Building a Movie Recommender System Using Collaborative Filtering and Content-Based Techniques

Michael Carnival, Hector R. Gavilanes

Department of Mathematics & Statistics

University of West Florida

Summer 2024

Introduction

The topic we were excited to explore, and tackle was how to use or build a recommender system to assist in making decisions based on user preferences or specific situational scenarios. Nowadays, recommender systems have become extremely popular in a short amount of time [1]. Social media platforms like Facebook, YouTube, and TikTok use them to help users find friends, create personalized feeds (FYP), and suggest videos to watch. E-commerce websites like Amazon and eBay employ recommender systems to suggest 'frequently bought together' items. In healthcare, doctors can provide personalized treatment plans, medication options, and lifestyle changes based on individual health profiles.

Moreover, Netflix's recommendation system plays a crucial role in guiding users' viewing choices. According to research, more than 75% of the content selected by Netflix users comes from recommendations provided by the platform's system [1]. This high percentage demonstrates the significant influence and effectiveness of recommender systems in shaping user behavior on streaming platforms. The ability to suggest relevant content not only enhances user experience but also contributes to Netflix's success in retaining subscribers and maximizing engagement with its vast library of series, sitcoms, and movies.

From these examples, it's clear that recommender systems enhance user experiences, boost sales, and simplify decision-making by suggesting relevant options. To demonstrate and learn, we will show how movies are recommended to users, marking our entry into the world of recommender systems. In our project, we will focus on using collaborative filtering to generate movie recommendations based on the preferences of other users.

Literature Review

Movie recommender systems are designed to suggest films that users are likely to enjoy based on various factors. There are three main techniques commonly used in these systems: Popularity-based, Content-Based, and Collaborative-based. Each approach has its own strengths and weaknesses, and they are often combined to create more effective hybrid systems.

Popularity-based Recommender

This technique suggests movies based on their overall popularity or trending status among users. It's simple to implement and effective for new users with no prior viewing history. However, it lacks personalization and may not cater to niche interests.

Content-Based Recommender

This approach recommends movies based on the characteristics and features of films a user has previously liked or interacted with. It's highly personalized and can recommend niche items like user preferences. Content-based filtering relies on detailed item metadata and feature extraction, such as genre, cast, director, and movie descriptions [1].

Collaborative-based Recommender

This method suggests movies based on the preferences and behaviors of similar users. It can discover novel and serendipitous recommendations by leveraging collective user behavior. Collaborative filtering is often implemented using matrix factorization techniques, such as Singular Value Decomposition (SVD) [2] [3].

Collaborative filtering can be further divided into two main types:

- **User-based** approach examines the similarities between users by comparing their ratings for the same items, allowing for the generation of tailored recommendations [4].
-
- **Item-based** method identifies items with similar rating patterns and recommends them to users who've liked similar items. Item based predictions are dependent on items similarities rather than neighbors of users [4].

Both user-based and item-based approaches are fundamental to collaborative filtering algorithms and are widely used in various domains, including e-commerce product recommendations and educational content suggestions.

Nonetheless, each of these techniques has its advantages and limitations. For example, collaborative filtering can suffer from the cold-start problem for new users or items, while content-based filtering may lead to over-specialization [3].

To overcome these limitations, many modern recommender systems employ hybrid approaches that combine multiple techniques. For instance, a system might use content-based filtering to address the cold-start problem and collaborative filtering to provide diverse recommendations [2].

Recent advancements in deep learning have also been applied to movie recommender systems, potentially improving their performance and ability to capture complex user preferences.

Dataset

The MovieLens dataset is a widely recognized and utilized resource in the field of recommender systems research. Developed by the academic community, it serves as a valuable tool for scientific research and algorithm development. The dataset has been employed by various platforms, including a non-commercial portal called pogledajfilm.info [1], which uses it to provide movie recommendations to users and conduct scientific research.

In our case, the dataset version contains 100,000 ratings and 3,600 tag applications for 9,000 movies from 600 users.

One of the most popular versions of the dataset is MovieLens 10M, which contains 10 million ratings for 10,000 movies from 72,000 users [1]. This extensive dataset has proven particularly useful in addressing the cold start problem, a common challenge in recommender systems where new users or items lack sufficient data for accurate recommendations.

MovieLens (www.movielens.umn.edu) itself is a web-based research recommender system that was launched in Fall 1997 [5]. It has since grown into a significant platform, attracting hundreds of users weekly who rate movies and receive personalized recommendations. As of the last available

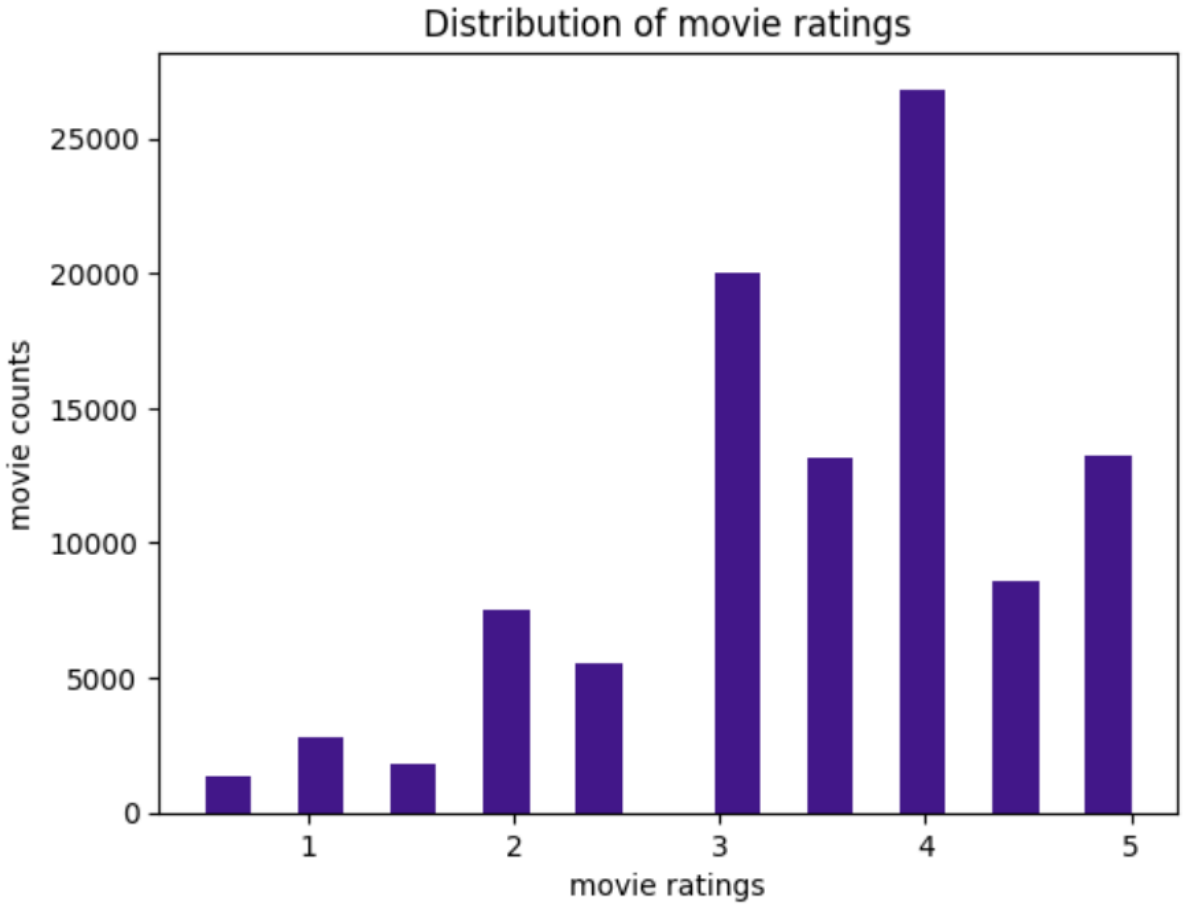
data, the site boasts over 35,000 users who have shared their opinions on more than 3,000 different movies [5].

However, it's worth noting that offline evaluations using this dataset should be approached with caution. Recent research has highlighted the importance of preserving the global timeline in offline evaluations to avoid data leakage and ensure realistic performance comparisons among recommendation models [6].

Exploratory Data Analysis

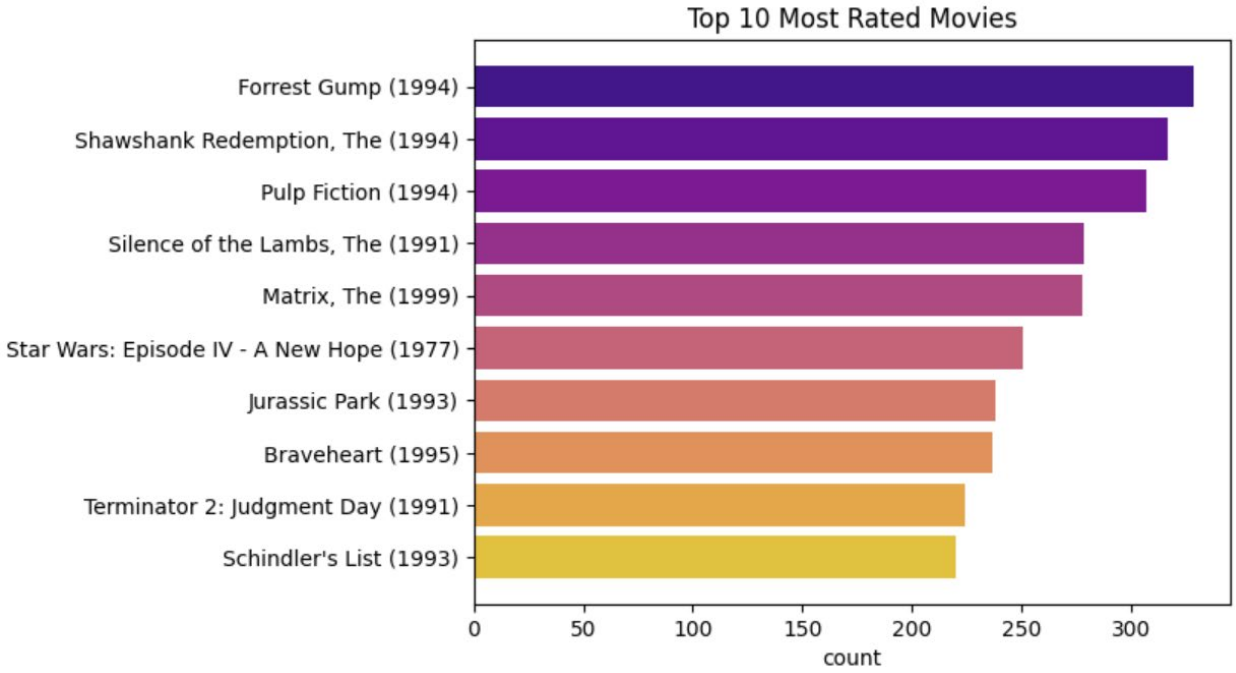
Rating Distribution

The distribution of movie ratings shows that most movies receive ratings of 3 or 4, with 4 being the most common. Ratings of 1, 2, and 5 are less frequent, indicating a tendency for moderate ratings.



On average, each user has contributed approximately 165.3 ratings, while each movie has received about 10.37 ratings. The distribution of movie ratings reveals a mean global rating of 3.5 and a mean rating per user of 3.66. The most frequently rated movies include "Forrest Gump" (1994) with 329 ratings, "The Shawshank Redemption" (1994) with 317 ratings, and "Pulp Fiction" (1994) with 307

ratings. Other notable films with high rating counts are "The Silence of the Lambs" (1991), "The Matrix" (1999), and "Star Wars: Episode IV - A New Hope" (1977).



The lowest average rating belongs to "Gypsy" (1962), while "Lamerica" (1994) may have the highest rating but only has 2 ratings, suggesting that a Bayesian average might be a better approach for evaluating movie popularity.

Popularity-Based Filtering

Bayesian Average

The Bayesian Average is particularly beneficial in scenarios requiring the ranking of items with varying numbers of ratings, as it effectively mitigates the influence of items with very few, potentially non-representative ratings. This method ensures a more balanced and reliable assessment by incorporating prior information and adjusting for the number of ratings, thereby providing a more accurate representation of an item's true quality.

The Bayesian Average formula [7] is:

$$BA = \frac{C \times M + \sum R}{C + N}$$

Where:

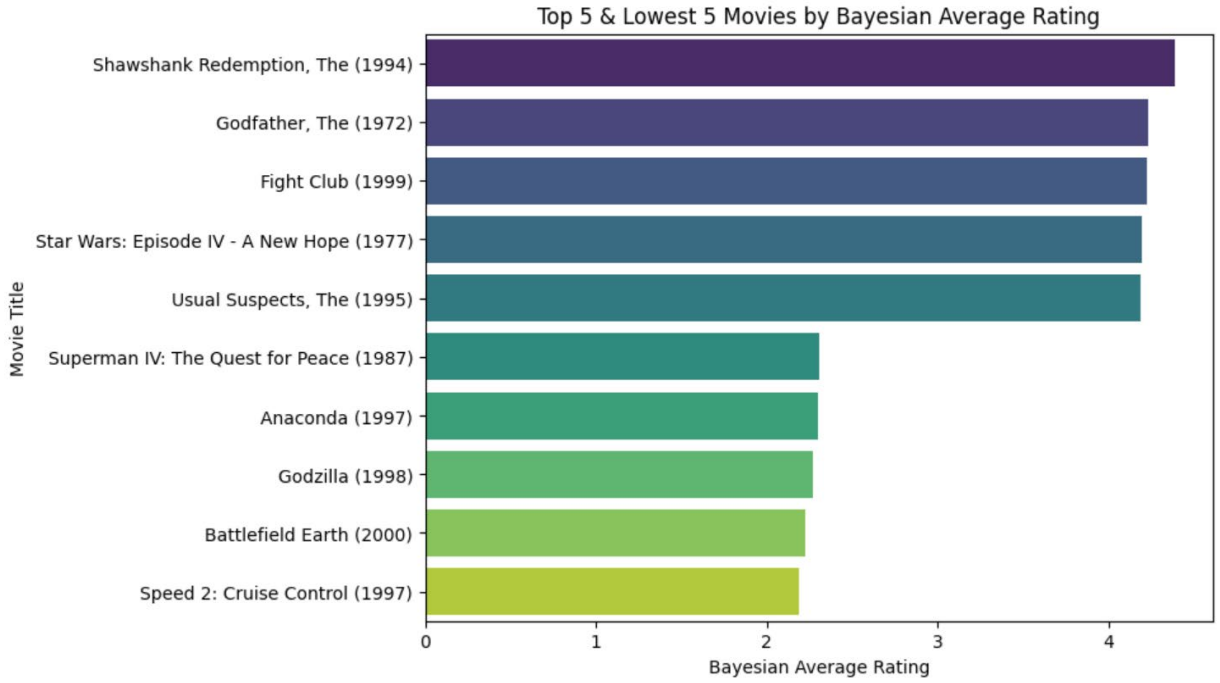
- C is the prior weight. Also called “confidence” factor
- M is the prior mean
- R is the sum of all ratings for the item
- N is the number of ratings for the item

The formula adjusts the average rating based on the number of ratings and prior belief about the overall distribution of ratings. Items with few ratings are pulled closer to the overall mean, while items with many ratings are less affected by the prior.

Insights from Bayesian Average

The average number of ratings received by each movie is 10.37, indicating how many users have rated the film. Meanwhile, the average rating score for these movies is 3.26, reflecting the overall quality assessment given by the users.

Using the Bayesian Average function, the rating for Lamerica decreased from 5.0 to 3.5. In contrast, the rating for Forrest Gump remained largely unchanged, staying close to its original score of 4.1.

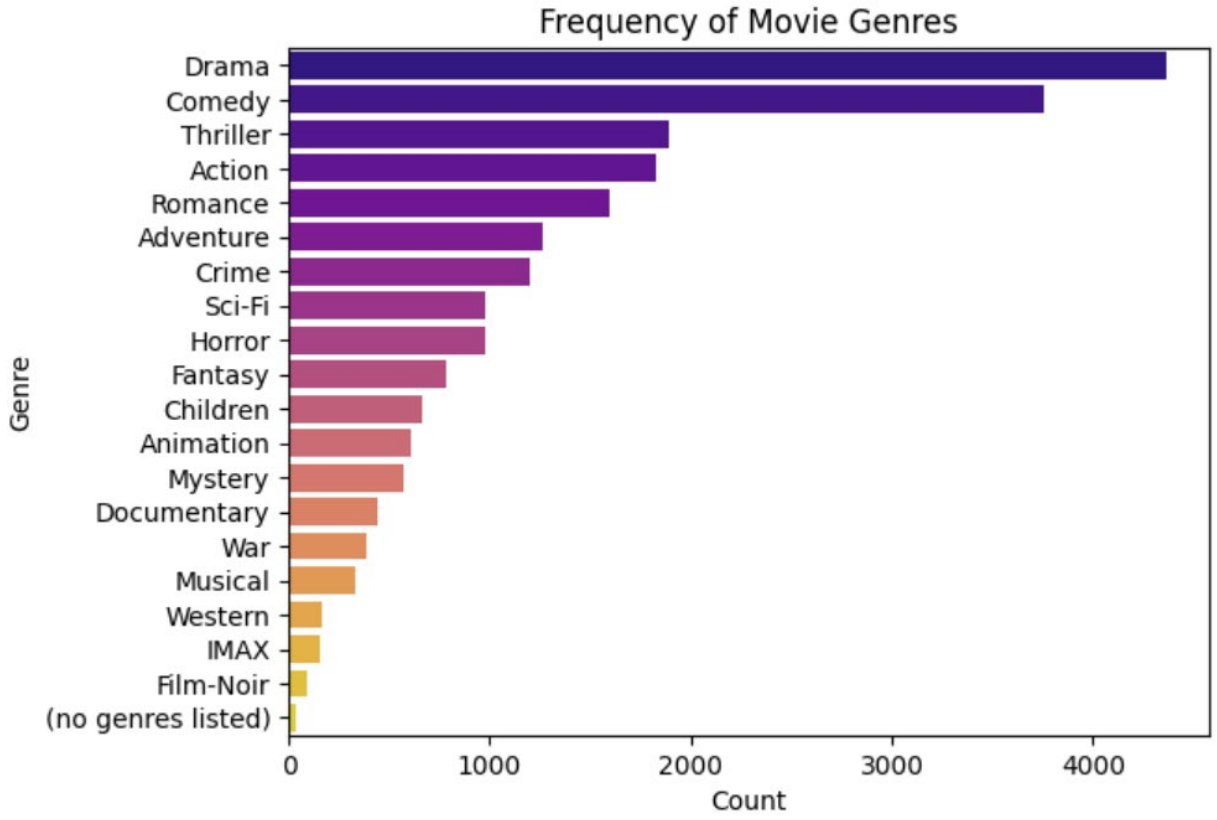


After applying the Bayesian average to the entire dataset, we observe that "The Shawshank Redemption," "The Godfather," and "Fight Club" are the highest-rated movies. This outcome is logical, given that these films are critically acclaimed. Conversely, applying Bayesian averaging reveals that "Speed 2: Cruise Control," "Battlefield Earth," and "Godzilla" are among the lowest-rated movies, suggesting that "Gypsy" is not as poorly rated as initially perceived. The results gain more credibility after applying the Bayesian Average, as the weighted calculation helps identify more relevant movies, thereby providing a more validated assessment.

Genre Frequency Analysis

In our analysis of the movie dataset, we have developed a frequency table for movie genres, which allows us to analyze the distribution of different genres in our dataset. The bar graph illustrates the frequency of each genre, highlighting the most and least common categories.

The frequency table reveals that Drama is the most common film genre, followed by Comedy and Thriller. These genres dominate the dataset, indicating their popularity among users. Conversely, Western and Film-Noir are the least common genres, suggesting they have a more niche audience.



Collaborative Filtering

User-Based - Collaborative Filtering

User-based collaborative filtering operates by identifying users with similar tastes in products or movies to the current user. To implement collaborative filtering, several steps are necessary. The initial step involves transforming our data into a user-item matrix, where rows represent users and columns represent movies. This configuration is illustrated below:

userId	movie_id1	movie_id2	...	movie_idn
1	rating1	rating2	...	...
2	rating1	rating2	...	...
...	...	...	...	...
n	...	...	...	...

To achieve this transformation, we need a function that generates a sparse matrix along with four mapping dictionaries. These include the **movie mapper**, which maps movie IDs to movie indices, and the **user mapper**, which maps user IDs to user indices. Additionally, a **movie inverse mapper** maps movie indices back to movie IDs, and a **user inverse mapper** maps user indices back to user IDs.

This approach ensures that the data is structured appropriately for collaborative filtering, facilitating the identification of similar users based on their interactions with various items.

Sparsity Evaluation

The sparsity of the user-item matrix is assessed to confirm that there are enough interactions for generating meaningful recommendations. The non-zero elements, or stored elements, represent the number of ratings in our dataset. For effective collaborative filtering, a sparsity level of 99.5% or lower is preferred. A high sparsity level indicates an extremely sparse user-item interaction matrix, with very few interactions recorded.

In our case study, a sparsity level of 98.3% strikes a balance between having sufficient data for accurate recommendations and maintaining computational efficiency.

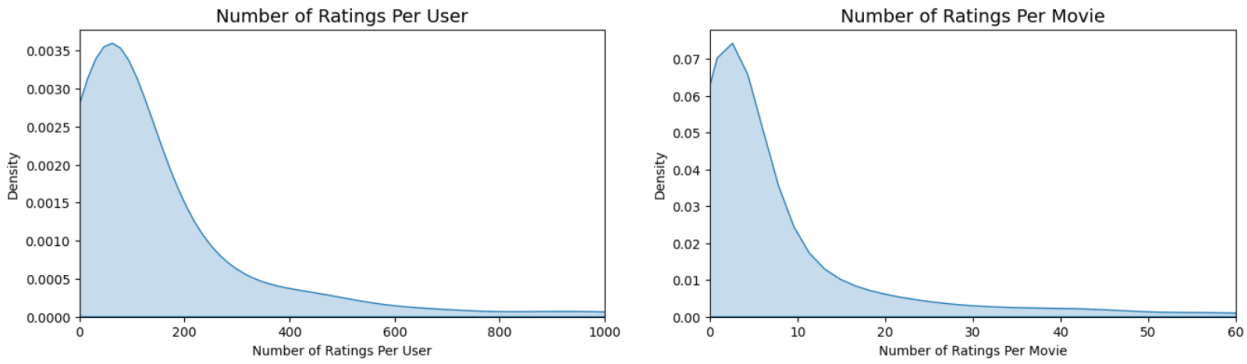
Insights from User and Movie Ratings

User Activity

- The most active user has rated 2,698 movies, indicating a high level of engagement with the platform. This user provides a rich dataset for understanding comprehensive movie preferences.
- The least active user has rated only 20 movies, which is significantly lower. This highlights a common challenge in recommender systems: dealing with users who have sparse interaction data.

Movie Ratings

- The most rated movie has received 329 ratings, suggesting it is widely viewed and possibly popular among users. Such movies can serve as anchor points in collaborative filtering models due to their extensive interaction data.
- The least rated movie has only 1 rating, which poses a challenge for recommendation systems as it provides minimal data to infer preferences or similarities.



The two graphs illustrate that the majority of users rate only a limited number of items, while most movies receive only a handful of ratings. This observation underscores the common issue of sparsity in user-item interaction datasets, where a small subset of users and movies generates the bulk of ratings.

Normalization

Normalization is applied to remove systematic biases from the ratings, ensuring that recommendations reflect genuine user preferences. Users may have different rating scales, and items may receive consistently high or low ratings due to factors unrelated to their quality. By normalizing the ratings, we can make the data more suitable for collaborative filtering. This process ensures that the recommendations are based on genuine user preferences rather than biases, leading to more accurate and fair recommendations.

Model Implementation

We use the k-Nearest Neighbors (kNN) algorithm with cosine similarity to find movies similar to a given movie.

In our movie recommender system, we employ collaborative filtering to suggest movies based on user preferences. A key component of this approach is the find_similar_movies function, which identifies the top-k similar movies to a given movie using the k-Nearest Neighbors (kNN) algorithm. The function first transposes the user-item matrix X and retrieves the vector corresponding to the specified movie_id. It then initializes and fits a kNN model using the "brute" algorithm and the specified distance metric (e.g., cosine similarity). The function finds the k+1 nearest neighbors to the movie vector, excluding the movie itself, and maps the indices of these neighbors back to their original movie IDs using the movie_inv_mapper dictionary.This implementation leverages the kNN algorithm to efficiently compute similarities between movies, allowing us to recommend films that are closely related in terms of user ratings. By using cosine similarity, we ensure that the recommendations are based on the angle between rating vectors, which is particularly useful in high-dimensional spaces where the magnitude of vectors is less informative.

As we previously described in the literature review section, the **cold-start problem** occurs in collaborative filtering when new users or items with no interactions are excluded from the recommendation system. This happens because collaborative filtering relies solely on user-item

interactions within the utility matrix. Content-based filtering can address this issue by generating recommendations based on user and item features.

Content-Based Filtering

In our analysis, we utilize a cosine similarity matrix to evaluate the similarities between different movie genres. This approach allows us to measure the angle between genre vectors, providing insights into how closely related various genres are based on their features.

Methodology for Calculating Cosine Similarity

Cosine similarity is a metric used to measure how similar two vectors are in an inner product space.

Vector Representation: The first step is to represent the items (e.g., documents, products) as vectors. In text analysis, this is often done using techniques like Term Frequency-Inverse Document Frequency (TF-IDF) vectorization, which converts text data into numerical vectors that capture the importance of words within documents [8].

Cosine Similarity Formula: The cosine similarity between two vectors **A** and **B** is calculated using the formula:

$$\text{Cosine Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Where:

- **A**·**B** is the dot product of the vectors
- ‖**A**‖ and ‖B‖‖**B**‖ are the magnitudes (or Euclidean norms) of the vectors.

The cosine similarity between two vectors results in a value between -1 and 1:

- **1 (*θ* =0∘)**: This indicates that the vectors are identical, meaning they point in the same direction.
- **0 (*θ*=90∘)**: This indicates that the vectors are orthogonal, meaning they are at right angles to each other and have no similarity.
- **-1 (*θ*=180∘)**: This indicates that the vectors are completely opposite, meaning they point in exactly opposite directions.

These interpretations are based on the angle ***θ*** between the two vectors, where cosine similarity is essentially the cosine of this angle.

Cosine Similarity Matrix for Movie Genres

To construct the cosine similarity matrix, we first represent each movie by its genre attributes. The genres are encoded into a feature matrix, where each row corresponds to a movie and each column represents a genre. The presence or absence of a genre is indicated by binary values.

Insights from Cosine Similarity Matrix

- **Similarity Measurement**: The cosine similarity matrix quantifies the similarity between genres by calculating the cosine of the angle between their corresponding vectors. A higher cosine similarity value indicates that two genres share more common features, suggesting a closer relationship.
- **Genre Clustering**: By analyzing the cosine similarity matrix, we can identify clusters of genres that frequently co-occur in movies. For example, genres like Drama and Thriller might exhibit high similarity due to their common thematic elements.

For our scenario, the cosine similarity scores are used to find the top 10 most similar movies to "Forrest Gump" based on their genre features.

```
Because you watched Forrest Gump (1994):
1730           Life Is Beautiful (La Vita è bella) (1997)
2262                 Train of Life (Train de vie) (1998)
6296    Tiger and the Snow, The (La tigre e la neve) (...
6624    I Served the King of England (Obsluhoval jsem ...
3                               Waiting to Exhale (1995)
10                        American President, The (1995)
47                               Mighty Aphrodite (1995)
52                     Postman, The (Postino, Il) (1994)
83                               Beautiful Girls (1996)
165                       Something to Talk About (1995)
```

This analysis of the cosine similarity matrix between movie genres provides valuable insights into the relationships between different genres, informing both the recommendation process and the exploration of diverse movie options.

Conclusion

This exploration into recommender systems offers an engaging journey into their pivotal role in enhancing user experiences across various platforms, such as social media, e-commerce, and healthcare. Utilizing the MovieLens dataset, we navigated through the processes of data loading, exploration, and visualization, uncovering patterns in movie ratings and identifying popular films. The project delved into user-based collaborative filtering, illustrating how to construct a user-item matrix, evaluate sparsity, and normalize ratings. Additionally, it briefly addressed the cold-start problem with content-based recommendations by using genre features to compute cosine similarity and identify similar movies. This experience sparks curiosity and encourages further investigation into the mechanics of recommendation systems, making it a valuable resource for application in other fields. Moreover, recommendations must remain timely and relevant as new content is continuously produced to ensure their effectiveness. Keeping pace with the influx of fresh material is essential for recommender systems to maintain user engagement and satisfaction.

References

[1] A. Handžić, "Online evaluation of recommender system with MovieLens dataset," Faculty of information Technologies Pan-European University, 2016.

[2] D. Bahl, V. Kain, A. Sharma and M. Sharma, "A novel hybrid approach towards movie recommender systems," *Journal of Statistics and Management Systems,* vol. 23, pp. 1049 - 1058, 2020.

[3] S. N. Salloum and D. Rajamanthri, "Implementation and Evaluation of Movie Recommender Systems Using Collaborative Filtering," *Journal of Advances in Information Technology,* 2021.

[4] N. a. G. S. Shrivastava, "Analysis on Item-Based and User-Based Collaborative Filtering for Movie Recommendation System," *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT),* pp. 654-656, 2021.

[5] B. M. Sarwar, G. Karypis, J. A. Konstan and J. T. Riedl, "Application of Dimensionality Reduction in Recommender System -- A Case Study," GroupLens Research Group / Army HPC Research Center.

[6] Y. Ji, A. Sun, J. Zhang and C. Li, "On Offline Evaluation of Recommender Systems," *ArXiv,* vol. abs/2010.11060, 2020.

[7] P. Masurel, "Of Bayesian Average and Star Ratings," 17 03 2013. [Online]. Available: https://fulmicoton.com/posts/bayesian_rating/. [Accessed 7 08 2024].

[8] I. Lumintu, "Content-Based Recommendation Engine Using Term Frequency-Inverse Document Frequency Vectorization and Cosine Similarity: A Case Study," *2023 IEEE 9th Information Technology International Seminar (ITIS),* pp. 1 - 6, 2023.