CS634 – Data Mining Course at NJIT: Term Project Final Report (December 2023)

Bruce Kirschbaum

ABSTRACT The movie recommendation system presented in this term project for the Data Mining course at NJIT utilizes content-based filtering techniques to provide personalized movie suggestions to users. The dataset used for this project is the Small MovieLens Latest Dataset recommended for education and development. The system analyzes tag genome scores to recommend movies based on similarities between a user-inputted movie and all of the other movies in the dataset. To enhance the user experience, the movie recommendation system also provides explanations for the recommendations by highlighting shared genres and shared relevant movie tags between the input and recommended movie. The system aims to deliver accurate movie suggestions and contributes to the broader field of recommendation systems as a whole.

I. INTRODUCTION

In today's age, there are so many options to choose from when it comes to searching for a movie to watch. This can be both a blessing and a curse. On the one hand, you can find an endless number of films for whichever genre you are interested in. On the other hand, the number of movies out there can make it hard to make a choice. This dilemma is the reason I decided on creating a movie recommendation system with explainability.

The project aims to address this problem. My goal is to build a movie recommendation system that not only suggests movies but also provides explanations for why a particular recommendation was made, enhancing user trust and understanding. In this case, I will be having the user select one or perhaps several movies that they have previously watched and enjoyed, and based on their inputs, produce movie recommendations using similarity analysis. The main attributes I will be looking at for the model include the genre, tags, and ratings of each movie. Along with the recommendations will be explanations as to why the model chose those recommendations. To produce these explanations in plain English, I will be utilizing natural language processing techniques, most likely utilizing Python's Natural Language Toolkit.

The data I am going to use for this project is the Small MovieLens Latest Dataset recommended for education and development. This dataset includes 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users. The files that are included in this dataset include:

movies.csv: movieId, title, genres

tags.csv: userId, movieId, tag, timestamp

ratings.csv: userId, movieId, rating, timestamp

There is some preprocessing I will have to perform on this data before using it to create a movie recommendation model. First, I must get rid of irrelevant attributes. In this case, I will not need the timestamp of each user's ratings and tag applications. Next, this dataset does not have tag genome data. The preliminary work will be for me to generate this tag genome data to have a more stable metric for the model. After this is completed, I will be ready to work on building the model for the movie recommendation system.

"I hypothesize that by applying similarity analysis techniques to user behavior and movie features, my recommendation system will achieve a high level of accuracy and relevance in movie recommendations. Additionally, the system will provide sufficient explanations for why certain movies were chosen, ultimately resulting in increased user satisfaction."

II. BACKGROUND / RELATED WORK

My first task at hand is to perform some exploratory research on work done by others on similar projects. This research is meant to ensure that my project is not merely a copy of any project that has already been completed by someone else. Here are some links to websites I found that detail some of the work others have done in this area:

https://labelyourdata.com/articles/movie-recommendation-with-machine-learning

https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/

https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109

Most of the work done by others, as seen on these websites, details two techniques to create the movie recommendation systems:

Content-based filtering:

> The core element in content-based filtering is that only one user's data is used to make predictions. This recommendation system analyzes past preferences of a user, and calculates similarity between the user and attributes of various movies in order to recommend movies for the user.

Collaborative filtering:

> In collaborative filtering, the strategy is to consider the user's preferences and compare them to other users in the system. If users A and B are similar, then movies that A has watched and enjoyed will be recommended to B, and vice versa.

I plan to build a model that uses content-based filtering, in which one user inputs a single movie or several movies that they enjoy, and the system will recommend similar movies for them to watch. Where my project differs from the rest of these projects is the output that I will be giving will not only be the movies, but a description of why the system chooses those specific movies for the user. This is something I have not explicitly encountered in my research.

As previously mentioned, the dataset used for this project is the Small MovieLens Latest Dataset recommended for education and development. There is some preprocessing to perform on the data. Irrelevant attributes are to be removed, user ratings are to be normalized, and the relevant tag genome is to be created. The tag genome will also range on a scale from 0 to 1, and thus will already be normalized. The reasons for normalization are to have consistency among all of the relevant attributes, make it easier to conduct data analysis and perform machine learning algorithms, and to reduce sensitivity to small differences in the data.

III. METHODOLOGY

A. RAW DATA

All raw data collected for this project can be found at this link: https://grouplens.org/datasets/movielens/

The data used is from the Small MovieLens Latest Dataset recommended for education and development. As mentioned, this includes three relevant .csv files:

-movies.csv contains the movieId, title, genres of 9,000+ movies.

- tags.csv contains userId, movieId, tag, timestamp of 3,600 tag applications
- ratings.csv contains userId, movieId, rating, timestamp of 100,000+ user ratings

Cleaning of this data is simple. Get rid of irrelevant attributes, such as timestamp and userId, as the model will not care about these.

B. PRE-PROCESSING / DATA GENERATION

There is a solid amount of pre-processing to perform on the data to get it to the point where the model can work with it.

The first and easiest part is normalizing the user-given ratings. They were given on a scale from 0.5 to 5 in increments of 0.5s. To normalize this data, just apply the following formula on the whole column for ratings:

$$X' = (X - 0.5) / (5 - 0.5)$$

Next, it will be beneficial to explain what a tag genome is. There are 1128 tags that are present in the dataset, and each tag has a certain relevance score based on how relevant it is to a certain movie. For example, Toy Story (1995) has a relevance score of 0.99950 for the tag "toys" and a relevance score of 0.05775 for the tag "apocalypse." This should give a good idea of how the relevance scores work in the tag genome.

The issue is that, for the Small MovieLens Latest Dataset recommended for education and development, there is no tag genome given. The good news is that there is a tag genome for the full-sized dataset. So, the process of generating the tag genome for the small-sized dataset was relatively simple. The process is to take the full-sized tag genome and create a Python script (filter_genome_scores.py) to filter the relevance scores to generate a .csv file that only includes the relevance scores of the 'movieId's that appear in movies.csv. After this, the last step is to filter movies.csv, because the original tag genome does not include all the movies in the small-sized dataset for some reason. Another Python script (filter_movies.py) solves this problem. After this, the data is all in its final state and the model can now work with it.

C. DATA FEATURES

The finalized data consists of the following:

- filtered-genome-scores.csv (9929784 rows)
    - Contains relevance scores for all 1128 tags for each of the 8803 movies
    - Heavily right skewed
        - Shows that most tags do not define movies well

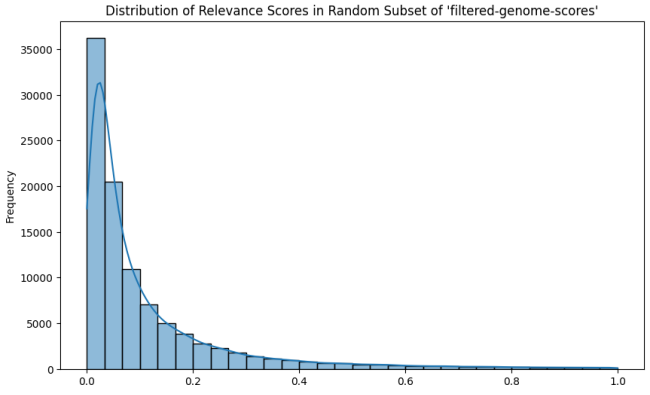

FIGURE 1. Distribution of relevance scores in a random subset (n=100000) of filtered-genome-scores.csv.

- filtered-movies.csv (8803 rows)
    - Contains movieId, title, and genres of each movie
- genome-tags.csv (1128 rows)
    - Contains tagId and tag (the name of the tag) for all 1128 tags
- normalized-ratings.csv (100836 rows)wor
    - Contains movieId and ratings applied over all of the movies
    - Moderately left skewed
        - Most movies get more higher ratings than lower ratings

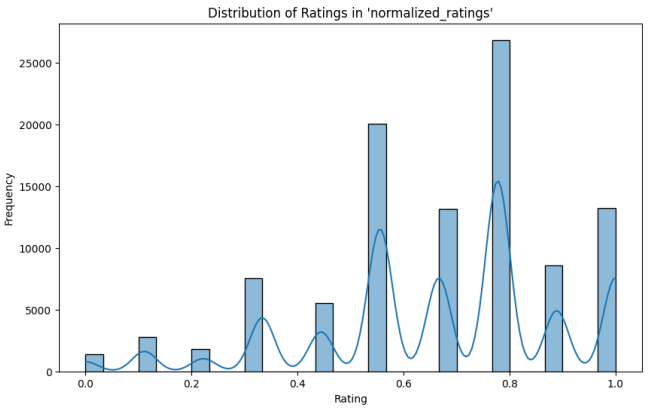

FIGURE 2. Distribution of ratings in normalized-ratings.csv.

C. MODEL

The medium through which I will be creating this project is Jupyter Notebook. Jupyter notebooks provide an interactive environment where you can run code in small chunks and see the results immediately. This is especially helpful for data exploration, iterative model development, and debugging. It is easy to quickly experiment with different algorithms and hyperparameters to see what produces the best result.

The movie recommendation system employs a content-based filtering approach, focusing specifically on the relevance scores of all the tags for each movie. The reasoning for choosing content-based filtering over collaborative filtering is primarily because of the dataset. Because of the all the numerical data regarding all the movies and the lack of data regarding specific user preferences, content-based filtering emerges as the best choice.

To utilize the tag genome to its fullest extent, the first step is to use the relevance scores assigned to different tags for each movie. These scores serve as quantitative values which capture the essence of each movie.

The selection of cosine similarity as the similarity metric is based on its ability to work with vectors and non-binary values, which aligns perfectly with the nature of the tag genome data. Cosine similarity provides a great metric for quantifying the similarity between two movies.

The model identifies the top 50 most relevant tags of the inputted movie based on all 1128 relevance scores for the given movie. These tags form a feature vector, and this vector is then compared against all other movies based on their feature vectors for the same 50 tags that were used for the inputted movie.

The resulting similarity scores show the model how similar the movie is to the inputted movie, and this is on a range from 0 to 1, 0 being completely dissimilar and 1 being completely similar. The top five movies are then recommended to the user.

Afterwards, the model provides an explanation as to why it chose the movie with the highest similarity score. This includes features such as the shared genres and shared relevant tags with the inputted movie. There is also some extra information generated, such as the average user rating for the top recommended movie and the top relevance scores for both the inputted movie and the top recommended movie.

Additionally, the model chooses a random movie to recommend to the user as a benchmark. It shows that the model is accurately recommending similar movies, as this random movie tends to have a significantly lower similarity score to the inputted movie than the actual recommended movies.

IV. RESULTS / EVALUATIONS

The results presented after thoroughly testing the finalized model for the project have exceeded expectations. The recommended movies for each one of the inputted movies have been great recommendations, not only because of the similarity scores being very high, but because of user feedback. I was able to test the model out with multiple users whom all agreed that the movies picked by the system were accurate and aligned with their preferences.

The cosine similarity between the top 5 ranked movies that the system would pick and the inputted movie was almost always above the threshold of 0.9, while randomly picked movies would usually have a similarity score somewhere in the range of 0.6 to 0.8. This shows that the recommendation system does a great job of recommending movies that closely match the inputted movie.

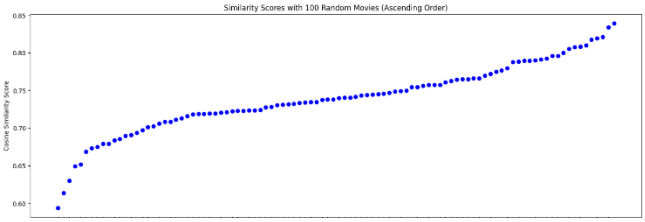

FIGURE 3. Similarity scores of a randomly selected subset of movies when compared to "The Cabin in the Woods (2012)."

Furthermore, the system's ability to produce a rationale as to why the recommendations were made played an important role in fostering increased user satisfaction. These explanations gave the users a reason to trust what the model was outputting.

This evaluation validates my hypothesis:

"I hypothesize that by applying similarity analysis techniques to user behavior and movie features, my recommendation system will achieve a high level of accuracy and relevance in movie recommendations. Additionally, the system will provide sufficient explanations for why certain movies were chosen, ultimately resulting in increased user satisfaction."

As the recommendation system consistently exhibited high accuracy, and users were content with the results, this project has been a success.

V. CONCLUSION

The movie recommendation system developed for this project successfully utilizes content-based filtering techniques to provide users with accurate and trustworthy movie suggestions. By analyzing the tag genome and computing cosine similarities between an inputted movie and all other movies in the dataset, the system offers recommendations with explanations as to why the model chooses its top recommended movie.

The project's aim to address the challenge of choosing a movie from such a vast array of options is met through this implementation of a recommendation system with explainability. The emphasis on content-based filtering, where user preferences are analyzed on an individual level rather than a collaborative level, along with the recommendation explanations distinguish this project from others like it.

The methodology involved much data preprocessing, as well as data generation, and finally, the actual implementation of the model itself. The choice of Jupyter Notebook as the platform promotes interactive development and is a great tool for visualization and an easy way to experiment with different algorithms and hyperparameters. It was very useful in producing the best results possible.

To summarize, the movie recommendation system does not only fulfill its primary objective of offering accurate movie suggestions but also contributes to the broader field of recommendation systems.

REFERENCES

https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/

https://grouplens.org/datasets/movielens/

https://labelyourdata.com/articles/movie-recommendation-with-machine-learning

https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109

VI. INSTRUCTIONS ON HOW TO RUN THE CODE

Start by downloading the dataset from MovieLens: https://grouplens.org/datasets/movielens/
You will need both ml-latest.zip and ml-latest-small.zip under "recommended for education and development."

All the code is available on my Git repository: https://github.com/Bruce241/Movie-Recommendation-System

First, you can perform an optional step. I cleaned the data by manually removing the rows I did not need, which include all userId columns and all timestamp columns.

Then, I also manually normalized the ratings in ratings.csv. Take the whole ratings column and apply the formula listed in Section III.B and save the file as normalized-ratings.csv.

Next, run the pre-processing Python programs filter_genome_scores.py and filter_movies.csv to generate filtered-genome-scores.csv and filtered-movies.csv respectively.

Then, the data you have has been finalized. All that is left to do is to use the Jupyter Notebook to run the model.

Don't forget to either upload filtered-genome-scores.csv, filtered-movies.csv, genome-tags.csv, and normalized-ratings.csv, or upload these files to your Google Drive and mount your Drive to the Jupyter Notebook.