JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 11, No. 1, March 2026, Pp.705-715

DEEP LEARNING SOLUTION FOR SPARSITY PROBLEM TO IMPROVE RECOMMENDATION QUALITY

Sri Lestari *[1)], Tiwuk Mariana[2)]
1. Institut Informatika dan Bisnis Darmajaya, Indonesia
2. Institut Informatika dan Bisnis Darmajaya, Indonesia

Article Info
Keywords: Recommendation System; Sparsity; Deep Learning

Article history:
Received 12 December 2024
Revised 12 February 2025
Accepted 18 March 2025
Available online 17 March 2026

DOI :
https://doi.org/10.29100/jipi.v11i1.7027

* Corresponding author.
E-mail address:
srilestari@darmajaya.ac.id

ABSTRACT
Recommendation systems have become indispensable across various platforms due to their ability to enhance personalized services. However, these systems face a critical challenge known as sparsity. Sparsity occurs when there are numerous gaps in data, making user preferences unknown. This results in less relevant recommendations, reducing system effectiveness and diminishing user satisfaction. Moreover, it can lead to missed business opportunities. The purpose of this study is to address the sparsity problem using Deep Learning to enhance recommendation quality. The research stages include literature review (SLR), data collection from the Netflix Prize dataset obtained from kaggle.com, data preprocessing, Deep Learning implementation, testing, analysis, and conclusions. The stages of this study are conducted literature study (SLR), data collection, data preprocessing, Deep Learning implementation, testing and analysis, and conclusions. The method of this study is carried out data preprocessing and imputaion using several existing methods by using the Netflix Prize dataset, data taken from kaggle.com. The result of this study shows that the Deep Learning method is able to solve the sparsity problem to improve the quality of recommendations, because the experimental results states that the Root Mean Squared Error (RMSE) value is the smallest compared to the Matrix-Factorization, SVD, KNN and other methods.

.

I. INTRODUCTION

IN the digital era, Various platforms have made integrated recommendation systems a part of them. The competition among companies to provide the most relevant and useful recommendations is increasing. Personalized service is crucial for a company to remain competitive. Recommendation systems can filter useful patterns from historical data to produce suggestions, thereby increasing the efficiency of information utilization. However, recommendation systems still face significant challenges, namely sparsity, cold start, and scalability.

Sparsity is a condition where there are a lot of data gaps, this is because many users do not provide ratings for a large number of products [1]. Cold Start is a condition where the recommender cannot make conclusions regarding users or goods due to lack of data [2]. This generally happens to new users or new products, whose interest direction is not yet known. Scability is a condition where the system fails to handle user or product increases and provide recommendations within a reasonable time [3].

There are two techniques for building a recommendation system, namely Content-Based Filtering and Collaborative Filtering. Content-Based Filtering focuses on specific user characteristics. This method works based on user preferences, does not compare other people's choices or similarities to recommend users, the same data is not needed to recommend to other users [2]. Meanwhile, Collaborative Filtering focuses on tastes for an item. This method is based on user behavior by comparing the similarities of others to recommend items to users.

Collaborative filtering in recommending items to users refers to other users' preference data. However, this process is greatly influenced by the condition of the available data. If the data is complete, the system will be able to recommend items accurately. Conversely, if there is a lot of missing data (sparsity), it will negatively impact the quality of the recommendations produced.

Sparsity also results in less relevant recommendations, making the recommendation system less effective and reducing user satisfaction. This can lead to the loss of business opportunities, such as increased sales, customer retention, and overall business growth. Therefore, sparsity is a crucial problem as it affects the quality of recommendations and diminishes service quality. Consequently, much research has been conducted to address sparsity issues, including the work by Hafidz, MF et al., using the K-Means and Weight Point Rank (WP-Rank) algorithms [4]. Next, G. Behera and N. Nain, proposed a deep collaborative recommender system to handle non-

linearityin data and sparsity [5]. Y. Yang et all, proposed GPS (Factorized group preference-based similarity method) which combines similarity-based methods and Markov chains to offer sequential recommendations [6].

In addition, Jia H, et al, proposed a graph neural network method and information enhancement to solve the sparse knowledge graph (KG) problem in short texts [7]. Meanwhile, Luo, Y at al, proposed a new method ESATInt to method Explicitly high-dimensional and sparse features can select meaningful high-level feature interactions and can eliminate irrelevant impacts [8]. Zhang W, et al proposed a new method, namely Deep Variational Matrix Factorization (DVMF), used to solve sparsity and scalability problems [9].

Research on handling sparsity has also been carried out by Lestari, et al by proposing Porat-Rank, namely a ranking-based approach to overcome sparsity problems. Porat-Rank successfully overcomes the sparsity problem by improving the quality of recommendations [10]. Next, combining the clustering approach and ranking approach to solve sparsity and scalability problems. The clustering algorithm used is K-Means and the ranking method used is Weight Point Rank (WP-Rank) [4]. Next is to use imputation techniques to overcome the sparsity problem. The imputation technique used is Hot Deck Imputation and was successful in solving the sparsity problem [1]. As well as a comparative analysis of the Borda and Copeland methods [11], developed a new method of NFR [12], and developed a new method WP-Rank [13].

Research to resolve sparsity problems has been conducted with the aim of improving the quality of recommendations, thereby supporting service quality enhancements and creating greater opportunities for company development. Building on this, the present research proposes using Deep Learning to address sparsity issues and improve recommendation quality.

With the unstoppable growth of data volume, we have a monumental task to filter and find the information that best suits the needs and preferences of users. If the Content-Based Filtering method focuses on item characteristics. This method works based on user preferences, not comparing the choices or similarities of other users. While Collaborative Filtering focuses on tastes for an item. This method is based on user behavior by comparing the similarities of other users to recommend items to users. Based on the recommendation system, both Content-Based Filtering and Collaborative Filtering, the process is greatly influenced by the condition of the available data. If the data is complete, the system will be able to recommend items accurately. Conversely, if there is a lot of missing data (sparsity), it will have a negative impact on the quality of the recommendations produced. The use of the Deep Learning method is used to overcome this problem. Missing data is studied more deeply with the Deep Learning method, so that accurate and relevant recommendation results are obtained.

The K-Means method is a grouping of items that are used as recommendations, of course it is not a solution to the sparsity problem. Likewise, the WP-Rank method only uses rating data to improve the quality of recommendations. Use of Deep Learning in this study will provide a more effective solution to overcome the problem of sparsity compared to other methods.

II. RESEARCH METHOD

This study proposes a Deep Learning method to solve the sparsity problem, aiming to improve the quality of recommendations. This study is conducted in several stages: literature review, data collection, data pre-processing, Deep Learning implementation, and evaluation. For more details, the research stages are illustrated in Fig. 1.

Fig. 1. illustrates the stages. The first step is the literature review, which involves collecting journal articles and conference papers related to recommendation systems, collaborative filtering, sparsity, and Deep Learning.

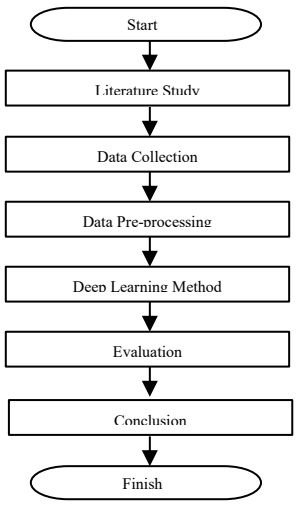


Fig. 1. Research Stages

The second stage is data collection. The dataset used in this study is the Netflix Prize Data, available at https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data. The Netflix Prize Data is a collection from Netflix's competition to improve recommendation algorithms. It contains over 100 million ratings from 480,000 randomly selected anonymous Netflix subscribers on 17,000 movie titles. The scale used is 1 to 5 stars.

The next stage is data pre-processing, which involves preparing the raw data into a structured form ready for analysis and removing any unnecessary data. Following this is the Deep Learning implementation stage. Here, the pre-processed dataset is processed using the Deep Learning method. For data that is still empty (sparsity) and has NaN values, the Deep Learning method replaces these NaN values with actual values, addressing the sparsity problem to improve recommendations. Next, the effectiveness of the method in addressing sparsity is evaluated using the Root Mean Square Error (RMSE) metric. A smaller RMSE indicates better performance in reconstructing the input data. Finally, after completing this study, conclusions are drawn, and a report is prepared.

A. Deep Learning

One branch of Machine Learning whose performance uses Artificial Neural Networks consisting of several layers, to automatically learn from data is called Deep Learning. Artificial Neural Networks in Deep Learning are employed to learn features found in complex and abstract data, as well as to make predictions and classifications on previously unseen data. In Deep Learning, the method learns through a hierarchical representation of data. In each network layer, information obtained from the previous layer is transformed and abstracted into more complex and abstract features. This representation is then used to predict or calculate the desired output. A mature scientific discipline in applying artificial intellegence to mine, analyze and recognize patterns from data [14].

Deep Learning is closely related to neural networks, namely an artificial neural network like the nerves of the human brain [15]. Deep Learning users in recommendation systems have effective capabilities. Deep Learning methods that have been used in recommendation systems include Convolutional Neural Network (CNN) [16].

Deep Learning has proven its performance capabilities in recommending. By using more complex abstract learning as an efficient and compact representation at higher layers and capturing complex relationships in data [9]. Methods that can learn robust features from input distributions and form high-level hierarchical paths [17].

B. Matrix Factorization

Matrix Factorization (MF) is a technique for calculating unknown data entries from some of the observed matrix interactions [5]. The MF method is the best choice for dealing with the problem of high levels of data sparsity in database system research. Apply Latent Sematic Index (LSI) and Singular Value Decomposition (SVD) reduction methods when using method-based approaches. SVD is the earliest recommendation algorithm in matrix decomposition [18]. Matrix factorization is a recommendation-based method that has good scalability. SVD, PMF, BPMF and VBMF are methods used in matrix factorization [19].

Matrix factorization aims to map items and users into a single space, representing the interaction of corresponding vectors. A matrix factorization technique that is often used in the recommendation system process which functions to extract latent vectors of user-item relationships and reduce the feature size by estimating missing values is usually called SVD [20]. Matrix factorization assumes that user preferences and item attributes are determined by only a small number of latent factors. so interaction information can then be mapped into a low-dimensional latent space and then unobserved ratings can be predicted by the product in the latent vector of users as well as items [9]. In Matrix Factorization, user behavior is determined by hidden factors commonly known as latent vectors, which are linear [5].

C. Collaborative Filtering

Collaborative filtering (CF) is a technique in recommendation systems that aims to provide personalization to users based on past behavior and preferences [18]. This method looks for similarities between users to make a prediction [21]. Metadata is used to overcome collaborative filtering data sparsity problems. The proposed strategy is evaluated based on MF techniques and Deep Learning approaches in predicting collaborative filtering tasks [5]. Algorithms used through collaborative filtering to create personalized recommendations to find similar users or items with similar preferences and characteristics [22]. Collaborative filtering utilizes historical user behavior, for example, providing ratings without requiring other information about the user or item [9]. CF-based methods focus on user ratings and discover unknown relationships [23]. CF carries out a filtering process for all users to obtain user information to provide recommendations, that works based on similarities in user characteristics that are able to provide information to users [24].

The neighborhood-based collaborative filtering method operates on the premise that users have similarities in their ratings of items. Similarly, items with similar rating patterns are considered alike. Therefore, vector similarity

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 11, No. 1, March 2026, Pp.705-715

calculations can be used to determine which users are similar to a particular user. When a user provides a recommendation for a film, other users can receive recommendations for films that are similar to the recommended one.

D. Cosine Similarity

Cosine Similarity is a method to measure how similar two users or two items are based on the ratings they give to the same item. The goal is to find users with similar preferences so that relevant recommendations can be provided. Items and nearest neighbors with highly predictive ratings are recommended to users with similar ratings [21].

The level of similarity between two users can be determined by comparing their respective vector devices. An N-dimensional vector can be created from user ratings to represent their similarity, and can be used to determine how similar two users' ratings are to each other [18]. User-based collaborative filtering aims to predict items that are potentially interesting to a user by leveraging the behavioral history and preferences of other users who show similar interests [25]. Equation 1, is used to calculate Cosine Similarity.

$$Cos\ \alpha = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i-1}^{n} A_i - B_i}{\sqrt{\sum_{i-1}^{n}(A_i)^2}\ x\ \sqrt{\sum_{i-1}^{n}(B_i)^2}} \quad (1)$$

A and B are vectors whose similarities will be compared. A • B is the dot product between vectors A and B. |A||B| is the cross product between |A| and |B|

E. Mean Absolute Error

Mean Absolute Error (MAE) is a commonly used evaluation method in data science. MAE works by calculating the average of the absolute difference between the predicted value and the actual value. MAE measures the average absolute error in prediction. The smaller the MAE value, the better the quality of the model. Therefore, it choosMAE) with Equation 2.

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

f_i is the forecasted value, y_i is the actual value, and n is the number of data.

F. Root Mean Square Error

Root Mean Square Error (RMSE) is the most frequently used matrix to measure recommendation system performance over time [18]. RMSE calculates the differences and error values that exist between actual data and estimated data. The RMSE value shows the level of accuracy of a method being built. The smaller the RMSE value, the higher the resulting level of accuracy [1]. Therefore, it chooses the Root Mean Square Error (RMSE) with Equation 3.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(X_i - y_i)^2}{N}} \quad (3)$$

N is the number of observations, xi is the observed value and yi is the expected value.

III. RESULTS AND DISCUSSION

This section explains the results and discussion. It is carried out after successfully collecting the required data. The first step in this study is to conduct data analysis to understand the characteristics of the data. This is useful for obtaining insights, information, and identifying any data errors from the data collection process.

Several experiments are conducted to evaluate the methods used. This study utilizes Google Colab with the Python programming language. The dataset has been sourced from Kaggle.com, specifically the Netflix Prize data. The movie rating file within the Netflix Prize data contains over 100 million ratings from 480,000 anonymous Netflix subscribers randomly selected from more than 17,000 movie titles, as illustrated in Table 1.

TABLE I
SHAPE MOVIE-TITLE

Id	Year	Name
13312	2003	Anastasia
7658	2004	Ray: Bonus Material
11522	2002	Queer as Folk: Season 2
15814	1941	Suspician
5494	1979	Connection 2

Table 1 represents five rows of data from movie titles in the Netflix Prize dataset. Next, the user data structure

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 11, No. 1, March 2026, Pp.705-715

needs to be processed to extract all ratings and form a matrix, as the file structure consists of a mixture of JSON and CSV files that are still unorganized. There are approximately 24 million different ratings by users, as shown in Table 2.

TABLE II
SHAPE USER-RATINGS

	User	Rating	Date	Movie
18153708	2532810	5.0	2005-05-02	3446
17649491	2243149	5.0	2004-03-04	3371
5096791	1952137	2.0	2000-03-14	1027
20604990	1900912	4.0	2002-08-24	3905
4024666	1688041	5.0	2005-10-18	760

Table 2 consists of four columns, namely User, Rating, Date and Movie and only displays five sample data (five rows). The rating distribution can be seen in Fig. 2.

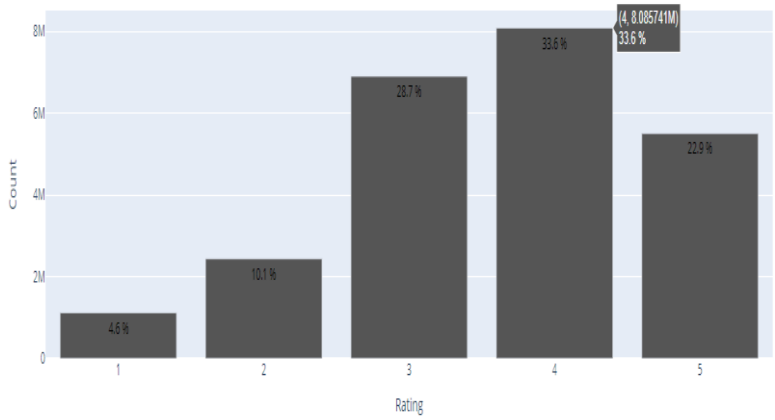


Fig. 1. Distribution of 24053764 Netflix-Rating

Netflix movies typically receive ratings rarely lower than three stars. Most ratings fall between three and four stars. As depicted in Fig. 2, the highest rating observed is four stars. Users may opt for a higher rating when they genuinely enjoy a film, potentially leading to customer retention. In contrast, casual viewers might not provide ratings at all.

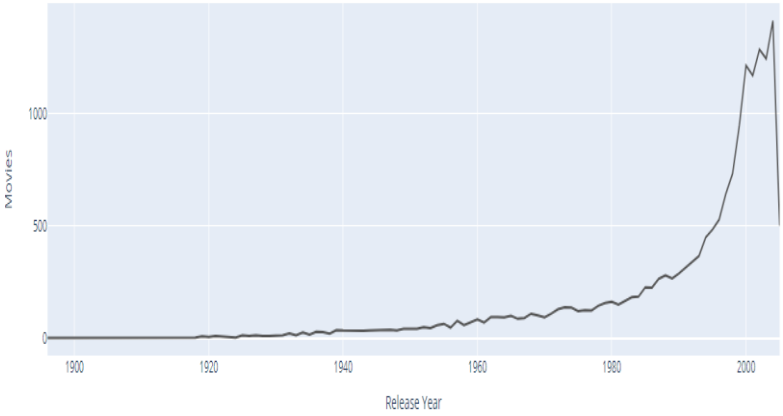


Fig. 2. 17434 Movies Grouped by Year of Release

Fig. 3. Categorizes films based on their release year, spanning from the late 19th century until the end of 2005. The peak period appears to be in early 2005.

There are several methods available for building recommender systems, one of which is the weighted average. This method calculates ratings while considering the varying levels of importance of data points in a dataset, assigning appropriate ratings to each film. The dataset contains abundant information that can be leveraged to create a robust recommendation system. Extracted information is combined using weighted averages to determine similarities between films.

It states that people choose films, then they consider not only the ratings but also the genre and popularity. A multi-objective approach incorporates film genres and ratings to predict suitable films for users.

To better visualize and understand the distribution of ratings per movie, the dataset is processed to limit the maximum number of ratings per movie to 10,000. Any column values exceeding 9,999 are capped at 9,999. The distribution involves grouping data by movie, calculating ratings counts, creating a histogram using Plotly, and

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 11, No. 1, March 2026, Pp.705-715

displaying the distribution of ratings per movie, as shown in Fig. 4. In Fig. 4 the highest rating count (928 ratings) falls within the range of 100 to 199 films. The second highest rating count (492 ratings) is in the range of 9,999 to 99,999 films.

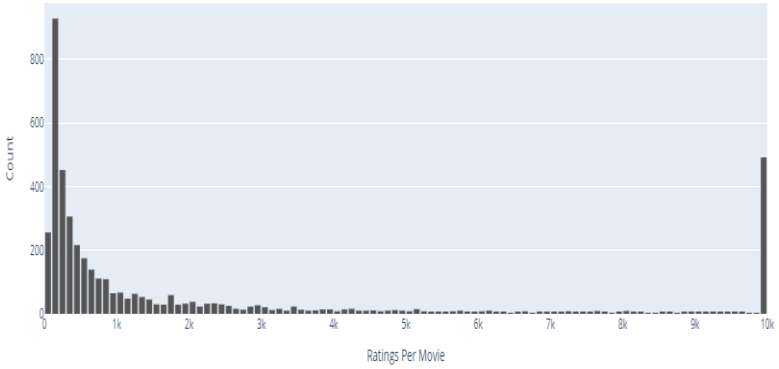


Fig. 3. Distribution of Ratings Per Movie (Clipped at 9999)

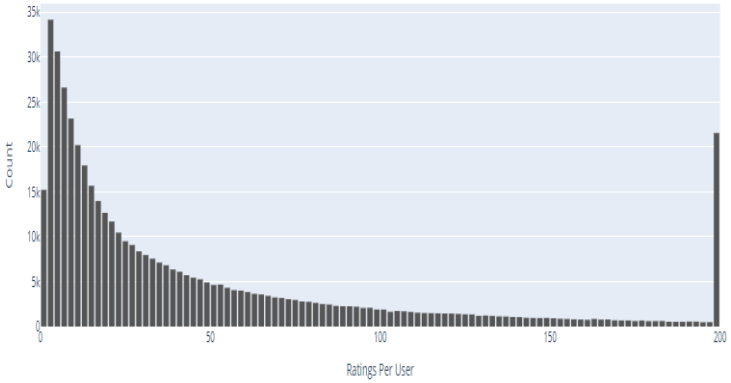


Fig. 4. Distribution of Ratings Per User (Clipped at 199)

The distribution of ratings per user is limited to 199 users to focus on higher values and enhance the readability of the histogram, emphasizing the most relevant range of values. Fig. 5 visualizes the distribution of the number of ratings given by each user in a histogram format. The highest number of ratings, approximately 34,000, falls within the range of 2 to 3 users. The second highest number of ratings, around 30,000, is in the range of 3 to 4 users.

Both the distribution of movie ratings and user ratings exhibit almost perfect exponential decay, indicating that only a few films and users receive many ratings.

In reducing the dimensionality of the dataset, movies that are rarely rated by users are filtered first. For simplicity, only movies with more than 10,000 ratings in the DataFrame are retained. Similarly, users with more than 200 ratings are filtered, resulting in a list of user IDs that have provided at least 200 ratings in the DataFrame. Subsequently, the DataFrame is filtered using these criteria to include only movies and users that meet the specified thresholds. After removing unnecessary temporary variables, the size of the DataFrame is printed before and after filtering. Initially, there are 24,053,764 movies, which reduced to 4,178,032 movies after filtering.

Once the data is filtered, it is shuffled and divided into a training set and a test set. Unnecessary columns are removed from both sets. The training set is used to train all methods, while the test set evaluates their effectiveness. For this study, the test set comprises 100,000 rows. In the training set, all rows except the last 100,000 are used, which are reserved for the test set evaluation. Empty values (sparsity) play a crucial role in this study. A sparse value indicates a movie that has not been rated but could potentially receive high ratings, making it a good recommendation for users. The goal is to predict these empty values to assist users in selecting recommended films. Next, a matrix is constructed where each row represents a user and their ratings, and each column represents a movie, as illustrated in Table 3. To display sparsity values, it uses the syntax in Fig. 6 and the results can be seen in Table 3.

```
df_p = df_train.pivot_table(index='User', columns='Movie', values='Rating')
print('Shape User-Movie-Matrix:\t{}'.format(df_p.shape))
df_p.sample(5)
```

Fig. 5. Sintax Shows Sparsity Value

TABLE III
SHAPE USER-MOVIE-MATRIX

Movie User	8	18	28	…	4392	4393	4402	4418
1717060	NaN	3.0	NaN	…	5.0	NaN	5.0	NaN
1753194	3.0	NaN	1.0	…	3.0	1.0	3.0	NaN
2382660	NaN	NaN	NaN	…	NaN	NaN	NaN	NaN
2150982	NaN	NaN	NaN	…	NaN	NaN	NaN	NaN
2132231	NaN	NaN	NaN	…	NaN	3.0	NaN	NaN

In Table 3, there are 6,148,516 NaNs in the total rating data, which should ideally consist of approximately 10 million movie ratings by users as sample data. This indicates that approximately 60% of the data is empty (sparse). The process of calculating the number of NaNs is depicted in Fig. 7.

```
total_nan_count = df_p.isna().sum().sum()
print("Total number of NaN values in DataFrame")
print(total_nan_count)
```

Fig. 7. Sintax Caluculate NaN Value

Calculating the average rating for all movies provides a single rating. These recommendations are uniform across all users and can be used when specific user information is not available. Alternatively, variations of this approach could involve creating separate rankings based on criteria such as country, year, or gender, and using these rankings individually to recommend movies to users.

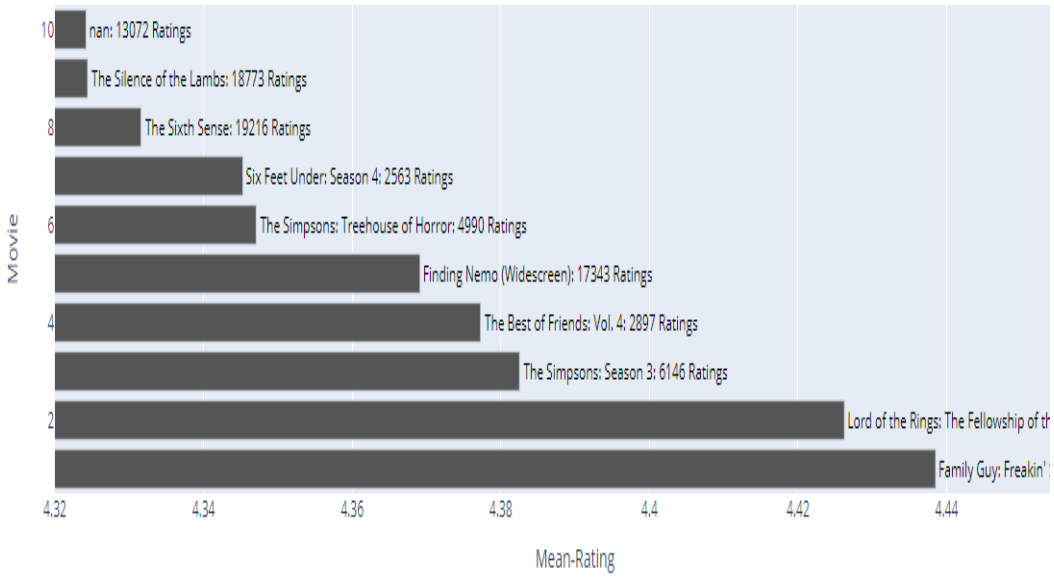


Fig. 8. Rangking of Top 10 Mean-Movie-Ratings

As depicted in Fig. 8, the top 10 movie titles with the highest average ratings have an RMSE of 0.9927. The figure shows that the film "Family Guy" received the highest rating despite having 2,790 ratings. In contrast, the lowest rating among the top 10 is for the film "The Silence of the Lambs," which received 18,775 ratings.

The weighted average rating is a method for calculating an average that considers the importance or weight of each element included in the calculation. It takes into account both the film's average rating and the number of reviews, unlike the simple average rating which only prioritizes the rating itself.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 11, No. 1, March 2026, Pp.705-715

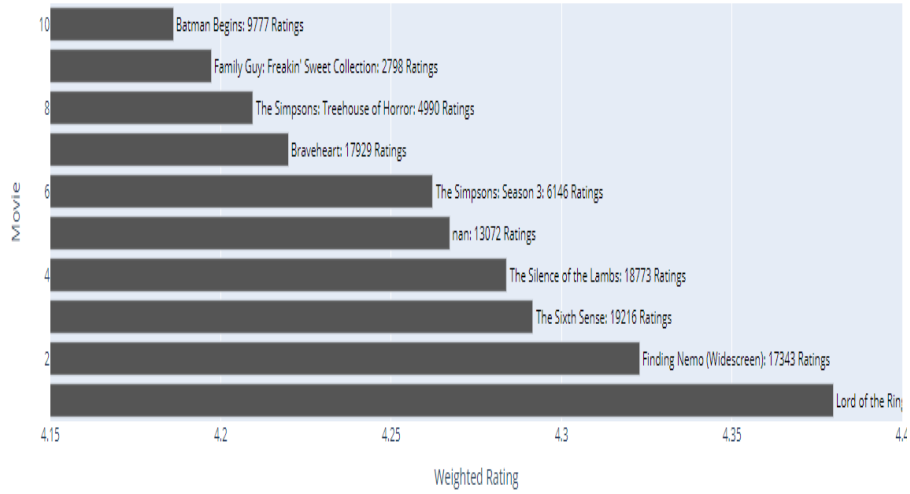


Figure 9. Ranking of Top Weighted-Movies-ratings

In Fig. 9, the highest ranking goes to "Lord of the Rings" with a rating count of 18,435. Although "Family Guy" has the highest average rating, it ranks 9th in the weighted rating due to fewer reviews. To better understand the difference between the mean and weighted ratings, refer to Fig. 8 and Fig. 9.

A. Cosine Similarity

In measuring similarity between users, cosine similarity can be used. In the matrix representation, each row represents a user, and each column represents a movie, with each cell containing the rating given by the user for that movie. Similarity between user vectors can be computed in treating each row of the matrix as a vector. This allows finding similar users who can provide recommendations to other users. However, if there are still empty values (NaNs) in the matrix, appropriate methods must be applied. One approach is to fill these empty values with each user's average rating. Subsequently, ratings from all similar users are aggregated and averaged. Movies that have not been rated by users are then recommended by identifying similar movie titles, calculated in a manner similar to user similarities while considering RMSE scores. By calculating user similarities in the dataset, it is filling missing values, and preparing recommendations based on the 100 most similar users, a movie recommendation system can be developed based on user similarities. The empty values (sparsity) in the DataFrame are filled with the average rating of each item to ensure there are no missing values in the data before calculating similarity. Fig. 10 illustrates the top 10 recommended movies for users based on user similarity, with an RMSE of 1.3336. After calculating the similarity based on the user, the films that are most similar to the selected film can be displayed in the form of a bar graph, which can also be used as recommendations. The initial step involves creating a matrix of movie descriptions and titles, calculating similarities, identifying similar movies, and creating a visualization of movie recommendations. In this case, this study selected two examples of movies, which can be seen in Fig. 11 and Fig. 12.

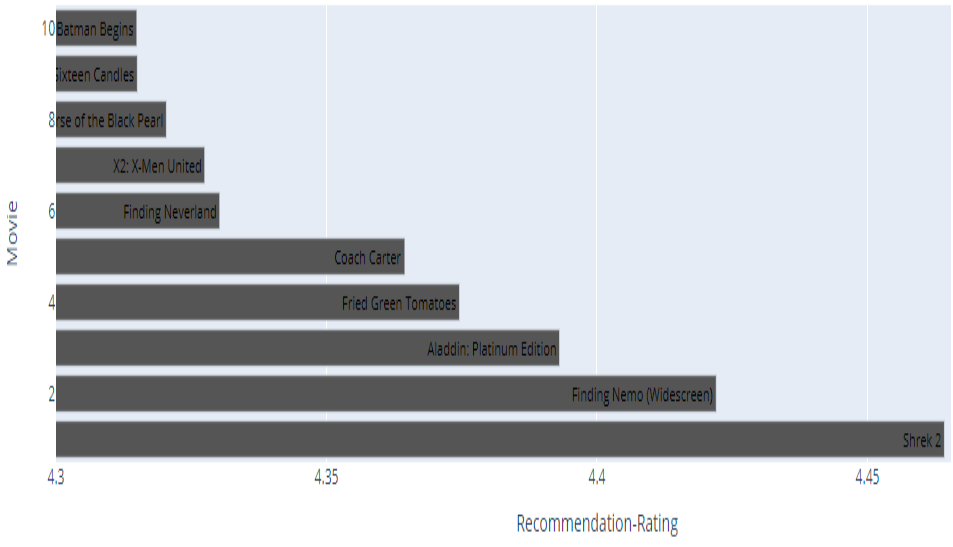


Fig. 10. Ranking of Top 10 Recommended Movies For A User Based On Similarity

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 11, No. 1, March 2026, Pp.705-715

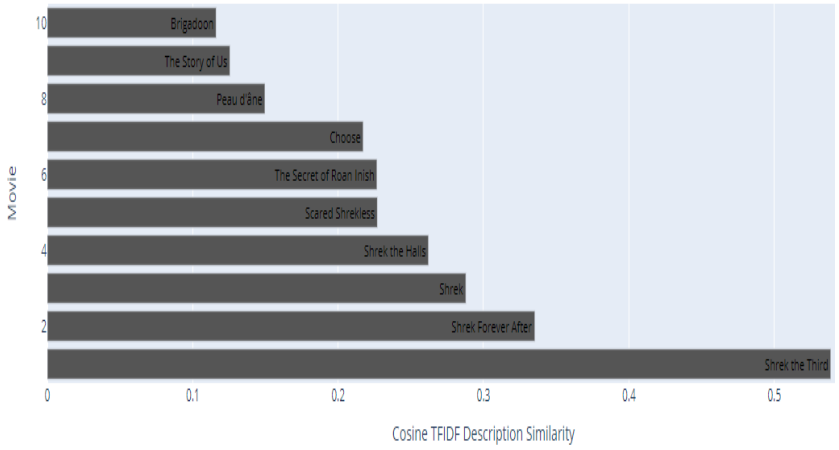


Fig. 11. Ranking of Top 10 Most Similar Movie Descriptions For "Shrek 2"



Fig. 12. Ranking Of Top 10 Most Similar Movie Descriptions For "Finding Nemo"

In Fig. 11, there are the top 10 films that have similar descriptions to the film Shrek 2, which can be used as recommendations. Likewise with Fig. 12, there are the top 10 films that have the same description as the film Finding Nemo.

B. Matrix Factorization With Keras

A matrix characterized by high user-movie ratings yet exhibiting sparsity can be transformed into dense data. Through the application of matrix factorization, a large matrix can be decomposed into two matrices of considerable length but reduced width. Subsequently, the matrix can be refined using gradient descent to accurately represent a given ranking. Latent variables that capture the underlying structure of the data set are identified against the gradient descent algorithm. Furthermore, to reconstruct the original matrix and predict the missing ratings for each user, one can use these latent variables.

In enhancing efficiency in data processing and the implementation of machine learning algorithms, a mapping is created to convert user IDs and movie IDs into numerical indices. This ensured consistency and prepared the data for further processing, ensuring that the data used for training and testing the method are aligned.

Next, the dimensions of the input variables required for the embedding method are specified, including the number of unique users and movies, and the embedding size. This step is crucial to prepare the data and structure the method to capture latent patterns in the recommendation data effectively.

The embedding process converts user IDs and movie IDs into vector representations, making them more meaningful and efficient for processing by machine learning algorithms. By leveraging embeddings, the method can learn latent relationships between users and movies, which is essential for accurate recommendations.

Following this, the embedding layer is reshaped, and the dot product of the embeddings is calculated to capture latent user and movie features, predicting the likelihood of a user preferring a movie based on method preferences and characteristics.

The method further refined its predictions using embeddings for users and movies, employing the "mse" loss function and the "adam" optimizer for training on rating data mapped to numeric IDs for users and movies. The training process included parameters such as batch size, number of epochs, validation data split, and data randomization to ensure efficient training and generalization on unseen data.

The method is then tested by predicting ratings based on test data and comparing predicted results with actual values to evaluate performance and generalization to unseen data. In the training phase, using 3,670,228 samples and validating on 407,804 samples, a loss of 2.1596 and a validation loss (val_loss) of 0.86 are achieved. The test results using hard matrix factorization yielded an RMSE of 0.9280.

C. Deep Learning With Keras

This approach is similar to matrix factorization, but it employs several dense layers to find an optimal combination. Different embedding sizes are used for users and movies, allowing the method to capture distinct feature representations based on the complexity of each feature entity. Predictions are made using test data by comparing the actual rating values from the test data and calculating the RMSE to evaluate the performance of the method used.

TABLE IV
COMPARISON OF METHODS BASED ON RMSE VALUES

Metode	RMSE
Deep Learning	0.907
Matrix-Factorization	0.927
SVDpp	1.005
SVD	1.006
BaselineOnly	1.006
KNNBaseline	1.006
KNNBasic	1.006
CoClustering	1.006
KNNWithMeans	1.006
NMF	1.006
KNNWithZScore	1.006
SlopeOne	1.006
NormalPredictor	1.006

This method can represent more complex interactions and provide more accurate rating predictions by combining user and movie embedding vectors and adding a dense layer. From the train set and test set processes used, It obtained a train set with 3,670,228 sample data and a validation set with 407,804 sample data, resulting in a loss of 0.8682 and a validation loss (val_loss) of 0.8230. The results of testing with deep learning methods yielded an RMSE of 0.907. This study uses other methods for comparison. The comparison results of several methods used, along with their RMSE values, can be seen in Table 4.

From the experimental results, the Mean Absolute Error (MAE) value is lower than the Root Mean Square Error (RMSE), but by using different methods, the MAE results are directly proportional to the RMSE, and the smaller the MAE value, the better the method used [5].

Future developments in recommendation systems involve the integration of more sophisticated technologies, such as more complex artificial intelligence (AI) and the use of more advanced machine learning techniques. Contextual concepts are also a focus of development, where systems can understand and respond to changes in user context to provide more timely and relevant recommendations. The combination of recommendation methods, such as hybrid approaches, will continue to develop to maximize user accuracy and engagement.

However, the author considers that the use of Deep Learning in this study will provide a more effective solution to overcome the problem of sparsity compared to other methods.

IV. CONCLUSION

Experimental results show that the deep learning method can overcome the sparsity problem, as indicated by the lowest RMSE value of 0.907 compared to other methods such as matrix factorization, SVDpp, SVD, and others. This study is limited to a single dataset, namely the Netflix Prize Data. Suggestions for further research include developing the study with the latest datasets to provide more accurate recommendations for users. Additionally, the method used can be further developed using a hybrid approach.

REFERENCES

[1] Sri Lestari, M. Elrico Afdila, and Yan Aditiya Pratama, “Imputation Missing Value to Overcome Sparsity Problems in The Recommendation

System," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 7, no. 6, pp. 1285–1291, 2023, doi: 10.29207/resti.v7i6.5300.
[2] M. Rahman, I. A. Shama, S. Rahman, and R. Nabil, "Hybrid Recommendation System To Solve Cold Start Problem," *J. Theor. Appl. Inf. Technol.*, vol. 100, no. 11, pp. 3562–3580, 2022.
[3] M. Singh, "Scalability and sparsity issues in recommender datasets: a survey," *Knowl. Inf. Syst.*, vol. 62, no. 1, pp. 1–43, 2020, doi: 10.1007/s10115-018-1254-2.
[4] Mohamad Fahmi Hafidz and Sri Lestari, "Solution to Scalability and Sparsity Problems in Collaborative Filtering using K-Means Clustering and Weight Point Rank (WP-Rank)," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 7, no. 4, pp. 743–750, 2023, doi: 10.29207/resti.v7i4.4543.
[5] G. Behera and N. Nain, "Handling data sparsity via item metadata embedding into deep collaborative recommender system," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9953–9963, 2022, doi: 10.1016/j.jksuci.2021.12.021.
[6] Y. Yang, D. Hooshyar, and H. S. Lim, "GPS: Factorized group preference-based similarity models for sparse sequential recommendation," *Inf. Sci. (Ny).*, vol. 481, pp. 394–411, 2019, doi: 10.1016/j.ins.2018.12.053.
[7] H. T. Jia *et al.*, "Application of graph neural network and feature information enhancement in relation inference of sparse knowledge graph," *J. Electron. Sci. Technol.*, vol. 21, no. 2, p. 100194, 2023, doi: 10.1016/j.jnlest.2023.100194.
[8] Y. Luo, W. Peng, Y. Fan, H. Pang, X. Xu, and X. Wu, "Explicit sparse self-attentive network for CTR prediction," *Procedia Comput. Sci.*, vol. 183, no. 2018, pp. 690–695, 2021, doi: 10.1016/j.procs.2021.02.116.
[9] W. Zhang, X. Zhang, H. Wang, and D. Chen, "A deep variational matrix factorization method for recommendation on large scale sparse dataset," *Neurocomputing*, vol. 334, pp. 206–218, 2019, doi: 10.1016/j.neucom.2019.01.028.
[10] M. Safitri, F. Rahmadani, E. Loniza, and S. Anggoro, *Simple Visible Light Spectrophotometer Design Using 620 Nm Optical Filter*, vol. 746 LNEE. 2021. doi: 10.1007/978-981-33-6926-9_54.
[11] S. Lestari, T. B. Adji, and A. E. Permanasari, "Performance Comparison of Rank Aggregation Using Borda and Copeland in Recommender System," *2018 Int. Work. Big Data Inf. Secur. IWBIS 2018*, no. March 2019, pp. 69–74, 2018, doi: 10.1109/IWBIS.2018.8471722.
[12] S. Lestari, T. B. Adji, and A. E. Permanasari, "NRF: Normalized Rating Frequency for Collaborative Filtering Paper," *Proc. ICAITI 2018 - 1st Int. Conf. Appl. Inf. Technol. Innov. Towar. A New Paradig. Des. Assist. Technol. Smart Home Care*, pp. 19–25, 2018, doi: 10.1109/ICAITI.2018.8686743.
[13] S. Lestari, T. B. Adji, and A. E. Permanasari, "WP-Rank: Rank aggregation based collaborative filtering method in recommender system," *Int. J. Eng. Technol.*, vol. 7, no. 4, pp. 193–197, 2018, doi: 10.14419/ijet.v7i4.40.24431.
[14] Aria Maulana, Muhammad Rivaldi Asyhari, Yufis Azhar, and Vinna Rahmayanti Setyaning Nastiti, "Disease Detection on Rice Leaves through Deep Learning with InceptionV3 Method," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 7, no. 5, pp. 1147–1154, 2023, doi: 10.29207/resti.v7i5.4344.
[15] F. Zamachsari and N. Puspitasari, "Penerapan Deep Learning dalam Deteksi Penipuan Transaksi Keuangan Secara Elektronik," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 5, no. 2, pp. 203–212, 2021, doi: 10.29207/resti.v5i2.2952.
[16] Umar Aditiawarman, Dimas Erlangga, Teddy Mantoro, and Lutfil Khakim, "Face Recognition of Indonesia's Top Government Officials Using Deep Convolutional Neural Network," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 7, no. 1, pp. 113–119, 2023, doi: 10.29207/resti.v7i1.4437.
[17] A. Priyatama, Z. Sari, and Y. Azhar, "Deep Learning Implementation using Convolutional Neural Network for Alzheimer's Classification," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 7, no. 2, pp. 310–217, 2023, doi: 10.29207/resti.v7i2.4707.
[18] A. Fareed, S. Hassan, S. Brahim, and Z. Halim, "Machine Learning with Applications A collaborative filtering recommendation framework utilizing social networks," *Mach. Learn. with Appl.*, vol. 14, no. January, p. 100495, 2023, doi: 10.1016/j.mlwa.2023.100495.
[19] W. Zhang, X. Zhang, H. Wang, and D. Chen, "Neurocomputing A deep variational matrix factorization method for recommendation on large scale sparse dataset," *Neurocomputing*, vol. 334, pp. 206–218, 2019, doi: 10.1016/j.neucom.2019.01.028.
[20] N. Heidari and A. Koochari, "An attention-based deep learning method for solving the cold-start and sparsity issues of recommender systems," no. September 2022, 2023, doi: 10.1016/j.knosys.2022.109835.
[21] Z. Romadhon, E. Sediyono, and C. E. Widodo, "Various Implementation of Collaborative Filtering-Based Approach on Recommendation Systems using Similarity," *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 4, no. 3, pp. 179–186, 2020, doi: 10.22219/kinetik.v5i3.1062.
[22] M. Hasan, "A Comprehensive Collaborating Filtering Approach using Extended Matrix Factorization and Autoencoder in Recommender System," vol. 10, no. 6, 2019.
[23] J. Jiang, W. Li, A. Dong, Q. Gou, and X. Luo, "A Fast Deep AutoEncoder for high-dimensional and sparse matrices in recommender systems," *Neurocomputing*, vol. 412, pp. 381–391, 2020, doi: 10.1016/j.neucom.2020.06.109.
[24] H. Yuwafi, F. Marisa, and I. D. Wijaya, "Implementasi Data Mining Untuk Menentukan Santri Berprestasi Di Pp . Manaarulhuda Dengan Metode Clustering Algoritma K-Means," *J. SPIRIT*, vol. 11, no. 1, pp. 22–29, 2019.
[25] Y. Zhang, H. Xu, and X. Yu, "The Recommendation Algorithm Based on Improved Conditional Variational Autoencoder and Constrained Probabilistic Matrix Factorization," *Appl. Sci.*, vol. 13, no. 21, p. 12027, 2023, doi: 10.3390/app132112027.