Journal of Computer Science

Research Article

Optimized Movie Recommendation Using Hybrid Jaccard-Based K-Means Clustering Using Apache Spark

Shiba Prasad Dash[1], Rajesh Kumar Sahoo[1], Ram Chandra Barik[2], Lipsa Priyadarshini Singh[3] and Debashreet Das[1]

[1]Department of Computer Science and Engineering, Biju Patnaik University of Technology, Rourkela, 769015, Odisha, India
[2]Department of Computer Science and Engineering, C. V. Raman Global University, Bhubaneswar, 752054, Odisha, India
[3]Department of Computer Science and Engineering, Centurion University of Technology and Management, Parlakhemundi, 761211, Odisha, India

Article history
Received: 10-01-2026
Revised: 04-07-2026
Accepted: 06-08-2026

Corresponding Author:
Ram Chandra Barik
Department of Computer Science and Engineering, C. V. Raman Global University, Bhubaneswar, 752054, Odisha, India
Email: ram.chandra@cgu-odisha.ac.in

Abstract: The need for recommendations specific to each person's interests is growing every day, making recommendation systems increasingly important in people's lives. Researchers have paid close attention to the rapid expansion of e-commerce businesses and online video streaming services like Hotstar, YouTube, and Netflix. The collaborative filtering-based RS system aims to suggest such films or videos to users based on their past viewing habits. However, RS has challenges with cold start, sparsity, and scalability, which the proposed hybrid system must handle. This data is often represented using a rating matrix. These scores, however, frequently differ since some individuals provide harsher evaluations while others are more forgiving. As a result, the RS cannot suggest customized films to demanding viewers. This research suggests a collaborative filtering recommendation system based on K-means clustering, movie clustering, and normalization in order to address the problem. K-means clustering and movie recommendation are carried out using Apache Spark. First, the algorithm clusters similar movies using Jaccard distance, and then normalization is carried out to eliminate the small ratings in the utility matrix. Further, the K-means technique is implemented to recommend the top possible movies to the users.

Keywords: Recommendation System, Clustering, K-Means, Collaborative Filtering, Clustering Based Normalization in Movie Recommendation, Apache Spark

Introduction

An emerging use of artificial intelligence is the recommendation system, which is used to suggest more items to actual consumers in digital marketing and e-commerce. The consumer is essentially presented with a variety of products by an AI-powered recommendation engine based on their past choices and how they stack up against other items in the same category. Based on the customer's previous purchases or opinions of any product, movie, or item, it offers recommendations. Recommendation systems serve Internet shops by recommending movies to their customers based on their past purchases, and they also provide recommendations for news items to online readers. Content Based (CB), hybrid, and Collaborative Filtering (CF) recommendation systems are the three general types of recommendation systems. The users are given recommendations using a utility matrix, which shows the user's rating for each movie or item. The top movies that people should think about seeing for the first time are suggested by this study. The Collaborative Filtering Recommendation System (CFRS), currently the most widely used RS algorithm, is used in this study. This CFRS appears after using content-based RS to improve accuracy. Several machine-learning methods are used in our work to suggest movies to viewers.

Compared to other recommendation system approaches, CF frequently offers better performance and accuracy when working with large and complex datasets. Early CF techniques for recommendation systems relied on association inferences, which had a high temporal complexity and were not very scalable (Bobadilla et al., 2013; Koren, 2010; Du et al., 2016; Yang et al., 2016). Collaborative filtering uses user ratings and preferences to propose movies to others based on their common interests and past viewings. Clustering is another auxiliary approach utilized in recommendation system development. The clustering approach groups a collection of movies so that

the movies in the same clusters are more comparable to one another than to those in different groupings. Jaccard distance is used to group comparable films together. A single movie cluster is created by grouping Jaccard distances smaller than 0.25. To get the optimum outcome, the K-means algorithm is also used on other datasets. The present technique creates a lot of clusters since the data is dispersed. However, by combining the data, the proposed method lowers the number of clusters. The hybrid framework with KNN clustering is utilized in addition to the k-means algorithm. The proposed recommendation system predicts a user's favorite movies based on a number of variables. The unwatched films in a cluster are suggested to the user if their average score for that cluster is higher than 2.5. Additionally, the K-means clustering technique is used to recommend unknown movies to viewers and evaluate the degree of similarity between movie clusters.

Related Work

According to Ahn (2008) a novel heuristic similarity measure known as Proximity Impact Popularity (PIP) was developed for collaborative filtering, whereby its effectiveness was tested against the traditional similarity measures, including Pearson and cosine correlations. Findings from this study revealed that the PIP-based approach outperformed the traditional measures regarding accuracy and effectiveness of recommendations. Nevertheless, the suggested strategy might not be the best course of action, and it's unclear whether it can be applied to other economic areas. Furthermore, a number of crucial elements that were emphasized in the previously described studies were mainly ignored during the standardization process. Bobadilla et al. (2013) have derived several RS equations that make use of memory and collaborative filtering. When designing and incorporating the suggestions into the e-learning program, they took the user's skill level into account. However, they haven't tested any e-learning databases. They created many memory-based collaborative filtering-based RS algorithms that provide suggestions and apply them to the e-learning RS while taking the users' experience level into consideration. Four categories of filtering algorithms, such as collaborative, hybrid, demographic, and content-based, as well as other collaborative filtering methods and recommendation systems, are discussed. Ahuja et al. (2019) suggested an RS model that made use of the K-Means Clustering and K-Nearest Neighbor algorithms. A range of tools and techniques have been used in the construction of recommender systems throughout their work. KNN, K-means clustering, Content Based (CB), and Collaborative Filtering (CF) are only a few of the techniques that have been fully detailed. Karidi et al. (2018) explored the use of Collaborative Filtering (CF) techniques in various Web 2.0 applications, including online marketing, advertising, and recommendation systems. In their study, they proposed the Multi-Collaborative Filtering Trust Network approach, an enhanced CF algorithm developed to improve the effectiveness of recommendation and personalization in Web 2.0 environments. Several versions of the KNN method for movie selection are studied by Airen and Agrawal (2022). To assess these differences, they have employed metrics like mean absolute error and accuracy. The study assesses the accuracy of several methods using real data from the MovieLens dataset. Du et al. (2016) created a trusted network and suggested trust relationships between various users in the 0-1 domain. In 2016, a trust model and Bayesian network-based movie recommendation system was created. Kim et al. (2016) used normalization and clustering to increase prediction accuracy. They normalized neighbor evaluations and retrieved statistical information from user ratings. Different clustering techniques used in movie recommendation are discussed in Table 1. Liu (2022) has proposed a movie recommendation system based upon NMF (or non-negative matrix factorization). The proposed recommendation system provides improved accuracy, recall, and overall satisfaction through the combination of a user's movie ratings with sentiment analysis of their previously reviewed films, rather than just relying solely on past film ratings or similar user-rated movies. Consequently, the author's NMF-based recommender significantly improves the traditional movie recommendation process.

A unique collaborative filtering recommender system for item suggestion based on normalization was presented by Panda et al. (2020). Their approach employed Min-Max normalization to address missing ratings and improve the recommendation process. Initially, the average ratings of individual users and movies were calculated. The missing ratings were subsequently estimated by applying Min-Max normalization to the users' average ratings.

Table 1: Comparison of different clustering techniques used in movie recommendation

References	Dataset	Clustering Techniques
Wang et al. (2014)	MovieLens 100k dataset	GA-Kmeans
Aditya et al. (2018)	TMDB 5000	K-means
Vilakone et al. (2018)	MovieLens 100k dataset	k-NN
Cho et al. (2012)	MovieLens 100k dataset	K-means
Dakhel and Mahdavi (2011)	MovieLens 100k dataset	K-means
Jayalakshmi et al. (2022)	MovieLens 100k dataset	GA-Kmeans
Markos et al. (2010)	MovieLens 100k dataset	PCA Neighborhood Formation
Widiyaningtyas et al. (2021)	MovieLens 100k dataset	K-means
Pitsilis et al. (2011)	MovieLens 100k dataset	K-means and k-NN
Dash et al. (2024)	MovieLens 100k dataset	Normalized k-NN

These estimated ratings were then used to identify and recommend the most suitable movies to each user. Furthermore, the authors incorporated a trust model to filter historical data and employed a Bayesian network to model user preferences. This combination was intended to enhance the robustness of the recommender system by reducing the influence of noisy or unreliable input data.

Many techniques, such as cosine distance, mean squared difference, modified cosine similarity, and Pearson correlation coefficient, can be used to assess how similar a user is to their objects. Movie recommendation systems give consumers a means of classifying other users who have similar interests. According to Karidi et al. (2018) a semantic suggestion can be used to show relevant tweets on users' timelines. This was accomplished using a knowledge graph, which displays concepts, examples, individuals, locations, and the connections between them. The neighborhood model and the latent component methodology are two distinct methods.

Nayak and Panda (2018) introduced a unique method that uses the greatest number of users an item has in common with other users to find k similar persons. Additionally, they make a useful comparison between the kNN methodology and their proposed approach. Despite being popular due to its simplicity and efficacy, the slope-one technique has problems with prediction accuracy and personalized recommendations. These issues are successfully addressed by hybrid models that combine enhanced K-means clustering with Genetic Algorithms (GAs) and Principal Component Analysis (PCA). Information overload is an issue when recommending the most pertinent items to customers based on a vast quantity of data. By identifying exactly identical neighbors among users or films based on their past shared ratings, online collaborative movie suggestions for media products try to help consumers access their favorite films (Wang et al., 2014). The accuracy and efficiency of traditional Collaborative Filtering (CF) algorithms have been challenged by the rapid growth of big data, which has increased information overload. Hybrid approaches, which combine CF with optimization techniques like swarm intelligence, have improved performance but still have problems like premature convergence. The Artificial Bee Colony (ABC) approach is used with K-means clustering to overcome these shortcomings. According to experimental data, the quality and efficiency of recommendations are improved by a lower MAE of 0.767 and faster computing (50 seconds). This hybrid approach presents encouraging developments for CF systems (Jing, 2022). Alhijawi and Kilani (2020) developed BLIGA, a collaborative filtering recommender system built on a genetic algorithm. With BLIGA, users will receive recommendations based on multiple factors, including Semantic Similarity, User Satisfaction Similarity, and Predicted Ratings, for each of the complete Recommendation Lists. By employing hierarchical filtering of the candidate Recommendation Lists to increase overall accuracy and prediction quality, BLIGA also removes the cold-start and sparsity issues related to conventional CF and other Genetic Algorithm-based techniques. Alhijawi et al. (2018) propose a hybrid approach for item recommendation that takes into account both the similarity of user ratings and the semantic characteristics of the items. This method minimizes cold-start and sparsity problems while producing a single collection of neighbor items that enhances prediction accuracy and recommendation quality. There are still certain issues with new films that have little or no user data, despite the earlier research's ability to propose films effectively and with greater accuracy. This paper proposes a hybrid recommendation framework that combines the K-means clustering method with hierarchical clustering to address this cold start issue. The novel contributions are:

1. A hybridization order: Jaccard-based neighborhood with K-Means clustering on the derived similarity space, then Min Max normalization for scoring is employed
2. An Apache Spark optimized implementation with cluster-wise parallel Jaccard computation is followed by K-Means clustering for movie selection
3. A cluster-aware recommendation score, which combines intra-cluster similarity and normalized rating information, is used instead of using raw similarity or raw ratings

Materials and Methods

The first step in any movie recommendation system is to create a utility matrix. For this stage, a utility matrix that incorporates user ratings is also taken into consideration. Pre-processing, movie clustering, the Clustering-based Normalized K-Means (CNKM) algorithm, and recommendation finding are the four primary phases of the suggested approach. The information should be used to get the users' chosen movies. To obtain the best recommendations, the suggested method iteratively processes the utility matrix. The suggested clustering-based K-means movie recommendation approach first determines the similarity between each film on the list. To make the utility matrix smaller, related films are grouped together. Here, the viewers are given movie recommendations based on the grouping of comparable movies using the k-means technique. Fig. 1 displays the data flow diagram for the suggested technique. In this part, each module of the suggested technique is briefly described. Table 1 shows different clustering techniques used for movie recommendations.

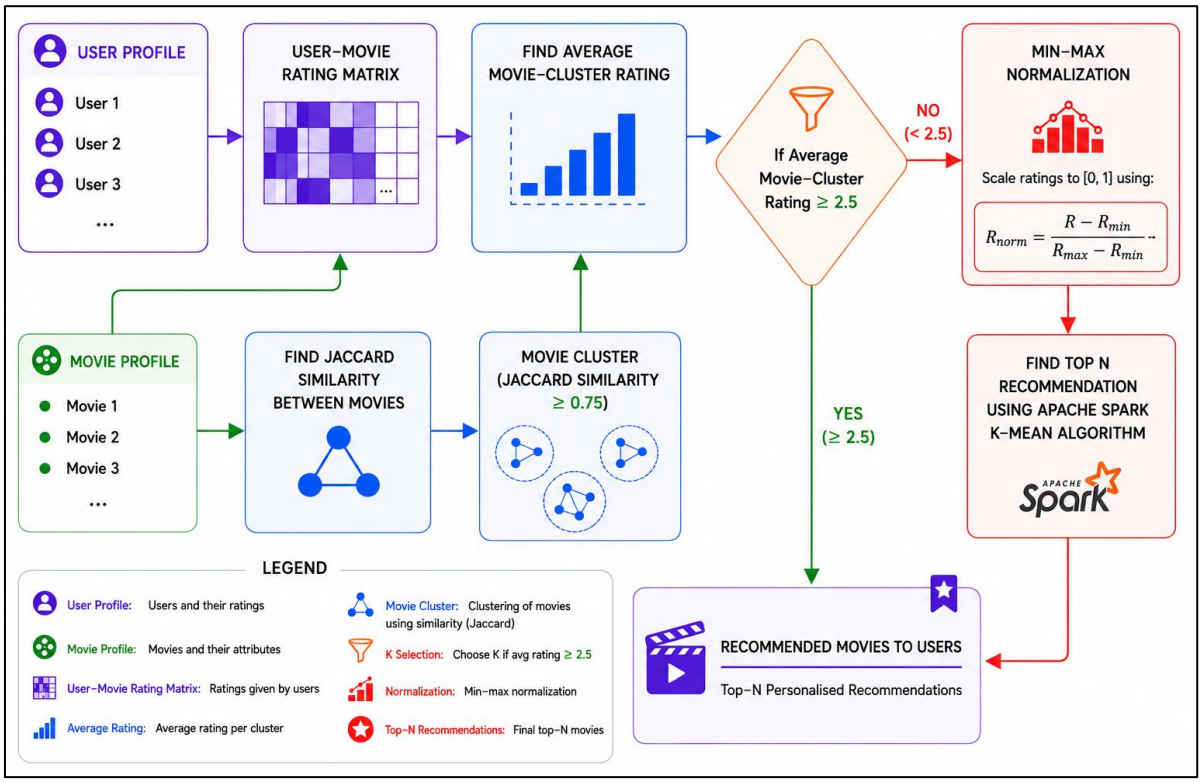

Fig. 1: Proposed clustering-based K-means movie recommendations

Data Collection

The initial step in the implementation process is gathering data. To make further computations easier, the right dataset has now been chosen. The movie recommendation system on the Kaggle website makes use of the MovieLens dataset, where 9742 movies have received 100836 user ratings from 610 users. Each rating ranges from 1 to 5. If a person has seen a specific film, they could have rated it. Otherwise, it is assumed that the viewer had not previously observed the movie if the rating is blank. Data preparation constitutes the second phase of the implementation process. At this stage, a utility matrix is constructed to represent the ratings provided by users for different movies. To accomplish this, the user-related information and movie-related information are initially extracted and organized into separate data frames. These two data frames are subsequently integrated to generate the utility matrix, which captures the relationships between users and movies through their corresponding ratings.

Movie Clustering

Following the creation of the utility matrix, relevant movies are grouped together using the hierarchical clustering method. During this process, the algorithm splits a large collection of movies into smaller groups. For the grouping process, we have used the popular Jaccard similarity measure. Its input consists of a rating utility matrix, an array of m movies, and n users. The cluster matrix is initialized in the first line of the clustering method, as seen in Algorithm 1. Assigning comparable film genres to each cluster is the aim of the cluster matrix. First, the system determines if a particular user has given ratings to a particular movie. If there is a rating, which is any number between 1 and 5, then the count will be incremented by 1. Recently identified movies are initially added to the cluster matrix and stored as a new cluster. The similarities between the movies are then compared using lines 3 through 10. Here, Jaccard's similarity is applied to determine how similar the two films are. Movies are regarded as a single cluster if their name similarity is more than 0.75. Additionally, the cluster's average rating is calculated by dividing the user's overall rating by the number of movies the user has rated in the cluster. Additionally, it indicates that the user likes that specific movie series if the average cluster rating is higher than 2.5. In this instance, it is feasible to suggest every movie in the cluster that the user hasn't yet viewed. All of the unwatched movies are suggested to the related viewer if the user's average rating for a movie cluster is more than 2.5. This indicates that the user enjoys the movie series or other comparable movie material. In order to eliminate the low ratings, the rating matrix is further normalized.

Algorithm 1: Movie Clustering using Jaccard Similarity

Input: User–Movie Rating Matrix R(U×M)
Array of Movie Titles T
Output: Movie Clusters with Average Ratings

1. Initialize MovieClusters ← ∅
2. Create CountMatrix to store the number of users who rated each movie.
3. For each movie Mi ∈ T do
4. For each movie Mj ∈ T, j > i do
5. Compute JaccardSimilarity (Mi, Mj)
6. If JaccardSimilarity (Mi, Mj) ≥ 0.75 then
7. Assign Mi and Mj to the same cluster
8. End If
9. End For
10. End For

11. For each cluster Ck ∈ MovieClusters do
12. For each user Ui do
13. AvgClusterRating (Ck) ← mean of all ratings of Ui for movies in cluster Ck
14. End For
15. End For

16. For each cluster Ck ∈ MovieClusters do
17. If AvgClusterRating (Ck) ≥ 2.5 then
18. Recommend all unrated movies in Ck to the corresponding user
19. End If
20. End For

21. Return MovieClusters and Recommended Movies

Normalization

The normalization stage begins after a large number of films have been sorted to form smaller clusters. In the normalization stage, the algorithm aims to normalize the number of users for each cluster and determine the average user rating for each movie cluster. Data normalization is an important preprocessing step in machine learning, particularly for models trained using gradient-based optimization, as it reduces the adverse effects caused by differences in the scale and range of input values. In the proposed approach, Min–Max normalization is employed to transform the original user-rating values into a common predefined range. This process requires determining the minimum and maximum values of the user ratings, along with the lower and upper bounds of the desired normalized range. In this instance, we also need the maximum and minimum user counts, as well as the lowest and highest normalization ranges. The rating matrix's minor ratings are eliminated using this normalization method. Algorithm 2 illustrates the Min-Max Normalization Process. In the subsequent stage, the suggested model finds similar movie clusters using K-means clustering algorithms.

Algorithm 2: Min-Max Normalization Algorithm.

Input: Cluster matrix with average user ratings, Array of movie titles T.
Output: Normalized movie-cluster ratings in the range [0,1].

1. Determine the minimum rating value:
Rmin ← min(C)
2. Determine the maximum rating value:
Rmax ← max(C)
3. For each rating R in C do
4. Compute normalized rating:
Rnorm ← (R − Rmin) / (Rmax − Rmin)

5. Store Rnorm in Cnorm
6. End For
7. Return Cnorm

Algorithm 3: K-Means Clustering with Top-N Recommendation

Input: User–Movie Rating Matrix R(U×M)
Number of Clusters K
Number of Recommendations N
Output: Top-N Recommended Movies for each user

1. Initialize K cluster centroids randomly.
2. Repeat
3. For each movie cluster Ci do
4. Compute the distance between Ci and all K centroids.
5. Assign Ci to the nearest centroid.
6. End For

7. For each cluster Ck do
8. Compute the mean of all movie clusters assigned to Ck.
9. Update centroid μk using the mean.
10. End For

11. Compute centroid shift:
Δ = distance (old centroid, new centroid)
12. Until Δ < Threshold or MaximumIterations reached

13. For each user Uu do
14. Identify clusters containing highly rated movies by Uu.
15. Predict ratings for unseen movies within those clusters.
16. Rank unseen movies in descending order of predicted ratings.
17. Select the Top-N highest-ranked movies.
18. Recommend the Top-N movies to user Uu.
19. End For
20. Return Top-N Recommended Movies.

The Traditional K-Means Algorithm

The simplicity and straightforwardness of this well-known algorithm allow it to address a wide range of issues. The main concept is to split the movie clusters into fragmented clusters by decreasing the similarity between the new cluster center and the target movie cluster. In this research, Apache Spark is used to perform the K-means clustering process. The k-means clustering step is shown in Algorithm 3. In the beginning phase, the algorithm uses

clusters of similar movies that are previously grouped using Jaccard similarity instead of individual movies. Similarly, instead of users' ratings for each movie, this algorithm uses the average user rating for each cluster.

A feature vector representing each movie with latent factors, genres, and normalized ratings for the recommendation system. Employing K-means clustering of similar groups of movies by minimizing:

$$J = \sum_{i=1}^{K} \sum_{m_j \in C_i} \left\| m_j - \mu_i \right\|^2 \quad (1)$$

In the above equation, m_j represents the feature vectors of a movie, μ_i is the centroid of movie cluster *i*, and *Ci* is the set of movies in cluster *i*. By using Jaccard similarity and kNN, the clustered movies are used to recommend similar movies.

Results and Simulation Analysis

In order to facilitate remote data processing and scalable computation, the experimental setup for this study was set up using Apache Spark on a Windows 11 operating system. In order to ensure that there were sufficient computer resources available to conduct the experiments, the configuration was done on a personal computer with an 11th Gen Intel(R) Core (TM) i9-11900K @ 3.50GHz with 32 GB of RAM and 1 TB SSD. The software environment included Python 3.10, Apache Spark 3.5.0, and Java Development Kit (JDK) 11, all of which are compatible with running PySpark applications.

Dataset Description

The performance of the proposed recommendation method is evaluated on the MovieLens dataset, obtained from Kaggle. The dataset comprises 100,836 user ratings assigned to 9,742 movies by 610 users. It consists of two primary files: Ratings and movies. The movies file provides metadata about the movies, including their titles and genres, whereas the ratings file records the ratings assigned by individual users to different movies.

Evaluation Metrics

Using various comparison criteria, we will assess our suggested CNKM methods in this chapter. First, we will look at a testing matrix and compare the proposed approach to the existing technique. For the evaluation of the proposed work, Mean Absolute Error (MAE) (Eq. 3), recall (Eq. 5), precision (Eq. 4), Root Mean Square Error (RMSE) (Eq. 2), and F-Score (Eq. 6) are used:

$$RMSE = \sqrt{\frac{1}{t}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(R_{ij} - P_{ij}\right)^2} \quad (2)$$

$$MAE = \frac{1}{t}\sum_{i=1}^{n}\sum_{j=1}^{m}\left|R_{ij} - P_{ij}\right| \quad (3)$$

$$Precision = \frac{true\ positives}{/(true\ positives + false\ positives} \quad (4)$$

$$Recall \frac{= true\ positives}{/(true\ positives + false\ negatives} \quad (5)$$

$$F - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (6)$$

Where:

R is the user-movie rating matrix
P is the predicted ratings matrix
n is the number of users
m is the number of cluster movies
t is the total number of ratings

Experimental Results

The suggested CNKM method is compared to the min-max normalization-based algorithm (NCFR) and the existing Clustering-based Normalization in the Movie Recommendation (CNMR) technology (Panda et al., 2020). The CNMR model recommends movies to users in two stages. In the first stage, it clusters the movies as the proposed model works, and in the second stage, it recommends the movies after applying min-max normalization to the movie clusters. The suggested model has an RMSE of 1.899 and an MAE of 1.603. Table 2 indicates that the suggested Clustering-based Normalized k-Means (CNKM) performs better than the other models, as Fig. 2 illustrates. Additionally, the findings are compared with clustering-based normalized kNN movie recommendation models. Fig. 3 shows that the hybrid movie recommendation system utilizing k-means is outperforming the recommendation system using the kNN method. The K-means technique is used in the simulation process to carry out the clustering and movie suggestions using Apache Spark. The ROC curve between the proposed CNKM model and the recommendation of the hybrid movie using kNN is shown in Fig. 4. The classification performance of the proposed CNKM model is further illustrated by the confusion matrix in Fig. 5.

Table 2: Comparison of Algorithms based on different accuracy measures

Algorithm	MAE	RMSE	Precision	Recall	F-Score
Panda et al. (2020)	1.692	2.009	0.571	0.444	0.500
Ranjan et al. (2024)	0.7211	0.9103	0.8704	0.7391	0.799
Ali et al. (2022)	0.6021	0.7759	1.0	0.653	0.790
Dash et al. (2025)	-	-	0.91	0.91	-
Proposed	1.603	1.899	0.634	0.573	0.601

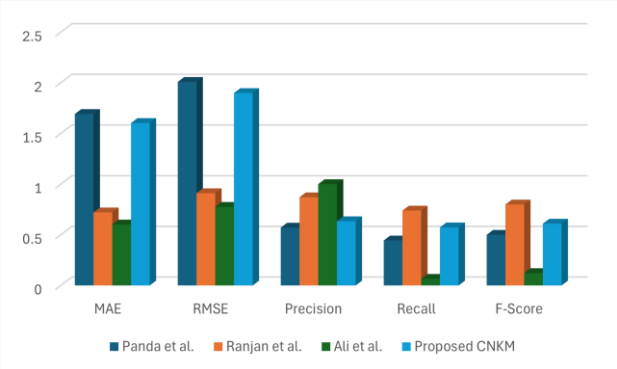

Fig. 2: Comparison of different movie recommendation models

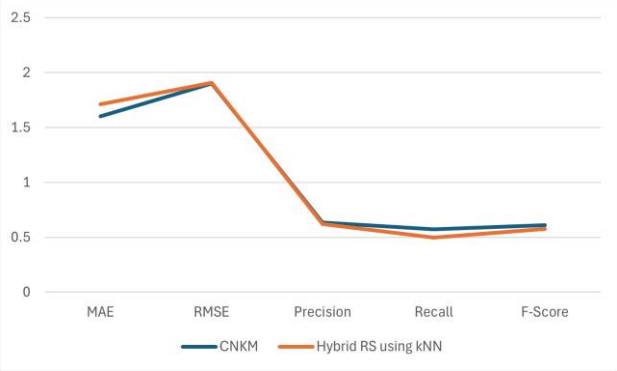

Fig. 3: Comparison of hybrid recommendation using kNN and K-means

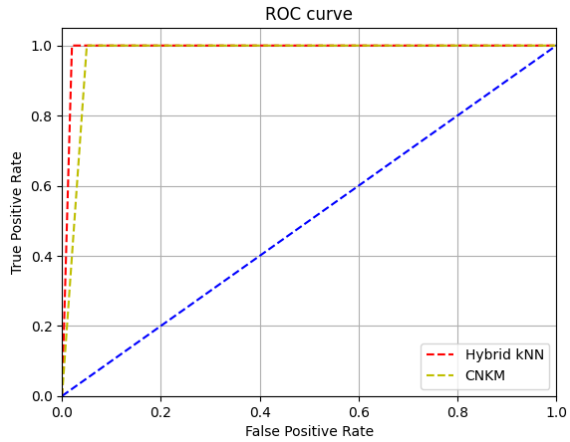

Fig. 4: ROC curves between hybrid recommendation using kNN and K-means (CNKM)

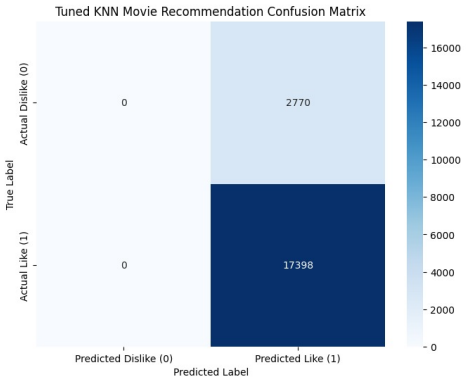

Fig. 5: Confusion matrix of the proposed CNKM model

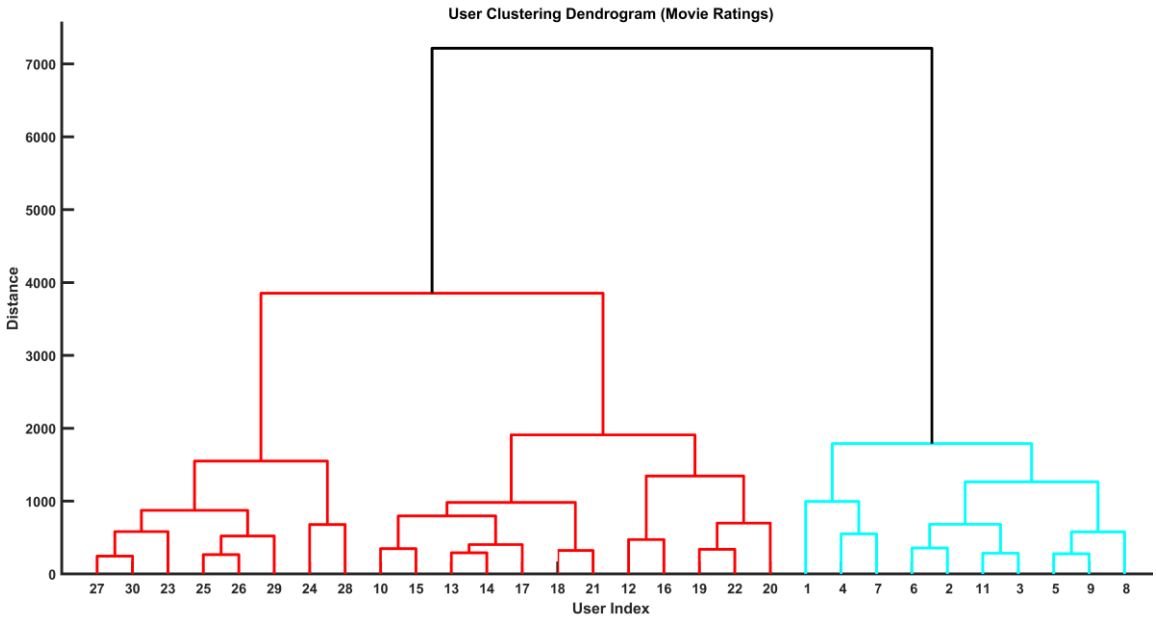

Fig. 6: Dendrogram Plot of User Preferences against the recommended movie

Dendrogram Analysis

The dendrogram visually represents user similarity based on their movie ratings, grouping users with similar preferences together as shown in Fig. 6. The x-axis corresponds to different users in the dataset, where each user is initially treated as an individual cluster. The y-axis represents the Euclidean distance between users, indicating how different their movie preferences are. The users with the most similar ratings are merged first at lower distances, while less similar users merge at higher distances. The height at which two branches merge indicates the level of dissimilarity; lower merges mean more similarity, while higher merges mean greater dissimilarity. The cut-off threshold in the dendrogram determines the number of user clusters, which can be used for personalized movie recommend

ations. Different colors in the dendrogram (if applied) represent distinct user clusters that share similar movie preferences. By identifying user clusters, we can suggest movies to users based on the preferences of others in the same cluster. Unlike k-means, hierarchical clustering does not require predefining the number of clusters, allowing a more flexible analysis of user behavior. Users grouped closely together can receive recommendations based on movies rated highly by their group members, improving the precision of the recommendations.

Conclusion

The present research proposes a Normalized K-Means Collaborative Filtering (NK-CF) strategy for personalized movie recommendation, integrating unsupervised machine learning, similarity-based clustering, and rating normalization. The proposed methodology is designed to address several limitations associated with conventional collaborative filtering approaches, particularly the high dimensionality and sparsity of user-item rating matrices, as well as the computational complexity of identifying relevant neighbors. To overcome these challenges, the proposed framework combines a hierarchical clustering strategy with Min-Max normalization, thereby improving the consistency and comparability of users' ratings while reducing the computational burden of the recommendation process. In the proposed approach, movies with similar characteristics are grouped into clusters, thereby reducing the effective size of the original rating matrix and narrowing the search space for potential recommendations. Jaccard similarity is employed to identify relationships among movies and facilitate the formation of meaningful clusters based on users' interaction patterns. Subsequently, the normalized ratings within each cluster are analyzed to identify users' preferences. When a user's average rating for a particular cluster exceeds the predefined threshold of 2.5, the proposed K-Means clustering algorithm is employed to identify the most relevant neighboring movies within that cluster. Unwatched movies associated with these highly relevant neighbors are then prioritized and recommended to the user.

Additionally, the suggested recommendation architecture makes use of Apache Spark to provide K-Means-based clustering and effective large-scale data processing. The system may produce customized suggestions while lowering computational overhead thanks to the combination of neighborhood identification, rating normalization, similarity-based clustering, and distributed computing. The previous section's experimental results show that the suggested method outperforms current collaborative filtering-based techniques in terms of efficiency and recommendation performance, underscoring its efficacy for scalable and customized movie recommendation.

AI Disclosure

The author(s) declare that no assistance was taken from generative AI to write this article.

Author's Contributions

Shiba Prasad Dash: Writing original draft, methodology, software, validation, investigation, resources, data curation.

Rajesh Kumar Sahoo: Conceptualization, investigation, methodology, software, formal analysis, supervision, writing review and edited.

Ram Chandra Barik: Supervision, writing review and edited.

Lipsa Priyadarshini Singh: Methodology, software.

Debashreet Das: Methodology, formal analysis.

Ethics

Ethical Approval and Consent to Participate

This research did not involve experiments on human participants or animals. All data used in this study were obtained from publicly available sources and were analyzed in accordance with relevant guidelines and regulations. Therefore, formal ethical approval and informed consent were not required.

Consent for Publication

Not applicable, as the manuscript does not contain any individual person's data in any form (including images, videos, or personal details).

Data Integrity and Research Ethics

The authors confirm that the research presented in this manuscript is original and has not been published previously, nor is it under consideration for publication elsewhere. All sources of information have been properly cited, and the study complies with accepted standards of academic integrity and research ethics.

Conflict of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

Aditya, T. S., Rajaraman, K., & Monica, M. S. (2018). Comparative Analysis of Clustering Techniques for Movie Recommendation. *MATEC Web of Conferences*, *225*, 02004. https://doi.org/10.1051/matecconf/201822502004

Ahn, H. J. (2008). A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, *178*(1), 37–51. https://doi.org/10.1016/j.ins.2007.07.024

Ahuja, R., Solanki, A., & Nayyar, A. (2019). Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor. *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 263–268. https://doi.org/10.1109/confluence.2019.8776969

Airen, S., & Agrawal, J. (2022). Movie Recommender System Using K-Nearest Neighbors Variants. *National Academy Science Letters*, *45*(1), 75–82. https://doi.org/10.1007/s40009-021-01051-0

Alhijawi, B., & Kilani, Y. (2020). A collaborative filtering recommender system using genetic algorithm. *Information Processing & Management*, *57*(6), 102310. https://doi.org/10.1016/j.ipm.2020.102310

Alhijawi, B., Obeid, N., Awajan, A., & Tedmori, S. (2018). Improving collaborative filtering recommender systems using semantic information. *2018 9th International Conference on Information and Communication Systems (ICICS)*, 127–132. https://doi.org/10.1109/iacs.2018.8355454

Ali, Y., Khalid, O., Khan, I. A., Hussain, S. S., Rehman, F., Siraj, S., & Nawaz, R. (2022). A hybrid group-based movie recommendation framework with overlapping memberships. *PLOS ONE*, *17*(3), e0266103. https://doi.org/10.1371/journal.pone.0266103

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109–132. https://doi.org/10.1016/j.knosys.2013.03.012

Cho, Y. S., Moon, S. C., Noh, S. C., & Ryu, K. H. (2012). Implementation of personalized recommendation system using k-means clustering of item category based on RFM. *2012 IEEE International Conference on Management of Innovation & Technology (ICMIT)*, 378–383. https://doi.org/10.1109/icmit.2012.6225835

Dakhel, G. M., & Mahdavi, M. (2011). A new collaborative filtering algorithm using K-means clustering and neighbors' voting. *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, 179–184. https://doi.org/10.1109/his.2011.6122101

Dash, S. P., Prusty, S., Barik, R. C., Yadav, D. K., & Sahoo, R. K. (2025). A neuro-inspired hybrid framework for personalized movie recommendation using deep spiking neural networks and collaborative filtering. *International Journal of Information Technology*, 1–18. https://doi.org/10.1007/s41870-025-02821-5

Dash, S. P., Sahoo, R. K., Singh, L. P., & Barik, R. C. (2024). A New Variant of Clustering based Normalized Knn Movie Recommendation System. *2024 International Conference on Intelligent Computing and Sustainable Innovations in Technology (IC-SIT)*, 1–6. https://doi.org/10.1109/ic-sit63503.2024.10862097

Du, Y., Du, X., & Huang, L. (2016). Improve the Collaborative Filtering Recommender System Performance by Trust Network Construction. *Chinese Journal of Electronics*, *25*(3), 418–423. https://doi.org/10.1049/cje.2016.05.005

Jayalakshmi, S., Ganesh, N., Čep, R., & Murugan, J. S. (2022). Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions. *Sensors*, *22*(13), 4904. https://doi.org/10.3390/s22134904

Jing, H. (2022). Application of Improved K-Means Algorithm in Collaborative Recommendation System. *Journal of Applied Mathematics*, *2022*, 1–10. https://doi.org/10.1155/2022/2213173

Kim, S.-C., Sung, K.-J., Park, C.-S., & Kim, S. K. (2016). Improvement of collaborative filtering using rating normalization. *Multimedia Tools and Applications*, *75*(9), 4957–4968. https://doi.org/10.1007/s11042-013-1814-0

Koren, Y. (2010). Collaborative filtering with temporal dynamics. *Communications of the ACM*, *53*(4), 89–97. https://doi.org/10.1145/1721654.1721677

Liu, C. (2022). Personalized Recommendation Algorithm for Movie Data Combining Rating Matrix and User Subjective Preference. *Computational Intelligence and Neuroscience*, *2022*, 1–11. https://doi.org/10.1155/2022/2970514

Markos, A. I., Vozalis, M. G., & Margaritis, K. G. (2010). An Optimal Scaling Approach to Collaborative Filtering Using Categorical Principal Component Analysis and Neighborhood Formation. *Artificial Intelligence Applications and Innovations*, 22–29. https://doi.org/10.1007/978-3-642-16239-8_6

Nayak, S. K., & Panda, S. K. (2018). A User-Oriented Collaborative Filtering Algorithm for Recommender Systems. *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 374–380. https://doi.org/10.1109/pdgc.2018.8745892

Panda, S. K., Bhoi, S. K., & Singh, M. (2020). A collaborative filtering recommendation algorithm based on normalization approach. *Journal of Ambient Intelligence and Humanized Computing*, *11*(11), 4643–4665. https://doi.org/10.1007/s12652-020-01711-x

Pitsilis, G., Zhang, X., & Wang, W. (2011). Clustering Recommenders in Collaborative Filtering Using Explicit Trust Information. *Trust Management V*, 82–97. https://doi.org/10.1007/978-3-642-22200-9_9

Karidi, D. P., Stavrakas, Y., & Vassiliou, Y. (2018). Tweet and followee personalized recommendations based on knowledge graphs. *Journal of Ambient Intelligence and Humanized Computing*, *9*(6), 2035–2049. https://doi.org/10.1007/s12652-017-0491-7

Ranjan, S., Pandey, T. N., Dash, B. B., Mishra, M. R., De, U. C., & Patra, S. S. (2024). Performance assessment of various machine learning algorithms in recommendation. *2024 Second International Conference on Inventive Computing and Informatics (ICICI)*, 292–297. https://doi.org/10.1109/ICICI62254.2024.00055

Vilakone, P., Park, D.-S., Xinchang, K., & Hao, F. (2018). An efficient movie recommendation algorithm based on improved k-clique. *Human-Centric Computing and Information Sciences*, *8*(1), 38. https://doi.org/10.1186/s13673-018-0161-6

Wang, Z., Yu, X., Feng, N., & Wang, Z. (2014). An improved collaborative movie recommendation system using computational intelligence. *Journal of Visual Languages & Computing*, *25*(6), 667–675. https://doi.org/10.1016/j.jvlc.2014.09.011

Widiyaningtyas, T., Hidayah, I., & Adji, T. B. (2021). Recommendation Algorithm Using Clustering-Based UPCSim (CB-UPCSim). *Computers*, *10*(10), 123. https://doi.org/10.3390/computers10100123

Yang, Z., Wu, B., Zheng, K., Wang, X., & Lei, L. (2016). A Survey of Collaborative Filtering-Based Recommender Systems for Mobile Internet Applications. *IEEE Access*, *4*, 3273–3287. https://doi.org/10.1109/access.2016.2573314