Content-Based Movie Recommendation System

Gaurab Baral
Dr. Junxiu Zhou
School of Computing and Analytics
Northern Kentucky University

Abstract

This Movie Recommendations project aims to create a delightful movie-watching experience for users by providing personalized movie suggestions based on their preferences and movie attributes. The project focuses on building a prototype that extracts data from diverse movie databases. The system prepares the data for analysis using the Bag-of-Words, TF-IDF, and hashing models. It then utilizes cosine and Jaccard similarity metrics to measure movie similarity by employing preprocessing techniques. Through comparing all six combinations (bag-of-words / TF-IDF / hashing + cosine, bag-of-words / TF-IDF / hashing + Jaccard), the most effective approach is determined. This user-friendly tool significantly enhances the movie-watching experience, enabling users to explore personalized movie recommendations catering to their unique preferences.

Movie Dataset

The movie recommendation system utilizes the IMDB 5000 movie dataset collected from Kaggle.com. The dataset comprised of 28 attribute columns for the movies including title, name of actors, genre and so on. The word cloud(1) below visually represents all the movie attributes present in the dataset.

Figure 1: Word Cloud of Movie Attributes

Hypothesis

To test which recommendation system runs the best and gives the optimal result among the six different approaches.

Data Cleaning and Preprocessing

1. Only meaningful attributes, including 'director_name', 'actor_1_name', 'actor_2_name', 'genres', 'movie_title', etc., were retained in the dataset in this project in order to reduce computational cost.
2. Empty rows were removed from the dataset during the data cleaning process. The dataset was also checked for duplicate movies with the same title, and a total of 126 such movies were removed from the dataset in total.
3. We then processed the text data such as removing all whitespaces, converting all uppercase data to lowercase, lemmatized, and remove stopwords such as "the" and "and" , etc., using Python's NLTK library.
4. We then kept all the data of all columns into one column and called it all_data.

A sample of the Dataframe column(2) is shown in the image below:

'm night Shyamalan seychelle Gabriel noah ringer aasif mandvi action adventure family fantasy last airbender avatar fire kingdom tribe water'

Figure 2: Sample of the Dataframe column

Vectorization(NLP)

To convert the processed text data into numerical representations suitable for further analysis, the following techniques were employed:

1. Tokenization

In this method, the text data is tokenized (Kaur et al.). meaning the text is split into individual words or tokens. The resulting matrix represents the frequency of words in each movie.

index	aaron	aasif	abandoned	abbie
127	0.0	1	0.0	0.0

Figure 3: Count /Tokenization Vectorizer

2. TF-IDF (Term Frequency-Inverse Document Frequency):

This (Kaur et al.). approach evaluates the importance of each word in a specific movie relative to its occurrence across the entire dataset. It is calculated by multiplying the term frequency (frequency of a word in a specific movie) with the inverse document frequency (reciprocal of the number of movies containing that word).

index	aaron	aasif	abandoned	abbie
127	0.0	0.3272830538930234	0.0	0.0

Figure 4: TF-IDF Vectorizer

3. Hashing:

Hashing (Kaur et al.). involved creating a hash function that mapped words from the movie attributes to integer values. By using this approach, we were able to efficiently represent the textual information without the need to store an extensive vocabulary, making the process memory efficient.

14	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	-0.324428422615251	0.0	0.0

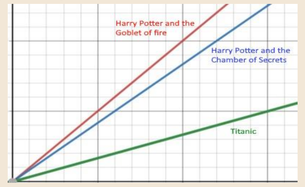
Figure 5: Hashing Vectorizer

Similarity Methods

In our Movie Recommendations project, we implemented two similarity methods.

1. Cosine Similarity:

In our project, each movie is represented as a vector using the numerical representations obtained from the bag-of-words, TF-IDF, and hashing. Cosine Similarity measures the cosine of the angle between these vector representations, indicating how similar the movies are based on their textual attributes. A cosine similarity score of 1 indicates that the two movies are identical in terms of their textual attributes, while a score of 0 signifies that they have no similarity (Goyani et al. 28).



$$\text{Cos}(A, B) = \frac{A \cdot B}{\|A\| * \|B\|}$$

Figure 6: Cosine Similarity

2. Jaccard Similarity:

Jaccard Similarity is another similarity metric we utilized to measure the similarity between movies. The Jaccard Similarity score was determined by calculating the size of the intersection between common words divided by the size of their union (all distinct words present in both movies). A score of 0 indicated no similarity between the two movies, while a score of 1 indicated that the movies shared the same set of words in their textual attributes (Goyani et al. 28).

$$\frac{\{\text{"Harry "}, \text{"}Potter\text{"}\}}{\{\text{"Harry"}, \text{"Potter"}, \text{"Goblet"}, \text{"Fire"}, \text{"Chamber"}, \text{"Secrets"}\}} = \frac{2}{6} = 0.333$$

Figure 7: Jaccard Similarity

Results:

- In terms of time of execution, the algorithm implemented using hashing and Cosine similarity used the least CPU time(3 µs) whereas all other approach used 5 µs.
- The wall time however was lowest for the approach implemented using TF-IDF and Cosine Similarity which was executed on 7.87 µs. The algorithm implemented using Hashing and Jaccard used had the highest wall time which was 10.5 µs. Note: This time was the average execution time for 25 trial runs.

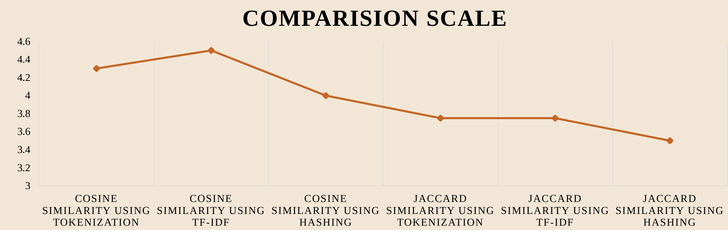


Figure 8: Comparision

Conclusion:

- It turns out the approach using TF-IDF and Cosine Similarity gave the best yield of recommendations among all approaches.
- This method was also executed at the lowest time making it the best algorithm among all others.

Future Directions:

We will continue enhance the recommendation performance of this project, by using the collaborative movie recommendation system, and/or deep learning recommendation systems. However, we'll still rely on content-based recommendations as the foundation for these innovative approaches.

References:

1. Goyani, Mahesh, and Neha Chaurasiya. "A review of movie recommendation system: Limitations, Survey and Challenges." *ELCVIA: electronic letters on computer vision and image analysis* 19.3 (2020): 0018-37.
2. Kaur, Sawinder, Parteek Kumar, and Ponnurangam Kumaraguru. "Automating fake news detection system using multi-level voting model." *Soft Computing* 24.12 (2020): 9049-9069.