Asymmetric response aggregation heuristics for rating prediction and recommendation

Shujuan Ji[1], Wei Yang[1], Shenghui Guo[1], Dickson K.W. Chiu[2], Chunjin Zhang[3], Xinyue Yuan[1]
[1] Key Laboratory for wisdom mine information technology of Shandong Province, Shandong University of Science and Technology, Qingdao, China
[2] Faculty of Education, The University of Hong Kong, China
[3] Network Information Center (NIC), Shandong University of Science and Technology, Qingdao, Shandong 266590, China

Corresponding author: Chunjin Zhang (e-mail: zhangchjin@163.com).

Abstract User-based collaborative filtering is widely used in recommendation systems, which normally comprises three steps: (1) finding the nearest conceptual neighbors, (2) aggregating the neighbors' ratings to predict the ratings of unrated items, and (3) generating recommendations based on the prediction. Existing algorithms mainly focus on steps 1 and 3, but neglect subtle treatment of aggregating neighbors' suggestions in step 2. Based on the discovery of psychology that (i) *users' responses to positive and negative suggestions are different,* and (ii) *users may respond differently from one another*, this paper proposes a Personal Asymmetry Response-based Suggestions Aggregation (PARSA) algorithm, which first uses the linear regression method to learn each user's response to negative/positive suggestions from neighbors and then uses the gradient descent algorithm for optimizing them. In addition, this paper designs an Identical Asymmetry Response-based Suggestions Aggregation (IARSA) baseline algorithm, which assumes that *all the users' responses to suggestions are identical* as references to verify the key contribution of the heuristics employed in our PARSA algorithm that *user responses differently to positive and negative suggestions.* Three sets of experiments are designed and implemented over two real-life datasets (i.e., Eachmovie and Netflix) to evaluate the performance of our algorithms. Further, in order to eliminate the influence of different similarity measures, this paper select three kinds of similarity measures to discover neighbors. Experimental results demonstrate that most people indeed pay more attention to negative suggestions and our algorithms achieve better prediction as well as recommendation performances than the compared algorithms under various similarity measures.

Keywords collaborative filtering, response, positive suggestions, negative suggestions, linear regression method

1. Introduction

With the recent exponential growth of e-commerce transaction volume and user feedback, discovering useful information by recommendation systems (RSs) from such Big Data has become increasingly difficult. At present, collaborative filtering (CF) algorithms are commonly employed in RSs, and user-based CF normally comprises three steps, i.e., finding the nearest neighbors (conceptually near in purchasing requirement and rating patterns), aggregating the neighbors' suggestions to predict the object user's rating, and generating a recommendation list based on the prediction. Currently, most existing recommendation-related algorithms are dedicated to the first and third steps. However, aggregating conceptual neighbors' suggestions is vital to rating prediction and recommendation algorithms, because once neighbors are found based on a certain similarity measure, collaborative recommendation systems could aggregate neighbors' suggestions and predict user ratings of other unrated items. However, most of the existing algorithms [1-2] aggregate neighbors' suggestions ignoring whether a rating is positive or negative. This contradicts with psychology research discovery [3-7] that *people often pay more attention to negative than positive information.* Therefore, to improve prediction accuracy and recommendation performances, we propose a new method to aggregate similar neighbors' suggestions with the following key contributions:

(1) To improve the traditional suggestion aggregation (TSA) method, we propose a novel method (called PARSA algorithm) for aggregating neighbors' suggestions based on psychology findings that (i) *users' responses to positive and negative suggestions are different,* and (ii) *users may respond differently from one another*.

(2) To compare against a baseline, we design an IARSA algorithm, which is an extreme case of PARSA. With the assumption that *all the users' responses to positive and negative suggestions are identical*, the IARSA algorithm can isolate the effect of our main contribution in employing the novel heuristic that *people have different responses to positive and negative suggestions*, which is the lower bound performance of our PARSA algorithm.

(3) For the effective implementation of PARSA and IARSA, we have chosen to use the linear regression algorithm to learn the response factors to negative and positive suggestions, and the gradient descent algorithm for optimizing them.

(4) Three sets of experiments over two real-life datasets (Eachmovie and Netflix) are designed and implemented to verify the prediction and recommendation performances of our algorithms. What's more, we select three kinds of similarity

measures to discover the nearest neighbors in all experiments. Experimental results show that most people indeed pay more attention to negative suggestions, and the prediction and recommendation performances of our algorithms (i.e., PARSA and IARSA algorithms) are better than the compared algorithm (i.e., TSA algorithm) no matter what kind of similarity measure is adopted.

We develop the rest of this paper as follows. Section 2 reviews related work. Section 3 explains the main idea of our algorithms. Section 4 describes the experimental details, and the evaluation criteria before we conclude this paper in section 5.

2 Related work

We concentrate on the review of models designed for recommendation, as a more general review of recommender systems is available by Lu et al. [38]. The most popular recommendation methods are based on collaborative filtering algorithms [8], which are generally classified into user-based [9] and item-based [10] ones. Since this paper is based on users, we only review user-based ones in this section. Generally, there are three important steps in the users-based collaborative filtering algorithms, i.e., finding the nearest neighbors, aggregating the neighbor's suggestions to predict the object user's rating, and generating recommendations based on the prediction. As this paper directly adopts the simple top-N recommendation strategy, the third step will not be reviewed in detail.

2.1 Nearest Neighbors Discovering Methods

The most common method to discover the nearest neighbors for a given user is to discover users who rated items similarly. The most frequently adopted rating similarity measure is Cosine similarity, Pearson similarity, and Modified Cosine similarity. In order to estimate neighbors more accurately, many researchers [11-24] focused on improving these similarity calculation methods.

2.1.1 **Improved Cosine or Pearson Similarity**

Some researchers improved the traditional Cosine or Pearson methods that are used to search for nearest neighbors. For example, Srikanth et al. [11] proposed a new distance measure by improving the Pearson correlation to better evaluate the correlation between users whose ratings are linearly related. Wu et al. [12] estimated the similarity between users suggested with a ratio-based approach. Li et al. [13] integrated the Jaccard coefficients into Cosine similarity and Pearson similarity respectively to get two new similarity measures. Zang et al. [14] considered not only the co-rated items set, but also items rated only by neighbors. Suryakant et al. [15] proposed a CjacMD similarity measure, which combined Cosine, Jaccard, and Mean Measure of Divergence for evaluating sparse datasets.

To address the problem when only few ratings are available for similarity estimation, Liu et al. [16] considered both local context information of user ratings and global preference of user behavior. Further, some researchers introduced the concept such as the co-rated items and the non-common rated items into similarity calculation [17-19]. For example, Wang et al. [17] integrated an asymmetric factor based on the ratio of the co-rated items to all the rated items by each user into similarity calculation. Li et al. [18] introduced another asymmetric factor according to the ratio of co-rated items to non-common rated items by each user. Hu et al. [19] integrated the similarity of items to improve the calculation of users' similarity in the memory-based collaborative filtering algorithms.

2.1.2 **Trust and Distrust-Improved Similarity**

To further solve the sparseness of ratings, some researchers introduced social relationships [1, 20-22] into users' similarity calculation. For example, Lee et al. [20] combined user ratings with social trust information in similarity calculation, and in particular, distrust links are used to refine the propagation of trust relationship. However, in some e-commerce platforms, there are no explicitly trust and distrust relationships among users. To address this problem, implicit trust relationship is discovered among users over their historical ratings and integrated with ratings to calculated users' similarities [1, 21-22].

2.1.3 **Improved Similarity Considering Negative ratings**

The above methods only consider users' positive ratings in similarity calculation. However, in the analysis of users' historical ratings, researchers found that some users gave only sparse negative ratings without positive ones. Therefore, traditional methods of finding neighbors are ineffective for them. To improve that, some researchers believed that negative ratings may contribute to a more accurate finding of neighbors, and hence considered both positive and negative ratings in similarity calculation. For example, Zeng et al. [23] used a parameter to combine the positive and negative rating matrices to find neighbors, and showed that negative ratings should be weighted more. Frolov et al. [24] also believed that negative ratings are more important than positive ratings in calculating users' similarity, especially for users who only gave sparse negative ratings in their history.

2.2 Aggregating the Neighbors' Suggestions

Once knowing the nearest neighbors, the recommendation system next aggregates these suggestions and predicts the ratings of the object user to each unrated item. Equation (1) shows a common implementation of the traditional suggestion aggregation method [2]:

$$P_{m,c} = \overline{R}_m + \frac{\sum_{j=1}^{n} sim(u_m, u_j)(R_{j,c} - \overline{R}_j)}{\sum_{j=1}^{n} sim(u_m, u_j)} \tag{1}$$

where, $P_{m,c}$ represents predicted ratings of user *m* to item *c*, $\overline{R}_m$ is the average rating given by user *m*, $sim(u_m, u_j)$ is the similarity between user *m* and *j*, $\overline{R}_j$ is the average rating given by user *j*, and $R_{j,c}$ is the rating that user *j* rated item *c*.

2.3 Summary

Summarizing the work in Sections 2.1 and 2.2, we can see that most of the existing algorithms focus on how to discover the nearest neighbors, while neglecting how to aggregate neighbors' suggestions in a better way by just using a simple method like Equation (1). To enhance that, our method presented in this paper differentiates the negative and positive characteristics of neighbors' suggestions. The positive suggestions help increase the target user's interest (i.e., rating) while the negative suggestions are the hindrance that decreases user interest. This is reasonable and consistent with the research results in the psychology field.

According to psychologists Pratto & John [3], negative information tends to be considered more diagnostic than positive information, as people are more responsive to negatively-toned messages than to positive ones in daily life [4]. Further studies [5-7] by psychologists concluded that negative information should be weighted more heavily than positive information. Furthermore, Costa et al. [30] showed that different people have different personality, especially in taking others' suggestions. For example, it is easier for some people to accept others' suggestions, while some are more difficult to be persuaded. Moreover, some people are good at aggregating others' advice, while some tend to trust only one side. Some people just like to hear compliments, while some are more open to criticism or opposition. In summary, people's responses to positive and negative suggestions are different from one another.

As the aim of personal recommendation systems is to accurately mine each person's individual character in order to make precise recommendations, *personal response to positive and negative suggestions* should be considered differently. To our knowledge, until now, such a useful heuristic has not been considered in the existing recommendation related literature. Therefore, the main contribution of this paper is our exploration of this heuristic in aggregating neighbors' suggestions for the rating of predictions and recommendations. Section 3 illustrates our methods in detail.

3 Proposed Methodologies

As people have different responses to positive and negative suggestions according to psychology research [3-7], we first propose the necessary assumptions and definitions in Section 3.1. Next, we propose an aggregation heuristic (called PARSA) in Section 3.2, which considers that *the responses of users to positive and negative suggestions are different and may also be different from one another*. Then, we set a baseline comparison algorithm (called IARSA) in Section 3.3, which assumes simply that *all the users' responses to positive and negative suggestions are identical*, so that we can use this hypothetical extreme case to analyze the lower bound of PARSA algorithm.

3.1 Definitions and Assumptions

Definition 1 (Rating matrix) $R_{m\times c} = [r_{ui}]_{m\times c}$ represents the rating matrix that users $User = \{u_1, u_2, u_3, \ldots, u_m\}$ rated items $Item = \{i_1, i_2, i_3, \ldots, i_c\}$, which can be represented as Equation (2).

$$R_{m\times n} = \begin{pmatrix} r_{11} & \mathrm{L} & r_{1c} \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ r_{m1} & \mathrm{L} & r_{mc} \end{pmatrix} \tag{2}$$

where r_{ui} is the rating that user *u* rated item *i*.

Definition 2 (Positive and negative suggestions) For each user, the positive and negative suggestions are those with ratings larger and smaller than the average of all ratings given by the user, respectively.

In our aggregation algorithms proposed in the following sections of this paper, these concepts such as the increment for positive suggestions and the decrement for negative suggestions will be frequently used. To specify these concepts explicitly, we define them formally as follows.

Definition 3 (Increment for positive suggestions) $\Delta R_P^{m,c}$ represents the increment for positive suggestions that are given by the neighbors of user *m* to item *c*, which is calculated according to Equation (3).

$$\Delta R_P^{m,c} = \frac{\sum_{j=1, r_{j,c} > \bar{r}_j}^{n} sim(u_m, u_j)(r_{j.c} - \bar{r}_j)}{\sum_{j=1}^{n} sim(u_m, u_j)} \quad (3)$$

$$\bar{r}_j = \frac{1}{num} \sum_{c=1}^{num} r_{jc} \quad (4)$$

where $\Delta R_P^{m,c}$ represents the increment for positive suggestions, $sim(u_m, u_j)$ represents the relationship between user *m* and user *j*, $r_{j.c}$ is the rating that user *j* rated item *c*, u_j is the neighbor of user *m*, *n* is the number of user *m*'s neighbors, $\bar{r}_j$ is the average rating of user *j* according to Equation (4), and *num* is the number of the ratings given by user *j*.

Definition 4 (Decrement for negative suggestions) $\Delta R_N^{m,c}$ represents the decrement for negative suggestions provided by the neighbors of user *m* to item *c*, which is defined as Equation (5).

$$\Delta R_N^{m,c} = \frac{\sum_{j=1, r_{j,c} < \bar{r}_j}^{n} sim(u_m, u_j)(r_{j.c} - \bar{r}_j)}{\sum_{j=1}^{n} sim(u_m, u_j)} \quad (5)$$

where $\Delta R_N^{m,c}$ represents the decrement for negative suggestions, $sim(u_m, u_j)$ represents the relationship between user *m* and user *j*, $r_{j.c}$ is the rating that user *j* rated item *c*, u_j is the neighbor of the user *m*, *n* is the number of user *m*'s neighbors, $\bar{r}_j$ is the average rating of user *j* according to Equation (4), and *num* is the number of the rating given by user *j*.

It should be noted that there is no historical ratings for new users and hence the $\bar{r}_j$ of each new user is zero. To make above definition feasible, we set the average rating of the new users to a neutral value (e.g. 3 in a 5-rank metric ranging from 1 to 5, or 0.6 in a 5-rank metric ranging from 0.1 to 1), which can then evolved with time. Besides, the increments for the positive suggestions of neighbors have a promotional effect on the object user's interest and purchase decision, while the decrements for the negative suggestions of neighbors have a blocking effect on object user's interest and purchase decision. For different persons, their individual responses to positive and negative suggestions are often different. Therefore, we make the following assumptions.

Assumption 1 $S_m^P \in [0,1]$ and $S_m^N \in [0,1]$ represent user *m*'s responses to positive and negative suggestions, respectively. Supposing that *people have different responses to positive and negative suggestions*. That is to say,

$$S_m^P \neq S_m^N \quad (6)$$

From Assumption 1, we know that people have different responses to negative and positive suggestions. To further investigate how such difference influences suggestions aggregation, we make the following Assumption 2 for our target algorithm (i.e., PARSA algorithm) and Assumption 3 for a baseline algorithm (i.e., IARSA algorithm), respectively.

Assumption 2 (PARSA) *The responses of users to positive and negative suggestions are different from one another.*

$$\forall m \neq j,\ (S_m^P = S_j^P \ and\ S_m^N \neq S_j^N)\ or\ (S_m^P \neq S_j^P \text{ and } S_m^N = S_j^N)\ or\ (S_m^P \neq S_j^P \text{ and } S_m^N \neq S_j^N) \quad (7)$$

where S_m^P $\left(S_m^P \in [0,1]\right)$ and S_m^N $\left(S_m^N \in [0,1]\right)$ represent the responses to positive/negative suggestions of user *m*, S_j^P $\left(S_j^P \in [0,1]\right)$ and S_j^N $\left(S_j^N \in [0,1]\right)$ represent the responses to positive/negative suggestions of user *j*, user *m* and user *j* are the members of *User*.

Assumption 3 (IARSA) *All the users' responses to positive/negative suggestions are identical.*

$$\forall m \neq j,\ S_m^P = S_j^P \ and\ S_m^N = S_j^N \quad (8)$$

where S_m^P $\left(S_m^P \in [0,1]\right)$ and S_m^N $\left(S_m^N \in [0,1]\right)$ represent the responses to positive/negative suggestions of user *m*; S_j^P $\left(S_j^P \in [0,1]\right)$; S_j^N $\left(S_j^N \in [0,1]\right)$ represents the responses to positive/negative suggestions of user *j*; and user *m* and user *j* are the members of *User*.

It is obvious that Assumption 3 is an extreme case, which is designed only for analyzing the bound of Assumption 2, in particular, the lower bound of PARSA algorithm that is constructed based on Assumption 2.

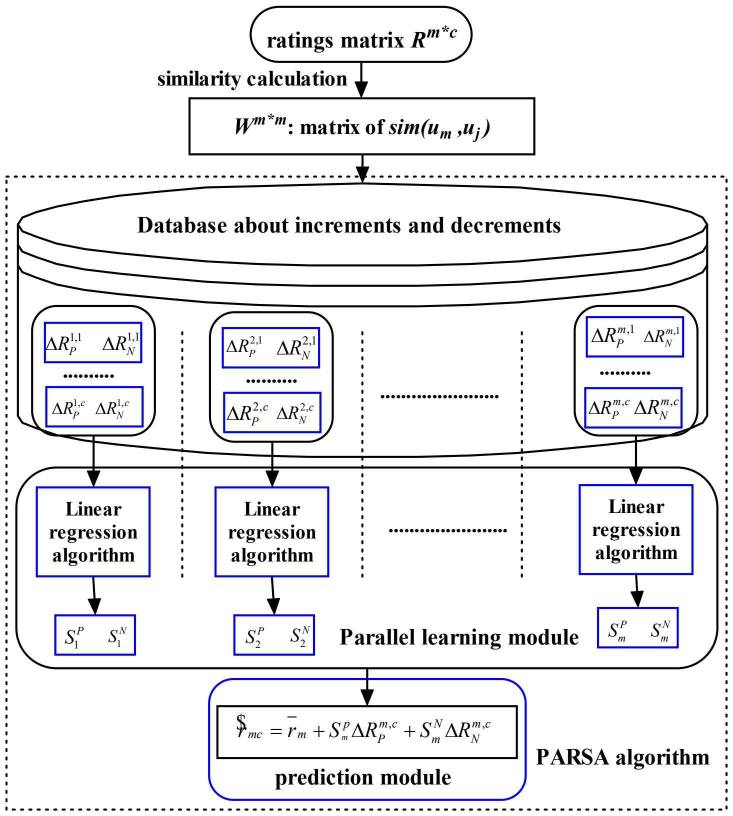
3.2 The Personal Asymmetry Response-based Suggestions Aggregation Algorithm (PARSA Algorithm)

Based on Assumptions 1 and 2, this section proposes a suggestion aggregation algorithm (see Figure 1) that each user gives personal responses to negative and positive suggestions (named PARSA algorithm).

Algorithm 1 describes the steps of the PARSA algorithm in details. This algorithm comprises four steps, i.e., calculation of weight matrix, calculation of increments for positive suggestions and decrements for negative suggestions, parallel learning of positive/negative responses, and prediction of ratings. In the first step, we calculate the similarity between any two users using an existing similarity calculation method (steps 1-5) to obtain a matrix of similarity (i.e., $W^{m\times m}$ in Figure 1). In the second step, based on Definitions 3 and 4, we can get the increments for positive suggestions and decrements for negative suggestions of each user (i.e., $\Delta R_P^{m,c}$ and $\Delta R_N^{m,c}$) generated according to the ratings that neighbor user *m* rated item *c* (see steps 6-11 in Algorithm 1) and store them in the database about increments and decrements (see Fig 1). In the third step, we learn each user's responses to positive/negative suggestions (i.e., S_m^P, S_m^N) (steps 12-17). Since the learning processes of users do not affect one another, we can use a parallel learning method, which is graphically described as the parallel learning module in Figure 1. Finally, we predict the rating that the user *m* may rate the item *c* according to Equation (9) and generate a recommendation list for each user (steps 18-23). The prediction module in Figure 1 represents the prediction process.

$$\hat{r}_{mc} = \bar{r}_m + S_m^P \Delta R_P^{m,c} + S_m^N \Delta R_N^{m,c} \tag{9}$$

where $\hat{r}_{mc}$ is the predicted rating that user *m* will rate item *c*, $\bar{r}_m$ is the average rating of user *m*, S_m^P is the response to positive suggestions of user *m*, S_m^N is the response to negative suggestions of user *m*, $\Delta R_P^{m,c}$ is the increment for positive suggestions of user *m* to the item *c*, and $\Delta R_N^{m,c}$ is the decrement for negative suggestions of user *m* to item *c*.



ratings matrix R^{m*c}
similarity calculation
W^{m*m}: matrix of $sim(u_m, u_j)$
Database about increments and decrements
$\Delta R_P^{1,1}$ $\Delta R_N^{1,1}$
$\Delta R_P^{1,c}$ $\Delta R_N^{1,c}$
$\Delta R_P^{2,1}$ $\Delta R_N^{2,1}$
$\Delta R_P^{2,c}$ $\Delta R_N^{2,c}$
$\Delta R_P^{m,1}$ $\Delta R_N^{m,1}$
$\Delta R_P^{m,c}$ $\Delta R_N^{m,c}$
Linear regression algorithm
Linear regression algorithm
Linear regression algorithm
S_1^P S_1^N
S_2^P S_2^N
Parallel learning module
S_m^P S_m^N
$\hat{r}_{mc} = \bar{r}_m + S_m^P \Delta R_P^{m,c} + S_m^N \Delta R_N^{m,c}$
prediction module
PARSA algorithm

FIGURE 1. Architecture of Personal Asymmetry Response-based Suggestions Aggregation (PARSA) algorithm

In detail, in the realization of the third step of the PARSA algorithm (i.e., the positive and negative responses learning module), we adopt a linear regression algorithm to approximate these data and a gradient descent algorithm to optimize the responses of user *m* to positive and negative suggestions (i.e., S_m^P and S_m^N, respectively). First, we define the cost function as Equation (10). Then, the optimization is performed through a gradient procedure which minimizes the cost function given in Equation (10). User *m's* gradients of $\Delta R_P^{m,c}$ and $\Delta R_N^{m,c}$ are formally defined as Equation (11). The learning formulae are further given in Equations (12) and (13).

$$E(m) = \frac{1}{2}\sum_{c=1}^{B_m} (r_{mc} - \hat{r}_{mc})^2 \tag{10}$$

$$\begin{cases} \nabla E_P(m) = -\frac{1}{B_m}\sum_{c=1}^{B_m}(r_{mc} - \hat{r}_{mc})\Delta R_P^{m,c} \\ \nabla E_N(m) = -\frac{1}{B_m}\sum_{c=1}^{B_m}(r_{mc} - \hat{r}_{mc})\Delta R_N^{m,c} \end{cases} \tag{11}$$

$$S_m^P \leftarrow S_m^P + \alpha \nabla E_P(m) \tag{12}$$

$$S_m^N \leftarrow S_m^N + \alpha \nabla E_N(m) \tag{13}$$

where $E(m)$ is the cost function of user *m*, $\nabla E_P(m)$ and $\nabla E_N(m)$ are the user *m's* gradients of $\Delta R_P^{m,c} / \Delta R_N^{m,c}$, B_m is the number of the items rated by user *m*, $\hat{r}_{mc}$ is the predicted rating that the user *m* will rate item *c*, r_{mc} is the true rating of user *m* to item *c*, S_m^P is the response of user *m* to positive suggestions, S_m^N is the response of user *m* to negative suggestions (with the initial value of S_m^P and S_m^N are generated at random), $\Delta R_P^{m,c}$ is the increment for positive suggestions of user *m* to item *c*, $\Delta R_N^{m,c}$ is the decrement for negative suggestions of user *m* to item *c*, and α is the learning rate in the gradient descent algorithm.

Algorithm1: PARSA (Personal asymmetry response-based suggestion aggregation) algorithm

Input: User-item rating matrix R^{m*c};
Output: Recommendation list for each user $m \in User$

```
1)  for $m \in User$ do
2)    for $j \in User$ and $m \neq j$ do
3)      Calculate $sim(u_m, u_j)$ according to a similarity calculation method
4)    end for
5)  end for
6)  for $m \in User$ do
7)    for $c \in Item$ do
8)      Calculate $\Delta R_P^{m,c}$ according to Equation (3)
9)      Calculate $\Delta R_N^{m,c}$ according to Equation (5)
10)   end for
11) end for
12) for $m \in User$ do
13)   repeat
14)     Learn $S_m^P$ according to Equation (12)
15)     Learn $S_m^N$ according to Equation (13)
16)   end repeat
17) end for
18) for $m \in User$ do
19)   for $c \in Item$ do
20)     Calculate $\hat{r}_{mc}$ according to Equation (9)
21)   end for
22)   get the top-N item as the recommendation list of user m
23) end for
```

3.3 An Identical Asymmetry Response-Based Suggestions Aggregations Aggregation Algorithm (IARSA Algorithm)

Based on Assumptions 1 and 3, this section outlines our baseline aggregation algorithm (see Figure 2 for the framework) that all users give identical responses to negative and positive suggestions (named IARSA). Similar to the PARSA algorithm, the IARSA algorithm also comprises four steps, i.e., calculation of weight matrix, calculation of increments for positive suggestions and decrements for negative suggestions, positive/negative responses learning, and rating prediction. The first, second, and fourth steps are identical to the ones in the PARSA algorithm. However, in the third step, according to Assumption 3, all the users give identical responses, therefore we can omit the subscript of S_m^P / S_m^N and represent all the users' positive/negative responses as S^P / S^N. The learning module in Figure 2 describes the learning process of S^P and S^N, with processing steps similar to the steps 12-17 in Algorithm 1. The only difference is that the IARSA algorithm needs not distinguish users in the process of learning. Besides, according to Assumption 1, *users' responses to positive suggestions are different to their responses to negative suggestions*, we can predict the rating that user *m* may rate the item *c* according to Equation (14). The prediction module in Figure 2 represents the prediction process.

$$\hat{r}_{mc} = \bar{r}_m + S^P \Delta R_p^{m,c} + S^N \Delta R_N^{m,c} \tag{14}$$

where, $\$_{mc}$ is the predicted rating that user *m* will rate item *c*, $\bar{r}_m$ is the average rating of user *m*, S^P is users' response to positive suggestions, S^N is users' response to negative suggestions, $\Delta R_P^{m,c}$ is the increment for positive suggestion of user *m* to item *c*, and $\Delta R_N^{m,c}$ is the decrement for negative suggestions of user *m* to item *c*.

In detail, in the realization of the third step of the IARSA algorithm (i.e., the positive and negative responses learning module), we adopt a linear regression algorithm to approximate these data and use the gradient descent algorithm to optimize the parameters (i.e., S^P and S^N). The optimization process is performed through a gradient procedure that aims at minimizing the cost function given in Equation (15), with the gradients defined as Equation (16).

$$E = \frac{1}{2}\sum_{m=1}^{A}\sum_{c=1}^{B_m}(r_{mc} - \hat{r}_{mc})^2 \tag{15}$$

$$\begin{cases} \nabla E_P = -\dfrac{1}{\sum_{m=1}^{A} B_m}\sum_{m=1}^{A}\sum_{c=1}^{B_m}(r_{mc} - \hat{r}_{mc}).\Delta R_P^{m,c} \\ \nabla E_N = -\dfrac{1}{\sum_{m=1}^{A} B_m}\sum_{m=1}^{A}\sum_{c=1}^{B_m}(r_{mc} - \hat{r}_{mc}).\Delta R_N^{m,c} \end{cases} \tag{16}$$

where E is the cost function, ∇E_P and ∇E_N are the gradients of $\Delta R_P^{m,c}$ and $\Delta R_N^{m,c}$, r_{mc} represents the true rating of user *m* to item *c*, $\$_{mc}$ is the predicted rating that user *m* will rate item *c*, *A* is the size of *User*, and B_m is the number of items that user *m* rated.

The learning formulae are defined as follows in Equations (17) and (18):

$$S^P \leftarrow S^P + \alpha \nabla E_P \tag{17}$$

$$S^N \leftarrow S^N + \alpha \nabla E_N \tag{18}$$

where S^P is the response to positive suggestions, S^N is the response to negative suggestions (with the initial value of S^P and S^N generated at random), $\Delta R_P^{m,c}$ is the increment for positive suggestions of user *m* to item *c*, $\Delta R_N^{m,c}$ is the decrement for negative suggestions of user *m* to item *c*, and α is the learning rate in the gradient descent algorithm.

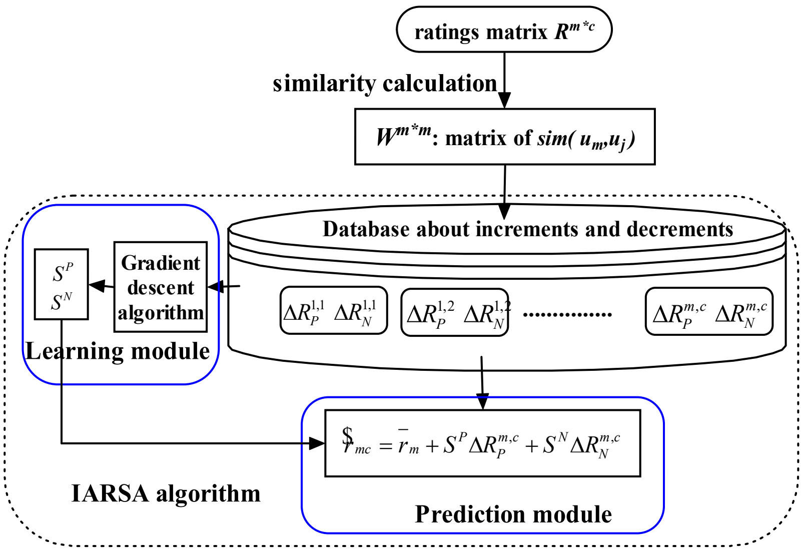

ratings matrix R^{m*c}
similarity calculation
W^{m*m}: matrix of $sim(u_m,u_j)$
Database about increments and decrements
S^P
S^N
Gradient descent algorithm
Learning module
$\Delta R_P^{1,1}$ $\Delta R_N^{1,1}$
$\Delta R_P^{1,2}$ $\Delta R_N^{1,2}$
....
$\Delta R_P^{m,c}$ $\Delta R_N^{m,c}$
$\$_{mc} = \bar{r}_m + S^P\Delta R_P^{m,c} + S^N\Delta R_N^{m,c}$
IARSA algorithm
Prediction module

FIGURE 2. Architecture of Identical Asymmetry Response-based Suggestions Aggregation (IARSA) algorithm

4. Experiments and Analysis

To verify the performance of the PARSA algorithm (which differentiates the responses to negative and positive characteristics of neighbors' suggestions in aggregating neighbors' suggestions) proposed in this paper, we design and implement three sets of experiments for evaluation. As the first set of experiments aim at verifying our Assumption 2 that *each user has different responses to suggestions,* we perform statistics on the responses of users to positive/negative suggestions on each kind of similarity. The second set of experiments target at verifying the performance of the personal asymmetry response-based suggestions aggregation algorithm (PARSA) with some benchmark similarity measures, by comparing these algorithms based on Mean Absolute Error (MAE). The setting of the third set of experiments is to demonstrate the recommendation performance of our proposed PARSA algorithm, in which we compare it with benchmark aggregation methods (i.e., IARSA and TSA algorithms) based on precision, recall, diversity, and coverage when making top-10 recommendations with different similarity measures.

In detail, in the second and third set of experiments, we compare the PARSA algorithm with the baseline IARSA algorithm proposed in section 3.3 and the traditional suggestion aggregation (TSA) algorithm defined in Equation (1). In these experiments, in order to verify the robustness of PARSA algorithm, we select three kinds of similarity measures to discover the nearest neighbors, such as traditional Pearson Correlation Coefficient (PCC) [13], TMFSF similarity [1], and a hybrid similarity (HySim)[17]. The main ideas of these similarity measures have been briefly reviewed in section 2.1.1 and 2.1.2. We select these measures because PCC is a widely used traditional similarity measure, TMFSF similarity considers also the trust relationship [1], and HySim considers almost all aspects to address the data sparseness problem [17]. Based on each kind of similarity measures (i.e., PCC, TMFSF, and HySim), we compare the PARSA algorithm with the IARSA and TSA algorithms. Therefore, we totally implement nine combinations of algorithms (see Table 1) in these two sets of comparison experiments. In particular, to test whether the performances of the PARSA algorithm and the baseline algorithms are affected by the sparseness and the randomness of dataset, we compare these algorithms on two sets of datasets, one of which is a small volume dataset with serious sparseness problem (i.e., Eachmovie with 5-rank values 0, 0.2, 0.4, 0.6, 0.8, 1.0, http://www.kumpf.org/eachtoeach/eachmovie.html), and the other is a larger volume one with less sparseness problem (i.e., Netflix with 5-rank values 0, 1, 2, 3, 4, 5, http://www.datatang.com/data/45455).

Table 1 Comparison algorithms with different similarity measures

Aggregation algorithm / similarity measures	TSA	IARSA	PARSA
PCC	PCC+TSA	PCC+IARSA	PCC+PARSA
TMFSF	TMFSF+TSA	TMFSF+IARSA	TMFSF+PARSA
HYSIM	HYSIM+TSA	HYSIM+IARSA	HYSIM+PARSA

4.1 Experiment Setting and Data Preprocessing

In our experiments, the parameters of the gradient descent algorithm and TMFSF algorithm should be assigned with proper values, which are shown in Table 2. α is the learning rating in the gradient descent algorithm, which is defined in equations (12), (13), (17), and (18). As many researchers set it to 0.15, we also set it to 0.15 in this paper. As for f, a parameter in calculating TMFSF similarity, we obtain its optimal value on different datasets via experiments. The process of learning them is shown in the Appendix.

As there are many inactive users in Eachmovie and Netflix, we must pre-process these two datasets. For the Eachmovie dataset, we filter out the users with less than 200 reviews. For the Netflix dataset, as it is greatly bigger and less sparse than Eachmovie, we filter out the users with less than 500 comments. The detail information about the raw and preprocessed datasets is shown in Tables 3 and 4, respectively. Each dataset is divided into two parts, 80% as the training set and the remaining 20% as the testing set. Cross-validation is adopted in our experiments.

Table 2 parameters setting in gradient descent algorithm and TMFSF algorithm

algorithms	parameters	meanings of parameters	Values
the gradient descent algorithm	α -used in equations (12) (13) (17) (18)	the learning rating in gradient descent algorithm	0.15
TMFSF algorithm	f -used in TMFSF algorithm proposed in [1]	a harmonic factor introduced to avoid the case where the denominator is zero in the similarity calculation	Eachmoive: 1.1, Netflix: 1.0

Table 3 Raw dataset and preprocessed dataset of Eachmovie

Eachmovie	Raw dataset	Preprocessed dataset
User	74424	1781
Item	1623	1618
Reviews	2811983	511950
Sparsity	97.9%	82.2%

Table 4 Raw dataset and preprocessed dataset of Netflix

Netflix	Raw dataset	Preprocessed dataset
User	480189	1530
Item	17770	2575
Reviews	2081556480	2287692
Sparsity	75.6%	41.9%

4.2 Evaluation Metrics

Since our algorithms aim at prediction and recommendation, we choose five widely-used evaluation metrics to compare the performance of three algorithms (i.e., TSA, PARSA, and IARSA): one (i.e., MAE [25]) for prediction accuracy evaluation and the other four for common metrics of recommendation accuracy evaluation, namely, precision [26] (the proportion of positive suggestions in the user recommendation list), recall [27] (the percentage of positive suggestions in the user recommendation list in the total number of his/her positive suggestions.), diversity [28] (measures the diversity between the recommendations given to different users), and coverage [29] (measures the proportion of recommended products to the total products). Equations (19) defines the concept of MAE. The smaller the value of MAE, the more accurate the prediction is. Equations (20)-(23) define the concepts of Precision, Recall, Coverage, and Diversity, respectively. All the ranges of these concepts are in [0, 1]. The larger these values, the better the recommendation performance is.

$$MAE = \frac{\sum_{(m,c)\in\Omega^{test}} \left|\hat{r}_{mc} - r_{mc}\right|}{\left|\Omega^{test}\right|} \qquad (19)$$

where Ω^{test} is the test dataset, $\left|\Omega^{test}\right|$ is the size of the test dataset, r_{mc} is the actual rating in the test dataset, and $\hat{r}_{mc}$ is the predicted rating corresponding to r_{mc}.

$$Precision = \frac{\left|L_u \cap B_u\right|}{\left|L_u\right|} \qquad (20)$$

$$Recall = \frac{\left|L_u \cap B_u\right|}{\left|B_u\right|} \qquad (21)$$

$$Diversity = \frac{2}{n(n-1)} \sum_{u,v\in User, u\neq v} \frac{\left|L_u \cap L_v\right|}{\left|L_u\right|} \qquad (22)$$

$$Coverage = \frac{\left|\bigcup_{u\in User} L_u\right|}{n} \qquad (23)$$

where the L_u is the recommendation list of user *u*, B_u is the collection of items that user *u* has rated for positive in test dataset, *User* is the sets of users in the dataset, and *n* is the size of user set *User* .

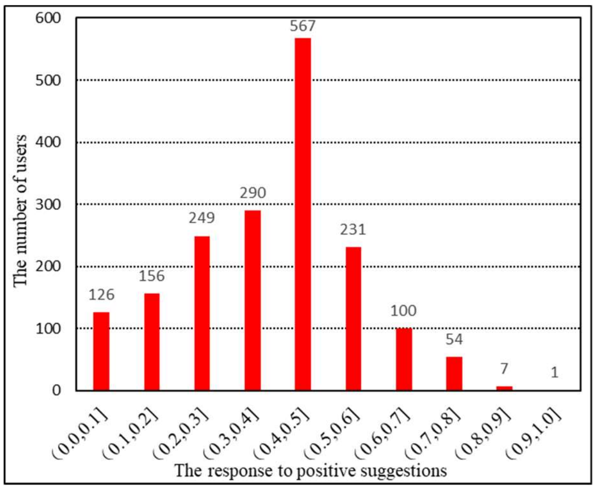
4.3 Statistical Analysis of Users' Responses to Positive/Negative suggestions

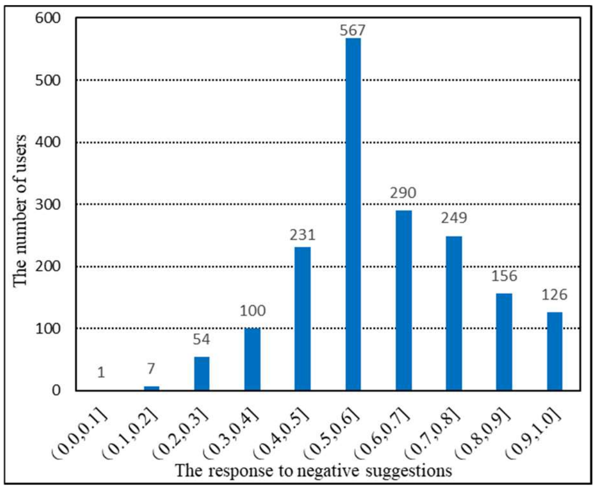
Recall Assumption 2, we assume that *the responses of users to positive and negative suggestions are different, and often different from one another*. To verify this assumption, in the first set of experiments, we first learn the responses of users to positive and negative suggestions on different kinds of similarity measures and different datasets (i.e., Eachmovie and Netflix) using formula (12) and (13). Next, we classify each user's responses to positive and negative suggestions into divided intervals such as (0, 0.1], (0.1, 0.2],… and (0.9, 1]. Finally, we compute the statistics of the user suggestions fell into each interval, which corresponds to the vertical axis values displayed in each interval.

Figure 3 shows the distribution of users under various response ranges on the Eachmovie dataset. From Figures 3(a) and (b), we can see that most people's responses to positive suggestions range from 0.2 to 0.5. In contrast, their responses to negative suggestions cluster around (0.5, 0.8]. Therefore, we can conclude that most people are more willing to believe in the negative suggestions from neighbors. When selecting the other two similarity measures (i.e., PCC and HySim), the results are similar, which can be seen from Figures 3(c) and (d) as well as Figures 3(e) and (f).

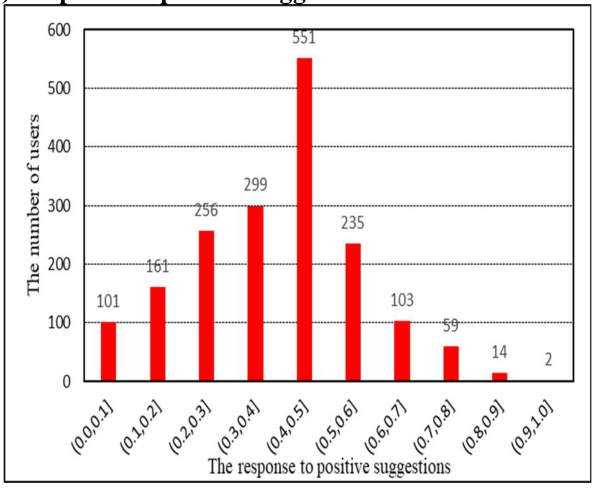
In order to analyze the influence of data sparseness problem and randomness, we perform statistics on the response ranges of users on the Netflix dataset (see Figure 4). Though Netflix is a large dataset with less sparseness, we can also find that the user distribution of the Netflix dataset is similar to that of the Eachmovie dataset, and the range of most people's responses to negative suggestions is (0.5, 0.8] whatever kind of similarity measure is adopted. Therefore, we can draw the following conclusion.

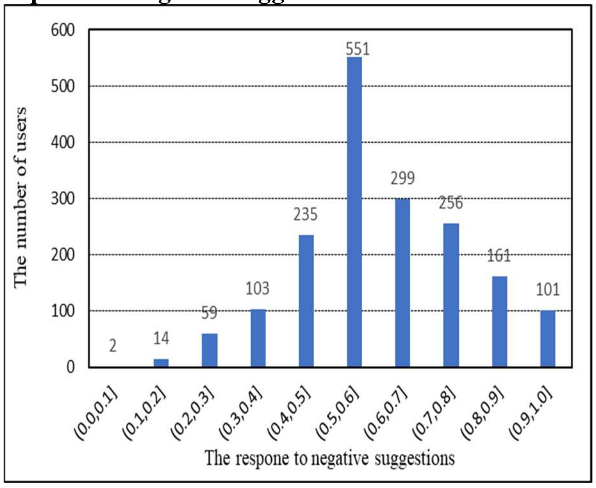
***Conclusion* 1** On both sparse and dense data sets, most users are more responsive to negative suggestions, which is consistent with the results of psychology.



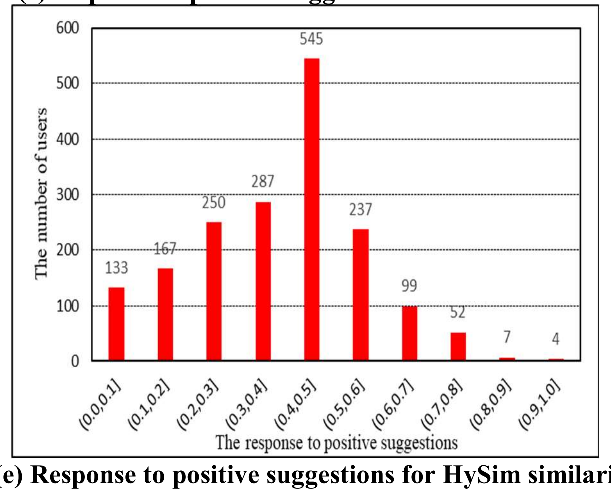


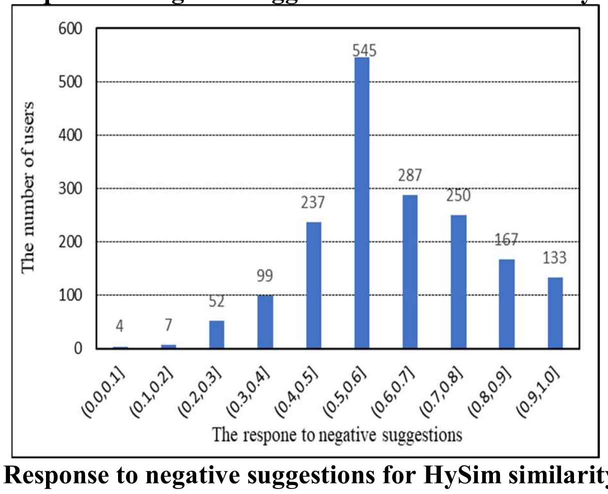
(a) Response to positive suggestions for TMFSF similarity (b) Response to negative suggestions for TMFSF similarity





(c)Response to positive suggestions for PCC similarity (d) Response to negative suggestions for PCC similarity



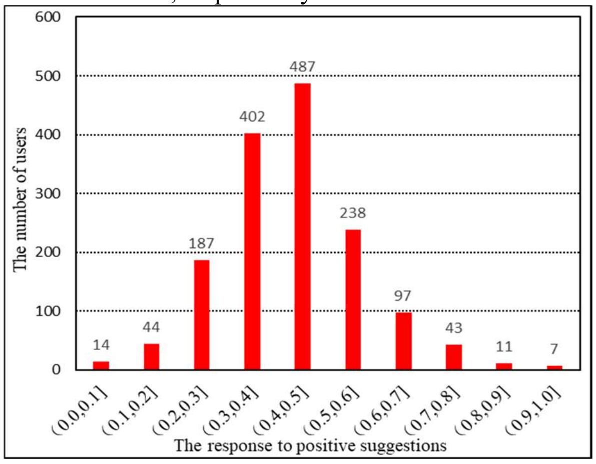


(e) Response to positive suggestions for HySim similarity (f) Response to negative suggestions for HySim similarity

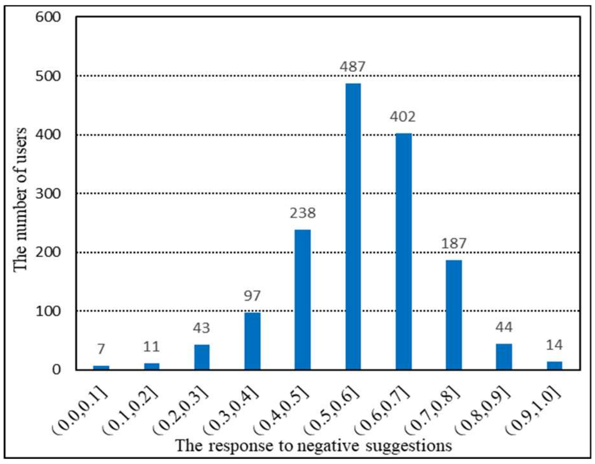
FIGURE 3. Statistical of users' responsive to positive/negative suggestions on three similarity measures over the Eachmovie dataset

4.4 Results and Analysis of Predicted Performance

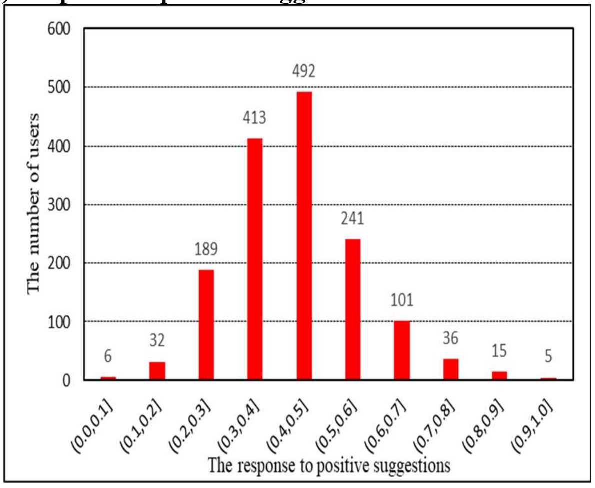
In the second set of experiments, we compare our PARSA algorithm with suggestion aggregation benchmarks (i.e., IARSA and TSA algorithms) on different similarity measures to evaluate the performance of prediction by using MAE defined in Equation (19) as the metric. As it is well-known that the number of selected nearest neighbors have an important impact on the quality of rating prediction, this set of experiments focus on analyzing how the performance of rating prediction varies with the increase of the numbers of neighbors when different suggestion aggregation methods are adopted. In this paper, the trend of the performance varying with the numbers of neighbors is more important than the point at which the performance can reach the optimal value. Therefore, this set of experiment compares the prediction performance of these algorithms when the number of neighbors is assigned with the values of 10, 30, 50, 70, 90, 110, 130, 150, and 170 over the Eachmovie dataset and Netflix dataset, respectively.



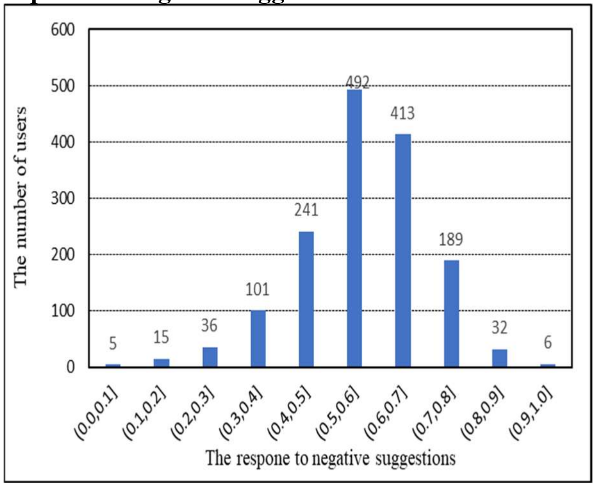
(a) Response to positive suggestions for TMFSF similarity



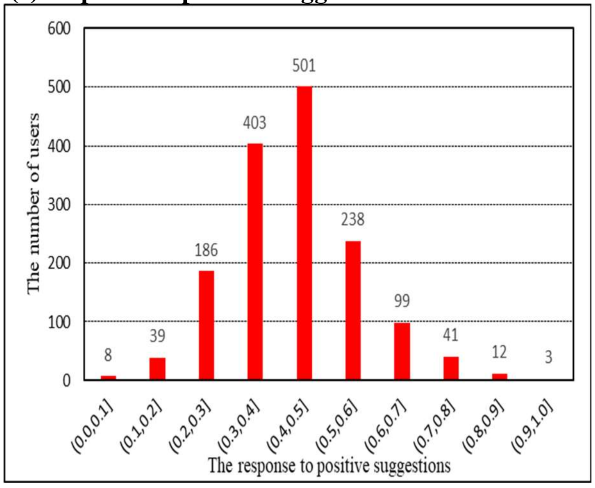
(b) Response to negative suggestions for TMFSF similarity



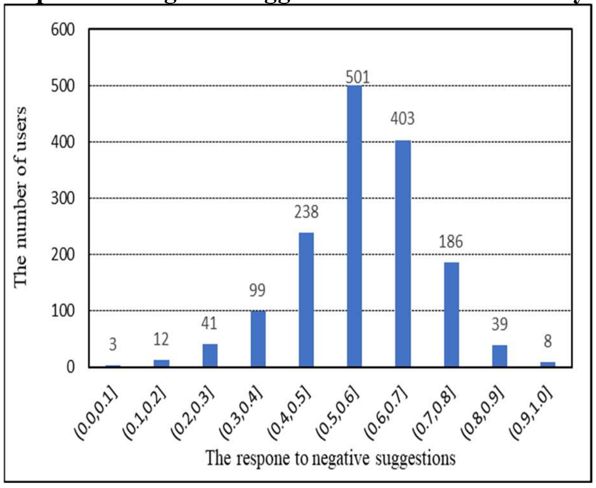
(c)Response to positive suggestions for PCC similarity



(d) Response to negative suggestions for PCC similarity



(e) Response to positive suggestions for HySim similarity

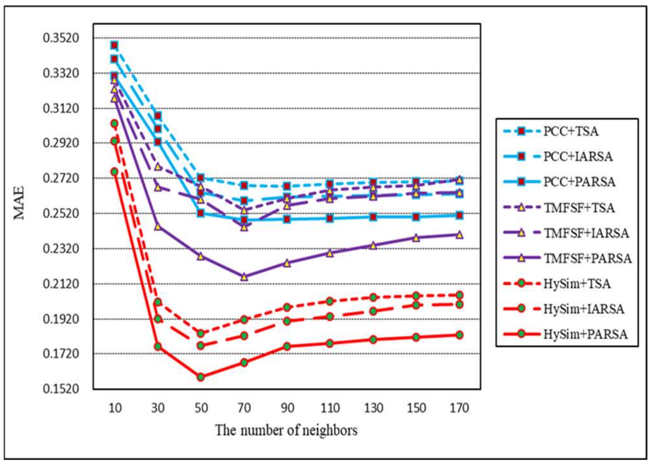


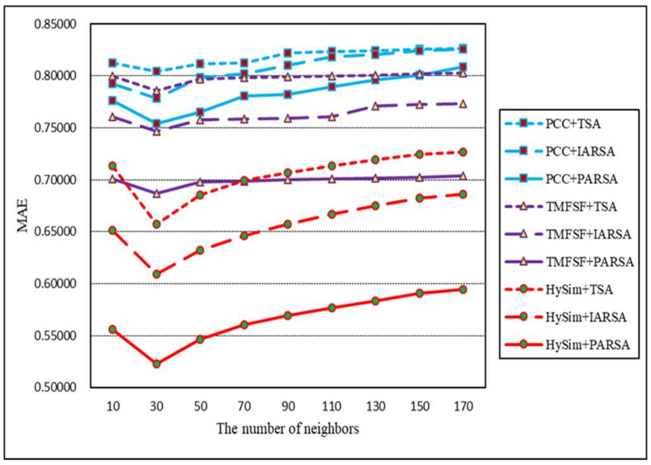
(f) Response to negative suggestions for HySim similarity

FIGURE 4. Statistical of users' responsive to positive/negative suggestions on three similarity measures over the Netflix dataset

Figure 5 shows the result of the prediction accuracy (i.e., MAE) on the two datasets. As for the influence of similarity measures and aggregation algorithms on the MAE values, we can see the following phenomena from the relationship of the curves. Generally, in both sub-figures, the sky blue curves of the PCC simulation measures are overall higher than that of the TMFSF measures, followed by the HySim measures. That is to say, the PCC simulation measures are worst, followed by TMFSF and HySim. For each simulation measure, the PARSA aggregation algorithm always achieves the lowest MAE (i.e., best performance), followed by IARSA and TSA.

In detail, the curves in Figures 5 (a) and (b) decrease and then increase with the increase of the number of neighbors on different similarity measures and datasets. From the result on Eachmovie shown in Figure 5(a), when applying the HySim similarity, we can see when the MAE values of all suggestion aggregation algorithms (i.e., TSA, IATSA, and PARSA) achieve the optimal values when the number of neighbors is 50. While we consider the other similarity measures (i.e., PCC and TMFSF), the MAE values of all suggestion aggregation algorithm (i.e., TSA, IATSA, and PARSA) achieve the optimal values when the number of neighbors is 70. The reason may be that the HySim similarity can solve the data sparseness problem, which can obtain adequate information from a much smaller number of neighbors. The result of the Netflix dataset is similar to that of Eachmovie (see Figure 5 (b)). The only difference is that the MAE values of all suggestion aggregation algorithm (i.e., TSA, IATSA, and PARSA) achieve the optimal values when the number of neighbors is 30 regardless of the similarity measures applied. This is because the Netflix dataset is less sparse, users may generally obtain enough information from a small number of neighbors.





(a) On Eachmovie dataset **(b) On Netflix dataset**

FIGURE 5. MAE among various algorithms on different similarity measures

From Figures 5 (a) and (b), we can also see that the MAE values of our algorithms (i.e., IARSA and PARSA) are lower than those of the TSA algorithms, regardless of what kind of similarity is adopted and the number of neighbors over both datasets. That is because our algorithms (i.e., IARSA and PARSA) consider users' asymmetry responses to negative and positive suggestions. Since the PARSA algorithm considered that all users giving different responses more realistically while the IARSA algorithm assumed that all users just give the same responses, the MAE values of the PARSA algorithm are lower than that of the IARSA algorithm no matter what kind of similarity is adopted. Besides, we can see that the MAE values over the Eachmovie dataset are much smaller than those over the Netflix dataset. That is because the values of ratings in Eachmovie dataset are 0, 0.2, 0.4, 0.6, 0.8, and 1.0, while the values of ratings in Netflix dataset are 0, 1, 2, 3, 4, and 5. From the above analysis, we can draw the following conclusions.

Conclusion **2** The PARSA algorithm proposed in this paper outperforms the IARSA and the TSA algorithms in rating prediction, no matter what kind of similarity measures and dataset are adopted. Moreover, the more advanced the similarity metrics, the more obvious the performance improvement is. The sparser the data, the smaller the improvement of prediction accuracy is.

4.5 Results and Analysis of Recommendation Performance

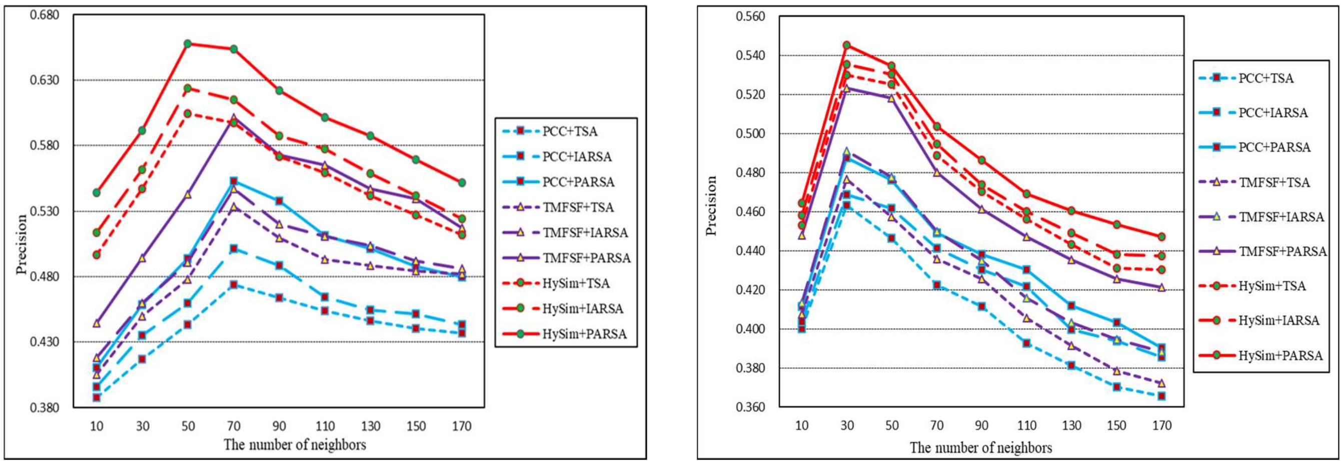
This section compares the recommendation quality among the TSA, IARSA, and PARSA algorithms on different similarity measures and different dataset (i.e., Eachmovie and Netflix). This paper uses precision, recall, coverage, and diversity as the evaluation criteria of recommendation. For similar reasons presented in rating prediction (see the first paragraph in 4.4), we assign the number of neighbors with values such as 10, 30, 50, 70, 90, 110, 130, 150, and 170 over the Eachmovie and Netflix dataset, respectively. We explain the experimental results in detail as follows.

4.5.1 Precision on Different Datasets

The comparison result of recommendation precision is shown in Figure 6. From Equation (20), we can see that the precision is different for each user. There are billions of empty ratings and millions of users in each dataset, which means on average millions of items can be recommended for each user. We may get good performance for some users, while bad performance for others. Therefore, in Figure 6, we show the average recommendation precisions of all the users. It shows that the improvement is very slight in precision and even for recall, diversity, and coverage. The larger and sparser the experimental data, the less improvement the performance is. Therefore, the improvement of (b) in Figures 6, 7, 8, and 9 is correspondingly lower than that of (a) in the same figure, respectively.

From Figure 6, we can see that the PARSA algorithm has the highest precision, while the TSA algorithm has the lowest precision on the same similarity, regardless of the number of neighbors over two datasets. Moreover, with the increasing number of neighbors, all curves of precision first increase and then decrease. For the Eachmovie dataset shown as Figure 6(a), when applying HySim similarity, the neighbors' number of all algorithms (i.e., TSA, IARSA, and PARSA) with optimal precision is 50. However, when selecting the other two similarity measures (i.e., PCC and TMFSF), the neighbors' number of all algorithms (i.e., TSA, IARSA, and PARSA) with optimal precision is 70. This is because the HySim similarity can find more similar (i.e., closer) neighbors than the other similarity measures. From Figure 6(b), we can see that the precision curves of the Netflix dataset are similar to that of the Eachmovie dataset. When the number of neighbors is 30, the precision of all algorithms (i.e., TSA, IARSA, and PARSA) achieves the best values on the same similarity. From Figure 6(a), we can see that the number of neighbors with optimal precision on HySim similarity is 50, while on the other two similarity measures, the number of neighbors with optimal precision is 70. From this figure, we can also see that the number of neighbors with optimal precision is similar than that of the Eachmovie dataset when the same similarity measure is applied. That is because the Netflix dataset is less sparse and facilitates more effective suggestions from fewer neighbors. However, the Eachmovie dataset with higher sparseness needs a larger number of neighbors to achieve a similar result. From the above analysis, we can draw the following conclusion.

Conclusion **3** When the same similarity measure is applied, the precision of PARSA algorithm is consistently the best one followed by IARSA and TSA algorithms, regardless of the number of neighbors and sparseness of the datasets. Moreover, the HySim similarity measure generally outperforms the TMFSF and PCC measures, no matter what kind of suggestion aggregation method is adopted. The sparser the dataset, the larger the improvement of recommendation precision is.



(a) On Eachmovie dataset **(b) On Netflix dataset**

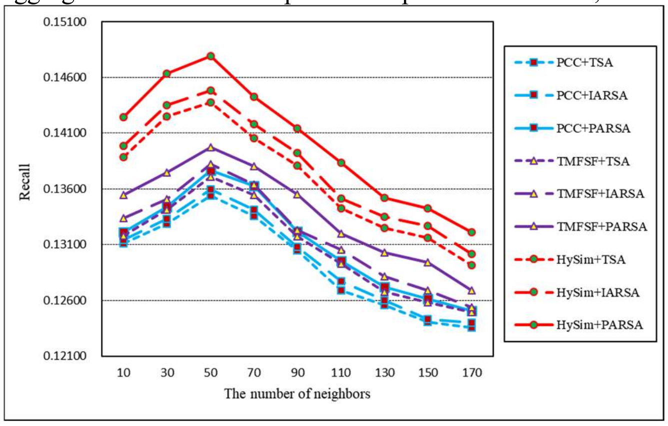
FIGURE 6. Precision comparison among various algorithms on different similarity measures

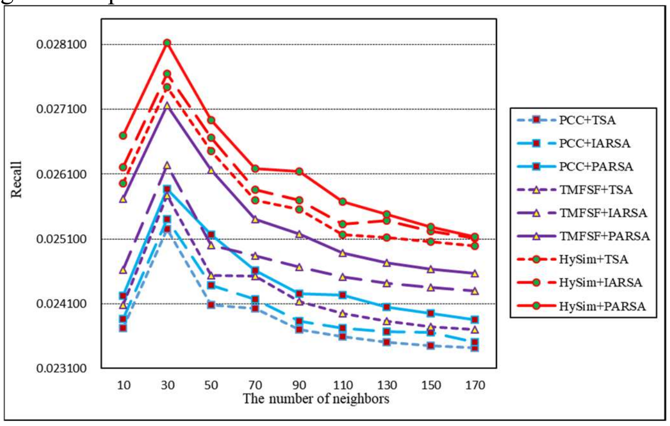
4.5.2 Recall on Different Datasets

The recall of all algorithms on Eachmovie and Netflix are shown in Figure 7. We can see that the PARSA algorithm has the best recall and the TSA algorithm has the worst recall, regardless of the number of neighbors when applying the same similarity measure.

From Figure 7(a), on the Eachmovie dataset, we can see that when the number of neighbors is 50, the recall of all algorithms (i.e., TSA, IARSA, and PARSA algorithms) achieves optimal values, no matter what kind of similarity is adopted. From Figure 7(b), on the Netflix dataset, we can see that the recall of all algorithms (i.e., TSA, IARSA, and PARSA) reaches their maxima when the number of neighbors is 30 values, no matter what kind of similarity is adopted. Comparing Figures 7(a) and (b), we can see that the trends of the recall curves on both datasets with different sparsity are similar, which first increase and then decrease. The difference is due to the fact that they get optimal recall values with different number of neighbors. Therefore, our algorithms proposed in this paper (i.e., IARSA and PARSA) provide a better recommendation performance on recall than the TSA algorithm on different similarity measures. According to the above results, we can draw the following conclusion, which is similar to conclusion 3.

***Conclusion* 4** When applying the same similarity measure, the recall of PARSA algorithm is consistently the best one followed by IARSA and TSA algorithms, regardless of the number of neighbors and sparseness of the datasets. Moreover, the performance of HySim similarity measure parallelly outperforms ones of TMFSF and PCC, no matter what kind of suggestion aggregation method is adopted. The sparser the data set, the larger the improvement of recommendation recall is.





(a) On Eachmovie dataset **(b) On Netflix dataset**

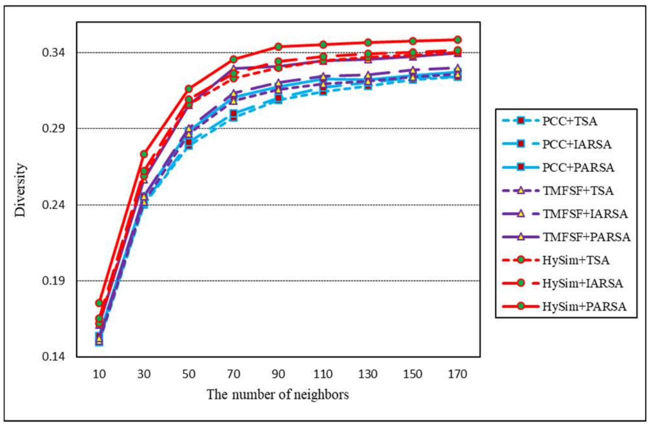
FIGURE 7. Recall comparison among various algorithms on different similarity measures

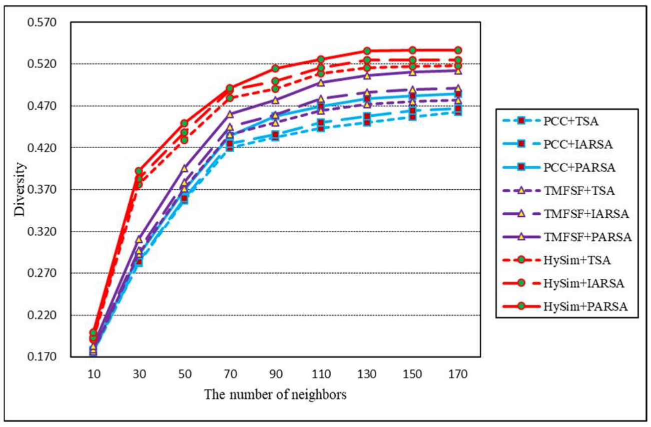
4.5.3 Diversity on Different Datasets

Figure 8 depicts the diversity results on Eachmovie and Netflix, which shows the PARSA algorithm has the best diversity, while the TSA algorithm has the worst one on the same similarity, regardless of the number of neighbors and sparseness of the datasets. Moreover, the diversity values of all suggestion aggregation algorithms (i.e., TSA, IARSA, and PARSA) increase constantly on both datasets with the increasing of the number of neighbors, no matter what kind of similarity is applied. This is because the kinds of recommendation items become diverse when the number of neighbors becomes larger.

From Figure 8(a), on Eachmovie dataset, we can see that the IARSA and PARSA algorithms achieve better diversity than the TSA algorithm when the same similarity is selected. When the number of neighbors is 90, the curves of diversity tend to be stable. From Figure 8(b), on the Netflix dataset, we can see that when the number of the neighbors is larger than 110, the diversity values of all algorithms converge. Comparing Figures 8(a) and (b), we can find the optimal diversity on the Netflix dataset is over 0.5 and higher than that on the Eachmovie dataset. This is because the latter dataset is sparser. From the above analysis, the diversity values of the IARSA and PARSA algorithms are higher than that of the TSA algorithm on the same similarity, regardless of the sparseness of the dataset. Therefore, we can get the following conclusion.

***Conclusion* 5** The algorithms proposed in this paper (i.e., IARSA and PARSA) converge to a larger diversity than TSA algorithm no matter how sparse the dataset is. The more advanced the similarity measure, the larger the diversity is.





(a) On Eachmovie dataset **(b) On Netflix dataset**

FIGURE 8. Diversity comparison among various algorithms on different similarity measures

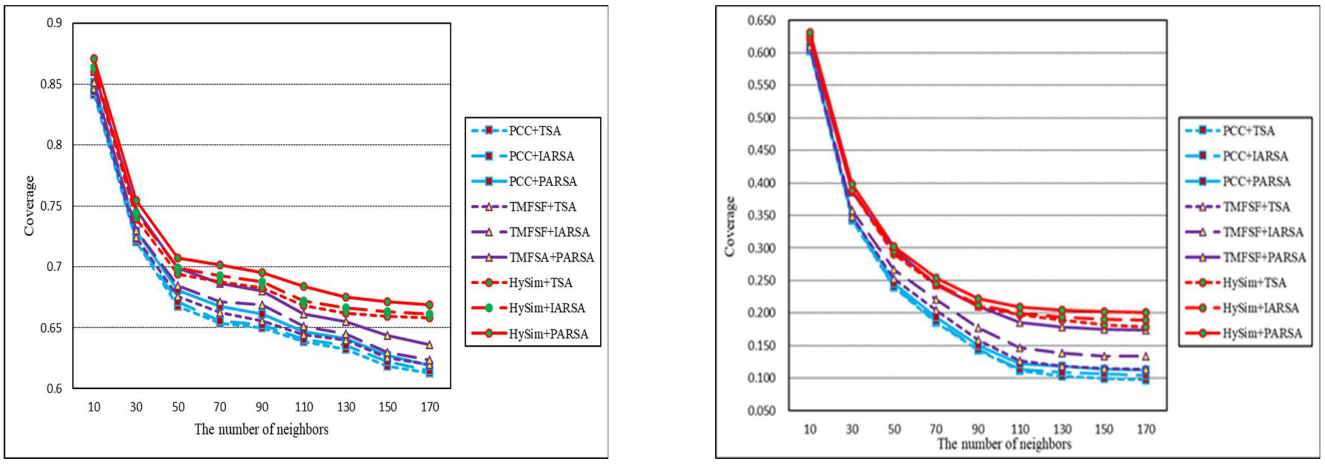
4.5.4 Coverage on Different Datasets

Finally, we verify the coverage of these algorithms on the Eachmovie and Netflix dataset. From Figure 9, we can see that all curves of coverage decrease constantly on two datasets with the increasing of the number of neighbors, and the coverage values of the IARSA and PARSA algorithms are larger than that of the TSA algorithm on the same similarity regardless of the number of neighbors and sparseness of the dataset. As we see from Figure 9(a), the curves of coverage are not stable, because the Eachmovie dataset is very sparse and the users provide less useful information. On the Netflix dataset as shown in Figure 9(b), when the number of neighbors is 110, the coverage curves of these algorithms become stable.

Comparing Figure 8 and Figure 9, we can see that the diversity increases while the coverage decreases with the increase of numbers of neighbors. That is because, for each object user u, if more number of users are selected as neighbors, more users' interest will be considered when generating the recommendation list. Therefore, u's recommendation list will be similar to that of much more users, hence the value of $\sum_{u,v\in User,u\neq v} |L_u \cap L_v|$ will increase and the value of $\left|\bigcup_{u\in User} L_u\right|$ will decrease. As both the value of $|L_u|$ and n is constant, the value of $\frac{2}{n(n-1)}\sum_{u,v\in User,u\neq v}\frac{|L_u \cap L_v|}{|L_u|}$ (equal to $\frac{2}{n(n-1)}\frac{\sum_{u,v\in User,u\neq v}|L_u \cap L_v|}{|L_u|}$) will also increase while the value of $Coverage = \frac{\left|\bigcup_{u\in User} L_u\right|}{n}$ will decrease. According to above results, we can draw the following conclusion.

Conclusion **6** The algorithms proposed in this paper (i.e., IARSA and PARSA) can converge to a larger value than the TSA algorithm no matter how sparse the dataset is. The more advanced the similarity measure, the larger the coverage is.

Based on the above analysis over various evaluation metrics, we can conclude that the PARSA and IARSA algorithms can enhance the performance of rating prediction with all the benchmark similarity measures (i.e., PCC, TMFSF, and HySim). On the basis of accurate prediction, the recommendation performances (i.e., precision, recall, diversity, and coverage) also achieve prominent values. That is to say, our algorithms (i.e., PARSA and IARSA algorithms) achieve good prediction performance as well as good recommendation performances on two real-life datasets with different sparseness, regardless of the similarity calculation method adopted. That is mainly because our algorithms proposed in this paper incorporates results from psychology (i.e., *people have different responses to positive and negative suggestions*) to learn individual's responses offline.



(a) On Eachmovie dataset **(b) On Netflix dataset**

FIGURE 9. Coverage comparison among various algorithms on different similarity measures.

4.5.5 COMPARISONS ABOUT THE TIME COMPLEXITIES OF RELATED ALGORITHMS

To further illustrate the performance of our algorithm, we summarize the time complexities of the similarity and suggestion aggregation algorithms on both offline and online aspects in Table 5. The time cost in offline aspects is used to learn users' characteristics and to discover the nearest neighbors and learn the responses to positive and negative suggestions, while the online time is spent in aggregating the suggestions of nearest neighbors. Since we select Top-N items to make recommendation according to prediction, we only summarize the time complexities of the algorithms for prediction in Table 5.

From Table 5, we can see that the time complexities of PCC and TMFSF similarity calculation methods are equal (i.e., $O(m^2c)$) in online and offline complexities, while the time complexities of HySim similarity calculation method is $O(m^2c^2)$. For suggestions aggregations, since the IARSA and PARSA algorithms need to learn the responses to negative and positive suggestions but TSA needs not do this, the offline time complexities of IARSA and PARSA algorithms are $O(m^2c)$ while the complexity of TSA is $O(1)$. Comparing the time complexities of suggestion aggregation algorithms and similarity, we can see that the offline time complexities of PARSA and IARSA algorithms are equal to those of the PCC and TMFSF similarity measures, and less than that of HySim similarity measure. From the above analysis, the following conclusion is drawn.

Conclusion 7 Our algorithms (i.e., PARSA and IARSA) need more time for offline training, though it is similar to the magnitude with some similarity measures. Even when applying the HySim similarity measure, the training time is acceptable. Moreover, after training, the online prediction and recommendation cost is similar to other algorithms while gaining high performance.

Table 5. The time complexities of related algorithms

Similarity calculation algorithms	offline	Suggestion Aggregation algorithm	offline	online
PCC	O(m^2c)	TSA	O(1)	O(mnc)
TMFSF	O(m^2c)	IARSA	O(m^2c)	O(mnc)
HySim	O(m^2c^2)	PARSA	O(m^2c)	O(mnc)

5 Conclusions

At present, user-based collaborative filtering algorithms are commonly employed in RSs. The traditional ones try to find the neighbors, and then aggregate their suggestions in the same way regardless of whether they are positive or negative. However, some researchers in psychology [20-24] showed that *people have different responses to positive and negative suggestions*. Therefore, this paper proposes two algorithms (i.e., IARSA and PARSA algorithms) considering users' different responses to positive and negative suggestions for aggregating neighbors' suggestions. First, this paper proposes the PARSA algorithm, which assumes that *the responses of users to positive and negative suggestions are different and may differ from one another*. Following that, this paper proposes the IARSA algorithm, which simply assumes that *all the users' responses to positive/negative suggestions are identical*. Obviously, the IARSA algorithm is an extreme case of PARSA algorithm, so that we take it as a baseline algorithm to analyze the lower bound of PARSA algorithm. Three sets of experiments are designed and implemented over two real-life datasets. Experimental results show that: (1) the assumption that *the responses of users to positive and negative suggestions are different and may differ from one another* is reasonable; (2) the PARSA algorithm performs best in rating prediction accuracy with different similarity measures and different dataset; (3) the PARSA algorithm performs best in recommendation over the evaluation criteria such as precision, recall, coverage, and diversity on different similarity measures and different dataset.

Though this paper shows that the PARSA and IARSA algorithms consider *people's different responses to positive and negative suggestions* and thus can achieve better performance according to various similarity metrics (PCC, TMFSF, and HySim). Neighbors' suggestions not only provide explicit ratings, but also imply other implicit information. Therefore, in our future work, we can proceed to mine users' hidden emotions and extend our model to further explore the impact of such information in the performance of prediction and recommendation. On the other hand, we may also consider fuzzy tool approaches [37]. Moreover, such methods can be applied to other social interaction applications that need to aggregate the suggestions of neighbors or friends, as well as reputation [31], reputation attacks defense [32-33], security risk evaluation [34], emergency resources allocation [35], alarm messages [36].

Acknowledgments

This paper is supported in part by the Natural Science Foundation of China (No. 71772107, 71403151, 61502281, 61433012, U1435215), Qingdao social science planning project (No. QDSKL1801138), the National Key R&D Plan (No. 2018YFC0831002), Humanity and Social Science Fund of the Ministry of Education (No. 18YJAZH136), the Key R&D Plan of Shandong Province (No.2018GGX101045), the Natural Science Foundation of Shandong Province (Nos. ZR2018BF013, ZR2013FM023), the Innovative Research Foundation of Qingdao (Grant No. 18-2-2-41-jch), Shandong Education Quality Improvement Plan for Postgraduate, the Leading talent development program of Shandong University of Science and Technology and Special funding for Taishan scholar construction project.

References

[1] L. Ye, C. Wu, and M. Li, "Collaborative Filtering Recommendation Based on Trust Model with Fused Similar Factor," in *MATEC Web Conf,* China, 2017, pp.00010.

[2] T. Yue, and H. Liang, "Rating prediction algorithm and recommendation based on user beahavior in IPTV," in *CECNet,* Yichang, China, 2012, pp.3373-3378.

[3] F. Pratto, and O. P. John, "Automatic vigilance: the attention-grabbing power of negative social information," *J PERS SOC PSYCHOL.*， vol. 61, No. 3, pp. 380-391, Oct. 1991.

[4] A. M. Frodi, M. E. Lamb, L. A. Leavitt, and W. L. Donovan, "Fathers' and mothers' responses to infant smiles and cries *," *INFANT BEHAV DEV.,* vol. 1, pp. 187-198, Jan. 1978.

[5] N. H. Anderson, "Cognitive algebra: integration theory applied to social attribution[1]," *ADV EXP SOC PSYCHOL.,* vol. 7, pp. 1-101, Oct. 1974.

[6] S. T. Fiske, "Attention and weight in person perception: the impact of negative and extreme behavior," *J PERS SOC PSYCHOL.,* vol. 38(6), pp. 889-906, Jun. 1980.

[7] D. L. Hamilton, and M. P. Zanna, “Differential weighting of favorable and unfavorable attributes in impressions of personality,” *J RES PERS.*, vol. 6, pp. 2-3, Dec. 1971.

[8] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, “Recommender systems survey,” *KNOWL-BASED SYST.*, vol. 46, pp. 109-132, July. 2013.

[9] Y. Shi, M. Larson, and A. Hanjalic, “Exploiting user similarity based on rated-item pools for improved user-based collaborative filtering,” *In RecSys '09,* NY, USA, 2009, pp.125-132.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *WWW '01, NY*, USA, 2001, pp.285-295.

[11] T. Srikanth, and M. Shashi, “A new similarity measure for user-based collaborative filtering in recommender systems，” *IJCT.,* vol. 14(9), pp. 6118, June. 2015.

[12] X. Wu, B. Cheng, and J. L. Chen, “Collaborative filtering service recommendation based on a novel similarity computation method，” *IEEE T SERV COMPUT.,* vol. 10(3), pp. 352-365, May-June. 2017.

[13] W. Li, H. Xu, M. Ji, Z. Xu, and H. Fang, “A hierarchy weighting similarity measure to improve user-based collaborative filtering algorithm,” in *ICCC,* Chengdu, China, 2017, pp.843-846.

[14] X. Zang, T. Liu, S. Qiao, W. Gao, J. Wang, and X. Sun, (2017). “A New Weighted Similarity Method Based on Neighborhood User Contributions for Collaborative Filtering,” in *DSC,* Changsha, China, 2017, pp.376-381.

[15] Suryakant, and T. Mahara, “A new similarity measure based on mean measure of divergence for collaborative filtering in sparse environment ☆，” *PCS.,* vol. 89, pp. 450-456, Aug. 2016.

[16] H. Liu, Z. Hu, A. Mian, H. Tian, and X. Zhu,“A new user similarity model to improve the accuracy of collaborative filtering ☆，” *KNOWL-BASED SYST.,* vol. 56, pp. 156-166, Jan. 2014.

[17] Y. Wang, J. Deng, J. Gao, and P. Zhang, “A hybrid user similarity model for collaborative filtering，” *INFORM SCIENCES.,* vol. 418-419, pp. 102-108, Dec. 2017.

[18] Q. Li, Z. Lin, and Z. Fei, “Similarity coefficient of collaborative filtering based on contribution of neighbors.,” in *ICMLC,* Jeju, South Korea, 2017, pp.226-232.

[19] Y. Hu, W. Shi, H. Li, and X. Hu, “Mitigating data sparsity using similarity reinforcement-enhanced collaborative filtering,” *TOIT.,* vol. 17(3), pp. 1-20, Jun. 2017.

[20] W. P. Lee, and C. Y. Ma, “Enhancing collaborative recommendation performance by combining user preference and trust-distrust propagation in social networks,” *KNOWL-BASED SYST.,* vol. 106, pp. 125-134, Aug. 2016.

[21] M. M. Azadjalal, P. Moradi, A. Abdollahpouri, and M. Jalili, “A trust-aware recommendation method based on pareto dominance and confidence concepts,” *KNOWL-BASED SYST.,* vol. 116, pp. 130-143, Jan. 2017.

[22] L. Li, Y. X. DONG, C. H. ZHAO, and W. J. CHENG, “Collaborative Filtering Recommendation Algorithm Combined with User Trust,” *J CHIN COMPUTE Sys.,* vol. 38(5), pp. 951-955, May. 2017.

[23] W. Zeng, and M. S. Shang, “Effects of negative ratings on personalized recommendation,” in *ICCSE,* Hefei, China, 2010, pp.375 – 379.

[24] E. Frolov, and I. Oseledets, “Fifty shades of ratings: how to benefit from a negative feedback in top-n recommendations tasks,” *in RecSys,* Boston, MA, USA, 2016, pp.91-98.

[25] W. Wu, J. Zhang, C. Zhang, F. Meng, Z. Zhang, and Y. Zhang Y, “Improving performance of tensor-based context-aware recommenders using bias tensor factorization with context feature auto-encoding”. *KNOWL-BASED SYST.,* vol. 128, no. 2017, pp. 71-77, April. 2017.

[26] X. Y. XU, J. M. LIU, “Collaborative Filtering Recommendation Algorithm Based on Multi-level Item Similarity,” *COMPUT SCI.,* vol. 43(10), pp. 262-265+291, Oct. 2016.

[27] P. Xia, J. Shuai, and D. O. Automation, “Research of Collaborative Filtering Recommendation Algorithm for Short Text,” JCC., vol. 02(14), pp. 59-66, Jan. 2014.

[28] C. N. Ziegler, S. M. Mcnee, J. A. Konstan, and G. Lausen, “Improving recommendation lists through topic diversification.” in *WWW,* Chiba, Japan, 2005, pp.22-32.

[29] M. Ge, C. Delgado-Battenfeld, and D. Jannach, “Beyond accuracy: evaluating recommender systems by coverage and serendipity,” in *RecSys,* Barcelona, Spain, 2010, pp.257-260.

[30] P. T. Costa, A. Terracciano, and R. R. Mccrae, “Gender differences in personality traits across cultures: robust and surprising findings,” in Journal of Personality & Social Psychology, 81(2):322, 2001.

[31] D. K. W. Chiu, H. F. Leung and K. M. Lam, “On the making of service recommendations: An action theory based on utility, reputation, and risk attitude,” *EXPERT SYST APPL.,* 36(2P2): 3293-3301, 2008.

[32] S. Ji, H. Ma, S. Zhang, H. Leung, D.K.W. Chiu, C. Zhang and X. Fang, "A pre-evolutionary advisor list generation strategy for robust defensing reputation attacks." *KNOWL-BASED SYST,* 103:1-18, April. 2016.

[33] S. Ji, H. Ma, Y. Liang, H. Leung, and C. Zhang, "Correction to: A whitelist and blacklist-based co-evolutionary strategy for defensing against multifarious trust attacks." *APPL INTELL,* to be published. DOI:10.1007/s10489-018-1195-1

[34] P.C.K. Hung, D.K.W. Chiu, W.W. Fung, W. Cheung, R. Wong, S.P.M. Choi, E. Kafeza, J. Kwok. J. Pun, and V.S.Y. Cheng, “End-to-End Privacy Control in Service Outsourcing of Human Intensive Processes: A Multi-layered Web Service Integration Approach,” *INFORM SYST FRONT.,* 9(1):85-101, March 2007.

[35] D.K.W. Chiu, D.T.T. Lin, E. Kafeza, M. Wang, H. Hu, H. Hu, and Y. Zhuang, “Alert based disaster notification and resource allocation,” *INFORM SYST FRONT.,* vol. 12, no.1, pp. 29-47. 2010.

[36] S. Meng, D.K.W. Chiu, E. Kafeza, W. Liu, and Q. Li, “Automated management of assets based on RFID triggered alarm messages,” *INFORM SYST FRONT.*, vol.12, no.5, pp.563-578, Nov.2010.

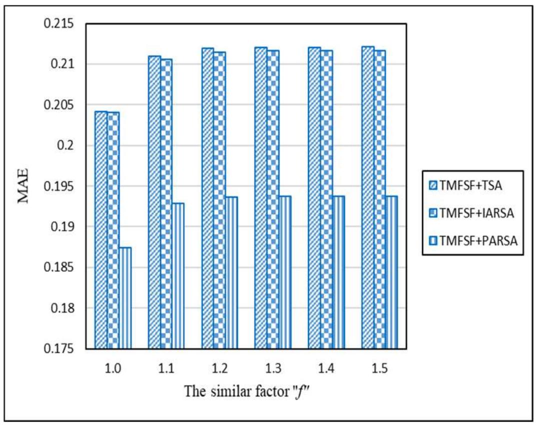
[37] R. Yera and L. Martínez, "Fuzzy Tools in Recommender Systems: A Survey ", International Journal of Computational Intelligence Systems, vol. 10, issue 1, pp. 776 - 803, 2017.

[38] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang. Recommender system application developments: a survey. Decision Support Systems, 74:12-32, 2015

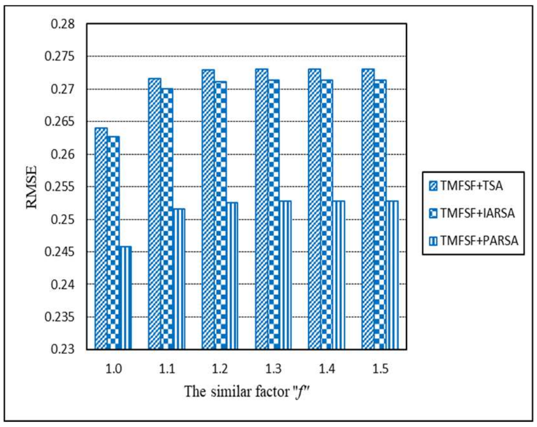
Appendix

In order to get optimal TMFSF similarity, we use an interpolation method to get the optimal value of f over two datasets, which is assigned a value of 1.0, 1.1, 1.2, 1.3, 1.4, and 1.5, respectively. We employ MAE and RMSE as evaluation criteria in this experiment. The smaller the values of MAE and RMSE, the more accurate the prediction is.

For the Eachmovie dataset, we obtain the MAE and RMSE values as shown in Figures A1(a) and (b), when f ranges from 1.0 to 1.5. Then we can see that, when f is 1.0, the values of MAE and RMSE achieve their minimums respectively. That means, the optimal value of f is 1.0. Similarly, as shown Figures A2(a) and (b), we obtain the optimal MAE and RMSE values on the Netflix dataset when f is 1.1. Therefore, we can conclude that the optimal values of f on the two datasets are 1.0 and 1.1, respectively.

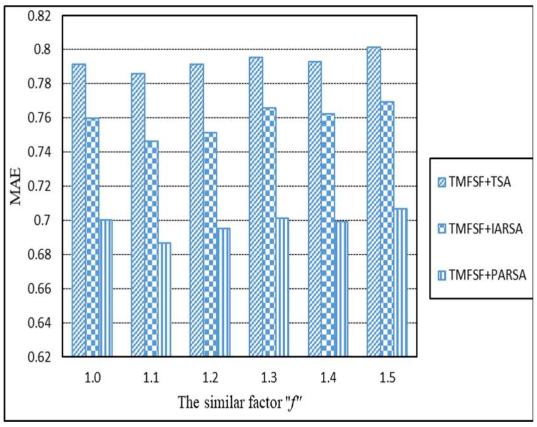


(a) MAE

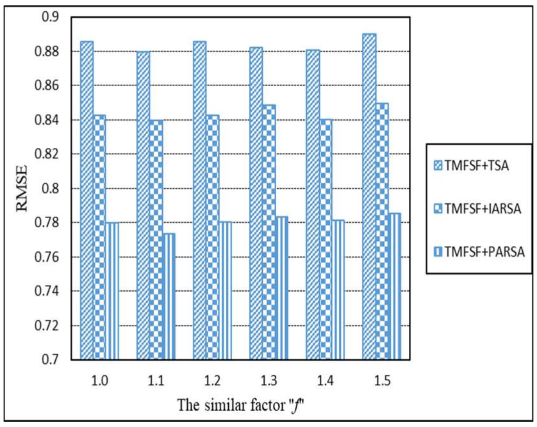


(b)RMSE

FIGURE A1. **The value of MAE and RMSE on Eachmovie**



(a) MAE



(b)RMSE

FIGURE A2. **The value of MAE and RMSE on Netflix**