Midterm Report

Project 19
Movie Recommendation System

By Jan Szachno and Aleksandra Głogowska

---

1. Task Description

The task is to develop a recommendation system for movies using the IMDB dataset. It has to cluster similar movies based on their description and other metadata.
A recommendation system in AI is a type of technology that predicts and suggests items to users based on various forms of input data. These systems are widely used to enhance user experience and engagement by personalizing content and suggestions according to individual preferences and behaviors. Having the user preferences data, we will prepare a movie recommendation system that will suggest a couple of different movies or tv series that the user has not seen yet.

2. Implementation Description

In the implementation we are using pandas and numpy for the data manipulation, as well as, sklearn, for the text processing and calculating similarities. The code starts with cleaning the dataset, by removing the “poster_link” column, dropping rows that- have the missing values, and removing duplicates, by the same names of the movies and tv series. We are using two databases. The first one is the IMDB database, and the second one provides the user's ratings.

After that, we are creating an instance of a TfidfVectorizer class. The TfidfVectorizer is a tool used to convert a collection of raw documents into a matrix of TF-IDF features. TF-IDF stands for Term Frequency-Inverse Document Frequency, which is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents. In our case, the words are scored against other words appearing in other records in the database. The TfidVectorizer will ignore common English stop words in the text. In our code, after creating the instance of the class, we are transforming the overview column in the database.

First, TF-IDF vectorization is applied to the 'Overview' column, converting it into a matrix of numerical values representing the importance of words in the context of the movie descriptions. Next, a new column, all_text, is created by concatenating genre, director, and the names of the top four stars for each movie. TF-IDF vectorization is then applied to this combined text to generate another feature matrix. These TF-IDF matrices are used to capture the textual and categorical information about the movies in a form that can be used to compute similarities between them. This process allows the model to understand and compare movies based on their descriptions and metadata.

The next step is to normalize the features from the database. The normalization of numerical features involves scaling the ‘IMDB_rating,’ ‘meta_score,’ and ‘no_of_votes’ columns to a range between 0 and 1 using ‘inMaxScaler’. This ensures that these numerical values are on a comparable scale, preventing any single feature from disproportionately influencing the model. The normalized numerical features are then combined with the previously generated TF-IDF matrices. This step is crucial for integrating different types of data (textual and numerical) into a single feature matrix for the subsequent similarity computations.

After that we combine the previously described features. Combining all features involves integrating the textual and numerical data into a single feature matrix. The TF-IDF matrices generated from the ‘Overview’ and ‘all_text’ columns are horizontally stacked with the normalized numerical features. This results in a comprehensive feature matrix that captures both the textual descriptions and key numerical attributes of the movies. The

combined feature matrix is then used to compute a similarity matrix using cosine similarity, which quantifies how similar each pair of movies is based on all available features. This similarity matrix is essential for the content-based filtering component of the recommendation system, as it allows the system to identify and recommend movies that are similar to a given movie.

The next thing that we do in the implementation is the computation of the user similarity matrix. It involves calculating how similar users are to each other based on their movie ratings. First, the user's ratings data is pivoted into a matrix format where each row represents a user and each column represents a movie, with the cell values being the ratings given by users to movies. Cosine similarity is then applied to this matrix to measure the similarity between each pair of users, resulting in a user similarity matrix. This matrix indicates how closely the preferences of different users align, with higher values representing more similar users. The similarity matrix is converted to a data frame for easier manipulation and is used in the collaborative filtering process to predict user ratings for movies they haven't rated yet. This user similarity matrix is crucial for recommending movies based on the preferences of similar users.

At the end of the code, we define two functions, 'predict_ratings' and 'get_combined_recommendations.' The latter one is the one called to execute the recommender program. The 'predict_ratings' function is included in the 'get_combined_recommendations.'

The 'predict_ratings' function estimates the ratings a specific user would give to movies they haven't rated yet. It starts by identifying similar users based on the user similarity matrix. The ratings of these similar users are then used to predict the target user's ratings. The prediction is calculated as a weighted average, where the weights are the similarity scores between the target user and other users. The predicted ratings are normalized by dividing by the sum of the similarity scores to ensure they are on a comparable scale. This function allows the system to infer the user's potential ratings for all movies, facilitating the collaborative filtering process. These predicted ratings are then used to recommend movies that the user is likely to enjoy based on the preferences of similar users.

The 'get_combined_recommendations' function provides personalized movie recommendations by merging content-based and collaborative filtering results. It begins with content-based filtering by finding movies similar to a specified movie using the content similarity matrix, generating a list of top similar movies. Concurrently, it predicts the user's ratings for all movies using the collaborative filtering approach, identifying the highest-rated movies according to the user's predicted preferences. The function then combines the scores from both methods, using a weighted average to balance content-based similarity and collaborative predictions. Afterward, it excludes movies the user has already watched to ensure the recommendations are new. Finally, the function returns a list of recommended movies. This dual approach should enhance the recommendation accuracy by leveraging movie content and user behavior data. The input is based on the name of the movie that the program has to give a similar recommendation to.

3. Experiments

Recommendation for movie 'Inception':
1. The Dark Knight
2. The Shawshank Redemption
3. The Godfather
4. Interstellar
5. Batman Begins

Recommendation for movie Interstellar:
1. The Dark Knight Rises
2. The Dark Knight
3. Inception
4. The Shawshank Redemption
5. The Godfather

Recommendation for movie The Lord of the Rings: The Return of the King:
1. The Lord of the Rings: The Two Towers
2. The Lord of the Rings: The Fellowship of the Ring
3. The Shawshank Redemption
4. Pulp Fiction
5. The Godfather

Recommendation for movie Pulp Fiction:
1. The Shawshank Redemption
2. The Godfather
3. The Dark Knight
4. The Lord of the Rings: The Return of the King
5. The Lord of the Rings: The Fellowship of the Ring

Recommendation for movie The Godfather:
1. The Godfather: Part II
2. Apocalypse Now
3. The Shawshank Redemption
4. The Dark Knight
5. Pulp Fiction

Recommendation for movie Mad Max 2:
1. Moneyball
2. North by Northwest
3. 12 Years a Slave
4. E.T. the Extra-Terrestrial
5. A Clockwork Orange

Experiment analysis:
Initially we tried using the approach based on content-based filtering. However, it gave us very generic results, mainly the movies with high ratings from the top of the list. However it was successfully able to recommend sequels of the movies. We suspect that putting more emphasis on genre will yield better results in the next iteration of the project.
We decided to try adding the collaborative filtering into the code, however it did not add any meaningful results to our experiments. We expect that it might be due to the limited quality of the dataset with user ratings, since it was manually created by us. We should consider finding a real dataset online to improve the quality of results.
We evaluated the results only manually by looking at them and comparing to what we expected. In further experiments we may consider adding an automatic method of evaluation of the results.

4. Changes from the initial plan

Initially we wanted to use only content-based filtering. However, to improve the quality of our results we decided to compare multiple approaches. We decided to not only try content-based filtering but also a combination of it with collaborative filtering. We hope that by comparing those solutions we will be able to better identify which approach will yield the best results. We decided to create a simple dataset of user ratings manually and assess the results. As mentioned earlier it did not yield satisfactory results. An online dataset should be found and evaluated.