Journal of Big Data

RESEARCH Open Access

# User profile correlation-based similarity (UPCSim) algorithm in movie recommendation system

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

*Correspondence: adji@ugm.ac.id
[1] Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada, Yogyakarta, Indonesia
Full list of author information is available at the end of the article

**Abstract**

Collaborative filtering is one of the most widely used recommendation system approaches. One issue in collaborative filtering is how to use a similarity algorithm to increase the accuracy of the recommendation system. Most recently, a similarity algorithm that combines the user rating value and the user behavior value has been proposed. The user behavior value is obtained from the user score probability in assessing the genre data. The problem with the algorithm is it only considers genre data for capturing user behavior value. Therefore, this study proposes a new similarity algorithm – so-called User Profile Correlation-based Similarity (UPCSim) – that examines the genre data and the user profile data, namely age, gender, occupation, and location. All the user profile data are used to find the weights of the similarities of user rating value and user behavior value. The weights of both similarities are obtained by calculating the correlation coefficients between the user profile data and the user rating or behavior values. An experiment shows that the UPCSim algorithm outperforms the previous algorithm on recommendation accuracy, reducing MAE by 1.64% and RMSE by 1.4%.

**Keywords:** Collaborative filtering, User rating value, User behavior value, UPCSim

## Introduction

The exponential growth of information on the internet causes users can get huge information resources to dig up and collect. This flood of information causes users difficulty accessing the desired information [1, 2]. Users have to spend more time and more energy finding the information they want, but users may not necessarily get satisfactory results. Fortunately, user behavior on e-commerce sites and other social networks can be recorded and be tracked, making it easier to analyze user interests [3, 4]. One of the tools to solve this problem in analyzing user interests is a recommendation system.

The recommendation system helps users get relevant items among millions of items in the database [5, 6]. The recommendation system's main task is to offer users personalized item recommendations through information filtering. This system has become a commercial platform that recommends users to select the desired items. The recommended items are useful to support users in various decision-making processes, such as what books to read, which locations to visit, what news to read, and more [7].

Based on the utilized data source and computation method, the recommendation system is divided into three approaches collaborative filtering, content-based filtering, and hybrid filtering [6, 8]. The collaborative filtering approach uses the collaborative power of ratings given by users to make recommendations. The content-based filtering approach uses descriptive attributes of items to make recommendations. Meanwhile, the hybrid filtering approach combines several filtering methods to get a list of items according to user preferences [9].

Collaborative filtering is the most popular, successful, and widely used among the above three recommendation system approaches [5, 10, 11] because it is simple, efficient, and has an acceptable accuracy level. Based on the advantages of this collaborative filtering, several real-world systems have used this method, such as Amazon, MovieLens, and Netflix [8, 10, 12].

The collaborative filtering approach is categorized into two approaches, viz ranking-oriented collaborative filtering and rating-oriented collaborative filtering [13, 14]. Ranking-oriented collaborative filtering directly provides a preference order of items for users without predicting the ratings of unrated items. In contrast, rating-oriented collaborative filtering predicts the ratings of unrated items based on rating information from other users [14]. The rating-oriented collaborative filtering approach is more widely popular because it is faster in generating recommendations than the ranking-oriented collaborative filtering approach.

The rating-oriented collaborative filtering approach is categorized into two methods, viz the model-based method and the memory-based method [8, 15, 16]. The model-based method uses a rating database to build a model and uses the model to predict ratings of unrated items [17]. Some of the techniques that are often used to build models include clustering [18, 19], Bayesian network [20], Markovian factorization [21], and Singular Value Decomposition (SVD) [22, 23]. Other sophisticated algorithms used model-based methods, such as combined deep learning and graph analysis approach [24] and deformable convolutional network [25]. This model-based method has the drawback that its computational complexity is very dependent on the model.

The memory-based method uses the rating database to calculate the similarity between users or similarity between items [26]. In its implementation, this method is divided into two techniques, namely User-Based Collaborative Filtering (UBCF) and Item-Based Collaborative Filtering (IBCF) [1, 2, 27]. The UBCF predicts ratings for all unrated items based on user similarity, while the IBCF predicts ratings based on item similarity [28]. Some of the frequently used traditional similarities are Cosine Similarity (COS), Pearson Correlation Coefficient (PCC), and Jaccard [2, 8].

Several similarity improvements are continuously being proposed to increase the recommendation accuracy. Among them are Bhattacharyya's similarity [16], the multi-level collaborative filtering similarity [29], the Triangle Multiplying Jaccard (TMJ) similarity [30], and the similarity integrating three impact factors, namely $S_1$, $S_2$, and $S_3$ [2]. These similarity algorithms only consider user rating values to calculate user similarity. A similarity algorithm was recently proposed in [31] that combines similarity based on user rating value and similarity based on user behavior value, which requires genre data. The problem with this approach is incorporating only the genre data in calculating the user behavior value. Thus, our motivation is to find the influence of other user behavior data on the recommendation accuracy.

Therefore, we aim to propose a new similarity algorithm called User Profile Correlation-based Similarity (UPCSim). Instead of only considering the genre data, this algorithm also examines other user behavior data or precisely user profile data (namely age, gender, occupation, and location) on giving weights to the similarity values. In this algorithm, the similarity weighting values are obtained by calculating the correlation coefficients between the user profile data and the user rating or user behavior values. Thus, the contribution of our study consists of two things:

1. The proposed UPCSim algorithm utilizes all user behavior data provided in the MovieLens 100K dataset to calculate the weights of similarity based on user rating value and similarity based on user behavior value.
2. The use of the UPCSim algorithm in a movie recommendation system is to classify similar user's preferences using $k$ nearest neighbors as a consideration in predicting unrated items.

The structure of this paper is as follows. In the "Related work" section, we describe the earlier research that used the similarity algorithms in the memory-based method. Then, the "Research method" section explains the proposed similarity algorithm in detail. The "Experiment" section present the experiment's results using the MovieLens dataset and its discussion. Finally, the "Conclusion and future work" section provides some conclusions and suggestions for further research development.

## Related work

Memory-Based method known as Memory-Based Collaborative Filtering (MBCF) is a prevalent recommendation method and is widely used in the commercial domain [16]. This method works by inputting the dataset, calculating similarity, determining the number of nearest neighbors who have similar interests, and calculating the predicted rating for unrated items. Among these processes, determining data similarity plays an essential part in improving an MBCF system.

The frequently used traditional similarity algorithms in recommendation systems are the Cosine similarity (COS) and the Pearson Correlation Coefficient (PCC) [32]. To formulate the similarities, we assume that the user and item sets are defined as $U = \{u_1, u_2, \ldots, u_m\}$ and $I = \{i_1, i_2, \ldots, i_n\}$. The user rating value matrix for the item set is denoted as $R = [r_{ui}]^{m \times n}$, where $m$ and $n$ are the number of users and the number of items, respectively, and $r_{ui}$ is the rating given by user $u$ on item $i$.

The Cosine similarity formula between user $u_1$ and user $u_2$ is stated in (1) and the PCC similarity formula is specified in (2).

$$Sim(u_1, u_2)^{COS} = \frac{\vec{r}_{u_1}.\vec{r}_{u_2}}{\|\vec{r}_{u_1}\|.\|\vec{r}_{u_2}\|} = \frac{\sum_{i \in I_{u_1} \cap I_{u_2}} r_{u_1 i}.r_{u_2 i}}{\sqrt{\sum_{i \in I_{u_1} \cap I_{u_2}} r_{u_1 i}^2}.\sqrt{\sum_{i \in I_{u_1} \cap I_{u_2}} r_{u_2 i}^2}} \quad (1)$$

$$Sim(u_1,u_2)^{PCC} = \frac{\sum_{i\in I_{u_1}\cap I_{u_2}}(r_{u_1i}-\overline{r}_{u_1}).(r_{u_2i}-\overline{r}_{u_2})}{\sqrt{\sum_{i\in I_{u_1}\cap I_{u_2}}(r_{u_1i}-\overline{r}_{u_1})^2}.\sqrt{\sum_{i\in I_{u_1}\cap I_{u_2}}(r_{u_2i}-\overline{r}_{u_2})^2} \tag{2}$$

In recent years, most researchers focused more on developing a similarity algorithm between users/items because of its simplicity in computation. For example, Patra et al. [16] proposed a similarity algorithm known as Bhattacharyya similarity. Their proposed similarity uses all ratings made by a pair of users. The relevance between a pair of rated items used Bhattacharyya measure. Furthermore, Polatidis et al. [29] proposed a similarity algorithm that increases PCC similarity, known as multi-level collaborative filtering. Their proposed similarity is an improvement in the PCC similarity by considering the number of co-rated items on several levels. Sun et al. [30] proposed a similarity algorithm by integrating the similarities of Triangle and Jaccard, known as TMJ similarity. The Triangle similarity calculates both the length and the angle of rating vectors between users, while the Jaccard similarity calculates unrated users. Feng et al. [2] proposed a new similarity algorithm by integrating three factors of similarity impact, namely $S_1$, $S_2$, and $S_3$. $S_1$ expresses the similarity between users, $S_2$ calculates the number of co-rated items less than the specified threshold, and $S_3$ explains the weight of each user rating value. The four similarity algorithms proposed by previous researchers perform similarity calculations based only on the user rating value data. However, using only user rating value data often causes the MBCF to result in improper inferences.

An illustration using the MovieLens dataset will explain how such MBCF can provide a wrong hypothesis when the inference process is only based on limited data/attributes. The MovieLens dataset stores some data, including movie data and rating data. Table 1 contains examples of movie data that records the genres of existing movies. Each movie can have several genres. Meanwhile, Table 2 includes rating data that records user rating values for the movies.

A scenario is given in Tables 1 and 2. User 1 likes to watch the animation genre and selects "Toy Story" movie, which is also categorized into two other genres, viz children and comedy. After watching the movie, User 1 gives a rating_value of 1 to the movie. The rating_value ranges from 1 (really dislike) to 5 (really like). If the MBCF infers a hypothesis solely based on the given low rating_value, the implication will be User 1 really dislikes the animation genre, which is not the case.

Thus, the MBCF may involve not only the rating_value data but also the behavior_value data for a better inference result. Table 3 summarizes the data in Tables 1 and 2 and provides information about the user, movie_title, rating_value given to movie_title, movie_title's genres, and movie_title genre's scores.

**Table 1** Examples of movie data

| id_movie | movie_title | Genre |
|---|---|---|
| *1* | *Toy Story* | *Animation*, Children, Comedy |
| 2 | Lion King | Animation, Children, Musical |
| 3 | Ghost in the shell | Animation, SciFi |
| 4 | Angels and Insect | Drama, Romance |
| 5 | Wallace and Gromit | Animation |

**Table 2** Examples of rating data

| id_user | id_movie | rating_value |
|---|---|---|
| *1* | *1* | *1* |
| 1 | 2 | 1 |
| 1 | 3 | 1 |
| 1 | 5 | 1 |
| 2 | 1 | 5 |
| 2 | 4 | 5 |

**Table 3** Summary of movie data and rating data

| id_user | movie_title | rating_value | Genre | Score |
|---|---|---|---|---|
| 1 | Toy Story | 1 | *Animation* | 1 |
| | | | Children | 1 |
| | | | Comedy | 1 |
| 1 | Lion King | 1 | *Animation* | 1 |
| | | | Children | 1 |
| | | | Musical | 1 |
| 1 | Ghost in the shell | 1 | *Animation* | 1 |
| | | | SciFi | 1 |
| 1 | Wallace and Gromit | 1 | *Animation* | 1 |
| 2 | Toy Story | 5 | Animation | 1 |
| | | | Children | 1 |
| | | | Comedy | 1 |
| 2 | Angels and Insect | 5 | Drama | 1 |
| | | | Romance | 1 |

Meanwhile, Table 4 accumulates each genre's scores in Table 3, showing the relationship between users and their genre preferences captured in the behavior_value data. The behavior_value data will be the new attribute in the recommendation process.

As previously mentioned, an MBCF that only uses rating_value data in the inference process using Tables 1 and 2 will give a wrong hypothesis of "User 1 really dislikes the animation genre". Therefore, using Table 4 that involves user behavior_value data, the MBCF will provide a better hypothesis of "User 1 likes the animation genre". It happenes because User 1 watches the animation genre four times.

Similarly, User 2 will get a wrong hypothesis of "User 2 really likes the animation genre" when MBCF only involves rating_value data (User 2 gives a rating_value of 5 to Movie 1) in the inference process using Tables 1 and 2. This statement contradicts the subsequent conclusion involving the behavior_value data in Table 4 that results in the hypothesis of "User 2 really dislikes the animation genre", because User 2 watches the animation genre one time.

Based on the previous scenario, MBCF will get better recommendation accuracy if it uses rating_value data and behavior_value data. To adopt this user behavior, Wu et al. [31] proposed a new similarity by combining similarity based on the user rating value and similarity based on user behavior value which extracts implicit user

**Table 4** User behavior values

| id_user | Genre | behavior_ value |
|---|---|---|
| *1* | *Animation* | *4* |
| 1 | Children | 2 |
| 1 | Comedy | 1 |
| 1 | Musical | 1 |
| 1 | SciFi | 1 |
| 2 | Animation | 1 |
| 2 | Children | 1 |
| 2 | Comedy | 1 |
| 2 | Drama | 1 |
| 2 | Romance | 1 |

behavior information in assessing the item type (genre). Their proposed similarity algorithm is known as the user score probability collaborative filtering (UPCF), as shown in (3).

$$Sim(u_1,u_2)^{UPCF} = \beta S_r(u_1,u_2) + (1-\beta)S_b(u_1,u_2) \tag{3}$$

$S_r$ ($u_1$,$u_2$) is the similarity between the user $u_1$ and user $u_2$ calculated by the UBCF method, $\beta$ is the threshold (which is between 0 to 1) that can be set to the average value of the similarity of all the users who are similar to the active user $u_1$, and $S_b$ is defined in (4) by referring to (2) .

$$S_b(u_1,u_2) = \frac{\sum_{g\in G_{u_1}\cap G_{u_2}}(P_{u_1g}-\overline{P}_{u_1}).(P_{u_2g}-\overline{P}_{u_2})}{\sqrt{\sum_{g\in G_{u_1}\cap G_{u_2}}(P_{u_1g}-\overline{P}_{u_1})^2}.\sqrt{\sum_{g\in G_{u_1}\cap G_{u_2}}(P_{u_2g}-\overline{P}_{u_2})^2}} \tag{4}$$

$G_{u_1}$ and $G_{u_2}$ are the set of item types (genres) rated by user $u_1$ and user $u_2$ respectively. $P_{u_1g}$ and $P_{u_2g}$ are the probability scores of item type $g$ assessed indirectly by user $u_1$ and user $u_2$ respectively. $\overline{P}_{u_1}$ and $\overline{P}_{u_2}$ are the average probability scores of all item types rated by user $u_1$ and user $u_2$ respectively, and $g$ is the type of item rated by both users.

The combination of the two similarities in the research conducted by Wu et al. [31] has several limitations. The calculation of similarity based on user behavior value only considers the item's genre data without capturing the user profile. Based on this problem, some researchers used user profile data to indicate similar users' preferences in recommender systems. For example, Al-Shamri [33] used three attributes of the user profile data (namely age, gender, and occupation) without involving the genre data to calculate the similarity between users in the demographic recommendation system. He conducted his experiment using the MovieLens 100K dataset. Each attribute of the user profile data was used to calculate its similarity and predictions. The final prediction was aggregated using a simple average of all individual prediction based on each attribute. The results of his research showed that the MAE and RMSE values are 0.91 and 1.22, respectively. Yassine et al. [34] used two user profile data attributes (namely gender and age). Their experiment utilized the MovieLens 100K dataset. They combined the collaborative filtering with $k$-means based on each user profile attribute.

The combined method was used to cluster movies into several clusters and the user profile was utilized to find user segmentation. The resulting segmented users were then recommended with the resulting clustered movies using model-based collaborative filtering with SVD. The result of the experiment showed that the *F*-measure of the *k*-means collaborative filtering on gender attribute and the *k*-means collaborative filtering on age attribute are 2.23 and 1.04, respectively.

Our study uses all user profile data (namely age, gender, occupation, and location) because age, gender, occupation, and location will influence the user's interest in the item. As an illustration, young users would have different preferences from older users. Female users would have different preferences from male users. Users with technician job would have different preferences from users with lawyer job, and users living on the coast would have different preferences from users living in cities. Thus, we will use all the user profile data to find the weights of similarities based on user rating and user behavior values. The similarity weighting values are obtained by calculating the correlation coefficients between the user profile data and the user rating or the user behavior values.

## Research method

In this study, a similarity algorithm called UPCSim is proposed. This UPCSim algorithm is embedded into an MBCF system, which was developed by Wu et al. [31]. Figure 1 details the proposed MBCF system.

We divide the system into four blocks: input, data preparation, the MBCF process, and output. The input block is the input dataset used in the MBCF system. The data preparation block consists of the data pre-processing stage, which results in a clean dataset. The data source includes the rating data and behavior data. While the behavior data used in Wu et al. [31] only employs the genre data (the green component in the data preparation block), our research also accommodates the user profile data (the red component in the data preparation block). The MBCF process block is the development of the MBCF method using the similarity weighting. The similarity weighting carried out by Wu et al. uses a threshold value ranging from 0 to 1 (the green component in the MBCF process). Our research's similarity weighting uses the correlation coefficients between the user profile data and the user rating or behavior values (the red component in the MBCF process). Finally, the output block evaluates the UPCSim algorithm in the MBCF method.

The detail of our proposed UPCSim algorithm is explained as a similarity calculation's component, which is presented in the "Similarity calculation" subsection. The detail of the developed MBCF system is described in "The developed MBCF system" subsection.

### Similarity calculation

In this study, we divide the similarity calculation between users into three components. The first s the $S_r$ similarity calculation component (shown in the dashed blue box). The second is the $S_b$ similarity calculation component (shown in the dashed green box). Finally, the UPCSim component (shown in the dashed red box) gives weights to both similarities. Figure 2 illustrates he three components in our similarity calculation.

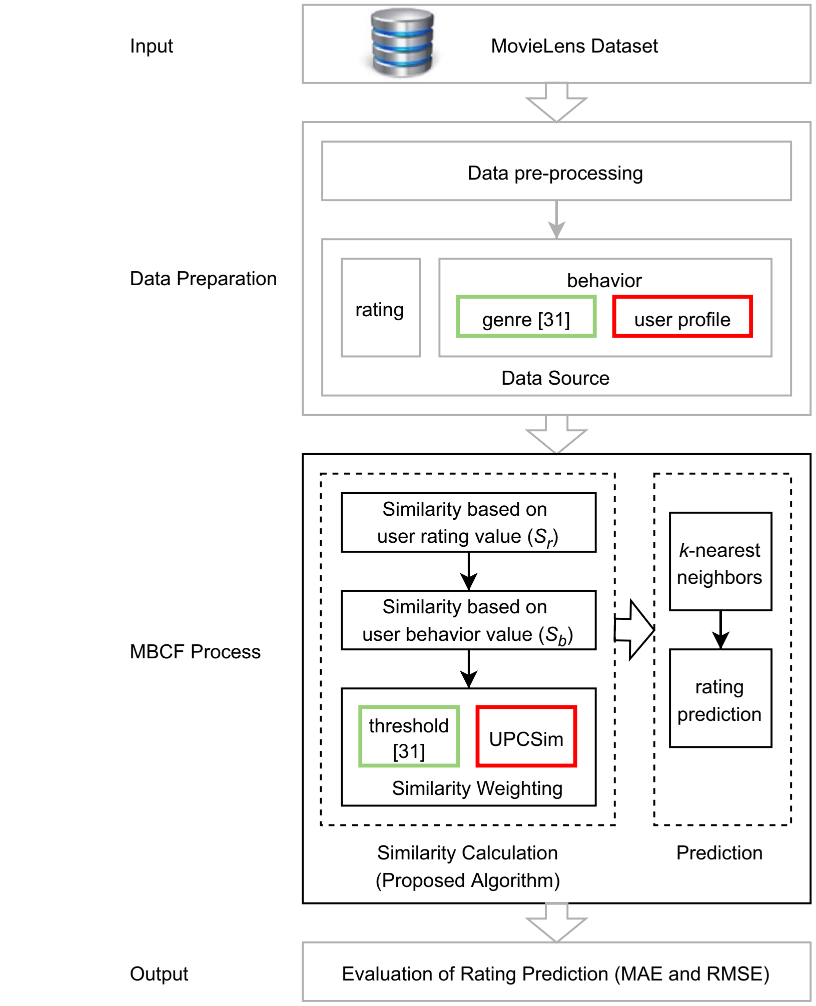

**Fig. 1** The developed MBCF system

Each component in Fig. 2 is described as follows.

### $S_r$ similarity

The $S_r$ similarity component is the similarity calculation based on the user rating value. As an example of the similarity calculation, we used the MovieLens 100K dataset. The initial stage in calculating the $S_r$ similarity is performed by reading the rating data from the resulted pre-processing data. Based on the rating data, we obtain a user rating value matrix of order 943 × 1682. The number 943 represents the number of users, and the number 1682 represents the number of movies in the dataset, shown as follow.

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{14} & \cdots & R_{1\_1682} \\ R_{21} & R_{22} & R_{23} & R_{24} & \cdots & R_{2\_1682} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ R_{943\_1} & R_{943\_2} & R_{943\_3} & R_{943\_4} & \cdots & R_{943\_1682} \end{bmatrix}$$

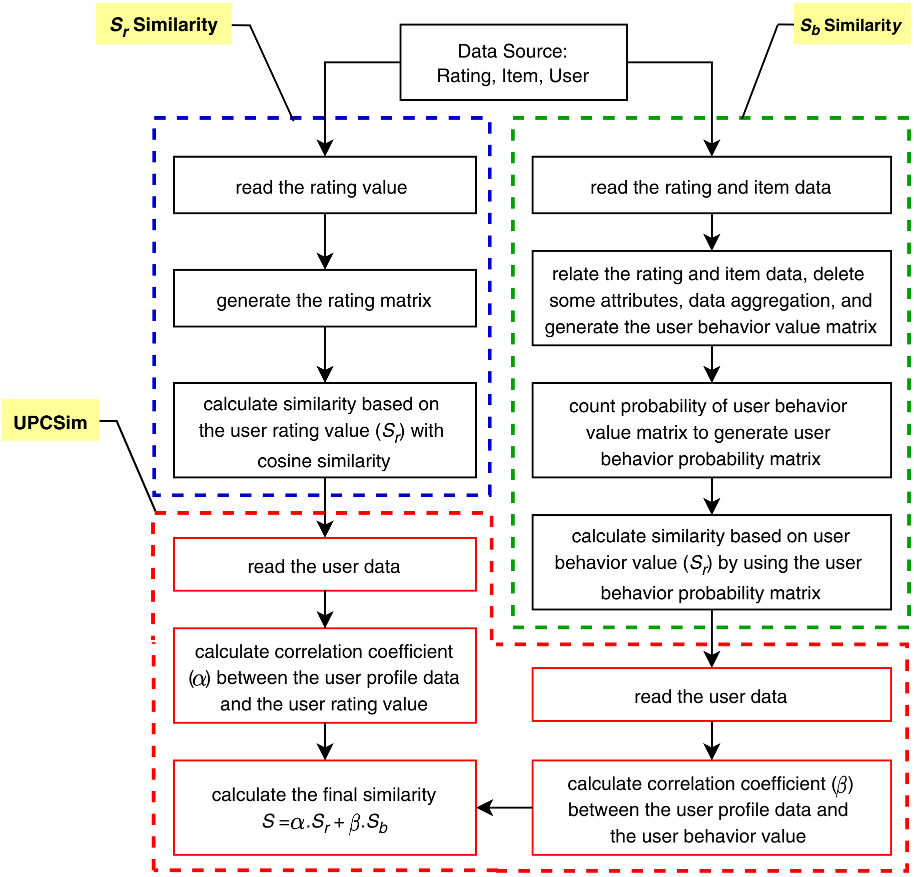
$S_r$ Similarity
Data Source: Rating, Item, User
$S_b$ Similarity
read the rating value
generate the rating matrix
calculate similarity based on the user rating value ($S_r$) with cosine similarity
read the rating and item data
relate the rating and item data, delete some attributes, data aggregation, and generate the user behavior value matrix
count probability of user behavior value matrix to generate user behavior probability matrix
calculate similarity based on user behavior value ($S_r$) by using the user behavior probability matrix
UPCSim
read the user data
calculate correlation coefficient ($\alpha$) between the user profile data and the user rating value
calculate the final similarity $S = \alpha.S_r + \beta.S_b$
read the user data
calculate correlation coefficient ($\beta$) between the user profile data and the user behavior value

**Fig. 2** Flowchart of similarity calculation

$R_{943\_1682}$ is the user rating value given by the 943rd user for the 1682nd item. The values of $R_{11}$ to $R_{9431682}$ range from 0 to 5, with a value of 0 indicates the user unrated the item.

After forming the user rating value matrix, the next step is to calculate the $S_r$ similarity using the Cosine similarity formula referred to (1). The final result of the $S_r$ similarity calculation forms the $S_r$ similarity matrix of order 943 × 943, shown as follow.

$$S_r = \begin{bmatrix} S_{11} & S_{12} & S_{13} & \cdots & S_{1\_943} \\ S_{21} & S_{22} & S_{23} & \cdots & S_{2\_943} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ S_{943\_1} & S_{943\_2} & S_{943\_3} & \cdots & S_{943\_943} \end{bmatrix}$$

$S_{1\_943}$ is the similarity value based on the rating between the 1st user and the 943rd user.

### $S_b$ *similarity*

The $S_b$ similarity component is the similarity calculation based on the user behavior value. This behavior value is obtained by finding the relationship between rating data and item data. In the MovieLens 100K dataset, the rating data describes the user rating value of each movie. Meanwhile, the item data represents the movie title data containing the genre information of each movie. Each movie title can include several genres. For example, the movie "Toy Story" has animation, children, and comedy genres.

After formulating the user behavior value, the next process is removing some unused attributes from the relationship between the rating data and the item data, and then performing data aggregation using the sum function grouped by user. These data aggregation results are illustrated in the user behavior value matrix of order 943 × 19. The number 943 represents the number of users, and the number 19 represents the number of genres, shown as follow.

$$B = \begin{bmatrix} B_{11} & B_{12} & B_{13} & \cdots & B_{1\_19} \\ B_{21} & B_{22} & B_{23} & \cdots & B_{2\_19} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ B_{943\_1} & B_{943\_2} & B_{943\_3} & \cdots & B_{943\_19} \end{bmatrix}$$

$B_{943\_19}$ is the 943rd user behavior value for the 19th genre, representing the total number of 19th genre watched by 943rd user. After forming the user behavior value matrix, the next stage is to calculate the probability of genre occurrence from the user behavior value matrix to produce a probability matrix of user behavior value using (5).

$$P = \frac{B(g)}{N} \tag{5}$$

$B(g)$ is the user behavior value for the target genre $g$, and $N$ is the total number of users who give rate to the target genre $g$. The illustration of the probability matrix of user behavior value is shown as follow.

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \cdots & P_{1\_19} \\ P_{21} & P_{22} & P_{23} & \cdots & P_{2\_19} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ P_{943\_1} & P_{943\_2} & P_{943\_3} & \cdots & P_{943\_19} \end{bmatrix}$$

$P_{943\_19}$ is the probability value of the 943rd user behavior for the 19th genre. The probability matrix of user behavior value is used for calculating the $S_b$ similarity referring to (4). The results of the $S_b$ similarity calculation forms a matrix of order 943 × 943, shown as follow.

$$S_b = \begin{bmatrix} S_{11} & S_{12} & S_{13} & \cdots & S_{1\_943} \\ S_{21} & S_{22} & S_{23} & \cdots & S_{2\_943} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ S_{943\_1} & S_{943\_2} & S_{943\_3} & \cdots & S_{943\_943} \end{bmatrix}$$

$S_{1\_943}$ is the similarity based on the user behavior value between the 1st user and the 943rd user.

### *UPCSim*

The UPCSim is a component of the similarity calculation using the UPCSim algorithm, which calculates the weights of both similarities ($S_r$ and $S_b$) based on the user profile attributes viz age, gender, occupation, and location as provided in the MovieLens 100K dataset. The weights of these two similarities are calculated based on the correlation coefficient ($R$) using multiple linear regression.

The general formula of the multiple linear regression and correlation coefficient ($R$) are defined in (6) and (7), respectively.

$$Y = a + b_1X_1 + b_2X_2 + b_3X_3 + \cdots + b_nX_n \tag{6}$$

$$R = \sqrt{\frac{b_1 \sum X_1Y + b_2 \sum X_2Y + b_3 \sum X_3Y + \cdots + b_n \sum X_nY}{\sum Y^2}} \tag{7}$$

$Y$ is dependent variable, $X$ is independent variables, $a$ is a constant, and $b$ is the regression coefficient for each independent variable. In our study, $Y$ represents user rating value or user behavior value, and $X$ denotes user profile data with four independent variables (namely age, gender, occupation, and location).

The weight of $S_r$ similarity is obtained by calculating the correlation coefficient between the user profile data (age, gender, occupation, and location) and the user rating value and is denoted by $\alpha$. The weight of $S_b$ similarity is obtained by calculating the correlation coefficient between user profile data (age, sex, occupation, and location) and the user behavior value and is symbolized by $\beta$.

After weighting both similarities, the next stage is to calculate the final similarity matrix by combining the weighted $S_r$ and $S_b$ similarities. The final similarity matrix $S$ of order $943 \times 943$ between user $u$ and user $v$ is defined in (8).

$$S(u,v) = \alpha S_r(u,v) + \beta S_b(u,v) \tag{8}$$

$S(u, v)$ is the final similarity between user $u$ and user $v$. $S_r(u,v)$ is the similarity based on user rating value between user $u$ and user $v$. $S_b(u,v)$ is the similarity based on user behavior value between user $u$ and user $v$. $\alpha$ is the weight of the similarity $S_r$, and $\beta$ is the weight of the similarity $S_b$.

### The developed MBCF system

Based on the illustration shown in Fig. 1, this subsection describes each block of the developed MBCF system.

#### *Input*

The first block of the developed MBCF system is the input dataset. In this paper, we used the MovieLens dataset collected by the "GroupLens Study Group of the University of Minnesota" [35]. The dataset consists of several versions, including ml-100K, ml-1M, ml-10M, ml-20M, etc. In this experiment, we chose the dataset used in a previous study [31], namely ml-100K (MovieLens 100K). This ml-100K dataset contains several data files. Our study used 3 data files: rating data, item data, and user data.

The rating data consists of 100,000 ratings as rated by 943 users on 1682 movies. Each user has rated at least 20 movies. The rating values given by the users range from 1 to 5. A score of 1 expresses that the user really dislike the movie, while a score of 5 describes that the user really likes the movie. This rating data has a sparsity of 93.7% and a density of 6.3%. This rating data structure consists of user-id, movie-id, rating, and timestamp.

Item data contains information about items (movies). This item data structure comprises 24 attributes: movie-id, movie title, release date, video release date, IMDb URL, and 19 attributes of movie type (genre). Each item/movie can have several genres.

User data contains information about the user profile. This user data structure comprises five attributes: user-id, age, gender, occupation, and zip code (which describes the user's location).

#### *Data preparation*

The second block is data preparation to perform data pre-processing by reducing irrelevant attributes. These irrelevant attributes are the timestamp of the rating data, movie title, release date, video release date, and IMDb URL of the item data.

#### *MBCF process*

The third block of the MBCF system is the MBCF process. The MBCF process contains two sub-blocks, namely the similarity calculation and the prediction.

The similarity calculation is the initial process used in the information filtering process using the MBCF approach. In this study, the similarity calculation consists of three components: $S_r$ similarity calculation, $S_b$ similarity calculation, and the UPCSim algorithm. Subsection "Similarity calculation" already details these three components.

The prediction is carried out to provide a predicted rating for items that had not been rated by active users. This prediction's initial stage is to determine the number ($k$) of the active user's nearest neighbors. $k$ is an integer number representing the number of neighbors, ranging from 10 to 100 [2, 29–31]. After determining the $k$ value , the next stage is to predict the ratings of unrated items.

The formula to predict the rating for an item ($i$) unrated by an active user ($u$) is shown in (9) [2, 16].

$$p_{ui} = \bar{r}_u + \frac{\sum_{v \in NNu} S(u,v).(r_{vi} - \bar{r}_v)}{\sum_{v \in NNu} |S(u,v)|}, v \neq u \tag{9}$$

$p_{ui}$ represents the predicted rating value of user $u$ to item $i$. $\bar{r}_u$ and $\bar{r}_v$ are the average ratings of user $u$ and user $v$, respectively. $r_{vi}$ is the rating value given by user $v$ to item $i$, $S(u, v)$ is the final similarity between user $u$ and user $v$, and $NNu$ is the set of nearest neighbors to user $u$.

#### *Output*

The fourth block of the MBCF system is the output block. This block evaluates the UPCSim Algorithm's performance in predicting ratings for items unrated by any active user.

The most prevalent MBCF system measures include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), precision, and recall. Jalili et al. [36] classified the metrics for evaluating recommendation systems into two categories: prediction and classification metrics. The MAE and RMSE primarily evaluate prediction [37, 38], whereas precision and recall evaluate classification, for example evaluating top-N recommendations [3].

In this study, we adopt the MAE and RMSE to measure the UPCSim Algorithm's prediction. The MAE is the most widely used metric in recommendation systems using a collaborative filtering approach. It is used to estimate the average absolute deviation between the actual and the predicted rating values. A lower MAE provides good recommendation quality [39]. The formula for calculating MAE is defined in (10).

$$MAE = \frac{1}{TN} \sum_{u \in U, i \in I} |p_{ui} - r_{ui}|. \tag{10}$$

RMSE reflects the degree of deviation between the predicted rating and the actual rating. A lower RMSE is associated with highlevel prediction [40]. The RMSE formula is expressed in (11).

$$RMSE = \sqrt{\frac{1}{TN} \sum_{u \in U, i \in I} (p_{ui} - r_{ui})^2} \tag{11}$$

$TN$ is the total number of predicted items. $p_{ui}$ and $r_{ui}$ represent the predicted rating and actual rating of the user $u$ to item $i$, respectively.

## Experiment

This section begins with the "Experiment design" subsection that describes the dataset statistics and the experiment's steps. Next, the "Experiment results and analysis" subsection explains the comparison between the proposed UPCSim algorithm and the previous similarity algorithms, namely the Cosine similarity algorithm and the UPCF similarity algorithm. The comparison utilized the MAE and RMSE values of the developed MBCF system. Finally, the "Discussion" section provides conclusions from the experiment results.

### Experiment design

We selected the ml-100K version of MovieLens in our study because this is a cleaned-up version dataset (MovieLens removes users who rate less than 20 items or who have no complete user profile). Based on a literature study conducted by Bansal et al. [41] on 108 research papers, 46% of research in recommendation system used the MovieLens dataset. The rest of the research used the Netflix dataset (8%), Last.fm (8%), Epinions (7%), Jester (7%), Amazon (4%), Microsoft Academic Search (4%), Flixter (4%), Yahoo WebScope (4%), Film Affinity (4%), and Movie Tweeting (4%).

The dataset constraint in our experiment is the dataset in the proposed algorithm does not contain only rating data but also information about the genre and the user profile data (namely age, gender, occupation, and location) to capture user's preferences. The statistics of the ml -100K dataset is shown in Table 5. The challenge is its very high sparsity which reaches 93.7% because users only rate a few items. Thus, we predicted the unrated items using UPCSim and $k$ nearest neighbor algorithm.

To evaluate the performance of the proposed UPCSim algorithm, the experiment design in this study implemented the following four steps:

The first step was to split the dataset into two parts, viz training data and testing data. The $k$-fold cross-validation method was applied with $k$=5 separating 80% of the dataset or training and the remaining 20% or testing. The training data are named train1, train2, train3, train4, and train5, and the testing data are called test1, test2, test3, test4, and test5.

The second step was to calculate the similarity matrix between users already explained in the "UPCSim" subsection. The $S_r$ similarity was obtained based on the user rating value matrix, while the $S_b$ similarity was acquired based on the user behavior value matrix. Both similarities were then weighted with correlation coefficients ($R$) of the multiple linear regression analysis processes.

The third step was to calculate the predicted ratings for the testing data. The nearest neighboring $k$ was selected based on the final similarity matrix. In this experiment, the $k$ value increased from 10 to 100 with a 10-step increment.

The fourth step was to measure the proposed UPCSim algorithm's prediction using the MAE and RMSE metrics.

The proposed and previous algorithms were coded using Python programming and were compiled using Jupiter notebook, which ran under Microsoft Windows 7 operating system. The experimental platform configuration is Intel® Core™ i7-4510U CPU @ 2.000 GHz (4CPUs), 2.6 GHz, 16 GB memory.

### Experiment results and analysis

This subsection aims to compare the proposed UPCSim algorithm's performance when utilized in MBCF with the traditional Cosine similarity and the UPCF similarity. Performance comparison of the three algorithms using iterations in the number of different neighbors (range from 10 to 100) would yield the MAE and RMSE values. The first iteration used the train1 and test1 datasets, then the second iteration used the train2 and test2 datasets, an repeated until the fifth iteration.

Table 6 shows the performance comparison of the average MAE of the three algorithms. MAE_c is the average MAE value of the UBCF experiment using Cosine similarity. MAE_p represents the average MAE value of the UBCF experiment using UPCF similarity, and MAE_ps explains the average MAE value of the UBCF experiment using the proposed UPCSim. MAE_ps-c denotes the difference between the MAE value using the proposed UPCSim and the MAE value using the Cosine similarity. Finally, MAE_ps-p is the difference between the MAE value using the proposed UPCSim and the MAE value using the UPCF similarity.

Based on Table 6, an increase in the number of nearest neighbors will decrease MAE values in the UPCF similarity algorithm and the UPCSim algorithm. Meanwhile, the Cosine algorithm's MAE value decreases from $k = 10$ to $k = 60$. Thus, it shows that the number of nearest neighbors affecting the algorithm's performance. The smallest MAE value is obtained in the UPCSim algorithm, which means that this algorithm's prediction error is the smallest. Thus, it can be said that the UPCSim algorithm is more reliable than others. Compared with the Cosine similarity, the UPCSim algorithm decreases MAE values ranging from 5.58 to 7.35%, with a decrease average MAE of 6.66% for all $k$ nearest neighbors. Compared with the UPCF similarity, the UPCSim algorithm decreases MAE values ranging from 1.23 to 1.84%, with a decrease average MAE of 1.64% for all $k$ nearest neighbors. The average MAE values of the three algorithms are illustrated graphically in Fig. 3.

**Table 5** The statistics of ml-100K dataset

| Dataset | #ratings | #users | #items | #genre | Ratings sparsity | Ratings density |
|---|---|---|---|---|---|---|
| MovieLens 100K | 100,000 | 943 | 1682 | 19 | 93.7% | 6.3% |

Figure 3 shows the three algorithms' MAE values decrease with an increasing number of nearest neighbors. At the beginning of the curves, the decline in MAE values is very sharp with the increase in the number of nearest neighbors, while at the end of the curves, the greater the number of nearest neighbors, the MAE values tend to be stable. It can be said that the number of nearest neighbors affects the MAE value, where the greater the number of nearest neighbors, the smaller the MAE value. With the same number of nearest neighbors, the MAE value of the UPCSim algorithm is always smaller than that of the other algorithms. In other words, the error between the actual rating and the predicted rating of the proposed UPCSim algorithm is the smallest.

Furthermore, Table 7 shows the comparison of the average RMSE values of the three recommendation algorithms. RMSE_c is the average RMSE value of the UBCF experiment using Cosine similarity. RMSE_p represents the average RMSE value of the UBCF experiment using UPCF similarity, and RMSE_ps explains the average RMSE value of the UBCF experiment using the proposed UPCSim. RMSE_ps-c is the difference between the RMSE value based on the proposed UPCSim and the RMSE value based on the Cosine similarity. Finally,RMSE_ps-p represents the difference between the RMSE value based on the proposed UPCSim and the RMSE value based on UPCF similarity.

As can be seen in Table 7, when the number of nearest neighbors is the same, the average RMSE values of the UPCSim are always smaller than those of the other algorithms. However, an increase in the number of nearest neighbors results in decreased RMSE values in the three algorithms. This result shows that the number of nearest neighbors influences the RMSE values. The UPCSim algorithm produces the smallest RMSE values, which means that the proposed algorithm's prediction error is the smallest and shows the UPCSim algorithm's superiority. Compared with the Cosine similarity, the UPCSim algorithm decreases RMSE values ranging from 6.24 to 8.17%, with a decrease average RMSE of 7.53% for all *k* nearest neighbors. Compared with the UPCF similarity, the UPCSim algorithm decreases RMSE values ranging from 0.42 to 1.79%, with a decrease average RMSE of 1.4% for all *k* nearest neighbors. The three algorithms' average RMSE values are illustrated graphically in Fig. 4.

Figure 4 illustrates the effect of changes in the number of nearest neighbors on the RMSE values. Three algorithms show a decrease in the RMSE value first and tend to be

**Table 6** Comparison of the average MAE values of the three algorithms

| Number of Neighbors | MAE_c | MAE_p | MAE_ps | MAE_ps-c | MAE_ps-p |
|---|---|---|---|---|---|
| 10 | 0.8227 | 0.7792 | 0.7669 | 0.0558 | 0.0123 |
| 20 | 0.8099 | 0.7631 | 0.7483 | 0.0616 | 0.0148 |
| 30 | 0.8051 | 0.7565 | 0.7410 | 0.0641 | 0.0155 |
| 40 | 0.8048 | 0.7551 | 0.7387 | 0.0661 | 0.0164 |
| 50 | 0.8043 | 0.7544 | 0.7369 | 0.0674 | 0.0175 |
| 60 | *0.8041* | 0.7535 | 0.7364 | 0.0677 | 0.0171 |
| 70 | 0.8049 | 0.7529 | 0.7359 | 0.0690 | 0.0170 |
| 80 | 0.8051 | 0.7526 | 0.7355 | 0.0696 | 0.0171 |
| 90 | 0.8056 | 0.7525 | 0.7347 | 0.0709 | 0.0178 |
| 100 | 0.8072 | *0.7521* | *0.7337* | *0.0735* | *0.0184* |
| Average | 0.8074 | 0.7572 | 0.7408 | 0.0666 | 0.0164 |

stable when the number of neighbors is greater than 50. The RMSE value of the UPCSim algorithm always shows the smallest value for each different number of neighbors. It indicates the UPCSim algorithm has the lowest error rate than the other two algorithms and confirms the UPCSim algorithm's advantage.

In addition to evaluating the MAE and RMSE, this study also performed the UPCSim, UPCF, and Cosine algorithms' execution time analysis. Table 8 shows the average execution time for these three algorithms. Note that the execution time in this experiment is the training time plus testing time. UPCSim requires a longer average execution time of 0.91 seconds for the Cosine algorithm and 0.54 seconds for the UPCF algorithm. This result occurred because UPCSim has a more complex algorithm than the Cosine and UPCF algorithms.

## Discussion

In this paper, we propose UPCSim algorithm in MBCF system. Our experiment results on the MovieLens 100K dataset show that UPCSim algorithm is comparable to UPCF algorithm with MAE and RMSE reductions of 1.64% and 1.4%, respectively. UPCSim algorithm also outperforms Cosine algorithm with MAE and RMSE reductions of 6.66% and 7.53%, respectively. These results can be obtained because UPCSim algorithm calculates the similarity weighting by involving more complete user behavior (genre and user profile) rather than only considers the genre attribute in UPCF algorithm. As a result, the prediction metric of the UPCSim algorithm is closer to the actual value.

UPCSim algorithm consists of three modules: the similarity calculation based on the rating value, the similarity calculation based on the user behavior value, and the weighting calculation for each similarity. In our experiment, these three modules run sequentially on a single personal computer. Consequently, this calculation process will take a long time and significantly affect the system's scalability. Several methods can overcome the system scalability problem for future works, such as parallel processing or distributed systems and

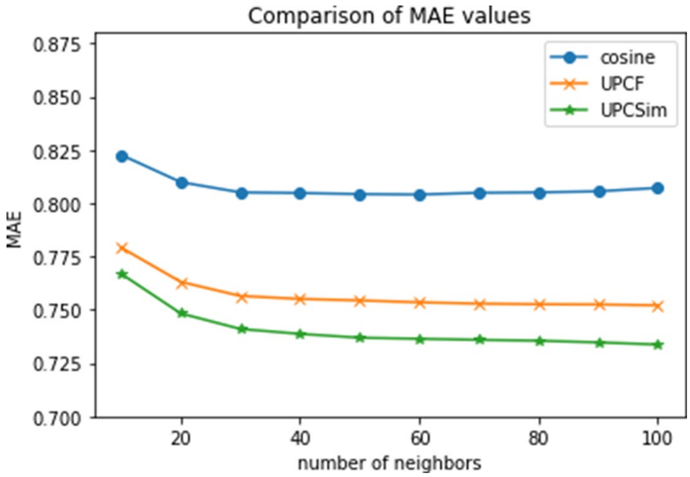

**Fig. 3** Comparison of the average MAE values of the three similarity algorithms using MovieLens 100K dataset

**Table 7** Comparison of the average RMSE values of the three algorithms

| Number of Neighbors | RMSE_c | RMSE_p | RMSE_ps | RMSE_ps-c | RMSE_ps-p |
|---|---|---|---|---|---|
| 10 | 1.0417 | 0.9835 | 0.9793 | 0.0624 | 0.0042 |
| 20 | 1.0248 | 0.9643 | 0.9541 | 0.0707 | 0.0102 |
| 30 | 1.0189 | 0.9574 | 0.9453 | 0.0736 | 0.0121 |
| 40 | 1.0163 | 0.9558 | 0.9427 | 0.0736 | 0.0131 |
| 50 | 1.0162 | 0.9556 | 0.9393 | 0.0769 | 0.0163 |
| 60 | *1.0154* | 0.9547 | 0.9389 | 0.0765 | 0.0158 |
| 70 | 1.0162 | 0.9544 | 0.9383 | 0.0779 | 0.0161 |
| 80 | 1.0170 | 0.9543 | 0.9381 | 0.0789 | 0.0162 |
| 90 | 1.0173 | 0.9542 | 0.9364 | 0.0809 | 0.0178 |
| 100 | 1.0176 | *0.9538* | *0.9359* | *0.0817* | *0.0179* |
| Average | 1.0201 | 0.9588 | 0.9448 | 0.0753 | 0.0140 |

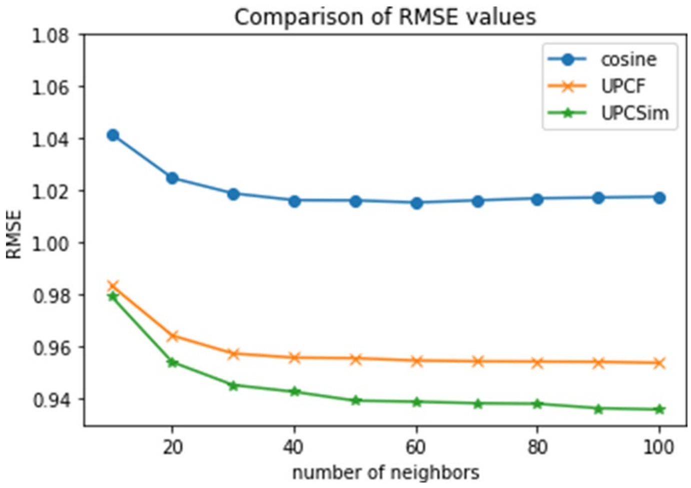

**Fig. 4** Comparison of the average RMSE values of the three similarity algorithms using MovieLens 100K dataset

clustering method, which reduces the large datasets. Distributed prediction models with big data technologies such as MapReduce, Apache Hadoop, and Apache Spark can also address large-scale data challenges.

The three modules in the UPCSim algorithm work using the static data-driven model principle. A data-driven model is a model of a system that is developed from the result of data analysis. The main concept is to find the relationships between the input and output without understanding the system's physical behavior [42]. The first and second modules use the principle of similarity whose formulations refer to (1) and (4). The third module uses the similarity weighting principle whose formulation refers to (7). Thus, all these three modules use similarity formulations based on metric distance without capturing the system's physical behavior. Therefore, these three modules can be categorized into data-driven models. In other words, all modules produce output relying solely on their data characteristics. Any added new variables will not change the algorithm model, adding another advantage to the proposed UPCSim algorithm.

Meanwhile, the static model is a model that represents a system at a certain point in time. It differs from the dynamic model that express system behavior over time [43]. As previously explained, UPCSim consists of three modules, all of which use static models.

The proposed UPCSim algorithm was also compared against the sophisticated model-based method, such as combined deep learning and graph analysis approach [24]. There are several differences between the model-based method and the proposed UPCSim algorithm. The datasets used in the model-based method were Epinions and Ciao, which were used in only 4% of recommendation system studies. Meanwhile, the proposed research used the MovieLens dataset is that was used in 64% of the recommendation system studies. The attributes used in the model-based method were rating and social trust data, while the proposed study used rating, genre, and user profile data. The size of the rating data in Epinions and Ciao are consecutively 6.65 times and 2.84 times larger than the rating data used in our study. These larger datasets led the model-based method to yield RMSE values of 0.9 for the Epinions dataset and 0.8 for the Ciao dataset. Nonetheless, our proposed research with a smaller dataset produced an RMSE of 0.94 and is still very competitive. Most researchers in deep learning-based recommendation systems as also used in [25] and [24] utilized large datasets such as Netflix, Amazon, Yelp, Ciao, and Epinions. Only a few researchers used the MovieLens dataset. As a result, the deep learning studies follow what deep learning researchers understand deep learning methods are more suitable for big data recommendation systems. Meanwhile, the proposed study suits handling data sparsity as the *k* nearest neighbor algorithm can predict unrated items. Furthermore, the deep learning studies have black box characteristics and cause difficulty finding reasons why certain users have preferences on specific items. Our proposed recommendation system successfully used the *k* nearest neighbor algorithm to segment users, making the system an explainable machine learning.

The weakness of the UPCSim algorithm is that a computational parallelization and distribution mechanism has not been developed. However, this can be done because the algorithms used in UPCSim are similarity, multiple linear regression, and *k* nearest neighbor. The three algorithms can naturally be parallelized and distributed.

**Table 8** Comparison of the average execution time of the three algorithms

| Number of Neighbors | time_cosine | time_UPCF | time_UPCSim |
|---|---|---|---|
| 10 | 3.67 | 4.51 | 4.78 |
| 20 | 3.91 | 4.48 | 4.82 |
| 30 | 4.21 | 4.60 | 5.02 |
| 40 | 4.42 | 4.70 | 5.27 |
| 50 | 4.82 | 5.01 | 5.68 |
| 60 | 5.08 | 5.28 | 5.99 |
| 70 | 5.10 | 5.37 | 6.02 |
| 80 | 5.25 | 5.38 | 6.10 |
| 90 | 5.29 | 5.38 | 6.15 |
| 100 | 5.25 | 5.81 | 6.28 |
| Average | 4.70 | 5.07 | 5.61 |

## Conclusion and future work

In this paper, we focus on improving the recommendation accuracy of the previous state-of-the-art algorithms. We proposed the UPCSim algorithm that accommodates user profile data for calculating the similarity weighting to capture the user preference more accurately. A new weighting schema is also generated for similarity based on user rating and user behavior values. The weighting values are obtained by calculating the correlation coefficients between the user profile data and the user rating value or user behavior value. The use of UPCSim algorithm in our MBCF system shows a recommendation improvement over the state-of-the-art algorithm with a decrease of MAE by 1.64% and RMSE by 1.4%.

The advantage of the UPCSim algorithm comes from the employment of three similarity modules (similarity based on user rating value, similarity based on user behavior value, and similarity weighting), which work based on metric distance without capturing the physical behavior of the system. Therefore, all modules produce output merely depending on their data characteristics. As a result, the UPCSim algorithm can add new variables without changing the system model as the user profile attribute increases. However, the UPCSim algorithm consumes more time compared with the previous algorithms.

In future studies, clustering methods can overcome scalability problems due to the larger number of datasets to reduce computation time. Besides that, it is also necessary to consider parallel processing and big data technology in large-scale data.

### Abbreviations

UPCSim: User Profile Correlation-based Similarity; MAE: Mean Absolute Error; RMSE: Root Mean Squared Error; SVD: Singular Value Decomposition; MBCF: Memory-Based Collaborative Filtering; UBCF: User-Based Collaborative Filtering; IBCF: Item-Based Collaborative Filtering; COS: Cosine Similarity; PCC: Pearson Correlation Coefficient; TMJ: Triangle Multiplying Jaccard; UPCF: User score Probability Collaborative Filtering.

### Acknowledgements

Not applicable.

### Authors' contributions

All authors contributed both the concepts and contents of this study. TW provided the manuscript under supervised by TBA and IH. All authors also performed discussion intensively for contents improvement. All authors read and approved the final manuscript.

### Funding

This research was funded by Indonesia Endowment Fund for Education (LPDP), Ministry of Finance of Republic of Indonesia: *Beasiswa Unggulan Dosen Indonesia - Dalam Negeri* $(BUDI - DN)$ Contract Number 20200421211035.

### Availability of data and materials

The datasets generated and analysed during the current study are available in the MovieLens dataset (https://grouplens.org/datasets/movielens/).

### Ethics approval and consent to participate

Not applicable.

### Consent for publication

Not applicable.

### Competing interests

The authors declare that they have no competing interests.

### Author details

[1] Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada, Yogyakarta, Indonesia.
[2] Department of Electrical Engineering, Universitas Negeri Malang, Malang, Indonesia.

Received: 2 November 2020 Accepted: 11 February 2021
Published online: 29 March 2021

## References

1. Xu G, Tang Z, Ma C, Liu Y, Daneshmand M. A collaborative filtering recommendation algorithm based on user confidence and time context. J Electr Comput Eng. 2019;2019:1–12. https://doi.org/10.1155/2019/7070487.
2. Feng J, Fengs X, Zhang N, Peng J. An improved collaborative filtering method based on similarity. PLoS ONE. 2018;13(9):1–18. https://doi.org/10.1371/journal.pone.0204003.
3. 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;56:156–66. https://doi.org/10.1016/j.knosys.2013.11.006.
4. Camacho LA, Alves-Souza SN. Social network data to alleviate cold-start in recommender system: a systematic review. Inf Process Manag. 2018;54:529–44. https://doi.org/10.1016/j.ipm.2018.03.004.
5. Sahu AK, Dwivedi P. User profile as a bridge in cross-domain recommender systems for sparsity reduction. Appl Intell. 2019;49:2461–81. https://doi.org/10.1007/s10489-018-01402-3.
6. Kumar P, Kumar V, Thakur RS. A new approach for rating prediction system using collaborative filtering. Iran J Comput Sci. 2019;2:81–7. https://doi.org/10.1007/s42044-018-00028-5.
7. Alonso S, Bobadilla J, Ortega F, Moya R. Robust model-based reliability approach to tackle shilling attacks in collaborative filtering recommender systems. IEEE Access. 2019;7:41782–98. https://doi.org/10.1109/ACCESS.2019.2905862.
8. Salah A, Rogovschi N, Nadif M. A dynamic collaborative filtering system via a weighted clustering approach. Neurocomputing. 2015;175:206–15. https://doi.org/10.1016/j.neucom.2015.10.050.
9. Aggarwal CC. Recommender systems. New York: Springer; 2016. https://doi.org/10.1007/978-3-319-29659-3.
10. Zhang J, Lin Y, Lin M, Liu J. An effective collaborative filtering algorithm based on user preference clustering. Appl Intell. 2016;45:230–40. https://doi.org/10.1007/s10489-015-0756-9.
11. Laishram A, Padmanabhan V, Lal RP. Analysis of similarity measures in user-item subgroup based collaborative filtering via genetic algorithm. Int J Inf Technol. 2018;10(4):523–7. https://doi.org/10.1007/s41870-018-0195-z.
12. Bagher R, Cami Hassanpour H, Mashayekhi H. User trends modeling for a content-based recommender system. Expert Syst Appl. 2017;87:209–19. https://doi.org/10.1016/j.eswa.2017.06.020.
13. Li G, Zhang Z, Wang L, Chen Q, Pan J. One-class collaborative filtering based on rating prediction and ranking prediction. Knowl Based Syst. 2017;124:46–54. https://doi.org/10.1016/j.knosys.2017.02.034.
14. Wang S, Huang S, Liu T-Y, Ma J, Chen Z, Veijalainen J. Ranking-oriented collaborative filtering: a listwise approach. ACM Trans Inf Syst. 2016;35(2):1–28. https://doi.org/10.1145/2960408.
15. Karabadji NEI, Beldjoudi S, Seridi H, Aridhi S, Dhifli W. Improving memory-based user collaborative filtering with evolutionary multi-objective optimization. Expert Syst Appl. 2018;98:153–65. https://doi.org/10.1016/j.eswa.2018.01.015.
16. Patra BK, Launonen R, Ollikainen V, Nandi S. A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data. Knowl Based Syst. 2015;82:163–77. https://doi.org/10.1016/j.knosys.2015.03.001.
17. Ocepeka U, Rugeljb J, Bosnića Z. Improving matrix factorization recommendations for examples in cold start. Expert Syst Appl. 2015;42(19):6784–94. https://doi.org/10.1016/j.eswa.2015.04.071.
18. Tran C, Kim JY, Shin WY, Kim SW. Clustering-based collaborative filtering using an incentivized/penalized user model. IEEE Access. 2019;7:62115–25. https://doi.org/10.1109/ACCESS.2019.2914556.
19. Bobadilla J, Bojorque R, Esteban AH, Hurtado R. Recommender systems clustering using Bayesian non negative matrix factorization. IEEE Access. 2018;6:3549–64. https://doi.org/10.1109/ACCESS.2017.2788138.
20. Vander Aa T, Chakroun I, Haber T. Distributed Bayesian probabilistic matrix factorization. In: Procedia of international conference on computational science, ICCS, 12–14 June 2017, Zurich, Switzerland; 2017. p. 1030–39. https://doi.org/10.1016/j.procs.2017.05.009.
21. Zhang R, Mao Y. Movie recommendation via Markovian factorization of matrix processes. IEEE Access. 2019;7:13189–99. https://doi.org/10.1109/ACCESS.2019.2892289.
22. Xian Z, Li Q, Li G, Li L. New collaborative filtering algorithms based on SVD++ and differential privacy. Math Probl Eng. 2017;2017:1–14. https://doi.org/10.1155/2017/1975719.
23. Guan X, Li CT, Guan Y. Matrix factorization with rating completion: an enhanced SVD model for collaborative filtering recommender systems. IEEE Access. 2017;5:27668–78. https://doi.org/10.1109/ACCESS.2017.2772226.
24. Kherad M, Bidgoly AJ. Recommendation system using a deep learning and graph analysis approach; 2020. p. 1–11. arXiv:2004.08100v5.
25. Li Z, Chen H, Lin K, Shakhov V, Shi L. Double attention-based deformable convolutional network for recommendation. In: Proceedings of the 2020 IEEE/CIC international conference on communications in China (ICCC); 2020. p. 1051–6. https://doi.org/10.1109/ICCC49849.2020.9238819.
26. Yue L, Sun XX, Gao WZ, Feng GZ, Zhang BZ. Multiple auxiliary information based deep model for collaborative filtering. J Comput Sci Technol. 2018;33(4):668–81. https://doi.org/10.1007/s11390-018-1848-x.
27. Shams B, Haratizadeh S. Item-based collaborative ranking. Knowl Based Syst J. 2018;152:172–85. https://doi.org/10.1016/j.knosys.2018.04.012.
28. Park Y, Park S, Jung W, Lee SG. Reversed CF: A fast collaborative filtering algorithm using a k-nearest neighbor graph. Expert Syst Appl. 2015;42(8):4022–8. https://doi.org/10.1016/j.eswa.2015.01.001.
29. Polatidis N, Georgiadis CK. A multi-level collaborative filtering method that improves recommendations. Expert Syst Appl. 2016;48:100–10. https://doi.org/10.1016/j.eswa.2015.11.023.
30. Sun SB, Zhang ZH, Dong XL, Zhang HR, Li TJ, Zhang L, Min F. Integrating triangle and Jaccard similarities for recommendation. PLoS ONE. 2017;12(8):1–11. https://doi.org/10.1371/journal.pone.0183570.
31. 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):1–13. https://doi.org/10.1186/s13638-019-1385-5.
32. Yu P. Collaborative filtering recommendation algorithm based on both user and item. In: Proceedings of 2015 4th international conference on computer science and network technology, ICCSNT 2015; 2015. p. 239–43. https://doi.org/10.1109/ICCSNT.2015.7490744.
33. Al-Shamri MYH. User profiling approaches for demographic recommender systems. Knowl Based Syst. 2016;100:175–87. https://doi.org/10.1145/2827872.

34. Yassine A, Mohamed L, Al Achhab M. Intelligent recommender system based on unsupervised machine learning and demographic attributes. Simul Model Pract Theory. 2020;107:1–9. https://doi.org/10.1016/j.simpat.2020.102198.
35. Harper FM, Konstan JA. The movielens datasets: history and context. ACM Trans Interact Intell Syst. 2015;5(4):1–19. https://doi.org/10.1145/2827872.
36. Jalili M, Ahmadian S, Izadi M, Moradi P, Salehi M. Evaluating collaborative filtering recommender algorithms: a survey. IEEE Access. 2018;6:74003–24. https://doi.org/10.1109/ACCESS.2018.2883742.
37. Zhang F, Gong T, Lee VE, Zhao G, Rong C, Qu G. Fast algorithms to evaluate collaborative filtering recommender systems. Knowl Based Syst. 2016;96:96–103. https://doi.org/10.1016/j.knosys.2015.12.025.
38. Zheng M, Min F, Zhang HR, Chen WB. Fast recommendations with the m-distance. IEEE Access. 2016;4:1464–8. https://doi.org/10.1109/ACCESS.2016.2549182.
39. Vellaichamy V, Kalimuthu V. Hybrid collaborative movie recommender system using clustering and bat optimization. Int J Intell Eng Syst. 2017;10(5):38–47. https://doi.org/10.22266/ijies2017.1031.05.
40. Fan X, Chen Z, Zhu L, Liao Z, Fu B. A novel hybrid similarity calculation model. Sci Program. 2017;2017:1–9. https://doi.org/10.1155/2017/4379141.
41. Bansal S, Baliyan N. A study of recent recommender system techniques. Int J Knowl Syst Sci. 2019;10(2):13–41. https://doi.org/10.4018/IJKSS.2019040102.
42. Solomatine DP, Ostfeld A. Data-driven modelling: some past experiences and new approaches. J Hydroinform. 2008;10(1):3–22. https://doi.org/10.2166/hydro.2008.015.
43. Buchadas A, Vas AS, Honrado JP, Alagador D, Bastos R, Cabral JA, Santos M, Vicente JR. Dynamic models in research and management of biological invasions. J Environ Manag. 2017;196:594–606. https://doi.org/10.1016/j.jenvman.2017.03.060.

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.