Final Project
2nd Semester 2020

1. Details

- The project will be developed in teams of 2-3 persons
- Project is due by Friday December 18, 2020
- For each problem, generate a .py file as **pf_01.py**, **pf_02.py** and **pf_03.py**
- Each person uploads to Teams four files. 1-3) the code files. 4) a .TXT file with the name of the team members, sorted in alphabetical order starting with the surname.
- Each team will present and explain their code personally online (shared screen).
- I will ask questions to each team member about the code implementation and the solution for the problem.
- The teams will present in a random order.
- The presentation will be on Friday December 18, 2929 at 14.00h

2. Problems

For the problems, you will use the three files we have collected and merged: posts.txt, labels.txt and users.txt

For the training phase of the first problem, the training set consists of all the FB posts grouped per user and the label for the corresponding user (h or m). The test phase will take as input any string (any FB post) and the program will output the predicted class (h or m) for such string (post).

For the second and third problems, only the posts file is needed (grouped per user), but the labels.txt file is not used.

1. **Implement the multinomial version of the Naïve Bayes classifier for training and testing.**

The training algorithm is as follows:

```
TRAINMULTINOMIALNB(C, D)
 1  V ← EXTRACTVOCABULARY(D)
 2  N ← COUNTDOCS(D)
 3  for each c ∈ C
 4  do N_c ← COUNTDOCSINCLASS(D, c)
 5     prior[c] ← N_c / N
 6     text_c ← CONCATENATETEXTOFALLDOCSINCLASS(D, c)
 7     for each t ∈ V
 8     do T_ct ← COUNTTOKENSOFTERM(text_c, t)
 9     for each t ∈ V
10     do condprob[t][c] ← (T_ct + 1) / (Σ_t' (T_ct' + 1))
11  return V, prior, condprob
```

Where C is the set of classes, i.e. C = {h, m}; and D is a collection of users, where each user is labelled with a class from C and represented as a bag of words (collection of words)

The testing algorithm is as follow:

```
APPLYMULTINOMIALNB(C, V, prior, condprob, d)
1  W ← EXTRACTTOKENSFROMDOC(V, d)
2  for each c ∈ C
3  do score[c] ← log prior[c]
4     for each t ∈ W
5     do score[c] += log condprob[t][c]
6  return arg max_{c∈C} score[c]
```

Where C is the set of classes, i.e. C = {h, m}; V, prior and condprob are the vocabulary, the priors and the conditional probabilities respectively, that were extracted in the training phase; and d is a new post to be classified represented as a bag of words (collection of words).

2. Implement the K-means clustering algorithm. The algorithm is shown below.

K-MEANS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)
1 $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)
2 **for** $k \leftarrow 1$ **to** K
3 **do** $\vec{\mu}_k \leftarrow \vec{s}_k$
4 **while** stopping criterion has not been met
5 **do for** $k \leftarrow 1$ **to** K
6 **do** $\omega_k \leftarrow \{\}$
7 **for** $n \leftarrow 1$ **to** N
8 **do** $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$
9 $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ *(reassignment of vectors)*
10 **for** $k \leftarrow 1$ **to** K
11 **do** $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ *(recomputation of centroids)*
12 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$

Where $\{\vec{x}_1, \vec{x}_2, \vec{x}_3, \ldots, \vec{x}_N\}$ is a collection of documents/users without any label, where each user is represented as a normalized vector $\vec{x}_i$ using tf-idf, N is the number of document/users, and K is the desired number of clusters. The stopping criterion is when the centroids does not change between iterations, or when a maximum number of iterations has been reached (defined the maximum number of iterations as 50). k controls for the cluster during iterations. $\vec{\mu}_k$ is the k centroid. There should be K centroids that indicate the center of each cluster.

3. Implement the Non-Negative Matrix Factorization (NMF) multiplicative algorithm.

In the NMF algorithm, the idea is to decompose a matrix in two matrices as follow:

$$X \approx WH$$

Where X is our term document matrix after applying tf-idf, normalization and being transposed (changing rows to columns), having each **column** representing a user as a normalized vector of p words. The idea is to construct the W and H matrices whose multiplication approximates the X matrix. For that, we must consider that X has a size of pxn (p words by n documents/users); W has size of pxr (p rows by r columns) and H has a size of rxn (r rows by n columns). The size r is an integer and a user input (try with values between 10 to 100).

The algorithm to find the matrices is as follow.

1. Initialize W and H at random (with float values)
2. Update matrices W and H during s iterations as:
   a. $H_{i,j} = H_{i,j} \frac{(W^T X)_{i,j}}{(W^T W H)_{i,j}}$
   b. $W_{i,j} = W_{i,j} \frac{(X H^T)_{i,j}}{(W H H^T)_{i,j}}$

The superscript T means the matrix transpose. The number s of iterations is defined by the user; try with different values for s, e.g., 20, 50, 70, 100, etc.

The resulting W matrix also represents a clustering of words, where the p words are organized in r clusters (the columns). As a final step, output the first 20 words for each cluster after sorting the values in each column in reverse order (major to minor).