Received: 23 February 2024 | Revised: 4 April 2024 | Accepted: 15 April 2024 | Published online: 20 April 2024

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2024, Vol. 3(3) 285–294
DOI: 10.47852/bonviewJCCE42022694

BON VIEW PUBLISHING

Time-Aware Similarity Integration for User-Based Collaborative Filtering

Soojung Lee[1,*]

[1]*Department of Computer Education, Gyeongin National University of Education, Republic of Korea*

Abstract: A time-aware collaborative filtering-based recommender system provides item recommendations for the current user by prioritizing recent items preferred by similar neighbors over past preferred items. The similarity measure critically affects the performance of the system, and this study focuses on measuring the similarity between users that changes over time. After dividing the users' rating time into intervals and computing similarity for each time interval, the final similarity is generated as a weighted sum by assigning lower weights to past similarity values and higher weights to more recent similarity values. Additionally, to ensure continuity of similarity measurement, consecutive time intervals are set to overlap. As a result of experiments applying the proposed method to the existing similarity measures, significant performance improvement was achieved in terms of some of the major performance metrics. In particular, the degree of coverage improvement was the highest, and the performance improvement effect was higher when the overlap size between time intervals was large rather than when it was small.

Keywords: time-aware collaborative filtering, similarity measure, collaborative filtering, recommender system

1. Introduction

E-commerce users have difficulty making choices due to the overwhelming number of products and services. Recommender systems provide personalized content by analyzing user preferences to aid the decision-making process. Currently, recommendation systems are commonly used in the e-commerce field, and the popular fields are generally music, news, shopping, movies, and books [1].

Several methods have been devised to implement a recommender system. Representative examples include content-based filtering and collaborative filtering (CF) [2]. Content-based filtering recommends items suited to the user's preferences by analyzing his/her past preference items. For example, document content, news information, weblogs, item characteristics, etc., are analyzed. Therefore, there is a problem in that recommendation accuracy cannot be guaranteed in the initial stage when there is no user preference information. In addition, there is a potential problem of continuously providing a recommendation list limited to the content of the user's past preference.

CF can solve these shortcomings of content-based filtering because it is based on the similarity between users' past preferences [2, 3]. For example, assuming that two users A and B have similar preferences for items and that user A has preferred item k, this item is most likely recommended to user B. Therefore, similarity measurement plays a critical role in the performance of CF systems, and various measures have been developed in the literature [4, 5].

CF has received much attention from academic and commercial sites due to the above advantages and is largely classified into memory-based technology and model-based technology. For model-based technology, in particular, various types of detailed algorithms have been developed, whose examples include Bayesian networks, clustering, and Markov decision process models [2]. Meanwhile, various information has been additionally utilized to improve recommendation accuracy for users. The context-aware recommender system (CARS) is a technology that utilizes the user's various situations to provide personalized services. Examples of context information include time, weather, and location information [6, 7].

In this study, we propose a new similarity measure for time context-aware memory-based CF. To measure similarity between users, changes in similarity over time are observed, and the final similarity is calculated as a weighted sum of similarity values where a small weight is assigned to past similarity and a higher weight to more recent similarity. This technique is differentiated from existing time-aware CF methods that reflect the time of the user's ratings for items [8–10]. In addition, the proposed method considers continuity of similarity measurement by overlapping the time intervals for calculating similarity, thereby reducing the possibility of co-rated items not being reflected due to separated time intervals. The performance of the proposed method was investigated from three perspectives. First, does the proposed time-aware CF technique show better performance in terms of various performance metrics than the previous one? Second, how does the overlap size of time intervals affect performance? Third,

*Corresponding author: Soojung Lee, Department of Computer Education, Gyeongin National University of Education, Republic of Korea. Email: sjlee@ginue.ac.kr

is the application of the proposed time-aware technique effective regardless of the type of existing similarity measures? As a result of experiments using open datasets widely used in academia, applying the proposed method to existing similarity measures showed excellent performance improvement in most performance metrics, and in particular, the degree of coverage improvement was significant. This improvement was better achieved when the time overlap size was larger than small.

The contributions of this paper are listed below.

1) It suggests a new strategy of reflecting users' rating time for time-aware CF, which is differentiated from previous methods in that it focuses on measuring the similarity between users that changes over time.
2) The similarity between users is generated as a weighted sum by assigning lower weights to past similarity values and higher weights to more recent similarity values.
3) It attempts to ensure continuity of similarity measurement, by overlapping consecutive time intervals for calculating similarity.
4) Performance experiments using a public dataset of two different data densities are conducted to demonstrate significant performance improvement of the proposed method in some major metrics, especially in coverage. Further, the effect of time overlap size on performance is investigated through the experiments.

The structure of the remaining paper is as follows. Section 2 describes existing related research results. Section 3 introduces the proposed method, followed by the results of performance experiments in Section 4. Section 5 concludes the paper.

2. Literature Review

Traditional CF-based recommender systems maintain user preference information in the form of the user-item matrix. This matrix maintains rating information for the user's items. However, one of the main problems of these systems is that they do not reflect the changing preferences of users over time. A CARS is suggested as a solution to this problem since it considers various contextual factors influencing the change in user preferences [11–15].

CARS-related research studies in the literature provided information on context modeling technology, context inclusion method and recommendation technology, evaluation metrics, and various applications of CARS [12]. Adomavicius et al. [16] proposed three types of approaches for integrating context information: pre-filtering, post-filtering, and context-specific modeling.

Recently, researchers have actively attempted to adopt deep learning technology to recommender systems [17]. He et al. [18] proposed NeuMF which is a deep learning technology using a matrix decomposition method. Their work modeled the interaction between users and items with two neural networks, but did not consider context information. da Costa and Dolog [19] used CNN to learn potential nonlinear characteristics by modeling the relationship among items, users, and time. Unger et al. [20] proposed a context modeling approach that extended NeuMF, utilizing nonlinear interactions among users, items, and context to integrate high-dimensional context information.

Time-aware CF, one of the context-aware CF techniques, determines the rating value prediction and recommendation list by additionally considering the time at which the user rating was assigned. Usually, an exponentially decreasing weight is imposed on past ratings [9, 10]. Alabduljabbar et al. [21] presented a literature review on time-aware recommender systems and the techniques to capture user preference changes over time. They also presented a quantitative assessment of the related works to provide information on publication time and types and the used dataset.

Yang and Li imposed [8] weights using a logistic function on past rating values to calculate the similarity of the Pearson correlation (COR). Rafeh and Bahrehmand [22] improved the traditional similarity measure and obtained the final similarity by integrating different types of information. This information includes the number of co-rated items between users, rating time and values, and the rating order differences. A new similarity measure was proposed through the normalization of this information. A study that considered information in addition to the rating value was also presented by Xu et al. [23]. They included not only the time context but also the user's confidence [23]. Additionally, Wangwatcharakul and Wongthanavasu [24] presented a recommendation method by applying a time function to user review information and analyzing it.

In a recent study, Sun and Dong [25] introduced clustering and a time influence index matrix considering changes in user preferences. Li and Han [26] developed a recommendation model that applied time-aware techniques to a hybrid method combining content-based and CF-based methods. In addition, Lu et al. [27] proposed a time-aware neural CF model that integrates the multidimensional features of papers to reflect the gradual decline of the influence of past papers in recommending academic papers.

Zou et al. [28] proposed a time-aware QoS prediction approach. Their method addresses the high sparsity of user service QoS calls and mines continuous temporal changes effectively. Hu et al. [29] utilized dynamic graphs and convolutional networks to model temporal interactions for time-aware QoS prediction. In a study by Wan et al. [30], a time-aware user and item-based CF algorithm and a time-aware latent factor model are integrated to propose a recommendation algorithm, where each model compensates for the shortcomings of the other.

Unlike the existing research and purposes described above, the proposed technique of this study applies to all types of system environments that manage items and user rating values only. In addition, it is not model-based using neural networks, clustering, or any others, but is memory-based and specifically a user-based time-aware CF technique that calculates similarity between users by reflecting changes in similarity over time.

3. Research Methodology

3.1. Motivation

The motivation for developing the proposed method is as follows. Let items x and y be unrated by the current user u. For rating prediction of these items, the system consults similar users of user u. Suppose that the system generates the prediction rating value of 3.5 for item x and that of 3 for item y, by accumulating ratings of similar users who have rated the corresponding item. If we do not take into account the rating time of these items, item x rather than item y should be recommended to user u. However, if the rating time of item y by similar users is more recent than that of item x, a typical time-aware CF system may produce a higher

prediction rating for item y, thus recommending item y instead of item x. This decision of course depends on the weighting function value over time.

However, assuming that similar users consulted for item x have much higher similarity with user u than those for item y, it may be more reasonable to recommend item x, even though the rating time for item y is more recent. This is because the predicted rating for item x can be considered more reliable. This decision is more valid when changes in user preferences depending on the characteristics of the item are insignificant over time.

Hence, we take an approach that does not consider the rating time of items by similar users but considers changes in similarity between users over time. That is, if two users' item ratings are more similar recently than in the past, their similarity is set to be larger than that in the opposite case. This method has a very different perspective in that existing studies considered the user rating time of each item in their time-aware CF [8–10]. In our method, a lower weight is given to past similarity values, and a higher weight is given to more recent similarity values to generate the final similarity.

To implement the proposed idea, similarity between users must be calculated at regular time intervals. If the time interval is set to be very long, the system will not be able to properly reflect changes in similarity values over time. As an extreme example, if the time interval is set to be equal to the whole rating time period of all users, the rating time context is not reflected at all. Conversely, if the time interval is set to be very short, the system load for calculating similarity for each cycle increases. Also in this case, the reliability of the generated similarity value would decrease, since the number of items co-rated by two users within a short period of time would be relatively small. This is because most existing similarity measures calculate similarity from the rating values of items co-rated by two users. Therefore, setting an appropriate time interval has a very important impact on the performance of the system.

3.2. Formulation

Figure 1 illustrates the time intervals for the proposed method. Assuming that the users in the system continue the rating activity from time 0 to time t, the entire time is divided into equal time intervals T_i, $I = 1, \ldots, n$. To generate similarity between any two users, the first step is to compute similarity based on the ratings given by the two users during each T_i time interval. Next, the weighted sum of all similarities generated for each time interval is calculated. At this time, the final similarity is determined by assigning a higher weight to the similarity value of the more recent time interval. The value of n, which determines the total number of time intervals, affects the system performance and is a parameter to be determined through experiment.

In implementing the proposed method, an additional factor to be considered is the overlap size of the time interval, as shown in Figure 1. In the figure, T_1 and T_2 are not independent of each other but overlap. Likewise, there is an overlap of the same size between other consecutive time intervals. The reason for allowing overlap in the proposed method is to take the continuity of similarity measurement into account. For example, let $t_{u,i}(t_{v,i})$ be the time when user $u(v)$ gives a rating to item i, as marked in the figure. In the figure, the time when user u rates item i is earlier than that by user v. If T_1 and T_2 are independent, the two users' ratings of item i belong to different time intervals and are not reflected at all in the final similarity. However, when T_1 and T_2 overlap as shown in the figure, both $t_{u,i}$ and $t_{v,i}$ do not belong to T_1, but belong to T_2, so the two rating values within T_2 are reflected in the similarity. Therefore, by providing overlap between consecutive time intervals, the number of ratings of co-rated items for calculating similarity would increase, thereby reducing the problem of data scarcity. The overlap size is one of the factors that determine the performance of the system, which is examined through experiments in the next section.

The similarity formula proposed in this study for time-aware CF is presented below. $sim_{u,v}(T_i)$ is the similarity between users u and v within the time interval Ti, and w(Ti) is the weight for the interval. The similarity between two users u and v is defined as follows.

$$sim_{u,v} = \frac{\sum_{i=1}^{n} \mathrm{w}(T_i) \cdot sim_{u,v}(T_i)}{\sum_{i=1}^{n} w(T_i)} \tag{1}$$

Any kind of similarity measure developed in the literature can be used to compute $sim_{u,v}(T_i)$. The weighting function for the time interval is usually an exponential function that gives exponentially higher weight to the more recent rating [8]. Therefore, in this study, we also follow this approach and define the weighting function as follows. λ is a parameter to be determined through experiment.

$$\mathrm{w}(T_i) = e^{\lambda i}, \quad \lambda > 0 \tag{2}$$

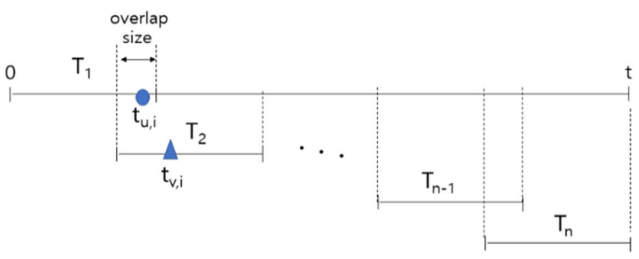
After calculating the similarity between any two users, the CF system predicts rating values for items that have not been rated by the current user and recommends the items in order of the highest predicted rating value. Rating prediction is based on the similarity and rating values of users similar to the current user, which is according to the principle of the CF system's assumption that similar users will give similar rating values for other items as well in the future. The set of similar users is usually called nearest neighbors. Since the rating prediction depends on which users would belong to the set of nearest neighbors, the similarity measure for determining the nearest neighbors is significant, thereby affecting the performance of the system.

To predict a rating for the current user's unrated item, the user-based prediction method is most widely known. This is defined as a weighted sum of the rating values for the item given by the nearest neighbors of the user [2]. Formally, the predicted rating of item x for user u is computed as follows:

$$r'_{u,x} = \bar{r}_u + \frac{\sum_v sim_{u,v}(r_{v,x} - \bar{r}_v)}{\sum_v |sim_{u,v}|} \tag{3}$$

where v is the nearest neighbor of u, $r_{v,x}$ the rating of item x given by v, and $\bar{r}_u$ the mean rating of u.

Figure 1
Illustration of time intervals for the proposed method



4. Performance Evaluation

4.1. Design of experiments

4.1.1. Dataset

For performance experiments, a dataset widely used for related research is selected. The dataset conditions for the experiment are that it must include user IDs, item IDs, rating values, and rating time. MovieLens dataset is operated non-commercially by the GroupLens Research Institute at the University of Minnesota and is most actively used in related academic fields [2]. This dataset is selected because it satisfies the experimental conditions of this study.

The PC environment used for this experiment contains an i7 Intel Core processor, 8 GB RAM, and a 64-bit Windows 11 operating system. We implemented our strategy and the baseline methods with C programs of a total size of 106 KB. The PC facilities had a very large performance burden to accommodate all the data provided in the dataset, so a portion of the original data was randomly extracted to conduct the experiment. The original dataset contains over one million rating records consisting of a total of 6,040 users and 3,952 items, where each user has at least 20 ratings. A record contains user-id, item-id, rating, and the time of rating. For the experiment, two datasets with different densities were prepared. Dataset *A* is composed of only those users with 150 ratings or more, resulting in a denser data environment than the original dataset. Dataset *B*, like the original dataset, includes users with more than 20 ratings. Both sets were configured identically in terms of the number of users and items, with a total number of 2,096 users and a total number of 3,952 items. The characteristics of the two datasets are summarized in Table 1.

Users of the MovieLens dataset participated in their rating activities for approximately 34 months starting in 2000, and the set was released in February 2003. For experimentation of the proposed method, the entire period of time must be divided into appropriate time intervals, similarity is to be calculated for each time interval, and the overlap size of consecutive time intervals must be determined. Regarding the length of the time interval, it affects not only the reliability of the generated similarity but also the prompt reflection of similarity changes over time, as discussed in Section 3.1. This situation occurs when the length is set to extreme values. Hence, determining a proper length is an important factor for system performance. In this experiment, the time interval was set to a medium value of 7 months, considering the whole rating time period of about 34 months, and performance was investigated for various overlap sizes.

Table 1
Dataset description

Feature	Dataset	Value
Number of users	A, B	2,096
Number of items	A, B	3,952
Rating range	A, B	1∼5 (integer)
Total number of ratings	A	744,072
	B	358,158
Least number of ratings/user	A	150
	B	20
Sparsity level	A	0.9102
	B	0.9568

4.1.2. Experimented methods

The proposed strategy is applicable to all similarity measures developed in the literature. In this experiment, we selected two measures with different characteristics, the Jaccard coefficient (Jaccard) and COR, which are among the most commonly used and cited similarity measures. Jaccard computes a relative number of items co-rated by two users, while COR takes the rating values of the co-rated items into account. Therefore, the effectiveness of the proposed method can be examined using the similarity types of the two unique features. In addition, Ding and Li's method (DL) [31] which is an early research result of the time-aware CF and became the basis for many subsequent studies, was included in our experiments.

Table 2 lists the meaning and values of the parameters used in the experiment. The exponential function presented in Equation (2) was used as a weighting function that assigns weight to each time interval. As λ increases, the influence of the similarity corresponding to the most recent time interval on the final similarity increases exponentially. θ is a parameter that determines whether the current user is satisfied with the recommendation list produced by the system. The items in the list are considered relevant if their real ratings of the user exceed θ. Considering that the rating range of the dataset is from one to five, θ was set to four, assuming that the user would prefer an item when its rating value is four or more.

Table 2
Description of the parameters for experiments

Parameter	Description	Value
TI	Time interval size	7 months
sz	Overlap size between time intervals	0, 1, 4, 5 months
λ	Parameter for the weight function	0.1
θ	Relevancy threshold	4.0

The similarity measures for CF used in our experiments are summarized below.

1) **Jaccard**: indicates the Jaccard coefficient. This calculates the ratio of the number of items rated in common out of all items rated by two users [32]. Many studies have been published showing that Jaccard improves performance by compensating for the shortcomings of existing similarity measures in sparse data environments [33, 34].
2) **P_JAC:** refers to the proposed method using the Jaccard coefficient as the similarity measure. The experiments of this method are conducted with several overlap sizes, 0, 1, 4, and 5.
3) **COR:** Pearson's correlation with no time-aware strategy employed.
4) **P_COR:** refers to the proposed method using Pearson's correlation as the similarity measure. The experiments of this method are conducted with several overlap sizes, 0, 1, 4, and 5.
5) **Ding and Li's method (DL)** [31]. This is a time-aware CF with the logistic time weighting function applied to past rating values.

4.1.3. Performance metrics

As for performance evaluation, two representative types are widely used in related research, i.e., prediction accuracy and recommendation accuracy [2]. To estimate prediction accuracy, several metrics have been developed, such as RMSE (root mean square error), MAE (mean absolute error), and relative root mean square error [35, 36]. Among them, we select the first two metrics as they are commonly used for evaluating recommender systems.

MAE refers to the average absolute value of the difference between the actual rating and the predicted rating estimated by the system. To estimate the predicted rating of an item unrated by the current user, we employed the formula of weighted sum. This formula accumulates the ratings given by users similar to the current user who have rated the item, where a higher weight is assigned to the user of higher similarity. Specifically, MAE is defined as $\text{MAE} = \frac{1}{n}\sum_u \sum_x |r_{u,x} - r'_{u,x}|$, where $r_{u,x}$ is the real rating given by user u for item x and $r'_{u,x}$ is its predicted rating. RMSE is a metric used for the Netflix prize and amplifies the error between the real rating and the prediction. It is defined as $\text{RMSE} = \sqrt{\frac{1}{n}\sum_u \sum_x (r_{u,x} - r'_{u,x})^2}$.

Precision is a representative metric for recommendation accuracy that measures the level of satisfaction the user has with the recommendation list produced by the system. If the user's actual rating value for the item in the list exceeds θ, it is considered relevant. Precision is formally defined as

$$\text{precision} = \frac{1}{|U|}\sum_{u\in U}\frac{|\{x \in L_u \mid r_{u,x} \geq \theta\}|}{|L_u|},$$

where L_u is the recommendation list of items for user u and U is the set of users.

Another important performance metric measures the proportion of items the system can generate their predictions for the current user. Note that prediction of an item can be made only when any one of the similar users has rated the item. Coverage is a measure for this purpose and is defined as the ratio of items for which the system can provide predictions for users out of all items unrated by them [37]. It is defined as follows:

$$\text{coverage} = \frac{1}{|U|}\sum_{u\in U}\frac{|\{x \in I \mid r_{u,x} = - \ \ \& \ \ V_{u,x} \neq \theta\}|}{|\{x \in I \mid r_{u,x} = -\}|},$$

where I is the set of items provided in the system, $V_{u,x}$ is the set of neighbors of u who have rated item x, and $r_{u,x} = -$ indicates that user u has not rated item x.

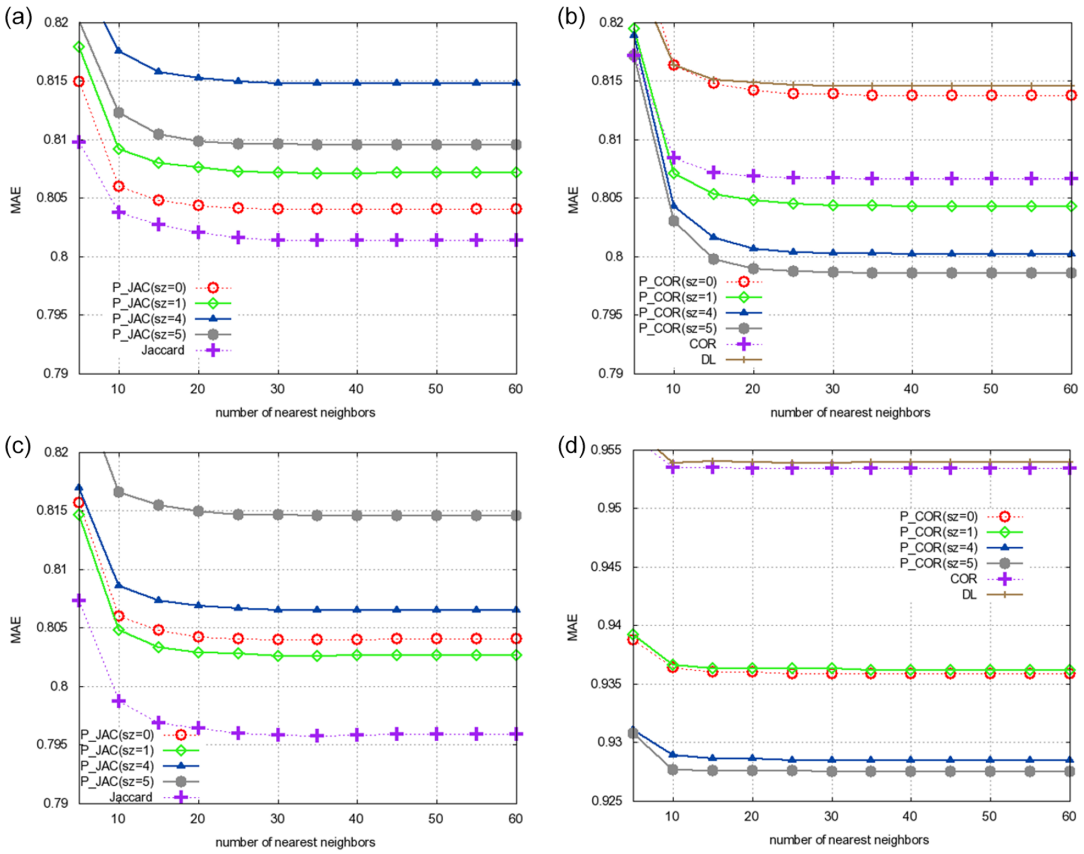
4.2. Results of experiments

4.2.1. Prediction accuracy

Figure 2 shows the performance results of MAE with varying numbers of nearest neighbors for different overlap sizes (sz). Figure 2(a) and (c) are the results of using the Jaccard coefficient

Figure 2

Mean absolute error for varying number of nearest neighbors using dataset *A* ((a) and (b)) and dataset *B* ((c) and (d))



(Jaccard) as a similarity measure, and Figure 2(b) and (d) are the results of using COR. All results show gradually stable errors as the number of nearest neighbors increases. However, the performance results for Jaccard and COR show different aspects. The biggest difference is that when Jaccard is used as a similarity measure, the proposed method leads to degradation in MAE performance regardless of the overlap size in both datasets, whereas when COR is used as a similarity measure, performance is improved with the overlap size greater than zero. Jaccard is known to be very advantageous in sparse data environment [33, 34], so its performance was superior to other methods in case of dataset B rather than A.

In Figure 2, P_JAC performance degrades with the large overlap size (sz = 4 or 5) than with the small size (sz = 0 or 1). On the contrary, P_COR performance improved rapidly as the overlap size increased, but when comparing the results of sizes 4 and 5, the degree of improvement appears to be limited. Meanwhile, DL showed the poorest performance even though it is equipped with a time-aware strategy. Their results are depicted along with COR and not with Jaccard, because they use COR for similarity measure.

Figure 3 presents the RMSE results for varying number of nearest neighbors. It can be said that the overall performance ranking of the methods is the same as in the MAE results. However, in the case of the denser dataset *A*, Jaccard and DL perform relatively worse than the others, in comparison with the MAE results. In other words, for this dataset Jaccard performs best as shown in Figure 2(a), but it performs comparably to the proposed method with the overlap size of zero, in terms of RMSE. Likewise, DL demonstrates a lot worse performance relative to the proposed with the size of zero in terms of RMSE, as seen in Figure 3(b).

4.2.2. Precision

Figure 4 shows the precision results. It is observed that the precision decreases as the number of recommended items increases. The precision results show that, unlike MAE, using Jaccard as similarity in the proposed method (Figure 4(a) and (c)) yields improved precision compared to the original Jaccard method (Jaccard) which employs no time-aware strategy. This is the case regardless of data sparseness. Additionally, this phenomenon remains the same even with varying overlap sizes.

On the other hand, the proposed method when using COR as similarity (Figure 4(b) and (d)) shows low precision with a slight difference from DL or the original COR. However, this varies a little depending on the number of recommended items. In particular, in the case of dataset *B*, it seems to be difficult to distinguish a method with clear superior performance as shown in Figure 4(d), which is believed due to the influence of sparse data. When the overlap size exceeds 0, the proposed method applied with COR shows comparable precision results, especially in the dense dataset *A*. Meanwhile, the experimental results using Jaccard show that the proposed method is advantageous in terms of precision, which is especially the case when the overlap size is large, regardless of the data set.

Overall, COR is found to yield much better precision performance than Jaccard. The results of Jaccard decrease more rapidly as the number of recommended items increases, but the decrease rate of COR results turns out to be relatively gentle. The

Figure 3

Root mean square error for varying number of nearest neighbors using dataset *A* ((a) and (b)) and dataset *B* ((c) and (d))

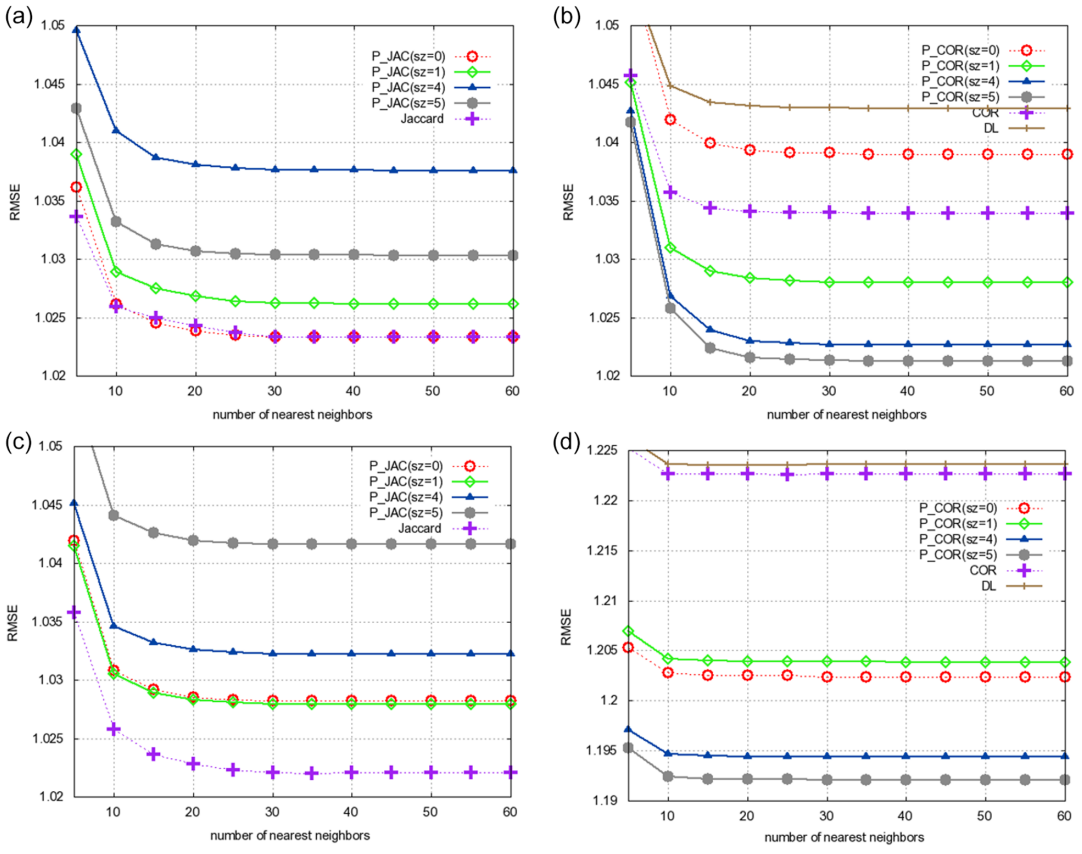


Figure 4

Precision for varying number of recommended items using dataset *A* ((a) and (b)) and dataset *B* ((c) and (d))

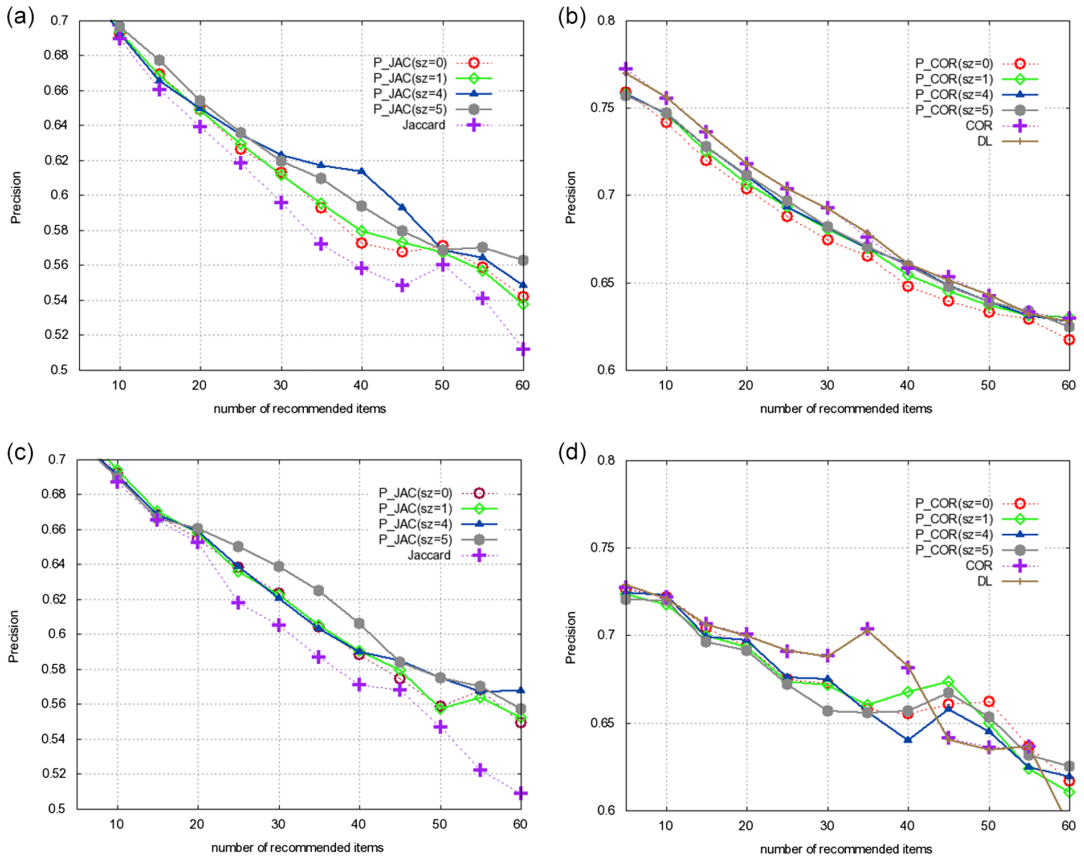
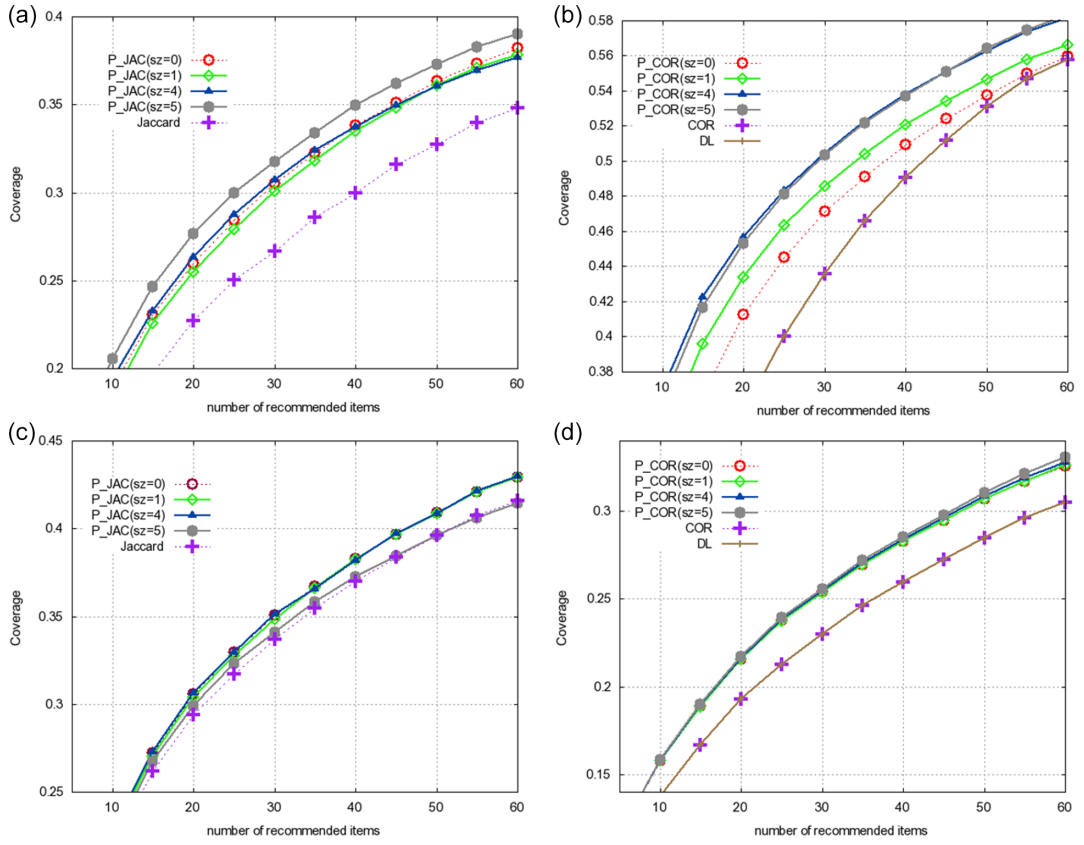


Figure 5

Coverage for varying number of recommended items using dataset *A* ((a) and (b)) and dataset *B* ((c) and (d))



reason might be that Jaccard calculates the similarity between two users from the number of co-rated items, making it less accurate than COR.

4.2.3. Coverage

Figure 5 shows the coverage results. Applying the proposed time-aware strategy to Jaccard or COR on both datasets is found to be effective, and performance was further improved in dense data environments of dataset *A*.

In particular, it is seen that in a dense data environment (Figure 5(a) and (b)), the performance difference becomes clearer depending on the overlap size. In other words, the larger the overlap size, the better the coverage, and this finding is almost similar to those of MAE. On the other hand, in a sparse data environment (Figure 5(c) and (d)), the impact of the overlap size turns out to be minimal. However, using either COR or Jaccard, the performance improvement by the proposed method in terms of coverage is significantly achieved, and in particular, the effect is greatest with a large overlap size.

To sum up, the effectiveness of the proposed method was found different depending on the type of similarity measure introduced in CF, the performance metric, and the overlap size of time intervals. It was shown that generally the same effects are obtained regardless of the density of the dataset. In the case of using COR for our strategy, setting the overlap size to be large was very effective in terms of MAE and coverage, but its effect on precision was minimal. On the other hand, when using the Jaccard coefficient for our method, MAE performance deteriorated regardless of the overlap size, but the improvement effect on precision and coverage was excellent when the overlap size was large. In particular, for both Jaccard and COR used with our strategy, notable performance improvement was achieved when the overlap size was large.

5. Conclusion and Discussion

This study proposed a new similarity calculation method for a time-aware collaborative filtering system. The proposed method reflects changes in similarity between users over time. It divides the users' rating time into intervals and calculates similarity for each time interval, then assigns a small weight to past similarity and a higher weight to more recent similarity to produce the final similarity. To ensure continuity of similarity measurement, consecutive time intervals were overlapped and performance changes with varying overlap sizes were examined.

To investigate the effectiveness of the proposed strategy, two widely known representative similarity measures with different characteristics, the Pearson correlation and the Jaccard coefficient, were selected among the existing similarity measures. Pearson correlation has been found to have excellent performance among several measures, and the Jaccard coefficient is known to be useful in a sparse data environment. For experiments in a fair data environment, two datasets with different sparsities were prepared. Regarding performance evaluation, we selected four metrics that measure different aspects of the system and are widely used for investigating the performance of collaborative filtering systems.

As a result of experiments of the proposed strategy, performance was improved in most performance metrics, but the results were different depending on the similarity measure employed. However, the effect on coverage performance was the most for both the Jaccard and Pearson correlation employed in our strategy. Additionally, it was more effective when the overlap size was large rather than when it was small.

The time-aware collaborative filtering strategy proposed in this study demonstrated its performance through the Jaccard coefficient and Pearson correlation similarity, but owing to the nature of the proposed strategy, any similarity measure can be used instead. The dataset used in our experiments is limited to one type with different data densities, so it is difficult to generalize the performance results presented in Section 4.

We plan to expand the experiments in the future to examine the effectiveness of the proposed method using different similarity measures in various data and experimental environments. We will focus on conducting experiments using other datasets having user rating time elements under various data density conditions. Performance was examined in this study using several metrics classically used in collaborative filtering research fields, but it would be interesting to investigate performance in terms of other metrics such as novelty, diversity, and rank-based ones. The experimental results presented in this study were obtained using a fixed size of time interval and four different overlap sizes. A lot of experimental work would be required to find the optimal values of time interval size and overlap size, which might be dependent on the data environment.

Lastly, this study assigns a higher weight to the latest similarity between users. However, weights for similarity may depend on user type, user rating behavior, item characteristics, etc. These factors need to be investigated and considered to determine more accurate weights for similarity over time. Moreover, this study limits its discussion to user-based collaborative filtering but can be easily extended to item-based or hybrid one. It is also worth studying how to apply the proposed strategy to a model-based collaborative filtering system rather than to a memory-based one.

Conflicts of Interest

The author declares that she has no conflicts of interest to this work.

Data Availability Statement

The MovieLens dataset that support the findings of this study are openly available at https://grouplens.org/datasets/movielens/1m/.

References

[1] Shao, B., Li, X., & Bian, G. (2021). A survey of research hotspots and frontier trends of recommendation systems from the perspective of knowledge graph. *Expert Systems with Applications*, *165*, 113764. https://doi.org/10.1016/j.eswa.2020.113764

[2] Li, Y., Liu, K., Satapathy, R., Wang, S., & Cambria, E. (2023). Recent developments in recommender systems: A survey. *arXiv Preprint: 2306.12680*. https://doi.org/10.48550/arXiv.2306.12680

[3] Gazdar, A., & Hidri, L. (2020). A new similarity measure for collaborative filtering based recommender systems. *Knowledge-Based Systems*, *188*, 105058. https://doi.org/10.1016/j.knosys.2019.105058

[4] Fkih, F. (2022). Similarity measures for collaborative filtering-based recommender systems: Review and experimental comparison. *Journal of King Saud University – Computer*

and Information Sciences, *34*(9), 7645–7669. https://doi.org/10.1016/j.jksuci.2021.09.014

[5] Khojamli, H., & Razmara, J. (2021). Survey of similarity functions on neighborhood-based collaborative filtering. *Expert Systems with Applications*, *185*, 115482. https://doi.org/10.1016/j.eswa.2021.115482

[6] Rostami, M., Farrahi, V., Ahmadian, S., Jalali, S. M. J., & Oussalah, M. (2023). A novel healthy and time-aware food recommender system using attributed community detection. *Expert Systems with Applications*, *221*, 119719. https://doi.org/10.1016/j.eswa.2023.119719

[7] Unger, M., & Tuzhilin, A. (2022). Hierarchical latent context representation for context-aware recommendations. *IEEE Transactions on Knowledge and Data Engineering*, *34*(7), 3322–3334. https://doi.org/10.1109/TKDE.2020.3022102

[8] Yang, H. Z., & Li, L. (2009). An enhanced collaborative filtering algorithm based on time weight. In *International Symposium on Information Engineering and Electronic Commerce*, 262–265. https://doi.org/10.1109/IEEC.2009.61

[9] Livne, A., Tov, E. S., Solomon, A., Elyasaf, A., Shapira, B., & Rokach, L. (2022). Evolving context-aware recommender systems with users in mind. *Expert Systems with Applications*, *189*, 116042. https://doi.org/10.1016/j.eswa.2021.116042

[10] Mohammadi, N., & Rasoolzadegan, A. (2022). A two-stage location-sensitive and user preference-aware recommendation system. *Expert Systems with Applications*, *191*, 116188. https://doi.org/10.1016/j.eswa.2021.116188

[11] Jeong, S. Y., & Kim, Y. K. (2022). Deep learning-based context-aware recommender system considering contextual features. *Applied Sciences*, *12*(1), 45. https://doi.org/10.3390/app12010045

[12] Kulkarni, S., & Rodd, S. F. (2020). Context aware recommendation systems: A review of the state of the art techniques. *Computer Science Review*, *37*, 100255. https://doi.org/10.1016/j.cosrev.2020.100255

[13] Sohafi-Bonab, J., Aghdam, M. H., & Majidzadeh, K. (2023). DCARS: Deep context-aware recommendation system based on session latent context. *Applied Soft Computing*, *143*, 110416. https://doi.org/10.1016/j.asoc.2023.110416

[14] Thomadsen, R., Rooderkerk, R. P., Amir, O., Arora, N., Bollinger, B., Hansen, K., . . . , & Wood, W. (2018). How context affects choice. *Customer Needs and Solutions*, *5*(1), 3–14. https://doi.org/10.1007/s40547-017-0084-9

[15] Villegas, N. M., Sánchez, C., Díaz-Cely, J., & Tamura, G. (2018). Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems*, *140*, 173–200. https://doi.org/10.1016/j.knosys.2017.11.003

[16] Adomavicius, G., Bauman, K., Tuzhilin, A., & Unger, M. (2022). Context-aware recommender systems: From foundations to recent developments. In F. Ricci, L. Rokach & B. Shapira (Eds.), *Recommender systems handbook* (pp. 211–250). Springer. https://doi.org/10.1007/978-1-0716-2197-4_6

[17] Chen, J., Wang, X., Zhao, S., Qian, F., & Zhang, Y. (2020). Deep attention user-based collaborative filtering for recommendation. *Neurocomputing*, *383*, 57–68. https://doi.org/10.1016/j.neucom.2019.09.050

[18] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, 173–182. https://doi.org/10.1145/3038912.3052569

[19] da Costa, F. S., & Dolog, P. (2019). Collective embedding for neural context-aware recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, 201–209. https://doi.org/10.1145/3298689.3347028

[20] Unger, M., Tuzhilin, A., & Livne, A. (2020). Context-aware recommendations based on deep learning frameworks. *ACM Transactions on Management Information Systems*, *11*(2), 8. https://doi.org/10.1145/3386243

[21] Alabduljabbar, R., Alshareef, M., & Alshareef, N. (2023). Time-aware recommender systems: A comprehensive survey and quantitative assessment of literature. *IEEE Access*, *11*, 45586–45604. https://doi.org/10.1109/ACCESS.2023.3274117

[22] Rafeh, R., & Bahrehmand, A. (2012). An adaptive approach to dealing with unstable behaviour of users in collaborative filtering systems. *Journal of Information Science*, *38*(3), 205–221. https://doi.org/10.1177/0165551512437517

[23] Xu, G., Tang, Z., Ma, C., Liu, Y., & Daneshmand, M. (2019). A collaborative filtering recommendation algorithm based on user confidence and time context. *Journal of Electrical and Computer Engineering*, *2019*, 7070487. https://doi.org/10.1155/2019/7070487

[24] Wangwatcharakul, C., & Wongthanavasu, S. (2021). A novel temporal recommender system based on multiple transitions in user preference drift and topic review evolution. *Expert Systems with Applications*, *185*, 115626. https://doi.org/10.1016/j.eswa.2021.115626

[25] Sun, B., & Dong, L. (2017). Dynamic model adaptive to user interest drift based on cluster and nearest neighbors. *IEEE Access*, *5*, 1682–1691. https://doi.org/10.1109/ACCESS.2017.2669243

[26] Li, H., & Han, D. (2021). A time-aware hybrid recommendation scheme combining content-based and collaborative filtering. *Frontiers of Computer Science*, *15*(4), 154613. https://doi.org/10.1007/s11704-020-0028-7

[27] Lu, Y., He, Y., Cai, Y., Peng, Z., & Tang, Y. (2021). Time-aware neural collaborative filtering with multi-dimensional features on academic paper recommendation. In *IEEE 24th International Conference on Computer Supported Cooperative Work in Design*, 1052–1057. https://doi.org/10.1109/CSCWD49262.2021.9437673

[28] Zou, G., Huang, Y., Hu, S., Gan, Y., Zhang, B., & Chen, Y. (2023). TRCF: Temporal reinforced collaborative filtering for time-aware QoS prediction. *IEEE Transactions on Services Computing*, 1–14. https://doi.org/10.1109/TSC.2023.3329110

[29] Hu, S., Zou, G., Zhang, B., Wu, S., Lin, S., Gan, Y., & Chen, Y. (2022). Temporal-aware QoS prediction via dynamic graph neural collaborative learning. In *Service-Oriented Computing: 20th International Conference*, 125–133. https://doi.org/10.1007/978-3-031-20984-0_8

[30] Wan, Y., Chen, Y., & Yan, C. (2021). An integrated time-aware collaborative filtering algorithm. In *Proceedings of 15th International Conference on Knowledge Management in Organizations*, 369–379. https://doi.org/10.1007/978-3-030-81635-3_30

[31] Ding, Y., & Li, X. (2005). Time weight collaborative filtering. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, 485–492. https://doi.org/10.1145/1099554.1099689

[32] Bag, S., Kumar, S. K., & Tiwari, M. K. (2019). An efficient recommendation generation using relevant Jaccard similarity.

Information Sciences, *483*, 53–64. https://doi.org/10.1016/j.ins.2019.01.023

[33] Kosub, S. (2019). A note on the triangle inequality for the Jaccard distance. *Pattern Recognition Letters*, *120*, 36–38. https://doi.org/10.1016/j.patrec.2018.12.007

[34] Park, S. H., & Kim, K. (2023). Collaborative filtering recommendation system based on improved Jaccard similarity. *Journal of Ambient Intelligence and Humanized Computing*, *14*(8), 11319–11336. https://doi.org/10.1007/s12652-023-04647-0

[35] Jin, B., & Xu, X. (2024). Price forecasting through neural networks for crude oil, heating oil, and natural gas. *Measurement: Energy*, *1*, 100001. https://doi.org/10.1016/j.meaene.2024.100001

[36] Jin, B., & Xu, X. (2024). Forecasting wholesale prices of yellow corn through the Gaussian process regression. *Neural Computing & Applications, 36*(15), 8693–8710. https://doi.org/10.1007/s00521-024-09531-2

[37] Al-Shamri, M. Y. H., & Al-Ashwal, N. H. (2014). Fuzzy-weighted similarity measures for memory-based collaborative recommender systems. *Journal of Intelligent Learning Systems and Applications*, *6*(1), 1–10. https://doi.org/10.4236/jilsa.2014.61001

How to Cite: Lee, S. (2024). Time-Aware Similarity Integration for User-Based Collaborative Filtering. *Journal of Computational and Cognitive Engineering*, *3*(3), 285–294. https://doi.org/10.47852/bonviewJCCE42022694