In this session:

- We will complete a program to build a recommender system based on a user-based collaborative filtering recommendation system.
- We will compare the proposed recommendation system with a naive classifier.
- We will compare the two methods using a heuristic validation based on genres.

Within this assignment, we will use the **MovieLens dataset**. The raw data is available through the following link. Note: we recommend using the educational and development dataset, specifically the smallest one: `ml-latest-small.zip`, with a size of 1 MB). In detail, the compressed file contains the following raw data files:

1. `movies.csv`: Contains a list of movies identified by a unique movieId, a title, and the various genres of the film.
2. `ratings.csv`: Contains user ratings for a specific movie, ranging from 1 to 5, where 1 indicates that the user did not like the movie and 5 indicates that the user enjoyed the movie.
3. `links.csv`: Contains a mapping between a movieId and references to IMDb and TMDb databases
4. `tags.csv`: Contains tags for different movies. For instance, the film Toy Story has tags like 'pixar,' 'fun,' and 'fantasy,' while the film Dead Man Walking has the tag 'death penalty'.

The first task of this assignment is to understand and provide a statistical description of the data provided. For example, you can examine the distribution of raring, number of unique users or identify the most frequently watched genres historically. To achieve this, we have already prepared a utility function to load the dataset (`utils.load_dataset_from_source`).

2 Building a naive recommender

The first recommender system to build in this session is a naive system that we use as a baseline. In this case, the naive recommender system will be based on making recommendations by proposing films with the highest ratings. Films with better rankings will be recommended to the users. We ask you to fill in the code for this recommender in the script `naive_recommender.py`. What is the time complexity of this recommender system? Could you envision the current limitation of this recommender system ?

3 Building an user-to-user recommender

In this section we ask you to build a user-based collaborative filtering recommendation system. The recommendation engine will estimate the potential interest of a user a on an item s based on be based on of other user in a set U and the degree of similiraity between user a and users in U. In particular, that interest is defined as

$$interest(a, s) = \bar{r}_a + \sum_b w(a, b)(r_{b,s} - \bar{r}_b), \tag{1}$$

where $w(a, b)$ is the normalized similarity between user a and b, $r_{b,s}$ is the rating of movie s by user b and $\bar{r}_a$ is the average rating of user a.

You have to decide how to define U and justify it in your report.

Here we ask to you to fill the script `user-based_recommender.py` following the next steps:

- Generate training and validation partitions using the split_users function. This function will enable you to split the dataset and fold a set of movies for each user as a validation partition (we will work with this partition during the next section).
- Complete the code in `generate_m`. This function should return a data structure `M`, such that `M[user][movi` yields the rating for a user and a movie.
- Complete the similarity function in `similarity.py`. This function should compute the similarity between two lists. You should determine which metric is better for the proposed problem.
- Complete the `user-based_recommender` function. (1) Determine, for the target user, the most similar users using the similarity metric that you proposed during the preliminary activities. (2) Determine the unseen movies by the target user. (3) Generate recommendations for unrated movies based on user similarity and ratings.

4 Validation based on genres

In this section, we request that you decide which recommender is more suitable based on accuracy and complexity constraints. For this purpose, we suggest comparing the top k movies retrieved from the two recommenders (rec1, rec2) against the validation set that we have folded for each user. To ease this comparison, we have prepared for you the function `matrix_genres` located in the utils scripts, which contains the relationship between movies and genres. Given a target user, we ask you to evaluate, each recommender system according the resemblance between the frequencies of each genre in the top k movies recommended by the system and the frequencies of each genre for the movies of the target user in the validation set. Please provide a set of experiments with different users; we do not expect an exhaustive evaluation for each user but the evaluation should be statistically robust.

Entrega: 21 Diciembre 2023