Collaborative Filtering for Movie Recommendation Systems

Sahithi Kodali
kodali1@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Sanskriti Motwani
smotwani@purdue.edu
Purdue University
West Lafayette, Indiana, USA

ABSTRACT

The film industry wields significant influence over cultural perceptions and global audiences through its diverse array of genres and narratives. Our aim is to design recommendation systems tailored to individual user preferences, leveraging historical data of similar users and movie-watching patterns. By categorizing movies based on attributes such as genre, directorial style, thematic elements, and audience reception, our aim is to unravel the nuanced patterns underpinning film recommendation systems. Our approaches encompass content and collaborative filtering methodologies (categorized on user-based and item-based recommendations) using the implicit and explicit feedback attributes within the movie dataset. We implemented three distinct models based on TF-IDF, kNN, and dense neural networks. These models were tasked with predicting the top k movies likely to resonate with the user, alongside estimating a rating indicative of the user's inclination to watch, where applicable. To evaluate and analyze the efficacy of these models, qualitative and quantitative metrics like genre comparison, cosine similarity, and the hit-ratio are utilized.

1 INTRODUCTION

This project aims to develop movie recommendation systems that can delve into the intricacies of user preferences and film attributes. Our objective is to develop a deeper understanding of the factors influencing movie recommendations and develop models that can give better accuracy of movie recommendations based on user interests. While there are many filtering methods for recommendations, this project is focused on exploring and implementing collaborative filtering approaches that can include a diverse range of feature information to make recommendations accordingly. By analyzing datasets comprising movie metadata and user ratings, we seek to predict and analyze the nuanced biases inherent in movie recommendations. Through this analysis, we aim to uncover how biases shape the recommendation algorithms and influence user choices.

1.1 Importance and Motivation

In today's digital age, movie recommendation systems hold significant sway over the cinematic experiences of audiences worldwide. These systems have the potential to impact the viewing habits and preferences of millions, shaping the cultural landscape and driving the success of films.

It is important to recognize that movie recommendation systems, like any other algorithmic recommendation system, may include biases. The algorithms used in these systems may be influenced by various factors, including user demographics, historical preferences, and promotional strategies. Consequently, the recommendations provided by these systems may not always align with the diverse tastes and preferences of individual users.

Our aim is to develop a movie recommendation system that balances user preferences but also encourages diversity in film experiences. By mitigating biases and promoting a variety of perspectives, our goal is to create a recommendation environment that enhances users' preferences and encourages openness to new narratives.

1.2 Literature review

With the rise of data, recommendation systems have evolved to develop efficient systems that filter recommendations based on the search history of users and items relevant to the search results to improve the quality of movie streaming. There are usually three types of recommendation systems, namely:

- Demographic Filtering: This type of recommendation system filters content based on demographic information such as age, gender, location, movie popularity, and genre. It aims to personalize recommendations according to the general user's demographic profile and does not include the specific interests of users.
- Content-based Filtering: This approach recommends items similar to those that a user has liked or interacted with in the past. It analyzes the attributes or features of items and recommends other items with similar characteristics. Content-based filtering does not rely on user demographics but rather focuses on item characteristics.
- Collaborative Filtering: Collaborative Filtering (CF) recommends items based on the preferences and behavior of similar users. It identifies users with similar tastes or preferences and recommends items that they have liked or interacted with. Collaborative filtering does not require explicit item features but relies on user-item interactions for recommendation generation.

There are many models implemented in previous studies related to these filtering methods, but the recent focus has been on collaborative and hybrid systems that include content and collaborative filtering.

CF excels in personalizing movie recommendations by leveraging the preferences and behaviors of a wide user base. This approach naturally encourages diversity, as it exposes users to movies liked by others with similar tastes, directly addressing our aim to balance personal preferences with a broadened cinematic experience. By focusing on the collective rather than individual demographics or historical data, CF inherently challenges biases in recommendations. This methodology aligns perfectly with our goal of mitigating

biases and promoting a variety of perspectives, ensuring our recommendation system enriches users' film explorations with new narratives and diverse viewpoints.

CF is categorized into two primary types: memory-based and model-based. Memory-based algorithms also referred to as neighbor-based algorithms, utilize the full database of ratings gathered by the service provider or vendor and use similarity/ranking metrics to measure their performance. On the other hand, model-based algorithms distinguish themselves from memory-based ones by first leveraging the database to learn or estimate a model, which is then used for making predictions. Typically, model-based algorithms tend to achieve greater accuracy compared to their memory-based counterparts [6].

The memory-based k-Nearest Neighbors (kNN) model is a good standard for a baseline model, but the recent state-of-the-art has been in model-based deep learning (DL) algorithms, specifically Recurrent Neural Networks (RNN) and graph-based neural networks (GNN). Hybrid methods that combine the perks of both memory-based and model-based are also on the rise in today's research.

A popular technique for CF is matrix factorization, which uses methods like Singular Value Decomposition (SVDs) and Stochastic Gradient Descent (SGD). Machine learning (ML) models like kNNs that include ontology and sequential patterns [3], Support Vector Regression (SVR), and other baseline ML models have been implemented and yielded good results when compared against their accuracy, F1-score, and other comparative metrics [5]. Additionally, a range of DL models have been experimented with, including (RNNs), Convolutional Neural Networks (CNNs), and Autoencoders, each offering unique strengths in capturing complex patterns within the data. In the recent state-of-the-art, the RNN-based Long short-term memory (LSTM) model, when combined with opinion mining techniques, has been shown to enhance the recommendation accuracy [4]. Along with that, recent graph-based models like Two-stage Siamese GNN [1] that tries to learn user and item side interaction information and knowledge-enhanced user-centric GNN [2] have shown significant improvement in recommendation accuracy.

2 DATASETS

The approaches and models discussed in Section 3 are implemented using two datasets rich in movies metadata: MovieLens 20M Dataset[1] and the Movies Dataset [2].

2.1 MovieLens 20M Dataset

The MovieLens 20M dataset is a comprehensive collection of movie ratings provided by GroupLens, a research lab at the University of Minnesota. It is widely used in the field of machine learning for building and evaluating recommender systems. The dataset encompasses 20 million ratings and 465,000 tag applications on 27,000 movies by 138,000 users approximately. This data spans a period from January 1995 to March 2015, providing a rich historical view of user preferences over two decades.

Within the MovieLens 20M dataset, the two key files used are 'movies.csv' and 'ratings.csv'. The 'movies.csv' file includes details about the movies, such as movie IDs (which uniquely identify each movie), titles, and genres. The genres are provided as a pipe-separated list, facilitating easy parsing and categorization of films into genres like Action, Comedy, Thriller, etc. On the other hand, the 'ratings.csv' file contains user ratings and includes attributes like user ID, movie ID, rating, and timestamp. Ratings are on a scale of 0.5 to 5 stars, recorded in half-star increments. The timestamp helps in analyzing the temporal dynamics of user preferences, making it possible to study changes in watching patterns over time.

2.2 The Movies Dataset

The Movies dataset consists of metadata for 45,000 movies listed in the Full MovieLens Dataset, which includes 26 million ratings from 270,000 users. The TMDB 5000 Movie Dataset is a comprehensive collection of movie data originally sourced from The Movie Dataset. This dataset includes detailed information on around 5,000 movies and is widely used for data analysis, machine learning, and recommendation system projects due to its rich metadata.

Contained within this dataset are several key attributes for each film, including but not limited to the movie's title, budget, revenue, release date, languages, production countries, and runtime. Each movie entry also contains a detailed description of the plot (overview), as well as metadata like genres, tags (keywords), and cast and crew details. This extensive set of features makes it ideal for exploring various aspects of movie data and how they affect user preferences as seen in approaches discussed further.

3 METHODOLOGY AND IMPLEMENTATION

Recommendation systems are generally built from two types of feedback, implicit and explicit. Implicit feedback is the data collected indirectly from user interactions, while explicit feedback is attained by direct and quantity data that is collected from the user. For example, in the movie data used, a user watching/interacting with a movie is implicit feedback, and the user liking/commenting/rating a movie is explicit feedback. In most cases, there is an abundant amount of implicit feedback, so most of the systems are built using this data along with explicit feedback data if needed. One disadvantage that comes with implicit feedback is assumptions made about the negative preferences of users when building recommendation systems. For example, if a user interacts with a movie, then we can definitely say the interaction is positive, but not everything that hasn't been interacted has a negative preference. However, to get this data while building recommendation systems, an approach of negative sampling is followed to include data with negative feedback, i.e, where there is no interaction between a movie and the user. In this Section, the algorithms and models implemented using TF-IDF, kNN, and neural network architectures are discussed.

3.1 TF-IDF

The first CF movie recommendation system implemented utilizes the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm by considering the content-based similarities between movies. Specifically, TF-IDF is applied to the genre descriptions of movies, where each genre is treated as a term. This method transforms the

[1] https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset
[2] https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset

categorical genre data into a numerical format, creating a TF-IDF matrix where each row represents a movie and each column corresponds to a genre. The TF-IDF for a genre j in a movie i is calculated as:

$$\text{TF-IDF}_{i,j} = \text{TF}_{i,j} \times \text{IDF}_j$$

where, $\text{TF}_{i,j}$ represents the term frequency of genre j in movie i, and IDF_j is the inverse document frequency of genre j calculated as:

$$\text{IDF}_j = \log(\frac{N}{n_j})$$

where, N is the total number of movies, and n_j is the number of movies that include genre j.

The TF-IDF values in this matrix reflect how important a genre is to a movie relative to the entire dataset, with higher values indicating greater importance. The cosine similarity is then computed based on the TF-IDF matrix to measure the content-based similarity between all pairs of movies. The similarity is computed as:

$$\text{similarity}(\vec{v}_i, \vec{v}_j) = \frac{\vec{v}_i \cdot \vec{v}_j}{\|\vec{v}_i\| \|\vec{v}_j\|}$$

where, $\vec{v}_i$ and $\vec{v}_j$ are the TF-IDF vectors of movies i and j respectively. This similarity metric provides a value between 0 and 1, where 1 signifies a perfect similarity.

During the recommendation phase, each movie's average content similarity score is calculated with respect to the movies watched by the user, and this score is combined with collaborative filtering scores (derived from user ratings and user similarity) to generate a final recommendation score. The combination is given by:

$$\text{Final Score} = \alpha \times \text{Content Similarity}$$

$$+(1 - \alpha) \times \text{Collaborative Filtering Score}$$

where, α is a parameter that balances the weight of content-based and collaborative filtering components. This hybrid approach leverages both the user's explicit interactions (ratings) and implicit preferences (content similarity) to recommend movies, thus providing a more personalized and contextually relevant set of recommendations.

For item-based movie recommendations, TF-IDF can be similarly applied to enrich the recommendation logic. In this approach, instead of focusing on user similarities, the system emphasizes similarities between items (movies) based on their content. By applying TF-IDF to the movie genres, each movie is represented as a vector in a multidimensional genre space. Cosine similarity is then used to calculate the similarity scores between all pairs of movies based on these vectors. This similarity matrix forms the core of the item-based recommendation system, where recommendations for a user are made by identifying movies that are most similar in content to those the user has rated highly. This method can particularly benefit users with specific genre preferences, as it directly suggests movies that share similar genre characteristics, thereby likely aligning more closely with the user's tastes and preferences. This item-based approach leverages content similarity independently of user interactions, making it robust against problems like cold start and sparsity, which are common in collaborative filtering systems.

3.2 kNN

The k-Nearest Neighbors (kNN) algorithm is a simple yet powerful machine-learning technique used in our movie recommendation system to identify movies with similar characteristics. kNN operates on the principle of proximity, predicting similarity between movies based on the closest items in the dataset. It functions by mapping each movie into a feature space, where each dimension represents a characteristic of the movie, such as genre or cast. When a recommendation is needed, kNN identifies the 'k' closest movies in this space to any given movie, leveraging these relationships to suggest similar films. The similarity between movies is determined by calculating the distance between their points in the feature space, with closer points indicating more similarity. This method is particularly effective for item-based recommendations as it allows the system to suggest movies that share significant similarities in content and style, thus enhancing the recommendation quality by focusing on the intrinsic properties of the movies themselves. This similarity is calculated as,

$$S(a, b) = \frac{a \cdot b}{\|a\| \|b\|}$$

where, a and b are the two movies being compared.

The kNN model operates by leveraging several key attributes of movies, namely genres, cast, and keywords. Each of these attributes plays a critical role in shaping a viewer's choice and perception of a film. For instance, genres offer a straightforward classification that helps viewers choose films aligned with their tastes or moods. The presence of particular actors or directors can influence the choice significantly, as established stars and directors often carry a quality expectation and a specific artistic style. Keywords serve as precise descriptors of the plot and thematic elements, providing insights into the unique aspects of the narrative.The selection of these attributes is strategic; they are known to strongly influence user preferences and are thus crucial for the system's accuracy. Genres, cast, crew, and keywords are encoded into binary vectors using one-hot encoding, to transform the categorical information into a format that can be processed by computational models.

Once the movies are represented as vectors, the model calculates the similarity between the movies using cosine similarity. This measure calculates the cosine of the angle between two vectors in a multi-dimensional space, where a cosine value of 1 indicates that the vectors are identical, and a value of 0 suggests no similarity.

$$\text{similarity score} = \frac{\text{genreSim} + \text{scoreSim} + \text{directSim}}{3.0}$$

where, similarity score is the overall cosine similarity.

By computing these values, the kNN model can assess which movies are most similar to those that a user has previously rated highly. The model then recommends the top movies that have the highest similarity scores, ensuring that the recommendations are tailored to each user's unique preferences. This sophisticated combination of kNN and cosine similarity provides a robust foundation for our movie recommendation system, enabling it to deliver personalized, relevant, and enjoyable movie suggestions to its users.

3.3 Neural Networks

While the traditional recommendation systems are built using methods like TF-IDF and KNN, deep learning-based recommendation systems represent the state-of-the-art in the field.

In this project, two user-based collaborative filtering neural network models are implemented - one based on explicit feedback and, the other uses explicit to convert to implicit feedback. A key challenge in building recommendation systems lies in effectively representing users, movies, and other features to capture their relationships. To address this, embeddings - low-dimensional vector representations of features - are utilized. User and movie IDs are represented as embeddings using embedding layers within a user and movie model defined in the architecture. These embeddings are determined based on the closeness of user preferences, i.e., ratings, thus capturing meaningful relationships between users and movies. The embedding size can be adjusted based on model complexity and computational resources.

For the model utilizing explicit feedback, the Tensorflow Recommenders (TFRS) [3] library of Tensorflow, built on Keras, is employed. This model predicts ratings, ranging from 1 to 5, using user and movie embeddings to recommend the top k movies for a user. This collaborative approach considers various user preferences in determining recommended movies. The user and movie (or item) vectors are fed to embedding layers to get the small dense vectors. These embeddings are concatenated to pass through a series of fully connected dense layers to predict the rating. The architecture of this model is illustrated in Figure 1.

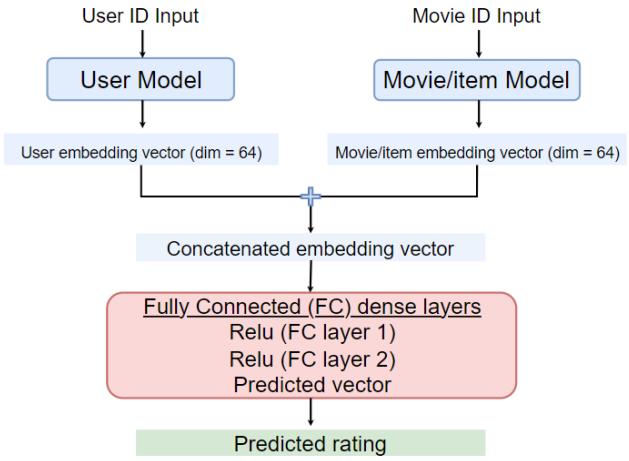

Figure 1: Architecture overview of explicit feedback model

The other model, which converts the explicit feedback to implicit, is built using PyTorch lightning [4] for the ease of training the model. In this approach, the input user and movie IDs are converted to embeddings using the embedding layer defined. For representing the interaction of a user with a movie, every movie that is rated by a user is determined as a positive interaction, which is labeled as 1. Since the data does not have a negative interaction representation, negative sampling is performed to create 5 negative samples for every positive rating sample. This is done by assuming that no user-movie interaction implies the movie is likely not watched, although this may not hold true in all real-world scenarios. However, it serves as a pragmatic approximation that generally proves effective. The data is split into train and test using the leave-one-out methodology, where for each user the most recent review is used for test data and the rest for training data, using the timestamp feature. The user and movie/item vectors are fed to embedding layers to attain small dense vectors. These embeddings are concatenated to pass through a series of fully connected dense layers and predict a vector, which is passed to sigmoid function to yield positive and negative interactions as 1 or 0, as seen in Figure 2.

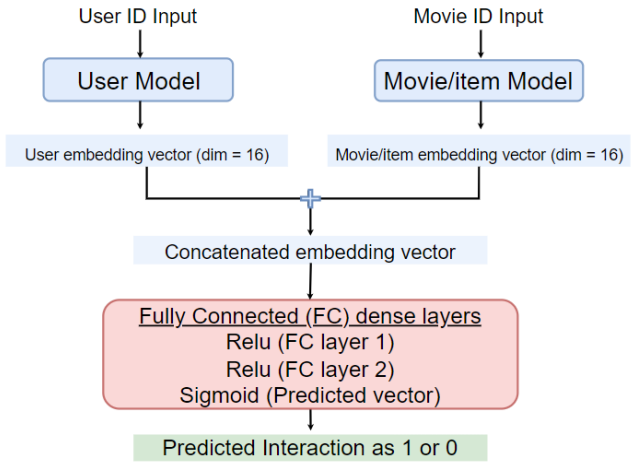

Figure 2: Architecture overview of implicit feedback model

Both the models predict the top k movies for a user, but the approaches and feature attributes used are different. Additionally, the model using explicit feedback can predict the rating a user might give to a movie. These results are evaluated further in Section 4.

4 RESULTS AND ANALYSIS

In this Section, the results of the models implemented are discussed and evaluated using various metrics.

4.1 Item-Based CF

In this item-based collaborative filtering approach using the MovieLens 20M Dataset, TF-IDF vectorization of movie genres was employed to create item profiles. This technique calculates the importance of each genre within the dataset, providing a numerical weight to each genre according to its frequency and relevance across all movies. While effective in distinguishing genres, TF-IDF primarily captures frequency rather than the intricate semantic connections between films. As a result, movie representations might have been somewhat superficial, without delving into deeper narrative or thematic links as seen in figure 3. This method, utilizing cosine similarity to measure the distance between genre vectors, might limit the system's ability to offer personalized recommendations that resonate more profoundly with individual user tastes.

[3] https://www.tensorflow.org/recommenders
[4] https://lightning.ai/docs/pytorch/stable/

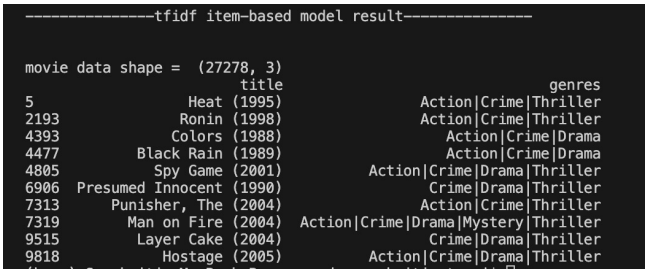

Figure 3: Item-based tf-idf result

The second method, a KNN model using cosine similarity, similarly focused only on genres like the first TF-IDF approach. However, it differed in the method of handling these genres. While the TF-IDF model weighted genres based on their frequency and relevance across the dataset, the KNN model treated genre features equally, calculating similarity directly through cosine distance between genre vectors without additional weighting. This straightforward approach surprisingly outperformed the TF-IDF method as can seen in figure 4, possibly due to its simplicity and directness in measuring genre-based similarities, which might have offered more clear-cut and consistent recommendations.

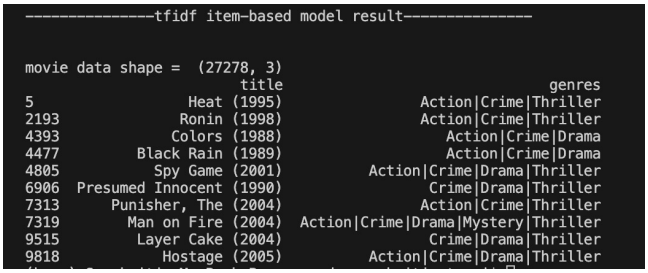

Figure 4: kNN model results

The third model we tested was an enhanced version of the KNN approach, incorporating not only genres but also cast and director information, aiming to utilize cosine similarity to capture a more detailed perspective of each movie. Despite initial expectations that this richer dataset would improve recommendation performance by identifying deeper content and stylistic similarities, the results did not show an improvement as seen in figure 5. However, the practical application was hindered by the smaller subset of the dataset where complete cast and director data were available, reducing the model's effectiveness. Additionally, the assumption that films with similar casts or directors would share thematic or genre consistency did not hold universally, as actors and directors often work across diverse film types. This diversity might have introduced noise into the similarity measures, complicating the detection of true cinematic parallels and thus not enhancing the model as anticipated.

4.2 User-Based CF

In the user-based TF-IDF model, recommendations are finely tuned to user preferences, as demonstrated by the selection of classic films like "The Philadelphia Story" and "The Graduate" as shown in

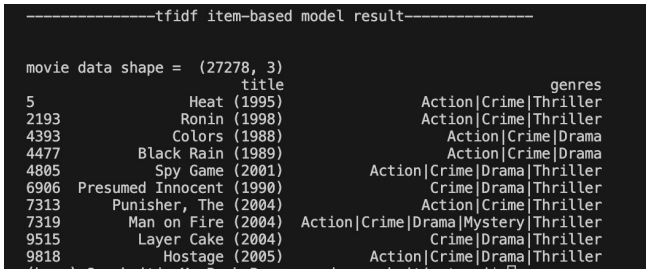

Figure 5: Enhanced kNN model results

Figure 6. The model combines user rating profiles with thematic content, reflecting a blend of genre considerations and viewer patterns. Notably, the weighted scores are uniform across the board, but the model discerns subtle differences in content scores, leading to a nuanced final ranking. With a focus on comedy, drama, and romance genres, the top picks exhibit a common thread in their genre composition, signifying that users' taste for multifaceted narratives in these genres is well captured and catered to by the recommendation algorithm.

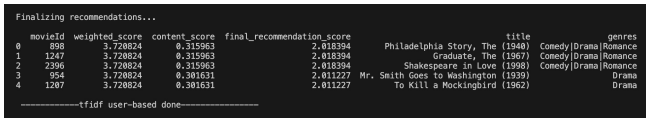

Figure 6: Sample of top-5 movies for a user by tf-idf

The models implemented using neural networks that use implicit and explicit feedback have performed almost similarly, with a little edge for the model using explicit feedback. This is as expected since the explicit feedback model is not based on any assumptions. However, this might fail when the explicit data available is scarce, so it is good practice to use implicit feedback foresaking the impact of generalized assumptions.

In the model built on implicit feedback, the evaluation metric employed is the Hit Ratio. This metric compares the predicted recommended movies for a user with the single movie that the user interacted with, as per the test data. If the interacted movie appears in the top-k predicted movies, it is considered a hit. Generally, if at least one movie achieves a hit in the test data, the metric signifies a successful recommendation with at least a 50% success rate of other recommended movies.

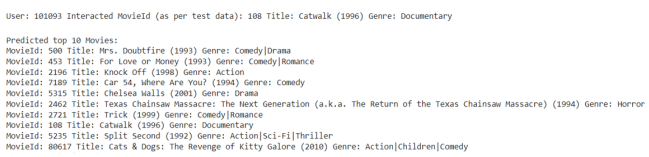

Figure 7: Sample of top-10 movies recommended for a user and hit ratio by the implicit feedback-based model

In Figure 7, the top 10 recommended movies for a user and the corresponding hit ratio achieved for 27,698 user ID and Movie ID

pairs in the test data are displayed. The hit ratio of 0.86 indicates that at least 86% of users have the movie they watched (according to the test data) included in the recommended top 10 movies. We can clearly see that the user-watched movie "Catwalk" is predicted as a recommended movie in the top 10 movies, which is considered as a hit.

In the model built on explicit feedback, the top-k movies of a user are predicted along with predicting the rating a user might give to a movie. In Figure 8, the top 5 recommended movies for a user can be seen.

```
Top 5 recommendations for user 123:
```

	original_title	genres	overview
1	Un long dimanche de fiançailles	Drama	In 1919, Mathilde was 19 years old. Two years ...
2	Dog Day Afternoon	Crime, Drama, Thriller	A man robs a bank to pay for his lover's opera...
3	The Greatest Story Ever Told	Drama, History	All-star epic retelling of Christ's life.
4	Kurz und schmerzlos	Drama, Thriller	Three friends get caught in a life of major cr...
5	Anatomie de l'enfer	Drama	A man rescues a woman from a suicide attempt i...

Figure 8: Sample of top-5 movies recommended for a user by explicit feedback-based model

Observing this we can see that the Drama genre is most liked by the user '123'. Now, we cross-evaluate its correctness by predicting the rating user '123' gives to two movies - 'Star Wars' in the genre 'Fiction|Fantasy..' and 'Dog Day Afternoon' in the genre 'Crime|Drama|Thriller'. As shown in Figure 9, the rating for the movie 'Dog Day Afternoon' in the drama genre is much higher than that for the movie 'Star Wars' in the genre of Fiction. This can also be cross-evaluated using Hit Ratio by storing the test dataset based on timestamp, but this cross-evaluation is shown to justify the recommendation accuracy.

```
1 predict_rating(123,'Star Wars')
```

```
Predicted rating for Star Wars: 3.3033976554870605
```

```
1 predict_rating(123,'Dog Day Afternoon')
```

```
Predicted rating for Dog Day Afternoon: 4.384540557861328
```

Figure 9: Predicted ratings by explicit feedback-based model

4.3 Analysis

In our analysis, we observed that user-based collaborative filtering yielded superior results. This can be attributed to the extensive range of user preferences captured within the dataset, which are more effectively modeled through user-based methods. These methods excel in identifying subtle variations in user rating patterns, which is particularly beneficial when user profiles are rich with ratings across many movies. Additionally, the use of TF-IDF for genre analysis, followed by the application of cosine similarity to these genre vectors, enriches the user-user similarity metrics. This can also be observed in the models implemented using neural networks, which exhibited better movie predictions than the baseline models.

On the other hand, item-based collaborative filtering seemed less effective, which is expected due to the sparsity of ratings across a vast array of movies, making it challenging to accurately compute item-item similarities. Furthermore, item-based methods might also suffer from a popularity bias, tending to recommend widely popular items and overlooking niche interests that might be highly rated by a specific subset of users. These factors, coupled with the dynamism in user preferences that user-based methods can adapt to more readily, explain the observed performance disparity and the enhanced performance of the user-based collaborative filtering approaches.

5 FUTURE WORK

As discussed earlier, RNN-based models represent the current state-of-the-art recommendation systems. However, due to time constraints, our implementation focused solely on neural network models using fully connected dense layers, without incorporating any RNN layers. Initially, we explored grouping movies by each user and intended to utilize RNNs to capture temporal patterns in movie-watching sequences. Although we commenced work on this model, we were unable to complete it within the project timeframe. Therefore, our future plans involve successfully implementing this RNN-based model to assess its performance against the best neural network results obtained thus far.

Additionally, we aim to enhance the recommendation system by incorporating more features such as movie cast and crew information, movie overviews, and potentially other metadata. This expansion will allow us to evaluate how these additional features impact movie recommendation patterns, potentially leading to more accurate recommendations.

Furthermore, we intend to explore the use of pre-trained models to assess movie recommendations while reducing computational complexity. This approach could streamline model training and inference processes, thereby improving efficiency without sacrificing recommendation quality.

REFERENCES

[1] Zhiwen JING, Yujia ZHANG, Boting SUN, and Hao GUO. Two-stage recommendation algorithm of siamese graph convolutional neural network. *Journal of Computer Applications*, 44(2):469, 2024.

[2] Guangyi Liu, Quanming Yao, Yongqi Zhang, and Lei Chen. Knowledge-enhanced recommendation with user-centric subgraph network, 2024.

[3] Ghulam Mustafa, Naveed Ahmad Jhamat, Zeeshan Arshad, Nadia Yousaf, Md. Nazmul Abdal, Mohammed Maray, Dokhyl Alqahtani, Mohamad Amir Merhabi, Muhammad Abdul Aziz, and Touseef Ahmad. Ontocommerce: Incorporating ontology and sequential pattern mining for personalized e-commerce recommendations. *IEEE Access*, 12:42329–42342, 2024.

[4] Luong Vuong Nguyen. Collaborative filtering-based movie recommendation services using opinion mining. In *2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*, pages 1–5, 2024.

[5] Ayesha Siddique, M Kamran Abid, Muhammad Fuzail, and Naeem Aslam. Movies rating prediction using supervised machine learning techniques. *International Journal of Information Systems and Computer Technologies*, 3(1):40–56, 2024.

[6] Ruisheng Zhang, Qi-dong Liu, Chun-Gui, Jia-Xuan Wei, and Huiyi-Ma. Collaborative filtering for recommender systems. In *2014 Second International Conference on Advanced Cloud and Big Data*, pages 301–308, 2014.

APPENDIX

Code Repository

https://github.com/sahithikodali1/NLP_Project