OMEIZA MARTIN OJAPAH

201891986

COMPUTER-BASED RESEARCH TOOLS AND APPLICATIONS

PROJECT REPORT – MOVIE RECOMMENDATION SYSTEM

CMSC- 6950

PROF. EBRAHIM KARAMI

In this project, I will be implementing a movie recommendation system. There are a lot of times when we watch movies, and we need to watch something else similar to that movie, maybe genre-wise, based on the ratings, or based on the director. There are two types of recommender systems that we will implement as we go.

- **Demographic Filtering-** This is provided to each user based on the popularity or genre of the film. This system recommends users with comparable demographic characteristics to the same movies. Because each user is unique, this technique is deemed too simplistic. The basic idea behind this system is that films that are more popular and critically acclaimed will have a higher probability of being liked by the average audience.

- **Content Based Filtering-** They offer similar things based on a specific item. Our system makes recommendations for movies based on metadata such as genre, director, description, actors, etc. The basic premise behind these recommender systems is that if a person likes one item, they will enjoy another similar.

Demographic Filtering:
Before getting started with this -

- we need a metric to score or rate movie
- Calculate the score for every movie
- Sort the scores and recommend the best-rated movie to the users.

We can use the average ratings of the movie as the score but using this won't be fair enough since a movie with 8.9 average rating and only 3 votes cannot be considered better than the movie with 7.8 as an average rating but 40 votes. So, I'll be using IMDB's weighted rating (wr):
$wr = (v/(v+m) * R) + (m/(m+v) * C)$
where,

- v is the number of votes for the movie;
- m is the minimum votes required to be listed in the chart;
- R is the average rating of the movie;
- C is the mean vote across the whole report

Content-Based Filtering:
The film's content (overview, cast, crew, keyword, tagline, etc.) is employed in this recommender system to determine its resemblance to other films. After that, the films that are most likely to be comparable are suggested.

As before, we'll make a matrix with each column representing a word from the overview vocabulary (all words that appear in at least one document) and each row representing a movie. This reduces the importance of words frequently occurring in plot overviews and, therefore, their significance in computing the final similarity score.
Fortunately, scikit-learn gives us a built-in TfIdfVectorizer class that produces the TF-IDF matrix in a couple of lines.

We are now in an excellent position to define our recommendation function. These are the following steps we'll follow:-

- Get the index of the movie given its title.
- Get the list of cosine similarity scores for that particular movie with all movies. Convert it into a list of tuples where the first element is its position and the second is the similarity score.
- Sort the list above of tuples based on the similarity scores; that is, the second element.
- Get the top 10 elements of this list. Please ignore the first element as it refers to self (the movie most similar to a particular movie is the movie itself).
- Return the titles corresponding to the indices of the top elements.