100 by using PCA. You can use the implementation of PCA from scikit-learn (or from any equivalent library in other programming languages).
Use this reduced dataset and again obtain 8 clusters using your implementations of Agglomerative Clustering and KMeans Clustering. Print the clusters into files **'agglomerative_reduced.txt'** and **'kmeans_reduced.txt'** respectively, in the same format as specified in part B.

E. Evaluation of the clusters: [20 points]

Implement Normalized Mutual Information (NMI) by writing a function that reads the dataset for the class labels, and one of the cluster files and prints the NMI score. Calculate this score for all the 4 sets of clusters you obtained from Tasks B, C and D; and include them in your **README** file. Note: Do NOT use any ML library for this part.

Submission Instructions

Submit separate codes for each task and put them in a folder called “src”. Comment your codes sufficiently, use meaningful variable names. Keep the datasets in a separate folder called “data”, and the clusters files in a folder called “clusters”. Also submit a README file which will contain the instructions on how to execute your codes, and the NMI scores asked for in Task E.

All source codes, result files and the README file must be uploaded via the course Moodle page, as a **single compressed file (.tar.gz or .zip)**. The compressed file should be named as: **{ROLL_NUMBER}_ML_A3.zip or {ROLL_NUMBER}_ML_A3.tar.gz**
Example: If your roll number is 16CS60R00, then your submission file should be named as 16CS60R00_ML_A3.tar.gz or 16CS60R00_ML_A3.zip

Note that the evaluators can deduct marks if the deliverables are not found in the way that has been asked for the assignment, or if your codes are not understandable.

You can use one of C / C++ / Java / Python for writing the codes; no other programming language is allowed. You need to submit **raw codes** in one of the specified languages, which can be executed on a Linux system from the console. Other platform-specific or software-specific formats such as ipython notebooks are not allowed.

You **cannot** use any library/module meant for Machine Learning for implementing your models, unless specifically allowed in the problem statement. You can use libraries for other purposes, such as generation and formatting of data. Also you **should not use any code available on the Web**. **Submissions found to be plagiarised or having used ML libraries (except for parts where specifically allowed) will be awarded zero marks.**

Submission deadline: March 29, 2020, 23:59 IST [hard deadline]

For any questions about the assignment, contact the following TAs:
1. Soham Poddar (sohampoddar26 @ gmail . com)
2. Paheli Bhattacharya (paheli.cse.iitkgp @ gmail . com)