Received: September 8, 2023. Revised: October 20, 2023.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Time Loss Function-based Collaborative Filtering in Movie Recommender System

Triyanna Widiyaningtyas[1]* Didik Dwi Prasetya[1] Heru Wahyu Herwanto[1]

[1]*Department of Electrical Engineering and Informatics, Universitas Negeri Malang, Indonesia*
* Corresponding author's Email: triyannaw.ft@um.ac.id

Abstract: Neighbor-based collaborative filtering is one of the prevalent recommendation approaches that apply similarity algorithms. Using a similarity algorithm to improve the accuracy of the recommendation system is one challenge in collaborative filtering. The computation of similarity now heavily relies on user rating and behavior scores. Users' preferences for each product kind (genre) demonstrate the user behavior's value. The algorithm's weakness is that it only considers genre data when estimating user behavior, regardless of when the user rated the item. Therefore, this study aims to develop a new similarity algorithm by considering the genre and the time weight of the item rating, which is called time loss function-based similarity (TLFSim). Newly assessed items have a higher weight than those estimated for a long time. Our experiment tested the TLFSim's performance compared to the user-score-probability-collaborative-filtering (UPCF) algorithm using the MovieLens 100k dataset. The experimental results demonstrate that the TLFSim algorithm surpasses the prior approach regarding recommendation accuracy, reducing mean absolute error (MAE) by 8.28% and root mean square error (RMSE) by 4.57%.

Keywords: Neighbor-based collaborative filtering, Similarity metric, TLFSim.

1. Introduction

Today, e-commerce is growing and is predicted to become a significant trend. One of the major benefits of e-commerce over traditional sales is the simplicity with which the number of products made available to users may be increased as no physical display space is required. However, because so many products are available on e-commerce systems, customers eventually give in to information overload [1–3]. For better outcomes, e-commerce should allocate resources to overcome this challenge.

Due to the vast number of users and products, recommender systems have been presented as a solution to the problem of information overload [4, 5]. These systems help users by providing personalized products they are more likely to be interested in by filtering out undesirable products and making suggestions based on their interests, preferences, or historical behavior. The tremendous sales of large e-commerce platforms like Amazon, Netflix, etc., are primarily due to recommender systems [6].

Various recommender systems, including collaborative, content-based, knowledge-based, demographic, utility-based, and hybrid filtering, combine other techniques [7–10]. Collaborative filtering is among the most well-known, practical, and frequently applied algorithms among the several recommender systems approach [4, 11, 12]. The collaborative filtering strategy focuses on the products that other users with similar preferences have previously enjoyed. The taste similarity of numerous users is specified based on the rating data similarity (on a scale of 1 to 5 for movies) or the history of users' browsing [13, 14].

Collaborative filtering contains two methods: model-based and neighbor-based [15–17]. The names of these two methods come from how they perform the learning activity based on the users' product preferences. The first method involves learning formulas and rules to predict the unknown preferences of the active user. The second method, in contrast, calculates the preference similarity

between each pair of users while offline [18]. The traditional similarities frequently applied in recommender systems are cosine, person correlation coefficient (PCC), and Jaccard Coefficient [19]. Based on the learning activity of the two methods, the model-based requires complex formulas in the learning process. In contrast, the neighbor-based method gives interpretability and simplicity to the recommendation.

Several studies on neighbor-based collaborative filtering have focused heavily on calculating the similarity of interest between each pair of users utilizing different approaches. Their objective is to constantly enhance the effectiveness of neighbor set identification for the active user to get better recommendation accuracy. Some of these similarity functions are proximity-significance-singularity (PSS) [20], newPCC [21], Bhattacharyya-similarity [22], PCCJaccard [23], multi-level-collaborative-filtering [24], item-frequency-based-similarity [25], triangle-multiplying-jaccard (TMJ) [26], and three-impact-factors-based-similarity [27]. These similarity functions only utilize the user rating data to compute the similarity between each pair of users. The user rating data indicate the score obtained from users when they rate the products.

Recent studies have presented the similarity functions by utilizing user behavior and rating data. The user rating data represents the rating score provided directly by users. Meanwhile, the user behavior data denote an accumulated score obtained from users indirectly in accessing genre data [28, 29]. The final similarity computation incorporates the similarity calculation using user rating data and the similarity calculation using user behavior data. The similarity functions are user-score-probability-collaborative-filtering (UPCF) [28] and user-profile-correlation-based-similarity (UPCSim) [29]. The evaluation of these similarity functions utilizes the MovieLens 100k dataset. The UPCF results in mean absolute error (MAE) and root mean square error (RMSE) values of 0.7572 and 0.9588. While the UPCSim results in MAE and RMSE values of 0.7408 and 0.9448. The results of these studies can improve the recommendation performance (especially in the accuracy of rating prediction, MAE and RMSE) compared to the traditional similarity (Cosine) that generates MAE and RMSE values of 0.8074 and 1.0201). Based on the evaluation results of these previous functions, there is still room to improve prediction accuracy by exploring other factors that influence prediction results.

The limitation with these similarity functions is that the similarity calculation only considers the genre data when attempting to estimate user behavior value, regardless of when the user rates the item. It means the predicted rating's accuracy is still high. Based on the limitation, our study proposes a new similarity algorithm by considering the genre and the time weight of the item rating. We assume that the newly assessed items have a higher weight than items that have been assessed for a long time. Our proposed model is called time loss function-based similarity (TLFSim).

The outline of this paper is organized as follows. First, section 2 discusses related works in the similarity algorithms in the recommender system. Then, the detail of our proposed method is described in section 3. Finally, experimental results are illustrated and discussed in section 4, followed by a conclusion section.

2. Related work

Neighbor-based methods employ historical user rating data to calculate the similarity between users or products. These methods aim to establish a similarity function between users or products and identify the most similar to suggest unrated products. The similarity function is the primary key of the neighbor-based method, which shows the user correlation. A higher similarity value indicates a higher correlation. The two conventional similarity functions typically used in recommender systems are cosine similarity and person correlation coefficient (PCC). The similarity between $user_x$ (u_x) and $user_y$ (u_y) utilizes cosine similarity and PCC formulated in Eqs. (1) and (2), respectively.

$$Sim(u_x, u_y)^{COS} = \frac{\vec{r}_{u_x} \cdot \vec{r}_{u_y}}{\|\vec{r}_{u_x}\| . \|\vec{r}_{u_y}\|} = \frac{\sum_{p \in P_{u_x} \cap P_{u_y}} r_{u_x p} . r_{u_y p}}{\sqrt{\sum_{p \in P_{u_x} \cap P_{u_y}} r_{u_x p}^2} . \sqrt{\sum_{p \in P_{u_x} \cap P_{u_y}} r_{u_y p}^2}} \quad (1)$$

$$Sim(u_x, u_y)^{PCC} = \frac{\sum_{p \in P_{u_x} \cap P_{u_y}} (r_{u_x p} - \bar{r}_{u_x})(r_{u_y p} - \bar{r}_{u_y})}{\sqrt{\sum_{p \in P_{u_x} \cap P_{u_y}} (r_{u_x p} - \bar{r}_{u_y})^2} . \sqrt{\sum_{p \in P_{u_x} \cap P_{u_y}} (r_{u_y p} - \bar{r}_{u_y})^2}} \quad (2)$$

$r_{u_x p}$ and $r_{u_y p}$ express the rating score to product p from $user_x$ (u_x) and $user_y$ (u_y), respectively. P_{u_x} and P_{u_y} indicate the set of products rated by $user_x$ (u_x) and $user_y$ (u_y), respectively. p is one of the products rated by both users. $\bar{r}_{u_x}$ and $\bar{r}_{u_y}$ represent the average rating of all products rated by $user_x$ (u_x) and $user_y$ (u_y), respectively.

In the last decade, some studies proposed the similarity functions to enhance the recommender

system performances. For example, Liu et al. [20] presented a new similarity function composed of three similarity factors (proximity, significance, and singularity). The first factor computes the absolute difference between the two ratings but also considers whether or not they agree. The second factor indicates that ratings have greater significance when they are further apart than the median rating. The third factor shows the variance between the two ratings compared to other ratings. Thus, the similarity function is called PSS similarity. Next, Sheugh and Alizadeh [21] introduced an extension of the PCC similarity function called newPCC similarity. The similarity aims to reduce NaN results for cases in which there is no similarity between users. Saranya et al. [23] offered a PCCJac similarity function that combined PCC similarity and Jaccard Coefficient. Hereafter, Zhang et al. [25] represented the item frequency-based PCC, which improves PCC's similarity. The similarity considers the weight of each item before calculating the correlation coefficient between users. Feng et al. [27] presented a novel similarity function by integrating three impact factors. The first impact factor aims to compute the similarity between users, the second factor identifies the user rating propensity, and the third factor describes each user's rating weight. Thus, the similarity function is named three-impact factors-based similarity. In general, these similarity functions only consider the user rating data to compute the similarity between users. Using only user rating data may cause the recommender system to produce incorrect inferences, thus affecting the recommendation's performance [29].

Furthermore, several studies have proposed the similarity functions by combining the user rating-based and behavior-based similarities. Their studies assume that users give a low rating to a product title, not necessarily that they dislike the product type/genre. The user rating-based similarity computes the similarity between users based on the user rating data that utilized the cosine similarity. The user behavior-based similarity calculates the similarity between users considering the user behavior data that adopted the PCC similarity (that replaces the user rating data with the user behavior data). The user rating data is the rating score given directly to the product. Meanwhile, the user behavior data is an accumulated score obtained indirectly in accessing genre data.

After giving weight to each similarity, The final similarity is computed with a combination of these two similarities. Examples of these similarities are user-score-probability-collaborative-filtering (UPCF) [28] and user-profile-correlation-based-similarity (UPCSim) [29]. These two similarities have different ways of assigning similarity weights. The UPCF similarity uses the threshold value for weighting. The UPCSim utilizes the correlation coefficient between user profile data (age, gender, occupation, and location) and user rating/behavior data. Eqs. (3) and (4) define these similarity formulas.

$$Sim(u_x, u_y)^{UPCF} = \beta . S_r(u_x, u_y) + (1 - \beta) . S_b(u_x, u_y) \quad (3)$$

$$Sim(u_x, u_y)^{UPCSim} = \alpha . S_r(u_x, u_y) + \delta . S_b(u_x, u_y) \quad (4)$$

$S_r(u_x, u_y)$ denotes the similarity based on user rating data between $user_x$ (u_x) and $user_y$ (u_y) that utilized the cosine similarity, which is formulated in Eq. (1). $S_b(u_x, u_y)$ states the similarity based on user behavior data between $user_x$ (u_x) and $user_y$ (u_y) that adopted the PCC similarity (by replacing the user rating data with the user behavior data), which formulated in Eq. (5). β is the threshold value that ranges from 0 to 1. α and δ indicate each similarity's weight computed based on the correlation coefficient (r) utilizing multiple linear regression analysis. Eq. (6) defines the multiple linear regression formula, and Eq. (7) states the correlation coefficient formula (r).

$$S_b(u_x, u_y) = \frac{\sum_{g \in G_{u_x} \cap G_{u_y}} (P_{u_x g} - \bar{P}_{u_x})(P_{u_y g} - \bar{P}_{u_y})}{\sqrt{\sum_{g \in G_{u_x} \cap G_{u_y}} (P_{u_x g} - \bar{P}_{u_x})^2} \cdot \sqrt{\sum_{g \in G_{u_x} \cap G_{u_y}} (P_{u_y g} - \bar{P}_{u_y})^2}} . \quad (5)$$

$P_{u_x g}$ and $P_{u_y g}$ indicate the probability score for product type/genre g from $user_x$ (u_x) and $user_y$ (u_y), respectively. G_{u_x} and G_{u_y} represent the product types rated by $user_x$ (u_x) and $user_y$ (u_y), respectively. g is one of the product types rated by both users. $\bar{P}_{u_x}$ and $\bar{P}_{u_y}$ express the average probability scores of all product types rated by $user_x$ (u_x) and $user_y$ (u_y), respectively.

$$y = a + b_1 x_1 + b_2 x_2 + b_3 x_3 + \cdots + b_n x_n \quad (6)$$

$$r = \sqrt{\frac{b_1 \sum x_1 y + b_2 \sum x_2 y + b_3 \sum x_3 y + \ldots + b_n \sum x_n y}{\sum y^2}} \quad (7)$$

y is a dependent variable that is represented by

the user rating value. x is an independent variable that is represented by the user profile data that consists of age (x_1), gender (x_2), occupation (x_3), and location (x_4). a is a constant, and b is the regression coefficient for each independent variable.

Using the MovieLens dataset, these two similarity functions (UPCF [28] and UPCSim [29]) can generate a rating prediction close to the actual rating. In other words, the UPCF [28] and UPCSim [29] can improve recommendation performance compared to similarity algorithms that only depend on user rating data. It happens because these two similarities involve the user behavior data to calculate the similarity between users. In addition, the experimental results show that UPCSim [29] can outperform UPCF [28] because UPCSim [29] uses more complete data in calculating similarity (namely, adding user profile data for computing the similarity weights).

Although both similarities are superior, both similarities still need to improve their accuracy because they do not consider the time factor when assessing items.

3. Research method

This work proposes an approach for making recommendations called TLFSim, which involve the time when assessing the item. Fig. 1 shows the four stages of our study: input, data preparation, neighbor-based process, and output.

3.1 Input

The input dataset is the first stage of the developed neighbor-based collaborative filtering system. In this paper, we used MovieLens 100k. There are 100,000 ratings in the MovieLens 100k dataset, which includes 1,682 films and 943 people. Each user in this dataset has rated at least 20 films using a scale from 1 to 5. A score of 1 means the user detests the film, while a score of 5 means the user enjoys it. The datasets have 93.7% data sparsity and 6.3% density [30]. This rating data structure consists of user-id, movie-id, rating, and timestamp. All items in the recommendation system must have ratings to obtain accurate user preference. Hence, we used rating prediction to anticipate the unrated items using the kNN algorithm.

3.2 Data preparation

The second stage is the data preparation. This stage performs data pre-processing by reducing irrelevant attributes. These irrelevant attributes are the movie title, release date, video release date, and

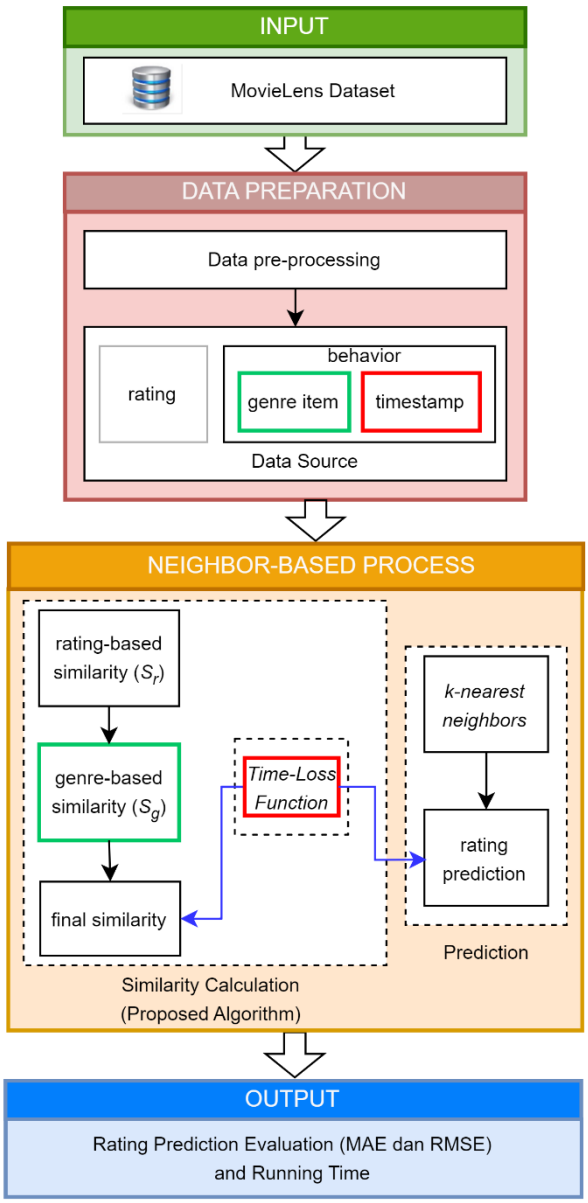

Figure. 1 Research stages

IMDb URL of the item data.

3.3 Neighbor-based process

The neighbor-based process is the third stage of this study. The two steps in this stage are similarity calculation and rating prediction. In this study, similarity calculation employs TLFSim, which considers the time-loss function to calculate similarity. The rating prediction method utilizes *k* nearest neighbor (kNN).

TLFSim combines rating-based and genre-based similarities by considering the time-loss function [31]. The rating-based similarity refers to Eq. (1), and the genre-based similarity refers to Eq. (3). Consideration of time-related ratings in the collaborative filtering approach is crucial because

the user's interests change over time. These ratings, which may be derived utilizing the right time loss functions, raise the system's efficiency. Two factors are critical from the standpoint of these functions: first, choosing the right time loss function, and second, the level at which the time loss functions are inferred. Two well-liked time loss functions: power and exponential. The power time loss function is formulated in Eq. (8).

$$TLF^{Power} = |T_{u_{x,p}} - T_{u_{y,p}}|^{-\varphi} \quad (8)$$

φ is a tuning parameter, $T_{u_{x,p}}$ and $T_{u_{y,p}}$ represent the timestamp of $user_x$ (u_x) and $user_y$ (u_y) on product p, respectively. The range of time function value is (0,1), which is reduced with time.

This study uses the power function to be applied in calculating similarity and rating prediction. The similarity between users employing the time loss function is calculated as follows:

$$Sim(u_x, u_y)^{TLFSim} = Sim(u_x, u_y)^{UPCF} * TLF \quad (9)$$

The nearest neighbors for each target user are identified using the similarity calculated by Eq. (9), which aids in producing the prediction. The prediction formula is given in Eq. (10) below.

$$\hat{r}_{u_x p} = \bar{r}_{u_x} + \frac{\sum_{u_y \epsilon NNu_x} Sim(u_x,u_y).\left(r_{u_y p} - \bar{r}_{u_y}\right)}{\sum_{u_y \epsilon NNu_x} |Sim(u_x,u_y)|} * TLF \quad (10)$$

$\hat{r}_{u_x p}$ is the predicted rating score of the $user_x$ (u_x) to product p. NNu_x is the set of nearest neighbors to $user_x$ (u_x). $Sim(u_x, u_y)$ is the final similarity between $user_x$ (u_x) and $user_y$ (u_y) using TLFSim. $\bar{r}_{u_x}$ and $\bar{r}_{u_y}$ is the rating average of $user_x$ (u_x) and $user_y$ (u_y), respectively. $r_{u_y p}$ is the rating score given by $user_y$ (u_y) to product p.

3.4 Output

Output is the last stage in this research. This stage evaluates how well-proposed algorithms perform compared to previous algorithms. In this study, the performance evaluation uses prediction and running time metrics. Prediction metrics utilize mean absolute error (MAE) and root mean square error (RMSE), which refer to Eqs. (9) and (10) [32, 33].

$$MAE = \frac{1}{N} \sum_{u_x \epsilon U_x, p \epsilon P} \left| r_{u_x p} - \hat{r}_{u_x p} \right| \quad (9)$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{u_x \epsilon U_x, p \epsilon P} \left(r_{u_x p} - \hat{r}_{u_x p} \right)^2} \quad (10)$$

$r_{u_x p}$ states the rating score given by $user_x$ (u_x) to product p. $\hat{r}_{u_x p}$ is the predicted rating score of $user_x$ (u_x) to product p.

4. Experiment

We start by outlining the experimental setting in this section. We then review the findings by contrasting the proposed algorithm with earlier algorithms. Finally, we discuss the results of our experiment.

4.1 Experimental setting

We employed MovieLens 100k to assess the effectiveness of the proposed TLFSim algorithm. The *k*-fold cross-validation (CV) method splits this dataset into training and testing data. The training data aims to create the similarity model, while the testing data functions to evaluate the recommendation method. We set the value of *k* with *k*=5 to get separate 80% of training data and 20% of testing data. We also set the number of nearest neighbors (*k*) with *k*=10 to *k*=100. We compared the neighbor-based collaborative filtering algorithms (Cosine, UPCF [28], and UPCSim [29]) with our proposed TLFSim algorithm.

This study utilized the computer specifications of Processor 11th Gen Intel® Core™ i7-1165G7 @ 2.80 GHz, 1690 MHz (4 Core), and Memory of 16 GB. The algorithms in Python were running under Microsoft Windows 7.

4.2 Experimental results

With training and testing data split (80%:20%) on the MovieLens 100k, this section compares the performance of our proposed TLFSim algorithm with the leading rating-based approaches. Metrics for evaluating performance use MAE, RMSE, and running time.

Table 1 compares of MAE values in MovieLens 100k with the distribution of training data and testing data of 80%:20%.

In all similarity algorithms, the average MAE values decline as the number of nearest neighbors grows. The average of MAE values using TLFSim reduces compared to other algorithms. The decrease in the average MAE values compared to UPCSim is 5.93%, UPCF is 8.28%, and Cosine is 15.45% in the MovieLens 100k dataset. It shows that the prediction accuracy using the time-loss function experienced an average increase compared to

Table 1. Comparison of the average MAE values of four neighbor-based algorithms in MovieLens 100k dataset

k	MAE			
	Cosine	UPCF [28]	UPCSim [29]	TLFSim
10	**0.8227**	**0.7792**	**0.7669**	**0.7135**
20	0.8099	0.7631	0.7483	0.7049
30	0.8051	0.7565	0.7410	0.7013
40	0.8048	0.7551	0.7387	0.7002
50	0.8043	0.7544	0.7369	0.6983
60	**0.8041**	0.7535	0.7364	0.6976
70	0.8049	0.7529	0.7359	0.6961
80	0.8051	0.7526	0.7355	0.6952
90	0.8056	0.7525	0.7347	0.6937
100	0.8072	**0.7521**	**0.7337**	**0.6923**
Avg	0.8074	0.7572	0.7408	0.6993

Table 2. Comparison of the average RMSE values of four neighbor-based algorithms in MovieLens 100k dataset

k	RMSE			
	Cosine	UPCF [28]	UPCSim [29]	TLFSim
10	**1.0417**	**0.9835**	**0.9793**	**0.9674**
20	1.0248	0.9643	0.9541	0.9500
30	1.0189	0.9574	0.9453	0.9166
40	1.0163	0.9558	0.9427	0.9116
50	1.0162	0.9556	0.9393	0.9107
60	**1.0154**	0.9547	0.9389	0.9096
70	1.0162	0.9544	0.9383	0.9074
80	1.0170	0.9543	0.9381	0.9069
90	1.0173	0.9542	0.9364	0.9058
100	1.0176	**0.9538**	**0.9359**	**0.9033**
Avg	1.0201	0.9588	0.9448	0.9169

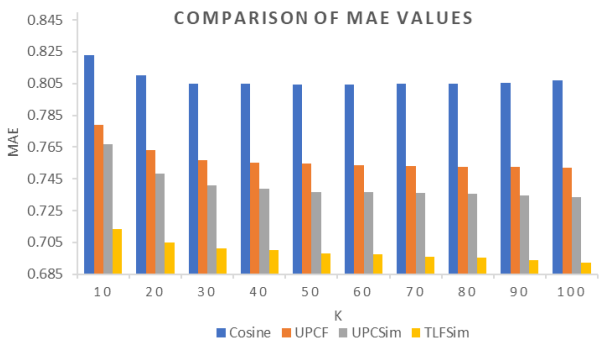

Figure. 2 Comparison of the average MAE values of the four similarity algorithms using MovieLens 100K dataset

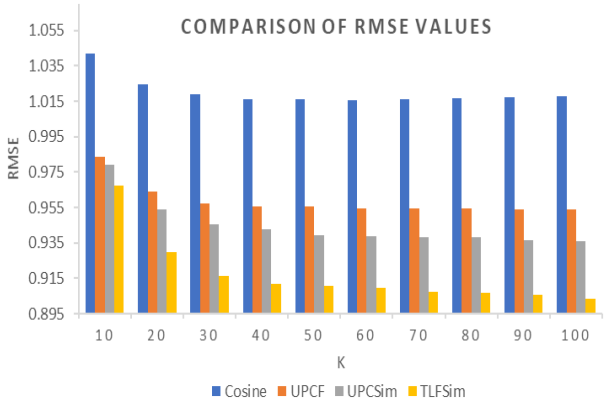

Figure. 3 Comparison of the average RMSE values of the four similarity algorithms using MovieLens 100K dataset

without the time-loss function.

Fig. 2 shows the MAE values of the four algorithms decrease with increasing number of nearest neighbors.

At the beginning of the curve, the decrease in the MAE value is very sharp as the number of nearest neighbors increases, while at the end of the curve, the greater the number of nearest neighbors, the MAE value tends to be stable. It can be said that the number of nearest neighbors influences the MAE value, where the greater the number of nearest neighbors, the smaller the MAE value. With the same number of nearest neighbors, the MAE value of the TLFSim algorithm is always smaller than that of other algorithms. In other words, the error between the actual and the predicted ratings of the proposed TLFSim is the smallest.

Table 2 compares the RMSE values in the MovieLens 100k dataset with the distribution of training data and testing data of 80%:20%. Increasing the value of *k* indicates a smaller RMSE value, which means that the number of nearest neighbors affects the method's performance.

Based on Table 2, the proposed algorithm always gives the smallest RMSE in each condition of the number of nearest neighbors compared to the other three similarity algorithms. The decrease in the average RMSE values compared to UPCSim is 3.05%, UPCF is 4.57%, and Cosine is 8.21% in the MovieLens 100k dataset.

It shows that the accuracy of TLFSim is the lowest, which means the proposed algorithm is superior and becomes the advantage of TLFSim. It happens because our proposed algorithm employs the time-loss function that gives the weight for the item period.

Fig. 3 illustrates how changing the number of nearest neighbors affects the RMSE value. The four algorithms show a decrease in the RMSE value first and are stable when neighbors exceed 50. The RMSE value of the TLFSim algorithm always indicates the smallest value for each different number of neighbors. It shows that TLFSim has the lowest error rate than the other two algorithms and

Table 3. Comparison of the average running time values of four neighbor-based algorithms in MovieLens 100k dataset

k	Algorithm			
	TLFSim	UPCSim [29]	UPCF [28]	Cosine
10	4.63	4.78	4.51	3.67
20	4.57	4.82	4.48	3.91
30	4.82	5.02	4.60	4.21
40	5.13	5.27	4.70	4.42
50	5.53	5.68	5.00	4.82
60	5.83	5.99	5.28	5.08
70	5.85	6.02	5.38	5.10
80	5.89	6.10	5.38	5.25
90	5.96	6.15	5.58	5.29
100	6.17	6.28	5.81	5.25
Avg	5.44	5.61	5.07	4.70

confirms the superiority of TLFSim.

In addition to measuring the MAE and RMSE values as recommendation metrics, this experiment also evaluated the running time of four algorithms to see the effect of the time-loss function on the running time of algorithms. Table 3 compares the running time on the MovieLens 100k dataset with the distribution of training data and testing data of 80%:20%.

Based on Table 3, TLFSim requires a longer average execution time, namely 1.08 seconds compared to Cosine and 0.71 seconds compared to UPCF [28]. However, TLFSim requires an average execution time of 0.17 seconds faster than UPCSim [29]. This result occurs because the TLFSim algorithm, apart from calculating the final similarity between users, also has an additional step in calculating the time loss function compared to the three previous algorithms.

4.3 Discussion

In this study, we propose a recommendation algorithm that utilizes a neighbor-based algorithm with a time-loss function called TLFSim. The neighbor-based method considers user ratings and behavior scores to accommodate user preferences. Besides that, this algorithm also employs time when a user assesses the item. In testing the method, we used the popular MovieLens 100k dataset. This dataset has sparsity levels of 93.7%, while density levels are 6.3%. The distribution of training data and testing data on the dataset is carried out using the *k*-folds CV method at *k*=5. Predictive accuracy (MAE and RMSE) and running time evaluate the algorithm's performances.

The experiment results showed that employing a time-loss function in neighbor-based collaborative filtering could improve the prediction performance by reducing MAE and RMSE compared to the state-of-the-art algorithms (i.e., UPCSim [29], UPCF [28], and Cosine). In addition, the performance of the recommendation processing time is faster than UPCSim [29]. However, it is lower than the other two algorithms (UPCF [28] and Cosine). It occurs because the similarity calculation also considers time data, requiring an additional step that consumes more time. It becomes the limitation of our work.

5. Conclusion

Based on the results and discussion in the previous section, applying the TLFSim algorithm produces a better predictive accuracy value than the earlier algorithms (UPCSim, UPCF, and Cosine). Still, it consumes more time compared to the UPCF and Cosine algorithms. However, the TLFSim is still faster than the UPCSim algorithm.

For further research, the system development can consider matrix factorization, clustering, and parallel processing to calculate user similarity and explore other hybrid methods to improve recommendation performance.

List of notations

$Sim(u_x, u_y)$ = similarity between $user_x$ (u_x) and $user_y$ (u_y).

$r_{u_x p}$ = the rating score from $user_x$ (u_x) to product p.

$r_{u_y p}$ = the rating score from $user_y$ (u_y) to product p.

P_{u_x} = the set of products rated by $user_x$ (u_x).

P_{u_y} = the set of products rated by $user_y$ (u_y).

$\bar{r}_{u_x}$ = the average rating of all products rated by $user_x$ (u_x).

$\bar{r}_{u_y}$ = the average rating of all products rated by $user_y$ (u_y).

$S_r(u_x, u_y)$ = the similarity based on user rating data between $user_x$ (u_x) and $user_y$ (u_y).

$S_b(u_x, u_y)$ = the similarity based on user behavior data between $user_x$ (u_x) and $user_y$ (u_y).

β = the threshold value that ranges from 0 to 1.

α = the similarity's weight for $S_r(u_x, u_y)$.

δ = the similarity's weight for $S_b(u_x, u_y)$.

g = the product type/genre.

$P_{u_x g}$ = the probability score from $user_x$ (u_x) to product type/genre g.

$P_{u_y g}$ = the probability score from $user_y$ (u_y) to

product type/genre g.

G_{u_x} = the set of product types rated by $user_x$ (u_x).

G_{u_y} = the set of product types rated by $user_y$ (u_y).

$\bar{P}_{u_x}$ = the average probability scores of all product types rated by $user_x$ (u_x)

$\bar{P}_{u_y}$ = the average probability scores of all product types rated by $user_y$ (u_y).

y = a dependent variable.

x = an independent variable that is represented by the user profile data that consists of age (x_1), gender (x_2), occupation (x_3), and location (x_4).

a = a constant.

b= the regression coefficient.

TLF^{Power}= the power time loss function

φ = tuning parameter.

$T_{u_{x,p}}$= timestamp of $user_x$ (u_x) on product p.

$T_{u_{y,p}}$= timestamp of $user_y$ (u_y) on product p.

$\hat{r}_{u_x p}$ = the predicted rating score of $user_x$ (u_x) to product p.

NNu_x = the set of nearest neighbors to $user_x$ (u_x).

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Conceptualization, TW and DP; methodology, TW, DP, and HWH; software, TW; validation, TW, DP, and HWH; formal analysis, TW and DP; investigation, TW and DP; resources, TW; data curation, TW; writing—original draft preparation, TW, DP, and HWH; writing—review and editing, TW, DP, and HWH; visualization, TW; supervision, DP and HWH; project administration, TW; funding acquisition, TW.

Acknowledgments

This research was supported by directorate general of higher education (Dikti), ministry of education, culture, research and technology, research grant: Penelitian dasar with contract number 140/E5/PG.02.00.PL/2023.

References

[1] P. Phorasim and L. Yu, "Movies recommendation system using collaborative filtering and k-means", *Int. J. Adv. Comput. Res.*, Vol. 7, No. 29, pp. 52–59, 2017, doi: 10.19101/IJACR.2017.729004.

[2] S. Setiowati, T. B. Adji, and I. Ardiyanto, "Context-based awareness in location recommendation system to enhance recommendation quality: A review", In: *Proc. of 2018 Int. Conf. Inf. Commun. Technol. ICOIACT 2018*, Vol. 2018-Janua, pp. 90–95, 2018, doi: 10.1109/ICOIACT.2018.8350671.

[3] J. Liu, Z. Yang, T. Li, D. Wu, and R. Wang, "SPR: Similarity pairwise ranking for personalized recommendation", *Knowledge-Based Syst.*, Vol. 239, 2022, doi: 10.1016/j.knosys.2021.107828.

[4] H. Khojamli and J. Razmara, "Survey of similarity functions on neighborhood-based collaborative filtering", *Expert Syst. Appl.*, Vol. 185, No. June, p. 115482, 2021, doi: 10.1016/j.eswa.2021.115482.

[5] K. V. Rodpysh, S. J. Mirabedini, and T. Banirostam, "Resolving cold start and sparse data challenge in recommender systems using multi-level singular value decomposition", *Comput. Electr. Eng.*, Vol. 94, No. June, p. 107361, 2021, doi: 10.1016/j.compeleceng.2021.107361.

[6] L. N. H. Nam, "Towards comprehensive approaches for the rating prediction phase in memory-based collaborative filtering recommender systems", *Inf. Sci. (Ny).*, Vol. 589, No. 227, pp. 878–910, 2022, doi: 10.1016/j.ins.2021.12.123.

[7] R. Abolghasemi, P. Engelstad, E. H. Viedma, and A. Yazidi, "A personality-aware group recommendation system based on pairwise preferences", *Inf. Sci. (Ny).*, Vol. 595, pp. 1–17, 2022, doi: 10.1016/j.ins.2022.02.033.

[8] F. Horasan, A. H. Yurttakal, and S. Gündüz, "A novel model based collaborative filtering recommender system via truncated ULV decomposition", *J. King Saud Univ. - Comput. Inf. Sci.*, Vol. 35, No. 8, 2023, doi: 10.1016/j.jksuci.2023.101724.

[9] Y. Gao, Z. W. Huang, Z. Y. Huang, L. Huang, Y. Kuang, and X. Yang, "Multi-scale broad collaborative filtering for personalized recommendation", *Knowledge-Based Syst.*, Vol. 278, p. 110853, 2023, doi: 10.1016/j.knosys.2023.110853.

[10] Y. Ali, O. Khalid, I. A. Khan, S. S. Hussain, F. Rehman, S. Siraj, and R. Nawaz, "A Hybrid group-based movie recommendation framework with overlapping memberships", *PLoS One*, Vol. 17, No. 3, 2023.

[11] R. Duan, C. Jiang, and H. K. Jain, "Combining review-based collaborative filtering and matrix factorization: A solution to rating's sparsity

problem", *Decis. Support Syst.*, No. December 2021, p. 113748, 2022, doi: 10.1016/j.dss.2022.113748.

[12] N. Bhalse and R. Thakur, "Algorithm for movie recommendation system using collaborative filtering", *Mater. Today Proc.*, No. xxxx, pp. 1–6, 2021, doi: 10.1016/j.matpr.2021.01.235.

[13] Y. Afoudi, M. Lazaar, and M. A. Achhab, "Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network", *Simul. Model. Pract. Theory*, Vol. 113, No. June, p. 102375, 2021, doi: 10.1016/j.simpat.2021.102375.

[14] Z. Z. Darban and M. H. Valipour, "GHRS: Graph-based hybrid recommendation system with application to movie recommendation", *Expert Syst. Appl.*, Vol. 200, No. November 2020, 2022, doi: 10.1016/j.eswa.2022.116850.

[15] A. Fareed, S. Hassan, S. B. Belhaouari, and Z. Halim, "A collaborative filtering recommendation framework utilizing social networks", *Mach. Learn. with Appl.*, Vol. 14, No. September, p. 100495, 2023, doi: 10.1016/j.mlwa.2023.100495.

[16] J. Liu, Y. Chen, Q. Liu, and B. Tekinerdogan, "A similarity-enhanced hybrid group recommendation approach in cloud manufacturing systems", *Comput. Ind. Eng.*, Vol. 178, No. October 2022, p. 109128, 2023, doi: 10.1016/j.cie.2023.109128.

[17] A. A. Amer, H. I. Abdalla, and L. Nguyen, "Enhancing recommendation systems performance using highly-effective similarity measures[Formula presented]", *Knowledge-Based Syst.*, Vol. 217, p. 106842, 2021, doi: 10.1016/j.knosys.2021.106842.

[18] T. Neammanee, S. Maneeroj, and A. Takasu, "Considering similarity and the rating conversion of neighbors on neural collaborative filtering", *PLoS One*, Vol. 17, No. 5, 2022.

[19] N. Sun, Q. Luo, L. Ran, and P. Jia, "Similarity matrix enhanced collaborative filtering for e-government recommendation", *Data Knowl. Eng.*, Vol. 145, No. October 2021, p. 102179, 2023, doi: 10.1016/j.datak.2023.102179.

[20] H. Liu, Z. Hu, A. Mian, H. Tian, and X. Zhu, "A new user similarity model to improve the accuracy of collaborative filtering", *Knowledge-Based Syst.*, Vol. 56, pp. 156–166, 2014, doi: 10.1016/j.knosys.2013.11.006.

[21] L. Sheugh and S. H. Alizadeh, "A note on pearson correlation coefficient as a metric of similarity in recommender system", In: *Proc. of 2015 AI Robot. IRANOPEN 2015 - 5th Conf. Artif. Intell. Robot.*, 2015, doi: 10.1109/RIOS.2015.7270736.

[22] B. K. Patra, R. Launonen, V. Ollikainen, and S. Nandi, "A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data", *Knowledge-Based Syst.*, Vol. 82, pp. 163–177, 2015, doi: 10.1016/j.knosys.2015.03.001.

[23] K. G. Saranya, G. S. Sadasivam, and M. Chandralekha, "Performance comparison of different similarity measures for collaborative filtering technique", *Indian J. Sci. Technol.*, Vol. 9, No. 29, 2016, doi: 10.17485/ijst/2016/v9i29/91060.

[24] N. Polatidis and C. K. Georgiadis, "A multi-level collaborative filtering method that improves recommendations", *Expert Syst. Appl.*, Vol. 48, pp. 100–110, 2016, doi: 10.1016/j.eswa.2015.11.023.

[25] F. Zhang, W. Zhou, L. Sun, X. Lin, H. Liu, and Z. He, "Improvement of Pearson similarity coefficient based on item frequency", *Int. Conf. Wavelet Anal. Pattern Recognit.*, Vol. 1, pp. 248–253, 2017, doi: 10.1109/ICWAPR.2017.8076697.

[26] S. B. Sun, Z. H. Zhang, X. L. Dong, H. R. Zhang, T. J. Li, L. Zhang, and F. Min, "Integrating triangle and jaccard similarities for recommendation", *PLoS One*, Vol. 12, No. 8, 2017, doi: 10.1371/journal.pone.0183570.

[27] J. Feng, X. Fengs, N. Zhang, and J. Peng, "An improved collaborative filtering method based on similarity", *PLoS One*, Vol. 13, No. 9, pp. 1–18, 2018, doi: 10.1371/journal.pone.0204003.

[28] C. Wu, J. Wu, C. Luo, O. Wu, C. Liu, Y. Wu, and F. Yang, "Recommendation algorithm based on user score probability and project type", *Eurasip J. Wirel. Commun. Netw.*, Vol. 2019, No. 1, 2019, doi: 10.1186/s13638-019-1385-5.

[29] T. Widiyaningtyas, I. Hidayah, and T. B. Adji, "User profile correlation-based similarity (UPCSim) algorithm in movie recommendation system", *J. Big Data*, Vol. 8, No. 1, 2021, doi: 10.1186/s40537-021-00425-x.

[30] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context", *ACM Trans. Interact. Intell. Syst.*, Vol. 5, No. 4, 2015, doi: 10.1145/2827872.

[31] G. Jain, T. Mahara, and S. C. Sharma, "Performance Evaluation of Time-based Recommendation System in Collaborative Filtering Technique", *Procedia Comput. Sci.*, Vol. 218, No. 2022, pp. 1834–1844, 2023, doi: 10.1016/j.procs.2023.01.161.

[32] M. Jalili, S. Ahmadian, M. Izadi, P. Moradi, and M. Salehi, "Evaluating Collaborative Filtering Recommender Algorithms: A Survey", *IEEE Access*, Vol. 6, pp. 74003–74024, 2018, doi: 10.1109/ACCESS.2018.2883742.

[33] X. Li and D. Li, "An Improved Collaborative Filtering Recommendation Algorithm and Recommendation Strategy", *Mob. Inf. Syst.*, Vol. 2019, 2019, doi: 10.1155/2019/3560968.