computers

MDPI

Article

Recommendation Algorithm Using Clustering-Based UPCSim (CB-UPCSim)

Triyanna Widiyaningtyas [1,2], Indriana Hidayah [1] and Teguh Bharata Adji [1,*]

[1] Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada, Yogyakarta 55281, Indonesia; triyanna2019@mail.ugm.ac.id (T.W.); indriana.h@ugm.ac.id (I.H.)
[2] Department of Electrical Engineering, Universitas Negeri Malang, Malang 65145, Indonesia
* Correspondence: adji@ugm.ac.id

Abstract: One of the well-known recommendation systems is memory-based collaborative filtering that utilizes similarity metrics. Recently, the similarity metrics have taken into account the user rating and user behavior scores. The user behavior score indicates the user preference in each product type (genre). The added user behavior score to the similarity metric results in more complex computation. To reduce the complex computation, we combined the clustering method and user behavior score-based similarity. The clustering method applies *k*-means clustering by determination of the number of clusters using the Silhouette Coefficient. Whereas the user behavior score-based similarity utilizes User Profile Correlation-based Similarity (UPCSim). The experimental results with the MovieLens 100k dataset showed a faster computation time of 4.16 s. In addition, the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) values decreased by 1.88% and 1.46% compared to the baseline algorithm.

Keywords: collaborative filtering; memory-based; similarity metrics; *k*-means clustering; Silhouette Coefficient

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

Academic Editor: George Angelos Papadopoulos

Received: 19 August 2021
Accepted: 26 September 2021
Published: 6 October 2021

1. Introduction

The increasing volume and complexity of online information make it difficult for users to obtain appropriate information. The recommendation system is the ultimate solution to deal with the information explosion [1,2]. This system is a valuable information filtering tool to assist users in finding a product or service from the many possibilities that exist.

Recommendation systems have developed rapidly, and various domains have used them, such as movies, music, news, books, restaurants, and other media. In addition, several researchers have developed recommendation systems with many existing approaches, including demographic filtering, content-based filtering, collaborative filtering, and hybrid filtering [3,4].

One of the most prevalent approaches to recommendation systems is collaborative filtering [5–8]. This approach is capable of generating recommendations based on the ratings provided by the users for several items. Collaborative filtering consists of two methods: model-based and memory-based. The first method uses a model built from the ratings to generate recommendations, while the second method utilizes similarity metrics to get the distance between two users/items [6,9].

In recent years, several researchers have proposed collaborative filtering using the similarity metrics approach to increase the accuracy of recommendations. Some of the proposed similarity metrics are Proximity-Significance-Singularity (PSS) [10], Bhattacharyya [11], multi-level collaborative filtering [12], item frequency-based similarity [13], Triangle Multiplying Jaccard (TMJ) [14], and three impact factors-based similarity [3]. However, these similarity metrics only consider the user rating score to calculate similarities between users. The user rating score is the value given directly by the user in assessing the selected or purchased product. This score ranges from 1 to 5, with a score of 1 indicating that the user

really dislikes the selected product and a score of 5 indicating that the user really likes the selected product.

Recently, the development of similarity metrics has considered the user rating score and user behavior score. The similarity metrics that have adopted user behavior scores in calculating similarity are User score Probability Collaborative Filtering (UPCF) [15] and User Profile Correlation-based Similarity (UPCSim) [16]. However, adding the user behavior score variable in the similarity calculation causes the computation to be more complex. Consequently, it consumes time with the increasing data.

Several studies [17–19] have utilized clustering methods by reducing large amounts of data to overcome the computational complexity of recommendation systems. The studies applied the partition-based clustering methods by determination of the number of clusters directly. The problem of these studies is determining the number of clusters without measuring the clustering quality to get the optimal number of clusters that affect the results of the recommendations.

Based on the above problems, our research proposes a recommendation system model that combines the clustering method and user behavior score-based similarity to reduce computational time, called Clustering-Based UPCSim (CB-UPCSim). The clustering method uses the partition-based method of *k*-means, measuring the clustering quality using the intrinsic model of the Silhouette Coefficient and grouping users based on user profile data (i.e., gender, age, job, and location). Meanwhile, the similarity metric uses UPCSim.

The remainder of this paper is organized as follows. Section 2 summarizes previous researches that consider similarity metrics and clustering in recommendation systems. Section 3 then presents the detail on the stages of the proposed method. Section 4 describes the experimental results and discussion. Finally, Section 5 concludes the results of our study and offers recommendations for further research.

2. Related Work

The memory-based method employs all rating data to generate a list of recommended products [11]. This method focuses on similarity metrics to count the similarity between products/users. The similarity metrics build upon the distance between two users or products. Pearson Correlation Coefficient (PCC) and Cosine Similarity (COS) are traditional similarity metrics that are frequently used in recommendation systems [20].

Several studies performed improvement of traditional similarity metrics to increase the performance of recommendation systems. Patra et al. [11] presented the Bhattacharyya coefficient in collaborative filtering (BCF) similarity. BCF utilized all rating scores assessed by pairs of users, combined with Jaccard similarity. Furthermore, Polatidis et al. [12] presented the improvement of PCC by using the number of co-rated products in some levels. This new similarity metric is then called multi-level collaborative filtering. Sun et al. [14] offered a new similarity metric called TMJ (Triangle Multiplying Jaccard) similarity. Triangle similarity considers the co-rated data, while Jaccard similarity provides information on ratings not assessed together. Finally, Feng et al. [3] developed a new similarity metric by considering three impact factors (S_a, S_b, and S_c). S_a describes the similarity between users, S_b states the tendency of users to give ratings, and S_c expresses the rating weight of each user. In general, these previous researchers only utilize the user rating score to count the similarity metrics.

Furthermore, Wu et al. [15] proposed a novel similarity metric by incorporating the user rating-based and behavior-based similarity score to improve recommendation accuracy. The user behavior score is the accumulated scores in assessing/viewing the product type (genre). In the movie recommender system, this score indicates how much users like a movie genre based on the movie title they watch. For example, User A watches the movie title "Aladdin" with the movie genre of animation, children, and comedy. Each genre will get a behavior score of "1". After that, User A watches the movie title "Get Shorty", which has the movie genre of action, comedy, and drama. Each genre will also get the behavior score by "1". Consequently, User A implicitly gives the user behavior

scores of animation = 1, children = 1, comedy = 2, action = 1, and drama = 1. The genre "Comedy" gets a score = 2 because the movie titles of "Alladin" and "Get Shorty" are included in the genre "Comedy." Therefore, the user behavior score ranges from 1 to *N*. If the user frequently accesses the product types, the value of *N* will be immense. Conversely, if the user rarely accesses the product types, the value of *N* will be smaller. Wu et al. applied the probability of the user behavior score to calculate the similarity metric. This similarity metric is called User score Probability Collaborative Filtering (UPCF). Their research results reduced the MAE and RMSE values by 1.51% and 0.94% compared to the traditional similarity metric, i.e., Cosine Similarity.

Recently, the research conducted by [16] presents a novel similarity metric known as User Profile Correlation-based Similarity (UPCSim). UPCSim improved UPCF, replacing the similarity weight in UPCF (threshold value) with the correlation coefficient between user profile factors and user rating/behavior score. The results showed that the MAE and RMSE values decreased by 1.64% and 1.4%.

Although UPCF and UPCSim improve recommendation performance, these similarity metrics still have shortcomings. Combining the two similarities makes the computation more complex. Therefore, the increasing data will consume time to produce recommendations.

Several researchers applied clustering methods to reduce the increasing data in their recommendation system. For example, Vellaichamy et al. [19] utilized clustering to overcome the scalability problem and improve recommendation quality. The clustering algorithm applied Fuzzy C-Means (FCM) with Bat optimization and determined the number of clusters to 16 groups. Bat algorithm serves to obtain the initial cluster position. Furthermore, PCC similarity calculates the similarity between users. The experimental results reduced the MAE value and increased the precision and recall values.

Meanwhile, Lestari et al. [17] applied *k*-means clustering to reduce the dataset by determining the number of clusters with 7. The cluster division utilized one of the user profiles factors, namely age. Each cluster then performs the recommendation process using a ranking-oriented collaborative filtering approach, known as WP-Rank. The results showed the Normalized Discounted Cumulative Gain (NDCG) increased by 0.022 and a longer running time of 0.026 s.

In the meantime, Tran et al. [18] implemented collaborative filtering based on clustering with an incentive/penalized user (IPU) model to overcome a large volume of data and improve the performance of the recommendation system. Their research combines spectral clustering and FCM algorithms by dividing the rating data into 10 clusters. After that, each product receives an incentive/penalty based on the user's tendency in the cluster. The experimental results showed a significant increase in the F1 score, precision, and recall.

The three previous studies [17–19] utilize the partition-based clustering methods, i.e., FCM and *k*-means. The FCM has $O(nk^2t)$ time complexity, whereas the *k*-means has $O(nkt)$. n, k, and t represent the number of data, the number of clusters, and the number of iterations, respectively [21]. These studies determined the number of clusters without measuring the clustering quality to obtain the optimal number of clusters. In addition, the study conducted by [17] only used the age factor in grouping users without considering other factors that affected the user preferences. Therefore, our study utilizes the intrinsic method of the Silhouette Coefficient to determine the optimal number of clusters and clustering the user and rating data based on all user profile factors (i.e., age, gender, job, and location). The clustering method in our study uses *k*-means clustering because the *k*-means method requires less computation time than FCM.

3. Research Method

This research proposes a recommendation algorithm that combines the clustering method and similarity based on user behavior scores (i.e., UPCSim), known as CB-UPCSim. The study consists of five stages: data collection, data preparation, clustering process, memory-based process, and evaluation, as illustrated in Figure 1. The following subsection presents the details of each stage.

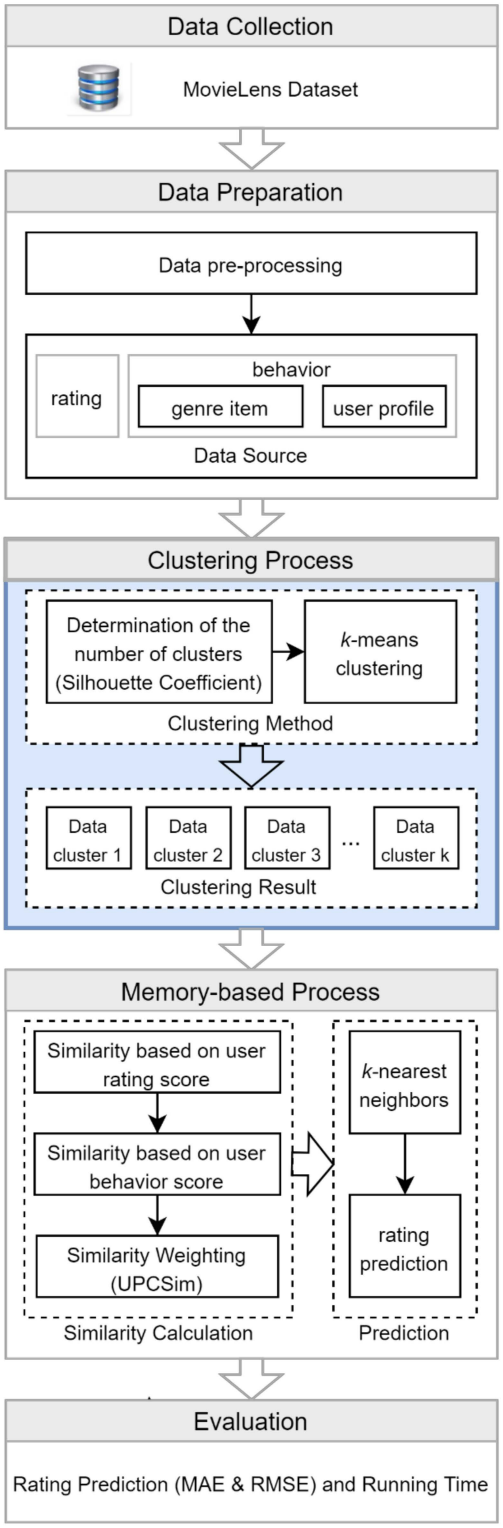

Figure 1. Research stages.

3.1. Data Collection

We utilized the MovieLens 100k dataset in this study. The dataset includes 100,000 ratings, 943 users, and 1682 movies collected by the "GroupLens Research Group of the University of Minnesota [22]. The rating scale is between 1 and 5. Score 1 indicates the user does not really like the movie, and score 5 indicates the user really likes the movie. The sparsity and density of this rating data are 93.7% and 6.3%. There are 19 genres in the

dataset, and each movie can contain several genres. Each user rates at least 20 movies and has information about the user profile (i.e., gender, age, job, and location).

3.2. Data Preparation

In this study, the data preparation stage is data pre-processing that prepares raw data before the following process (clustering and memory-based processes) to obtain clean data. One way to data pre-processing is to reduce irrelevant attributes. For example, there are timestamps, movie titles, release dates, video release dates, and IMDb URLs as irrelevant attributes in the MovieLens dataset.

3.3. Clustering Process

This stage consists of two steps. The first step is to determine the number of clusters to get the optimal number of clusters using the Silhouette Coefficient. The second step is to group the user data using the *k*-means algorithm, where *k* is the number of clusters with the maximum Silhouette Coefficient value. The details of each method are presented as follows.

3.3.1. Determination of the Number of Clusters

This process aims to measure the clustering quality to obtain the optimal number of clusters (*k*). The clustering quality measurement consists of two methods: extrinsic and intrinsic [21]. The extrinsic method compares a clustering result with the ideal clustering made by experts. If there is no ideal clustering from experts, we can use the intrinsic method, which evaluates the clustering quality by testing how far apart the clusters are and how dense the clusters are.

One of the metrics used in the intrinsic method is the Silhouette Coefficient. This method measures an object's similarity to its cluster (cohesion) compared to other clusters (separation). The following steps describe how to count the Silhouette Coefficient's value [21]:

1. Calculate the average distance from one document to another in a cluster using the formula defined in Equation (1).

$$a(i) = \frac{1}{|A| - 1} \sum_{j \epsilon A, j \neq i} d(i, j) \tag{1}$$

   j is another document in one cluster A, and $d(i, j)$ is the distance between document i and document j.
2. Calculate the average distance from the document i to all documents in other clusters, using the formula defined in Equation (2). Then, find the minimum average distance using Equation (3).

$$d(i, C) = \frac{1}{|C|} \sum_{j \epsilon C} d(i, j) \tag{2}$$

$$b(i) = min_{c \neq A}\, d(i, C) \tag{3}$$

3. Calculate the Silhouette Coefficient value using Equation (4).

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{4}$$

The value of $a(i)$ represents the density of the cluster containing object i. The smaller the value of $a(i)$, the denser the cluster. Meanwhile, the value of $b(i)$ indicates how far object i is apart from other clusters. The greater the value of $b(i)$, the further apart object i is from other clusters. If the value of $a(i)$ is minimal and the value of $b(i)$ is immense, then the Silhouette Coefficient of object i will be close to 1. It means that the cluster containing object i is very dense, and object i is far from other clusters. Conversely, if the value of $a(i)$ is immense and the value of $b(i)$ is minimal, then the Silhouette Coefficient of object i will

be close to −1. It means that the cluster containing object i is not congested, and object i is very close to other clusters.

The clustering results are said to be good if the Silhouette Coefficient value is positive. The Silhouette Coefficient value equal to 1 is the maximum value, which states that the number of clusters produced is perfect. This study uses the intrinsic Silhouette Coefficient method to obtain the optimal number of clusters (k).

We need to determine the optimal number of clusters because previous researches [17–19] determined the number of clusters directly without measuring the clustering quality to obtain the optimal number of clusters that will affect the similar user preferences in the same cluster.

3.3.2. Data Clustering

Clustering is a technique in data mining that groups the same objects in one cluster and different objects into different clusters [23–26]. Clustering is known as unsupervised learning because there is no class label. The clustering consists of four methods: hierarchical-based, grid-based, density-based, and partition-based.

The partition-based method works by dividing the data into several non-overlapping groups, and each data is in precisely one cluster [21]. This method is also known as the center-based method or the representative-based method [27]. Some of the algorithms included in the partition-based method are k-means, k-medoids, k-modes, and fuzzy c-means. k-means is one of the prominent algorithms in recommender systems [28].

The k-means algorithm aims to group data by maximizing data similarity in one cluster and minimizing the similarity inter clusters. The similarity measurement utilizes a distance function. The distance between data p in the C_i cluster and c_i centroid in the k-means algorithm uses the Euclidean distance. The shortest distance between the data and the centroid point indicates the maximum data similarity. The output of the k-means is highly dependent on the initial centroid that is randomly determined [26,29–31].

This study applies a partition-based method, k-means clustering, because this method is simple, easy to implement, and has fast computation time. Therefore, it is suitable for solving complex computational problems in the similarity metrics of recommendation systems. The clustering process works by grouping user and rating data based on all user profile factors.

3.4. Memory-Based Process

The memory-based method consisted of two processes: similarity calculation and rating prediction. In this study, the similarity calculation applies UPCSim [16], which refers to Equation (5).

$$S(x,y) = \alpha\, S_r(x,y) + \beta\, S_b(x,y) \tag{5}$$

$S(x,y)$ represents the final similarity between users x and y. $S_r(x,y)$ denotes the user rating score-based similarity between users x and y, whose formula refers to Equation (6). $S_b(x,y)$ states the user behavior score-based similarity between users x and y, whose formula refers to the Equation (7). Finally, α and β are correlation coefficients between user profile attributes and user rating/behavior scores, calculated using multiple linear regression [16].

$$S_r(x,y) = \frac{\sum_{p \in P_x \cap P_y} \left(r_{xp} - \overline{r}_x\right)\left(r_{yp} - \overline{r}_y\right)}{\sqrt{\sum_{p \in P_x \cap P_y} \left(r_{xp} - \overline{r}_x\right)^2} \cdot \sqrt{\sum_{p \in P_x \cap P_y} \left(r_{yp} - \overline{r}_y\right)^2}} \tag{6}$$

P_x and P_y express the set of products rated by user x and user y, respectively. Next, r_{xp} and r_{yp} state the rating values on product p by user x and user y, respectively. Furthermore,

$\overline{r}_x$ and $\overline{r}_y$ describe the rating averages for users x and y. Finally, p is one of the co-rated products by users x and y.

$$S_b(x,y) = \frac{\sum_{g \in G_x \cap G_y} \left(P_{xg} - \overline{P}_x\right)\left(P_{yg} - \overline{P}_y\right)}{\sqrt{\sum_{g \in G_x \cap G_y} \left(P_{xg} - \overline{P}_x\right)^2} \cdot \sqrt{\sum_{g \in G_x \cap G_y} \left(P_{yg} - \overline{P}_y\right)^2}} \quad (7)$$

G_x and G_y denote the set of product types assessed by user x and user y, respectively. Next, P_{xg} and P_{yg} express the probability of product type g given by users x and y. Furthermore, $\overline{P}_x$ and $\overline{P}_y$ indicate the average probability of product type from users x and y. Finally, g is a co-rated product type of users x and y.

The illustration of the similarity calculation between users can be explained as follows. The initial step is generating a rating matrix. For example, matrix R shows the user rating score given by five users on seven products. The blank value of matrix R indicates the sparseness of the matrix.

$$R = \begin{bmatrix} 5 & 3 & & 4 & 2 & & \\ 4 & & 2 & 4 & & 4 & \\ 4 & 4 & & & & 5 & 3 \\ & & 2 & 3 & 1 & & 5 \\ 1 & & 1 & 4 & & & 1 \end{bmatrix}$$

After generating the rating matrix, the next step calculates the similarity based on the user rating score (S_r) by referring to Equation (6). Matrix S_r shows the results of similarity S_r.

$$S_r = \begin{bmatrix} 1 & 0.9939 & 0.9701 & 0.9899 & 0.7954 \\ 0.9939 & 1 & 0.9939 & 0.9923 & 0.8642 \\ 0.9701 & 0.9939 & 1 & 1 & 0.9899 \\ 0.9899 & 0.9923 & 1 & 1 & 0.7265 \\ 0.7954 & 0.8642 & 0.9899 & 0.7265 & 1 \end{bmatrix}$$

After calculating the similarity S_r, the next step calculates the user behavior score-based similarity (S_b). Equation (7) explains this similarity S_b using the probability matrix of user behavior scores and the similarity S_b formula. The user behavior score is the total score given by the user in accessing the product type. Table 1 shows the product type data of the seven products rated by the previous five users. In this case, each product can be part of several products types. For example, product p_1 (movie title "Alladin") includes the product's types of animation, children, and comedy.

Table 1. Data product.

Product	Product Type							
	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8
p_1	0	1	1	1	0	0	0	0
p_2	1	0	0	1	0	1	0	0
p_3	0	0	0	0	0	1	1	0
p_4	0	0	0	0	0	1	0	0
p_5	0	0	0	0	1	0	0	1
p_6	0	0	0	0	1	0	0	1
p_7	0	0	0	1	0	0	0	0

p_1 = Aladdin, p_2 = get shorty, p_3 = the day, p_4 = white balloons, p_5 = seven, p_6 = usual suspects, p_7 = clerks, G_1 = action, G_2 = animation, G_3 = children, G_4 = comedy, G_5 = crime, G_6 = drama, G_7 = Sci-fi, G_8 = thriller.

Based on matrix R and Table 1, the illustration of user behavior scores can be explained as follows. If user 1 accesses product p_1 (the movie title "Aladdin"), then user 1 also accesses the product types (animation, children, and comedy), each of which will get a user behavior score of 1. Furthermore, user 1 accesses products p_2, p_4, and p_5. As a result, user 1 will

access the product type of action = 1, animation = 1, children = 1, comedy = 2, crime = 1, drama = 2, and thriller = 1. These values are called the user behavior scores. In the same way, we calculate the user behavior scores for user 2, user 3, user 4, and user 5. Matrix B shows the final results of calculating the user behavior scores of five users on eight product types.

$$B = \begin{bmatrix} 1 & 1 & 1 & 2 & 1 & 2 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 3 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 2 & 1 & 1 \\ 0 & 1 & 1 & 2 & 0 & 2 & 1 & 0 \end{bmatrix}$$

The next step is to calculate the probability score of user behavior by dividing the user behavior score by the number of users who access the product type. For example, based on matrix B, user 1 and user 3 accessed the product type of G_1. Thus, the probability of user behavior score for user 1 and user 3 on product type of G_1 is 0.5. Matrix P shows the probability score of user behavior from the five users of eight product types.

$$P = \begin{bmatrix} 0.5 & 0.25 & 0.25 & 0.4 & 0.25 & 0.4 & 0 & 0.25 \\ 0 & 0.25 & 0.25 & 0.2 & 0.25 & 0.4 & 0.33 & 0.25 \\ 0.5 & 0.25 & 0.25 & 0.6 & 0.25 & 0.2 & 0 & 0.25 \\ 0 & 0 & 0 & 0.2 & 0.25 & 0.4 & 0.33 & 0.25 \\ 0 & 0.25 & 0.25 & 0.4 & 0 & 0.4 & 0.33 & 0 \end{bmatrix}$$

Based on matrix P and Equation (7), we can calculate the similarity based on user behavior score (S_b) as shown in matrix S_b.

$$S_b = \begin{bmatrix} 1 & 0.4029 & 0.6675 & 0.3363 & 0.9997 \\ 0.4029 & 1 & 0.5416 & 0.9893 & 0.3272 \\ 0.6675 & 0.5416 & 1 & 0.6651 & 0.4688 \\ 0.3363 & 0.9893 & 0.6651 & 1 & 0.1453 \\ 0.9997 & 0.3272 & 0.4688 & 0.1453 & 1 \end{bmatrix}$$

Finally, the final similarity is calculated by combining the similarity S_r and similarity S_b with their weighting. Matrix S shows the final similarity between users, which can be used as a similarity metric model in the rating prediction. Note that we assume the weights of two similarities by 0.3 and 0.4 in this example.

$$S = \begin{bmatrix} 1 & 0.8402 & 0.8914 & 0.8200 & 0.8485 \\ 0.8402 & 1 & 0.8763 & 0.9915 & 0.7246 \\ 0.8914 & 0.8763 & 1 & 0.9129 & 0.8544 \\ 0.8200 & 0.9915 & 0.9129 & 1 & 0.5754 \\ 0.8485 & 0.7246 & 0.8544 & 0.5754 & 1 \end{bmatrix}$$

After calculating similarity, the following process is the rating prediction. This process aims to predict the rating score for unrated products by active users. Before making a rating prediction, it is necessary to determine the number of nearest neighbors (k). In this study, the k value ranges from 10 to 100, incremented by 10 [3,12,14–16].

The formula for calculating the predicted rating for the unrated products is expressed in Equation (8) [3,11].

$$\hat{r}_{xp} = \bar{r}_x + \frac{\sum_{y \epsilon NNx} S(x,y) \cdot (r_{yp} - \bar{r}_y)}{\sum_{y \epsilon NNx} |S(x,y)|} \quad (8)$$

$\hat{r}_{xp}$ is the predicted rating score from user x to product p. $y \epsilon NNx$ represents the set of users who have the nearest similarity to the user x. $S(x,y)$ denotes the final similarity between users x and y. $\bar{r}_x$ and $\bar{r}_y$ are the rating score average of users x and y, respectively. Finally, r_{yp} is the given rating score by user y to product p.

3.5. Evaluation

The final stage in this study is evaluation. This stage evaluates the recommendation system's performance that combines the *k*-means clustering method and user behavior-based similarity. The recommendation system's performance was measured by predictive metrics and running time. In this study, the predictive metrics utilize Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) [32,33].

The MAE calculates the average absolute deviation between the actual and the predicted rating scores. At the same time, RMSE computes the deviation degree between the actual and the predicted rating scores. A lower MAE and RSME represent good recommendation quality [19,34].

The formulas of MAE and RMSE refer to Equations (9) and (10).

$$MAE = \frac{1}{N} \sum_{x \epsilon U, p \epsilon P} \left| \hat{r}_{xp} - r_{xp} \right| \tag{9}$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{x \epsilon U, p \epsilon P} \left(\hat{r}_{xp} - r_{xp} \right)^2} \tag{10}$$

N is the total of predicted products. The r_{xp} and $\hat{r}_{xp}$ denote the actual and predicted ratings of the user x to product p, respectively.

4. Experiment Result and Discussion

This section provides the experiment results, starting from determination of the number of clusters, the data distribution after clustering, comparison of MAE and RMSE, and comparison of the running time. Finally, the discussion explains the findings of this study.

Experiments in the MovieLens 100k dataset used the computer specifications of Intel® Core™ i7-4510U CPU @ 2.000 GHz (4CPUs), ~2.6 GHz, and RAM of 16 GB. The UPCSim and CB-UPCSim algorithms were programmed using Python, running under Microsoft Windows 7.

4.1. Result of Silhouette Coefficient

The experiment for determining the optimal number of clusters begins with selecting the number of clusters to be evaluated, ranging from 2 to 19. The minimum number of clusters is 2, based on the smallest possible clusters. Meanwhile, the maximum number of clusters is 19, referring to the optimal value after the largest clusters used in previous studies [17–19]. At k = 18, the Silhouette Coefficient yields the second highest optimal value, and at k = 19, the Silhouette Coefficient value decreases again. We do not continue to the next k because the higher k needs more computation time.

Figure 2 shows the results of the clustering evaluation using the Silhouette Coefficient method. Note that the Silhouette Coefficient value is an average value taken from 5 experiments. Based on Figure 2, the number of clusters (k) equal to 3 gets the maximum value of the Silhouette Coefficient, showing the optimal number of clusters. Therefore, the number of clusters k equal to 3 will be used as the basis for the clustering process.

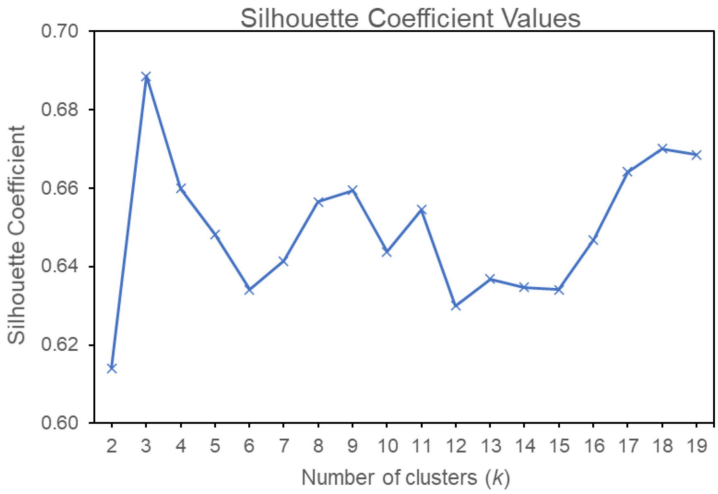

Figure 2. Result of Silhouette Coefficient.

4.2. Result of k-Means Clustering

The *k*-means clustering process with a value of $k = 3$ works to group 943 users and rating data on the MovieLens 100k dataset into 3 clusters based on the similarity of user profiles. Figure 3 shows the clusters formed, i.e., cluster 0, cluster 1, and cluster 2.

```
clusterNum = 3
k_means = KMeans(init = "k-means++", n_clusters = clusterNum, n_init = 12)
k_means.fit(X)
labels = k_means.labels_
print(labels)
```

```
[2 2 1 0 1 2 2 1 1 2 1 1 1 0 2 1 1 0 1 2 1 0 0 2 0 1 1 0 2 0 1 2 1 0 0 2 0
 0 1 1 2 1 1 0 0 0 1 1 2 0 1 0 0 0 1 0 2 0 1 1 1 2 2 0 0 2 0 1 0 0 2 2 0 0
 1 1 1 0 1 0 1 1 0 0 1 0 2 1 0 2 1 2 1 2 1 2 2 2 0 2 1 1 0 0 2 0 0 0 0 2 2
 0 2 2 1 2 1 2 0 1 2 1 1 0 1 1 2 1 1 0 1 2 2 2 0 2 2 0 1 1 0 0 1 1 0 2 1 2
 1 1 0 0 0 0 1 1 0 1 0 0 0 1 2 2 0 0 0 2 0 1 2 1 1 0 1 1 1 0 1 0 1 1 1 2 2
 0 1 1 2 2 2 2 0 1 2 0 2 0 1 2 0 0 1 1 1 0 2 1 2 1 1 0 0 1 0 1 1 1 0 2 1 1
 0 0 0 2 0 1 1 1 1 2 2 2 1 0 0 0 2 1 1 1 0 2 0 1 1 0 2 2 2 1 1 0 1 0 2 2 0
 2 2 2 0 2 2 2 2 1 0 0 1 0 1 0 2 2 0 1 2 1 1 1 0 2 0 1 1 1 2 2 0 2 0 2 0 1
 2 1 0 0 0 2 1 2 2 2 0 2 0 2 2 1 0 1 1 2 1 1 1 1 0 0 1 1 1 1 1 1 1 1 2 0 0
 1 1 0 1 0 0 2 0 2 1 2 2 2 2 0 0 2 0 0 1 0 0 2 2 1 0 2 0 2 2 1 1 0 0 2 2 1
 2 0 0 2 0 1 1 1 2 0 2 1 2 0 1 0 0 0 2 2 2 0 2 2 0 2 1 0 2 2 2 2 1 0 1 2 1
 0 2 2 0 1 2 0 2 2 0 1 0 1 0 2 2 0 0 1 2 0 1 2 2 0 1 0 0 1 1 0 1 0 0 2 1 0
 2 0 0 1 0 1 0 1 1 2 2 1 1 0 1 0 2 1 2 2 1 2 2 1 2 1 2 2 2 2 2 0 1 1 0 2 0
 0 0 1 2 2 2 1 0 0 0 2 0 0 1 0 0 0 2 2 1 1 1 2 1 1 1 1 1 2 1 1 1 1 0 1 0 2
 0 1 1 0 1 1 1 2 1 0 0 2 2 2 1 1 2 1 1 1 2 2 2 0 2 1 1 1 1 2 0 2 0 0 0 0 1
 1 1 0 1 1 0 1 1 2 0 1 1 1 2 1 1 1 0 0 0 2 2 0 0 1 2 2 1 1 2 2 1 2 2 2 2 2
 0 0 1 2 2 0 0 1 2 1 0 1 1 0 1 1 0 1 2 2 1 2 1 0 1 0 0 2 0 1 0 1 1 1 1 1 2 0
 0 0 0 0 0 1 0 2 1 1 0 0 2 0 0 0 0 0 2 0 2 1 1 0 2 0 1 2 2 0 2 2 1 1 2 0 0
 1 1 1 1 2 2 1 0 1 1 2 0 0 2 2 0 1 0 0 1 1 0 0 0 0 1 2 1 0 2 2 1 2 2 0 2 0
 2 0 0 1 2 0 2 1 0 1 0 2 0 2 0 0 1 1 1 2 0 2 2 2 2 0 1 0 2 0 0 2 2 2 2 2 1
 0 1 2 0 0 1 2 2 2 1 2 1 2 0 2 2 0 0 0 1 2 2 1 0 1 1 0 1 1 1 1 1 1 0 2 2 0 1
 1 0 2 0 1 2 2 1 1 2 1 0 2 1 1 2 0 1 1 0 0 1 0 2 1 2 0 1 1 2 2 1 2 2 2 1 1
 1 1 0 1 0 0 2 0 0 1 1 2 2 2 2 2 0 1 2 2 0 1 1 0 1 2 0 0 2 0 2 2 0 0 1 1 2 1
 2 0 0 1 2 0 1 1 0 1 1 0 0 1 0 2 0 1 0 0 2 1 0 0 1 2 2 0 2 0 0 0 0 2 0 1 1
 2 2 1 0 2 2 1 2 1 0 0 1 0 2 1 0 1 0 2 0 0 1 0 1 2 1 0 0 1 0 1 2 2 1 0 1 0
 1 0 0 0 1 1 1 0 1 0 1 2 0 1 1 2 2 2]
```

Figure 3. Result of *k*-means clustering for 943 users in MovieLens 100k.

Based on Figure 3, the distribution of user data after clustering is in cluster 0 with 319 users (33.83%), cluster 1 with 336 users (35.63%), and cluster 2 with 288 users (30.54%). Meanwhile, the distribution of rating data is in cluster 0 with 32,906 ratings (32.91%), cluster 1 with 34,370 ratings (34.37%), and cluster 2 with 32,724 ratings (32.72%). The sparsity and density of each cluster are 93.87% and 6.13% (in cluster 0), 93.92% and 6.08% (in cluster 1), and 93.24% and 6.76% (in cluster 2). Table 2 shows the details of the statistical data before and after the clustering process.

Table 2. Dataset statistics before and after the clustering process.

Data	Before Clustering	After Clustering		
		Cluster 0	Cluster 1	Cluster 2
#users	943	319	336	288
#ratings	100,000	32,906	34,370	32,724
% users	100%	33.83%	35.63%	30.54%
% ratings	100%	32.91%	34.37%	32.72%
sparsity of ratings	93.70%	93.87%	93.92%	93.24%
density of ratings	6.30%	6.13%	6.08%	6.76%

4.3. Result of MAE and RMSE

This subsection aims to compare the results of the recommendation system performance using a combination of memory-based and clustering methods with the memory-based method (without clustering) in the previous study. The results of the recommendation system performance are measured based on the MAE and RMSE values.

The experiment was performed by dividing the dataset of each cluster into several parts using the *k*-fold cross-validation method. In machine learning, the value of *k* is generally 5 or 10. Both of these values have been empirically proven to produce estimates of test error rates that are neither too high bias nor very high variance [35]. We select $k = 5$ from these values to divide the dataset because $k = 5$ consumes less time than $k = 10$. In addition, many previous studies [1,3,9,10,12,16] in recommender systems split the dataset into 80%:20% as the training data and testing data.

The value of *k* in this study is five. Hence, there are five training data (train_1, train_2, train_3, train_4, and train_5) and five testing data (test_1, test_2, test_3, test_4, and test_5). Therefore, there are five iterations performed in each cluster. The first iteration uses the train_1 and test_1 datasets, the second iteration uses the train_2 and test_2 datasets, repeated until the fifth iteration. The number of nearest neighbors ranges from 10 to 100. Table 3 compares the average MAE values before and after the clustering process in the MovieLens 100k dataset.

Table 3. Comparison of the average MAE values before and after the clustering process.

N	MAE			
	Before Clustering	After Clustering		
	UPCSim	Cluster 0	Cluster 1	Cluster 2
10	0.7669	0.7450	0.7469	0.7439
20	0.7483	0.7321	0.7329	0.7320
30	0.7410	0.7287	0.7303	0.7261
40	0.7387	0.7194	0.7220	0.7180
50	0.7369	0.7187	0.7194	0.7167
60	0.7364	0.7162	0.7188	0.7159
70	0.7359	0.7152	0.7167	0.7142
80	0.7355	0.7148	0.7161	0.7138
90	0.7347	0.7145	0.7156	0.7135
100	0.7337	0.7139	0.7152	0.7131
Average	0.7408	0.7219	0.7234	0.7207

Based on Table 3, the average of MAE values decreases as the number of nearest neighbors (*N*) grows. The average of MAE values after the clustering process reduces compared to before the clustering process. The decrease in the average of MAE values in cluster 0 is 1.89%, cluster 1 is 1.74%, and cluster 2 is 2.01%. It shows that the prediction accuracy after the clustering process experienced an average increase of 1.88% compared to before.

Figure 4 shows a graphic illustration of the average MAE values before the clustering process (using the UPCSim algorithm) and after the clustering process (using the CB-UPCSim algorithm in each cluster).

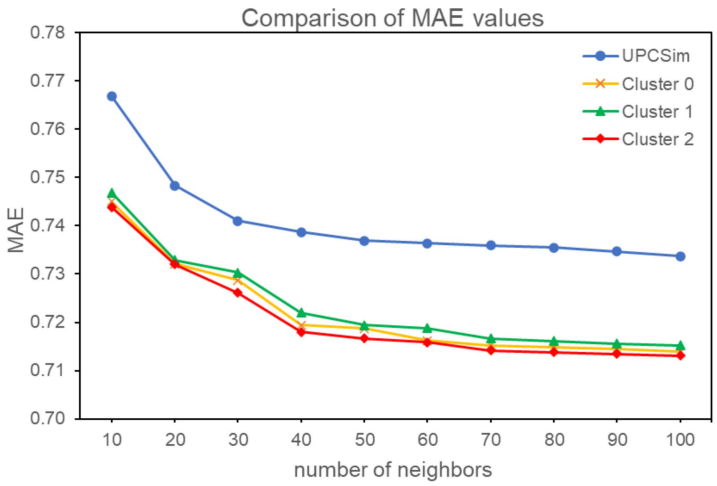

Figure 4. Comparison of the average MAE values in the UPCSim algorithm and the three clusters with the Movielens 100k dataset.

Figure 4 illustrates that the average of MAE values generated in the UPCSim algorithm and the combination of UPCSim and clustering (cluster 0, cluster 1, and cluster 2) decreases very sharply at the beginning of the curve. Meanwhile, at the end of the curve, the average of MAE values tends to be stable. In the same number of nearest neighbors, the average MAE values in each cluster shows a smaller value than the UPCSim algorithm. It shows that the clustering process results in the performance of the recommendation algorithm increases, especially in rating prediction.

Table 4 compares the average RMSE values before and after the clustering process using the MovieLens 100k dataset. The average RMSE value in each cluster is always smaller than the RMSE generated before the clustering process. In other words, there is a decrease in the RMSE value after the clustering process compared to before the clustering process. The reduction in the average of RMSE values in cluster 0 is 1.45%, cluster 1 is 1.27%, and cluster 2 is 1.65%. Thus, the recommendation performance generated after the clustering process obtained an average increase of 1.46%.

Table 4. Comparison of the average RMSE values before and after the clustering process.

N	RMSE			
	Before Clustering	After Clustering		
	UPCSim	Cluster 0	Cluster 1	Cluster 2
10	0.9793	0.9548	0.9557	0.9532
20	0.9541	0.9432	0.9451	0.9399
30	0.9453	0.9354	0.9367	0.9340
40	0.9427	0.9338	0.9348	0.9274
50	0.9393	0.9261	0.9287	0.9235
60	0.9389	0.9240	0.9250	0.9226
70	0.9383	0.9229	0.9243	0.9211
80	0.9381	0.9222	0.9241	0.9208
90	0.9364	0.9208	0.9236	0.9207
100	0.9359	0.9203	0.9233	0.9199
Average	0.9448	0.9304	0.9321	0.9283

Figure 5 shows the graphic illustration of the average RMSE value before the clustering process (using the UPCSim algorithm) and after the clustering process (in each cluster).

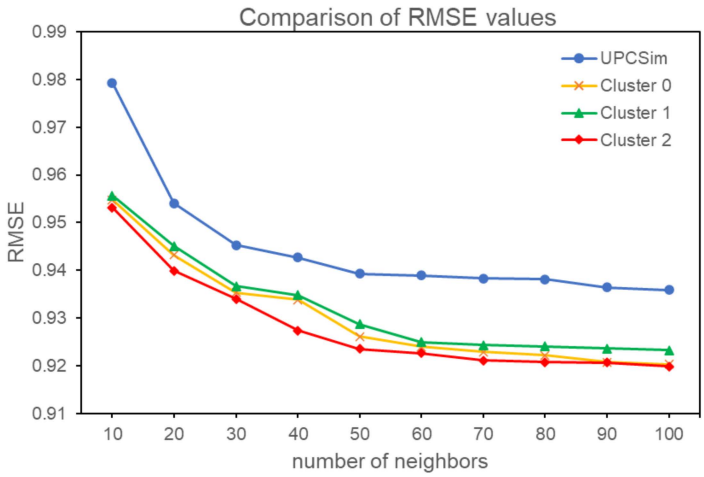

Figure 5. Comparison of the average RMSE values in the UPCSim algorithm and the three clusters with the Movielens 100k dataset.

Figure 5 shows that the increase in the number of nearest neighbors affects the resulting RMSE values. The four scenarios exhibit a decline in the value of RMSE initially, then stabilize as the number of nearest neighbors grows. The RMSE value generated in each cluster shows a smaller value than the RMSE value before the clustering process. It shows that the clustering process affects the resulting recommendation performance.

4.4. Result of Running Time

In addition to measuring the MAE and RMSE values as recommendation metrics, this experiment also computed the resulting execution time to see the effect of the clustering process on the running time of algorithms. Table 5 presents the average running time before and after the clustering process. Please note that the running time calculated in this study is a combination of training and testing times.

Table 5. Comparison of running time before and after the clustering process.

N	Running Time (Seconds)			
	Before Clustering	After Clustering		
	UPCSim	Cluster 0	Cluster 1	Cluster 2
10	4.78	0.70	0.75	0.67
20	4.82	0.81	0.85	0.75
30	5.02	0.91	0.89	0.81
40	5.27	0.89	0.95	0.86
50	5.68	0.96	0.98	0.89
60	5.99	0.98	1.00	0.87
70	6.02	0.99	1.01	0.90
80	6.10	1.01	1.03	0.92
90	6.15	1.03	1.02	0.94
100	6.28	1.07	1.06	0.94
Average	5.61	0.94	0.95	0.86

The running time after the clustering process in each cluster is faster than before the clustering process. The running time in cluster 0 shows 4.14 s faster, cluster 1 shows 4.12 s faster, and cluster 2 shows 4.22 s faster. Overall, the average running time is 0.91 s (decreased by 4.16 s or 5.5 times faster than before the clustering process). It shows that the performance of the execution time after the clustering process is better than before the clustering process, in the sense that the clustering process helps speed up the execution time to generate recommendations. It occurs because the amount of data in each cluster executed is less than before the clustering process.

We also measured the computation times for determining the optimal number of clusters using the Silhouette Coefficient and *k*-means clustering. Both methods need computation times of 1.03 s and 30.32 milliseconds, respectively.

4.5. Discussion

In this study, we propose a recommendation algorithm that combines memory-based and clustering methods. The memory-based method considers user rating scores and user behavior scores to accommodate user preferences. Meanwhile, the clustering method is *k*-means clustering by determining the number of clusters based on the Silhouette Coefficient to obtain the optimal number of clusters.

Choosing the number of clusters has become a consideration by researchers. There are two methods available, i.e., extrinsic and intrinsic methods. The extrinsic methods need expert judgment, while the intrinsic methods use algorithms to find the best number of clusters. We chose the intrinsic method, i.e., the Silhouette Coefficient, because there is no expert involved in our work. In addition, our experiment also evaluates the results using another well-known intrinsic method (i.e., Davies Bouldin Index). Both intrinsic methods result in the same optimal number of clusters (i.e., $k = 3$).

The results showed that combining memory-based and clustering methods could improve the prediction performance by reducing MAE by 1.88% and RMSE by 1.46% compared to the baseline method (UPCSim). In addition, the performance of the recommendation processing time after clustering improved 5.5 times faster than before clustering. It occurs because users with the same preferences are in one cluster, and the similarity calculation only considers data in one cluster without processing data in other clusters.

Furthermore, we also evaluated the performance of UPCSim and CB-UPCSim in another dataset (i.e., MovieLens 1M). Testing in MovieLens 1M dataset produces the average MAE and RMSE values of 0.6993 and 0.8921 for UPCSim and then 0.6857 and 0.8784 for CB-UPCSim. The testing results on the larger dataset show that CB-UPCSim also outperforms UPCSim, reducing MAE and RMSE by 1.94% and 1.53%, respectively. In addition, the larger dataset (MovieLens 1M) results in lower MAE and RMSE than the MovieLens 100k. It shows that the CB-UPCSim yields a low prediction error in a larger dataset, which is our research advantage.

Recently, some studies [36–38] also proposed sophisticated collaborative filtering to improve recommendation performances. The study conducted by [36] suggested new collaborative filtering using cognitive similarity. Their experimental results show that for the lower number of nearest neighbors (*k*) (i.e., $k = 10$ and $k = 20$), CB-UPCSim outperforms cognitive similarity. It becomes the advantage of our method because fewer nearest neighbors will need less computation time. However, for the higher *k* (i.e., $k = 30$ and $k = 50$), cognitive similarity outperforms our method and becomes the advantage of the cognitive similarity method. We still need further study to compare both algorithms for higher *k* (i.e., $k = 60$ up to $k = 100$) because the cognitive similarity did not measure them. In addition, Nguyen et al. [37] use word embedding to improve their proposed collaborative filtering. Hence, implementing word embedding in CB-UPCSim can be another option to obtain better performance. Furthermore, Logesh et al. [38] presented user-based collaborative filtering using a new bio-inspired clustering ensemble (BICE). This method was evaluated to large-scale datasets (i.e., Yelp and TripAdvisor). The clustering process in BICE will also be the next consideration to obtain the optimal cluster in CB-UPCSim.

Although the proposed system can improve recommendation performance (both rating prediction and processing time), it still has drawbacks where the similarity calculation has to work serially. This similarity calculation starts from calculating the user rating score-based similarity, calculating the user behavior score-based similarity, and finally calculating both similarity weightings. The computation time of our system consists of the overhead computation time and the prediction computation time. The overhead computation time of this research is the pre-processing data time that includes the dataset reading (130.11 milliseconds), attribute reduction (20.05 milliseconds), and attribute conversion

(16.35 milliseconds). This overhead is usually out of the researchers' consideration because only the prediction computation time will be taken into account in the operational system. Meanwhile, the prediction computation time consumes 0.91 s (the average running time from Table 4).

Besides the similarity calculation working serially, there are three other limitations in our proposed method. First, our method only uses MovieLens with sizes 100 k and 1 M. Further research must investigate MovieLens with sizes 10 M, 20 M, and 25 M. Second, in the pre-processing stage, we convert three attributes of user profile data (gender, occupation, and location) into a numeric type, transform the gender (M, F) into (1, 2) and convert the occupation into 1 to 21. We may explore other conversion techniques to gain a better performance. At the same time, there is a conversion from the location to the first digit of location. We assume that the conversion into two digits of location will increase the prediction result. Finally, in the post-processing, we only measure the rating prediction error without measuring the quality of top-N recommendations.

5. Conclusions

This paper focuses on improving recommendation performance from a previous similarity algorithm involving user behavior scores. We propose a combined clustering and memory-based method by using *k*-means clustering and UPCSim. The clustering method based on the user profile similarity can speed up the processing time of the recommendation system by 4.16 s. In addition, the method can increase the system performance with a decline of MAE and RMSE by 1.88% and 1.46% in the MovieLens 100k dataset. In a larger dataset (MovieLens 1M), our method yields better prediction performance.

For further research, the system development can consider parallel processing to calculate the similarity between users and explore other clustering methods to improve recommendation performance. Moreover, the pre-processing stage can be extended by considering two digits of location, and the post-processing can involve measurement of the top-N recommendation.

Author Contributions: Conceptualization, T.W. and T.B.A.; methodology, T.W., I.H. and T.B.A.; software, T.W.; validation, T.W., I.H. and T.B.A.; formal analysis, T.W.; investigation, T.W.; resources, T.W.; data curation, T.W.; writing—original draft preparation, T.W.; writing—review and editing, I.H. and T.B.A.; visualization, T.W.; supervision, I.H. and T.B.A.; project administration, T.W.; funding acquisition, T.W. and T.B.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Indonesia Endowment Fund for Education (LPDP), Ministry of Finance of Republic of Indonesia: *Beasiswa Unggulan Dosen Indonesia-Dalam Negeri* (*BUDI-DN*) with contract number 20200421211035 and Directorate General of Higher Education (Dikti), Ministry of Education, Culture, Research and Technology, Research Grant: *Penelitian Disertasi Doktor* with contract number 2313/UN1/DITLIT/DIT-LIT/PT/2021.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated and analyzed during the current study are available in the MovieLens dataset (https://grouplens.org/datasets/movielens/, accessed on 15 November 2020).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kherad, M.; Bidgoly, A.J. Recommendation system using a deep learning and graph analysis approach. *arXiv* **2020**, arXiv:2004.08100.
2. Sejwal, V.K.; Abulaish, M. Jahiruddin Crecsys: A context-based recommender system using collaborative filtering and lod. *IEEE Access* **2020**, *8*, 158432–158448. [CrossRef]
3. Feng, J.; Fengs, X.; Zhang, N.; Peng, J. An improved collaborative filtering method based on similarity. *PLoS ONE* **2018**, *13*, 1–18. [CrossRef] [PubMed]
4. Su, Z.; Lin, Z.; Ai, J.; Li, H. Rating Prediction in Recommender Systems based on User Behavior Probability and Complex Network Modeling. *IEEE Access* **2021**, *9*, 30739–30749. [CrossRef]

5. Sardianos, C.; Papadatos, G.B.; Varlamis, I. Optimizing parallel collaborative filtering approaches for improving recommendation systems performance. *Information* **2019**, *10*, 155. [CrossRef]
6. Ortega, F.; Mayor, J.; López-Fernández, D.; Lara-Cabrera, R. CF4J 2.0: Adapting Collaborative Filtering for Java to new challenges of collaborative filtering based recommender systems. *Knowl.-Based Syst.* **2020**, *215*, 106629. [CrossRef]
7. Zhang, F.; Qi, S.; Liu, Q.; Mao, M.; Zeng, A. Alleviating the data sparsity problem of recommender systems by clustering nodes in bipartite networks. *Expert Syst. Appl.* **2020**, *149*, 113346. [CrossRef]
8. Alhijawi, B.; Al-Naymat, G.; Obeid, N.; Awajan, A. Novel predictive model to improve the accuracy of collaborative filtering recommender systems. *Inf. Syst.* **2021**, *96*, 101670. [CrossRef]
9. Wang, D.; Yih, Y.; Ventresca, M. Improving neighbor-based collaborative filtering by using a hybrid similarity measurement. *Expert Syst. Appl.* **2020**, *160*, 113651. [CrossRef]
10. Liu, H.; Hu, Z.; Mian, A.; Tian, H.; Zhu, X. A new user similarity model to improve the accuracy of collaborative filtering. *Knowl.-Based Syst.* **2014**, *56*, 156–166. [CrossRef]
11. Patra, B.K.; Launonen, R.; Ollikainen, V.; Nandi, S. A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data. *Knowl.-Based Syst.* **2015**, *82*, 163–177. [CrossRef]
12. Polatidis, N.; Georgiadis, C.K. A multi-level collaborative filtering method that improves recommendations. *Expert Syst. Appl.* **2016**, *48*, 100–110. [CrossRef]
13. Zhang, F.; Zhou, W.; Sun, L.; Lin, X.; Liu, H.; He, Z. Improvement of Pearson similarity coefficient based on item frequency. *Int. Conf. Wavelet Anal. Pattern Recognit.* **2017**, *1*, 248–253. [CrossRef]
14. Sun, S.B.; Zhang, Z.H.; Dong, X.L.; Zhang, H.R.; Li, T.J.; Zhang, L.; Min, F. Integrating triangle and jaccard similarities for recommendation. *PLoS ONE* **2017**, *12*, e183570. [CrossRef] [PubMed]
15. Wu, C.; Wu, J.; Luo, C.; Wu, Q.; Liu, C.; Wu, Y.; Yang, F. Recommendation algorithm based on user score probability and project type. *Eurasip J. Wirel. Commun. Netw.* **2019**, *2019*, 80. [CrossRef]
16. Widiyaningtyas, T.; Hidayah, I.; Adji, T.B. User profile correlation-based similarity (UPCSim) algorithm in movie recommendation system. *J. Big Data* **2021**, *8*, 52. [CrossRef]
17. Lestari, S.; Adji, T.B.; Permanasari, A.E. WP-Rank: Rank Aggregation based Collaborative Filtering Method in Recommender System. *Int. J. Eng. Technol.* **2018**, *7*, 193–197.
18. Tran, C.; Kim, J.Y.; Shin, W.Y.; Kim, S.W. Clustering-Based Collaborative Filtering Using an Incentivized/Penalized User Model. *IEEE Access* **2019**, *7*, 62115–62125. [CrossRef]
19. Vellaichamy, V.; Kalimuthu, V. Hybrid collaborative movie recommender system using clustering and bat optimization. *Int. J. Intell. Eng. Syst.* **2017**, *10*, 38–47. [CrossRef]
20. Yu, P. Collaborative filtering recommendation algorithm based on both user and item. In Proceedings of the 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), Harbin, China, 19–20 December 2015; pp. 239–243. [CrossRef]
21. Jiawei, H.; Micheline, K.; Jian, P. *Data Mining: Concepts and Techniques Preface and Introduction*; Elsevier: New York, NY, USA, 2012; ISBN 9780123814791.
22. Harper, F.M.; Konstan, J.A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* **2015**, *5*, 1–19. [CrossRef]
23. Aggarwal, C.; Reddy, C. *Data Clustering: Algorithms and Applications*; Taylor & Francis Group, LLC.: Abingdon, UK, 2014; ISBN 9781466558229.
24. Garg, T.; Malik, A. Survey on Various Enhanced K-Means Algorithms. *Int. J. Adv. Res. Comput. Commun. Eng.* **2014**, *3*, 8525–8527. [CrossRef]
25. Indhu, R.; Porkodi, R. Comparison of Clustering Algorithm. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.* **2018**, *3*, 218–223.
26. Awawdeh, S.; Edinat, A.; Sleit, A. An Enhanced K-Means Clustering Algorithm for Multi-Attributes Data. *Int. J. Comput. Sci. Inf. Secur.* **2019**, *17*. Available online: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=An+Enhanced+K-means+Clustering+Algorithm+for+Multi-+attributes+Data&btnG= (accessed on 14 April 2021).
27. Zaki, M.J.; Meira, W. *Data Mining and Analysis: Fundamental Concepts and Algorithms*; Cambridge University Press: New York, NY, USA, 2014.
28. Ulian, D.Z.; Becker, J.L.; Marcolin, C.B.; Scornavacca, E. Exploring the effects of different Clustering Methods on a News Recommender System. *Expert Syst. Appl.* **2021**, *183*, 115341. [CrossRef]
29. Raval, U.R.; Jani, C. Implementing & Improvisation of K-Means Clustering Algorithm. *Int. J. Comput. Sci. Mob. Comput.* **2016**, *55*, 191–203. Available online: http://www.ijcsmc.com/docs/papers/May2016/V5I5201647.pdf (accessed on 14 April 2021).
30. Jose, J.T.; Zachariah, U.; Lijo, V.P.; Gnanasigamani, L.J.; Mathew, J. Case study on enhanced K-means algorithm for bioinformatics data clustering. *Int. J. Appl. Eng. Res.* **2017**, *12*, 15147–15151.
31. Bangoria Bhoomi, M. Enhanced K-Means Clustring Algorithm To Reduce Time Complexity for Numeric Values. *Int. J. Adv. Eng. Res. Dev.* **2014**, *5*, 876–879. [CrossRef]
32. Zhang, F.; Gong, T.; Lee, V.E.; Zhao, G.; Rong, C.; Qu, G. Fast algorithms to evaluate collaborative filtering recommender systems. *Knowl.-Based Syst.* **2016**, *96*, 96–103. [CrossRef]
33. Zheng, M.; Min, F.; Zhang, H.R.; Chen, W. Bin Fast Recommendations with the M-Distance. *IEEE Access* **2016**, *4*, 1464–1468. [CrossRef]

34. Fan, X.; Chen, Z.; Zhu, L.; Liao, Z.; Fu, B. A Novel Hybrid Similarity Calculation Model. *Sci. Program.* **2017**, *2017*, 4379141. [CrossRef]
35. Kuhn, M.; Johnson, K. *Applied Predictive Modeling*; Springer: New York, NY, USA, 2013; ISBN 9781461468493.
36. Nguyen, L.V.; Hong, M.S.; Jung, J.J.; Sohn, B.S. Cognitive similarity-based collaborative filtering recommendation system. *Appl. Sci.* **2020**, *10*, 4183. [CrossRef]
37. Nguyen, L.V.; Nguyen, T.H.; Jung, J.J.; Camacho, D. Extending collaborative filtering recommendation using word embedding: A hybrid approach. *Concurr. Comput.* **2021**, e6232. [CrossRef]
38. Logesh, R.; Subramaniyaswamy, V.; Malathi, D.; Sivaramakrishnan, N.; Vijayakumar, V. Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method. *Neural Comput. Appl.* **2020**, *32*, 2141–2164. [CrossRef]