Journal of Management and Service Science, 2024,
Vol. 04, Iss. 01, S. No. 066, pp. 1-11
ISSN (Online): 2583-1798

Movie Recommender System: Collaborative Filtering Methods

Abhishek Kumar Rai[1], Pooja Khanna[2], Pawan Singh[3]

[1,2,3]Department of Computer Science and Engineering, Amity University Uttar Pradesh, Lucknow, India
[1]abhishek008rai@gmail.com, [2]pkhanna@lko.amity.edu, [3]psingh10@lko.amity.edu

How to cite this paper: A. K. Rai, P. Khanna, P. Singh, "Movie Recommender System: Collaborative Filtering Methods," *Journal of Management and Service Science (JMSS)*, Vol. 04, Iss. 01, S. No. 066, pp. 1-11, 2024.

https://doi.org/10.54060/a2zjournals.jmss.66

Received: 22/011/2023
Accepted: 10/03/2024
Online First: 25/04/2024
Published: 25/04/2024

Abstract

In the current digital era, personalized movie recommendation systems have become essential tools for aiding users in navigating the vast array of available cinematic content. This paper offers an in-depth examination of the development and implementation of a Python-based movie recommender system, integrating content-based and collaborative filtering methodologies. Utilizing a comprehensive dataset encompassing diverse movie attributes and audience feedback metrics, our research aims to construct a resilient recommendation engine capable of delivering customized movie suggestions to users. Through a meticulous analysis and evaluation of various similarity metrics, including Pearson Correlation Coefficient, Spearman Rank Correlation Coefficient, Mean-Squared Distance, and Cosine Similarity, we assess the system's effectiveness in generating personalized recommendations. Our findings provide valuable insights into the strengths and weaknesses of each technique, thereby guiding future research and enhancements in movie recommendation systems.

Keywords

Collaborative filtering, Cosine Similarity, Mean-Squared Distance, mean absolute error.

1. Introduction

Within the fast-paced digital realm, personalized movie recommendation systems have become invaluable tools, enabling users to navigate the extensive collection of cinematic content. This paper offers a comprehensive examination of the development and implementation of a Python-based movie recommender system, integrating content-based filtering and collaborative filtering methodologies.

Utilizing a comprehensive dataset comprising movie attributes such as budgets, genres, production details, and audience feedback metrics, this research strives to construct a robust recommendation engine capable of providing customized movie suggestions to users. By integrating both content-based and collaborative filtering techniques, the system aims to enhance user experiences and satisfaction by delivering relevant and engaging movie recommendations [1].

This study's primary objectives are two-fold: first, to develop an efficient movie recommendation system leveraging the available data; and second, to evaluate the system's performance in generating personalized recommendations.

Through extensive analysis of the methodologies employed, including data pre-processing, algorithm development, and performance evaluation, this paper aims to contribute to the advancement of movie recommendation systems. The insights gained from the evaluation will inform discussions on the system's strengths, weaknesses, and potential areas for further research and improvement [2].

In the following sections, we delve into the research methodologies, presenting our approach to data pre-processing, algorithm development, and performance evaluation. Additionally, we explore the implications of our findings and propose future directions for enhancing the movie recommendation system.

2. Literature Review

In the domain of collaborative filtering research, several ground-breaking studies have supplied invaluable insights into the design, evaluation, and optimization of recommendation systems. Among these, Breese, Heckerman, and Kadie's (1998) work from Microsoft Research stands out for its empirical analysis of predictive algorithms for collaborative filtering. By examining the performance of memory-based and model-based strategies across diverse datasets, Breese et al. elucidated the strengths and limitations of different algorithms, offering crucial guidance for researchers and practitioners [1].

Further enriching the empirical understanding of collaborative filtering algorithms, Herlocker, Konstan, and Riedl (2002) performed a comprehensive analysis of design choices in neighborhood-based collaborative filtering [3]. Their research, published in Information Retrieval, scrutinizes aspects such as neighbourhood size and similarity metrics, providing invaluable insights into optimal algorithm configurations.

In the quest to enhance recommendation accuracy, Melville, Mooney, and Nagarajan (2002) proposed a content-boosted collaborative filtering approach. By integrating content-based features into the collaborative filtering framework, their work demonstrates the potential for enhancing recommendation quality by incorporating additional contextual information [4].

Collectively, these studies have significantly advanced our comprehension of collaborative filtering algorithms and their practical applications in recommendation systems [5]. By addressing key research questions and providing empirical evidence, they serve as foundational pillars in the field, guiding future research efforts and informing the development of more effective recommendation systems.

3. Background

3.1. Vocabulary

This paper examines algorithms employed in literature for movie recommendation systems. To facilitate understanding of these algorithms, it is crucial to establish key terminology:

- **User:** Refers to an individual for whom a record of past movie rating history is available.
- **Item:** Represents a movie with in the realm of the recommendation system.

- **Rating:** Denotes a user's preference for a movie, typically expressed on a scale from 0.5 to 5, with 5 indicating the highest level of preference.
- **Intersection Set:** For two users, denoted as u and v, the intersection set comprises the movies that both users have rated.
- **Active User:** Signifies the user for whom predictions of ratings are aimed for movies they have not yet viewed or rated.

3.2. Collaborative Filtering

Collaborative filtering, as explained by Extstrand et al., serves as a cornerstone in recommendation systems, where predictions and suggestions are derived based on the past ratings or behaviours of other users within the system. This paradigm operates on the fundamental assumption that users with similar tastes or preferences will exhibit comparable behaviours when interacting with items or content. Consequently, collaborative filtering techniques strive to identify and capitalize these patterns of similarity to generate personalized recommendations for users [6].

In the context of collaborative filtering, a common strategy is the neighbour-based method, as proposed by Altman et al., which seeks to identify a subset of users closely resembling the active user in terms of their past interactions with items. By identifying the k-nearest neighbours in the feature space, these algorithms aim to project the preferences of the active user based on the observed behaviours of their peers [7]. Consider the following hypothetical scenario:

Table 1. Collaborative filtering hypothetical scenario.

Movie	Genre	User 1	User 2	User 3	User 4	User 5
The Matrix	Action/Sci-Fi	5	4	5	?	3
Inception	Action/Thriller	4	?	5	3	4
Interstellar	Sci-Fi	?	5	3	4	?

Suppose User4 is the active user, with no rating provided for The Matrix. A proficient collaborative filtering algorithm would analyze the preferences of similar users, such as User1 and User3, who have shown a consistent liking for movies similar to The Matrix and Inception. Consequently, The Matrix would likely be recommended to User4 with a predicted rating of 5, aligning with the preferences of their neighbours.

Alternatively, if User5 is the active user, lacking a rating for Interstellar, the collaborative filtering approach would again identify similar users, such as User2 and User4, who have exhibited preferences aligned with Interstellar. Therefore, Interstella would be predicted to receive a high rating of 5 for User5, based on the observed behaviours of their neighbours.

Collaborative filtering encompasses two main variants: user-user collaborative filtering and item-item collaborative filtering.

While both approaches aim to identify patterns of similarity, this study specifically focuses on user-user collaborative filtering, where recommendations are generated based on the preferences of users similar to the active user.

4. Methods

4.1. Programming Tools

The movie recommendation engine was developed using Python programming language. The vast Python ecosystem was utilized, encompassing the scikit-learn module for machine learning tasks, NumPy for numerical calculation, and the Pandas

library for data processing. Additionally, Matplotlib and Seaborn were employed for data visualization to gain insights into the dataset.

4.2. Data

Two datasets were utilized in this study:

Dataset 1: Movie-Lens review dataset (ml-latest-small), augmented with additional data from IMDB and TMDB. It encompasses:

- Ratings: 200,000
- Movies: 20,000
- Users: 600
- Tags: 100,000

To ensure data integrity and comprehensiveness, the dataset was integrated with publicly available data from IMDB and TMDB. This integration enhanced the richness and diversity of the dataset, providing additional context and information about the movies.

Dataset 2: The Movie-Lens dataset, which includes user ratings and tags for films, is the second dataset. The user ID is anonymized but consistent across ratings.csv and tags.csv files. Additionally, movie IDs are consistent across ratings.csv, tags.csv, movies.csv, and links.csv.

Ratings Data File Structure (ratings.csv): The ratings data file includes user ratings for movies and is organized as user- Id, movie-Id, rating, and timestamp. The rating system uses a five-star scale with half-star increments, and the timestamps indicate the number of seconds since January 1, 1970, at midnight Coordinated Universal Time (UTC).

Tags Data File (tags.csv): The tags data file contains user-generated metadata about films, structured as user-Id, movie-Id, tag, and timestamp. Typically, tags consist of a single word or brief phrase, and the timestamps indicate the number of seconds from January 1, 1970, at midnight Coordinated Universal Time (UTC).

Movies Data File Structure (movies.csv): Movie information is stored in the movies.csv file, organized as movie-Id, title, and genres. Titles may include the year of release in parentheses, and genres are represented as a pipe-separated list from a predefined set.

To ensure accurate evaluation, both datasets were divided into training and testing sets using an 80-20 split based on user IDs. This partitioning scheme ensures that both training and testing sets contain a representative sample of users, allowing for a comprehensive analysis of the recommendation engine's performance.

4.3. General Algorithm

The movie recommendation engine's approach is as follows:

1. Selection of Active User: Choose an active user for whom we aim to predict the rating for a particular movie.
2. Similarity Weighting: Employ a weighting technique to evaluate the similarity between the active user and all other users in the dataset.
3. User Selection: Select a subset of users based on their closest similarity to the active user.
4. Prediction Making: Make a rating prediction for the movie based on the subset of users selected in the previous step.

In the process of selecting the user and movie for prediction (step 1), predictions are made sequentially for each user in the dataset. For each user, all movies rated by the active user are considered, except for the 10th to 20th rated movies, which are excluded to compute the similarity between the active user and others. These excluded movies are the ones for

which predictions will be made.

In step 2, four different similarities weighting schemes are explored: Pearson Correlation Coefficient, Spearman Rank Correlation Coefficient, Mean-Squared Distance, and Cosine Similarity. These schemes are analysed to observe their impact on determining similar users. The parameter 'k' specifies the number of users chosen in step 3, representing the k users with the highest similarity value to the active user. Finally, in step 4, the rating prediction for a movie 'i' is computed based on a weighted average determined by the similarity values of the selected users.

$$\hat{y} = \bar{r}_u + \frac{\sum_{v \in N} S(u,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in N} |S(u,v)|} \quad (1)$$

such that

- *u* - active user
- *v* - any other user
- N-set of k most similar users to u
- r&,/- user v's rating of movie i
- $\bar{r}$ - user's average movie rating
- *s(u, v)* - similarity between active user u and user v

4.4. Evaluation

To assess the performance of our recommendation engine, we employed the mean absolute error (MAE) metric. The mea absolute error quantifies the discrepancy between the predicted ratings and the actual ratings. Mathematically, it is defined as the absolute difference between the predicted rating (ŷ) and the true rating (y) for a given movie 'i'.

$$MAE = \frac{1}{n} \sum_{n}^{i=1} |\hat{y}_i - y_i| \quad (2)$$

5. Implementation

5.1. Pearson Correlation Coefficient

The Pearson Correlation Coefficient serves as a measure of the similarity between two users in collaborative filtering. It quantifies the degree of linear relationship between the ratings given by two users to a common set of movies [8].

To calculate the Pearson Correlation Coefficient, we first identify the intersection (*Iu* ∩ *Iv*) of the movies rated by the active user 'u' and another user 'v'. Then, we compute the statistical correlation of the ratings between these two users within this intersection set. This correlation captures how the ratings given by user 'u' and user 'v' vary together, indicating their level of agreement or disagreement in preferences for shared movies.

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}} \quad (3)$$

In essence, the Pearson Correlation Coefficient provides a numerical measure of similarity between users based on their rating patterns, aiding in the process of identifying users with comparable tastes and preferences.

5.2. Spearman Rank Correlation Coefficient

The Spearman Rank Correlation Coefficient is a similarity metric used in collaborative filtering to assess the relationship between two users' preferences [8]. Similar to the Pearson approach, it evaluates the degree of association between users' ratings; however, the Spearman method considers the ranking of movies rather than their actual ratings.

In the Spearman weighting scheme, each movie rated by a user is assigned a rank based on its position relative to other rated movies. Specifically, the highest-rated item receives rank 1, and subsequent items are ranked accordingly, with ties resolved by assigning the average rank for their position.

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v}(k_{u,i} - \bar{k}_u)\ (k_{v,i} - \bar{k}_v)}{\sqrt{\sum_{i \in I_u \cap I_v}(k_{u,i} - \bar{k}_u)^2}\sqrt{\sum_{i \in I_u \cap I_v}(k_{v,i} - \bar{k}_v)^2}} \quad (4)$$

The Spearman Correlation Coefficient quantifies the similarity between user 'u' and user 'v' by comparing the rankings of shared movies. It is computed in a manner similar to the Pearson Correlation, but with ratings replaced by rankings. Mathematically, the Spearman Correlation Coefficient is calculated by ranking the ratings for each user within the intersection set and then computing the Pearson Correlation Coefficient between the two sets of rankings. This method allows us to assess the similarity between users based on their relative preferences for movies, even when their actual ratings differ [9].

5.3. Mean-Squared Distance

The mean-squared distance similarity scheme is a method utilized to quantify the dissimilarity between two users' preferences in collaborative filtering. As its name suggests, it calculates the average squared difference between the ratings given by user 'u' and user 'v' for the movies they have both rated [8].

Mathematically, the mean-squared distance is computed by summing the squared differences between the ratings of user 'u' and user 'v' for each common movie, and then dividing by the total number of common movies. This metric place greater emphasis on penalizing deviations in tastes between users rather than rewarding similarities.

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v}(k_{u,i} - k_{v,i})^2}{|I_u \cap I_v|} \quad (5)$$

By focusing on the magnitude of differences in ratings, the mean-squared distance scheme provides a measure of dissimilarity that is sensitive to variations in user preferences [10]. It is particularly useful for identifying users with contrasting tastes, which can be valuable in diversifying recommendations and accommodating a wider range of user preferences.

5.4. Cosine Similarity

The mean-squared distance similarity scheme is a method utilized to quantify the dissimilarity between two users' preferences in collaborative filtering [8]. As its name suggests, it calculates the average squared difference between the ratings given by user 'u' and user 'v' for the movies they have both rated.

Mathematically, the cosine similarity metric is defined as the dot product of the rating vectors of user's 'u' and 'v', divided by the product of their magnitudes. This calculation yields a value between -1 and 1, where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates perfect dissimilarity [11].

$$s(u,v) = \frac{r_u . r_v}{||r_u||\ ||r_v||} \quad (6)$$

In essence, cosine similarity measures the cosine of the angle between the rating vectors of two users, providing a measure of similarity that considers both the direction and magnitude of their preferences. This approach is particularly effective for capturing similarities in user preferences across a wide range of movie ratings.

6. Findings

In the existing body of literature, it is commonly agreed upon that the Pearson Correlation Coefficient tends to outperform other similarity metrics in collaborative filtering tasks. Studies conducted by Herlocker et al have specifically highlighted the efficacy of Pearson correlation as a similarity measure. Their research indicated that for those who are considering using a neighbourhood-based prediction algorithm to perform automated collaborative filtering, the following recommendations apply: "Use Pearson correlation as a similarity measure: it remains the most accurate technique for computing similarity."
This recommendation was derived from experiments involving 60 neighbors [12]. In our study, we conducted experiments comparing the performance of four similarity metrics, including Pearson correlation, across varying numbers of neighbours ranging from 10 to the maximum number of neighbours. The results of these experiments are presented below.

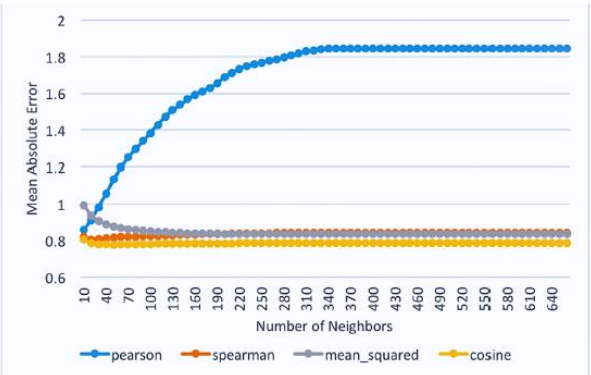

Figure 1. Comparison of Similarity Techniques from 10 Neighbours to All Neighbours

From the analysis depicted in Figure 1, it is evident that the performance of the Pearson Correlation Coefficient diminishes substantially as the number of neighbours increases. Notably, its performance plateaus at around 310 neighbours, representing approximately half of the dataset, where the mean absolute error reaches approximately 1.8. This performance is deemed unsatisfactory since a mean absolute error of 2 indicates a substantial degree of inaccuracy in the recommendation system, where predictions are often far from the actual ratings. The remaining three filtering techniques exhibit a more favourable trend, with improved accuracy as the number of neigbours increases. Among these techniques, the Mean-Squared Distance method initially performs the least favourably with a smaller number of neighbours [13]. However, as the number of neighbours increases, the Spearman Rank Correlation Coefficient and Cosine Similarity techniques exhibit comparable performance, with the latter slightly surpassing the former in accuracy. Notably, as the number of neighbours exceeds 10, Cosine Similarity emerges as the preferred approach, showing a steady plateau in mean absolute error at approximately 0.78, while the other techniques plateau at around 0.84.

Figure 2 further illustrates the performance trends, providing a comprehensive view of the comparative efficacy of each filtering technique, particularly when the number of neighbours considered is less than 10. From the insights provided in Figure 2, it becomes evident that the Pearson Correlation Coefficient exhibits superior performance when the number of neighbours considered is limited.

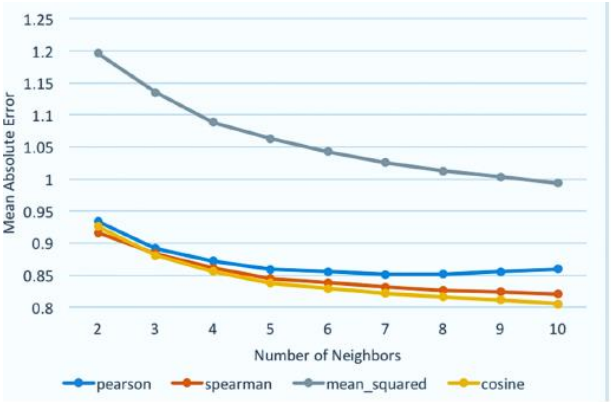

Figure 2. Comparison of Similarity Techniques from 2 Neighbours to 10 Neighbours

Its optimal mean absolute error (MAE) is achieved at 7 neighbours, beyond which the MAE tends to increase. Conversely, the Mean-Squared Distance method consistently performs poorly when the number of neighbours is fewer than 10, maintaining a relatively high MAE throughout [14].

Both the Spearman Rank Correlation Coefficient and Cosine Similarity techniques demonstrate comparable performance, with Cosine Similarity exhibiting slightly higher accuracy as the number of neighbours increases. This trend reinforces the superiority of Cosine Similarity, which consistently outperforms the other methods in terms of minimal MAE. Moreover, Cosine Similarity offers favourable computational efficiency, facilitated by NumPy optimizations enabling rapid and efficient calculation of dot products and Euclidean norms.

Given these observations, we recommend the adoption of Cosine Similarity as the preferred technique for movie recommendation engines, as it demonstrates superior performance across varying numbers of neighbours considered, ensuring both accuracy and computational efficiency.

7. Analysis of Result

In our analysis, the Pearson Correlation Coefficient consistently underperforms in comparison to other algorithms when various numbers of neighbours are considered. This may be due to its tendency to compute high similarity for users who have few items in common. For instance, consider the scenario where two users, u and v, have only rated two movies in common:

User	The Fugitive	Up
U	5	1
V	5	1

Users u and v's similarity is calculated using the Pearson Correlation Coefficient in the following way:

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v}(r_{u,i} - \bar{r}_u)\,(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v}(r_{u,i} - \bar{r}_u)^2}\ \sqrt{\sum_{i \in I_u \cap I_v}(r_{v,i} - \bar{r}_v)^2}}$$

$$s(u,v) = \frac{\sqrt{(1-3)^2 - (5-3)^2}}{\sqrt{(1-3)^2 - (5-3)^2}}$$

$$s(u,v) = \frac{8}{\sqrt{8} * \sqrt{8}} = 1$$

Here, r_(u,i) represents movie i's rating among user u's ratings, and $\bar{r}_u$ represents the average rating given by user u. The resulting similarity score of 1 indicates a perfect match between users' u and v, despite having rated only two movies in common [15].

To address this issue, Ekstrand et al. suggest setting a threshold on the number of co-rated items necessary for full agreement and scaling the similarity when the number of co-rated items falls below this threshold. However, determining the optimal threshold remains a subject of debate, warranting further empirical studies.

In contrast, the other three algorithms—Spearman Rank Correlation Coefficient, Mean-Squared Distance, and Cosine Similarity— demonstrate comparable performance without statistically significant differences. However, Cosine Similarity stands out as the preferred choice for several reasons. It consistently achieves a mean absolute error below 0.8, offers ease of implementation and testing, and benefits from NumPy optimizations, enabling efficient operations on large datasets. Thus, we recommend the adoption of Cosine Similarity for its robust performance and computational efficiency.

8. Analysis of Result

8.1. Achievement of Goals

- **Must Achieve:** Our primary objective was the implementation of collaborative filtering using three distinct weighting similarity techniques. We successfully realized this goal by developing algorithms based on Pearson Correlation Coefficient, Spearman Rank Correlation Coefficient and Mean-Squared Distance. These techniques underwent thorough evaluation to determine their effectiveness in generating accurate movie recommendations.
- **Expected to Achieve:** In addition to the statistical methods, we aimed to implement a non-statistical technique known as cosine similarity. This approach treats users as vectors and measures similarity based on cosine distance. We successfully incorporated cosine similarity into our framework and compared its performance with traditional statistical methods, providing valuable insights into its effectiveness.
- **Would Like to Achieve:** Although we did not explore additional weighting similarity techniques such as Constrained Pearson correlation and Pearson Correlation with threshold-based damping, we conducted an extensive comparative analysis of the implemented algorithms. This analysis allowed us to assess algorithm performance across different parameters and make informed recommendations based on the findings. While we did not develop our own similarity metric, our study provides valuable insights into existing techniques.

8.2. Analysis of Achievements

Our project successfully achieved the core objectives outlined in the proposal. By implementing and evaluating four different similarity metrics for collaborative filtering, we gained valuable insights into their performance characteristics. Through rigorous analysis, we identified the strengths and weaknesses of each technique, enabling us to make informed recommendations for algorithm selection based on various parameters. Although we did not fulfil all the objectives outlined in the proposal's "Would Like to Achieve" section, our study provides a comprehensive evaluation of existing similarity metrics and their suitability for movie recommendation systems. Overall, our project contributes to the advancement of collaborative filtering techniques and provides a foundation for future research in this domain.

9. Conclusion

In this research endeavour, a Python-based movie recommender system was developed and evaluated, utilizing both content-based and collaborative filtering techniques. By conducting an extensive examination of various similarity metrics, in-

cluding Pearson Correlation Coefficient, Spearman Rank Correlation Coefficient, Mean-Squared Distance, and Cosine Similarity, valuable insights into their performance characteristics have been gained. The outcomes of this study indicate that while Pearson Correlation Coefficient tends to exhibit underperformance, Cosine Similarity emerges as the preferred technique, demonstrating superior accuracy and computational efficiency across varying numbers of neighbours considered. Moreover, the analysis emphasizes the significance of addressing the shortcomings of traditional similarity metrics, such as the inclination to compute high similarity for users with few items in common. Potential future research directions may include investigating threshold-based damping techniques and developing novel similarity metrics to further improve the effectiveness of movie recommendation systems. Ultimately, this study contributes to the progression of recommendation algorithms and provides a solid foundation for enhancing user experiences in movie selection.

Reference

[1]. J. S. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” *arXiv [cs.IR]*, 2013.

[2]. M. Kumar, et al. "A movie recommender system: Movrec." *International journal of computer applications*, vol. 124, no. 3, 2015.

[3]. J. L. Herlocker, J. A. Konstan, and J. Riedl, “Explaining collaborative filtering recommendations,” in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, 2000.

[4]. R. J. Prem and R. Mooney, “Content-boosted collaborative filtering for improved recommendations,” *Aaai/iaai*, vol. 23, pp. 187–192, 2002.

[5]. M. Gupta, et al. "Movie recommender system using collaborative filtering." *2020 international conference on electronics and sustainable communication systems (ICESC). IEEE*, 2020.

[6]. R. Katarya and O. P. Verma, “An effective collaborative movie recommender system with cuckoo search,” *Egypt. Inform. J.*, vol. 18, no. 2, pp. 105–112, 2017.

[7]. Y. Koren, S. Rendle, and R. Bell, “Advances in collaborative filtering,” in *Recommender Systems Handbook*, New York, NY: Springer US, 2022, pp. 91–142.

[8]. X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” *Adv. Artif. Intell.*, vol. 2009, pp. 1–19, 2009.

[9]. B. Justin, and T. Hofmann. "Unifying collaborative and content-based filtering." *Proceedings of the twenty-first international conference on Machine learning*, 2004.

[10]. S. Airen and J. Agrawal, “Movie recommender system using K-nearest neighbors variants,” *Natl. Acad. Sci. Lett.*, vol. 45, no. 1, pp. 75–82, 2022.

[11]. R. H. Singh *et al.*, “Movie recommendation system using cosine similarity and KNN,” *Int. J. Eng. Adv. Technol.*, vol. 9, no. 5, pp. 556–559, 2020.

[12]. R. Ahuja, A. Solanki, and A. Nayyar, “Movie recommender system using K-means clustering AND K-nearest neighbor,” in *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2019.

[13]. A. Azaria, et al. "Movie recommender system for profit maximization." *Proceedings of the 7th ACM conference on Recommender systems*, 2013.

[14]. M. Goyani and N. Chaurasiya, “A review of movie recommendation system: Limitations, Survey and Challenges,” *ELCVIA: electronic letters on computer vision and image analysis*, vol. 19, pp. 18–37, 2020.

[15]. A. Odić, M. Tkalčič, J. F. Tasič, and A. Košir, “Predicting and detecting the relevant contextual information in a movie-recommender system,” *Interact. Comput.*, vol. 25, no. 1, pp. 74–90, 2013.

[16]. H. Xiangnan, et al. "Neural collaborative filtering." *Proceedings of the 26th international conference on world wide web*, 2017.

[17]. D. Heckerman, “Dependency networks for inference, collaborative filtering, and data visualization,” *Journal of Machine Learning Research*, vol. 1, pp. 49–75, 2000.

[18]. J. Herlocker, J. A. Konstan, and J. Riedl, “An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms,” *Information retrieval*, vol. 5, pp. 287–310, 2002.

[19]. J. Lee, M. Sun, and G. Lebanon, “A comparative study of collaborative filtering algorithms,” *arXiv [cs.IR]*, 2012.

[20]. Y. Koren, S. Rendle, and R. Bell, “Advances in collaborative filtering,” in *Recommender Systems Handbook*, New York, NY: Springer US, pp. 91–142, 2022.