Assignment 2

The deadline for the second assignment is May 2, before class. Turn in the code, with instructions on how to run it. The report should include detailed observations on the results. For late submissions the late policy on the page of the course will be applied. Details for the turn-in, and how to write reports are on the Assignments web page of the course. The will be an oral examination of the Assignment.

Question 1 (Singular Value Decomposition)

First, recall that the eigenvalue decomposition of a real, square and symmetric matrix B (of size $n \times n$) can be written as the following product:

$$B = Q\Lambda Q^T$$

where $\Lambda = diag(\lambda_1, \dots, \lambda_n)$ is a diagonal matrix that contains the eigenvalues of B (which are always real), and Q is an orthonormal matrix containing the eigenvectors of B as its columns.

Also, the Singular Value Decomposition (SVD) of a real matrix M (of size $n \times d$) can be written as:

$$M = U\Sigma V^T$$

where $\Sigma = diag(\sigma_1, \dots, \sigma_k)$ is a $k \times k$ diagonal matrix that contains the singular values of M (which are always real), U is a $n \times k$ orthonormal matrix containing the left singular vectors of M as its columns, and V is a $n \times k$ orthonormal matrix containing the right singular vectors of M as its columns.

1. Show that the matrices $M^T M$ and MM^T are real, square and symmetric
2. Show that matrices $M^T M$ and MM^T have the same eigenvalues. That is, if there are λ, x such that $M^T Mx = \lambda x$ then there is y such that $MM^T y = \lambda y$. Are x and y the same?
3. If $M = U\Sigma V^T$ is the singular value decomposition of matrix M write a simplified expression of $M^T M$ in terms of the matrices V, V^T and Σ.
4. You are given the matrix M:

$$M = \begin{bmatrix} -1 & 1 & 0 \\ -7 & -8 & -6 \\ -4 & -1 & -2 \\ 5 & 5 & 4 \end{bmatrix}$$

   Compute the SVD of M (use the *scipy.linalg.svd* function in Python and set *full_matrices=False*). Look at the singular values in Σ. What do you observe? Can we approximate M with a low-rank matrix? What is the error in the approximation?

Question 2 (Min-Hashing and LSH)

In this question you will implement and Min-Hashing and Locality Sensitive Hashing and you will apply it on the Twitter dataset. Our goal is to find very similar users. We suspect that such users are likely to be bots or trolls and we want to ban them from the system.

You will use again the file "tweets_dataset.txt" that you used for Assignment 1. Do the same preprocessing as before, but this time create a record for each user in the dataset that contains the **set** of hashtags and handles used by the user. To make the data denser, remove the users that have less than 20 hashtags/handles and the hashtags/handles with less than 20 appearances. Apply this pruning process iteratively until no more users or hashtags/handles and be removed. The dataset resulting from pruning is the dataset that we will use for this question. (Hint: The data is large, so it is important to use efficient data structures for the pruning process).

Compute the min-hash signatures for all users. In your implementation you will need to transform the hashtag/handle strings into 32-bit integers. Use the function hashString provided in the Assignments page. For the min-hashing functions, use hash functions of the form $h(x) = (a * x + b)\%R$ where a, b are random 32-bit integers, and R is a prime number larger than $2^{32} - 1$ (you can use the number $R = 4294967311$).

To evaluate the min-hash signatures that you created, compute the true Jaccard similarity between the users. Then compute the approximate Jaccard similarity as we described in class, using 10, 50, 100, και 200 hash functions. Evaluate the accuracy of the approximation using the sum of square errors (SSE) $\sum_{i,j}\left(J_{ij} - \overline{J_{ij}}\right)^2$, between the true Jaccard similarity (J_{ij}) and the approximate Jaccard similarity ($\overline{J_{ij}}$) for each pairs of users i, j. Also, find how many pairs of users have Jaccard similarity greater than 0.8 in the real data, and according to the min-hash signatures. Compute the number of false positives (pairs of users that in the min-hash signatures they have similarity greater than 0.8 but in reality they have similarity less than 0.8) and the number of false negatives (pairs of users that in the min-hash signatures they have similarity less than 0.8 but in reality they have similarity greater than 0.8). Create plots that show that these metrics change as we change the number of min-hash functions.

Then, implement Locality Sensitive Hashing, using b bands and r hash functions per band. Our goal is to find pairs of users with similarity greater than 0.8. Use the formula that gives the probability of creating a candidate pair as a function of the similarity of the pair to decide the values for b and r (include the plot of the function for these values in your report). Try different values for b and r, and report the number of candidate pairs and the number of false positives and false negatives.

Finally, for the values of b and r that you consider the best, look at the candidate pairs and the tweets that they have made. Do you think that some of these pairs correspond to bots or trolls? Justify your answer.

Question 3 (Recommendation systems)

In this question we will try to predict and recommend to Twitter users, hashtags or handles that they may be interested in.

Use the pruned data you created in the previous question. Select randomly 10% of the users, and remove randomly one of the hashtags/handles from their set. Our goal is to predict the hashtag/handle that we removed.

You will consider different algorithms for this task. The algorithms will produce a score for each hashtag/handle, and we will recommend to the user the k hashtags/handles with the largest score.

1. **MostPopular**: For the hashtags/handles that are not in the set of the user compute the score as the number of users that use them.
2. **User-based Collaborative Filtering** (**UCF**): For a user u find the m most similar users ($N_m(u)$) using Jaccard similarity, and compute the score of a hashtag/handle h as $\sum_{v \in N_m(u)} JSim(u,v) I(v,h)$, where $JSim(u,v)$ is the Jaccard similarity between u, v and $I(u,h)$ is 1 if the user u has the hashtag/handle h in her set, and 0 otherwise.
3. **Item-based Collaborative Filtering** (**ICF**): For a user u, and a candidate hashtag/handle h, find the m most similar hashtag/handles in the set of u ($N_m(h)$) using Jaccard similarity, and compute the score of h as $\sum_{h' \in N_m(u)} JSim(h,h')$, where $JSim(h,h')$ is the Jaccard similarity between hashtags/handles h and h'.
4. **Singular Value Decomposition** (**SVD**). Compute the Singular Value Decomposition of the 0/1 user-hashtag/handle matrix M.Take an m-rank approximation $M_r$of matrix M, and compute the score of hashtag/handle h for user u as $M_r(u,h)$.

Using any of the above algorithms, for some k, we can find the k hashtags/handles with the highest score. We will evaluate our algorithms by computing the accuracy of the prediction. We consider a prediction to be successful if the hashtag/handle we are trying to predict is within the k suggested hashtags/handles of the algorithm. The accuracy of the prediction of an algorithm is the fraction of users for which the prediction was successful.

Create a plot with the accuracy of the algorithms for values of k ranging from 1 to 20. In order to set the value of m for the algorithms **UCF** and **ICF** fix $k = 10$ and make a plot of the precision for different m ranging from 10 to 100 in steps of 5. For the **SVD** algorithm create a plot of the singular values and set m by finding the value for which the plot has a "knee". Include all plots in your report. Comment on your results.

Bonus: For every user-hashtag/handle pair (u,h) we have the number of times that the user use this hashtag/handle. Consider this to be the rating of the user for the hashtag/handle and modify the above algorithms to use cosine similarity instead of Jaccard similarity, as we described them in class. Compare with the implementation that uses 0/1.