In recent years, the number of web services grows explosively. With a large amount of information resources, it is difficult for users to quickly find the services they need. Thus, the design of an effective web service recommendation method has become the key factor to satisfy the requirements of users. However, traditional recommendation methods often tend to pay more attention to the accuracy of the results but ignore the diversity, which may lead to redundancy and overfitting, thus reducing the satisfaction of users. Considering these drawbacks, a novel method called DivMTID is proposed to improve the effectiveness by achieving accurate and diversified recommendations. First, we utilize users’ historical scores of web services to explore the users’ preferences. And we use the TF-IDF algorithm to calculate the weight vector of each web service. Second, we utilize cosine similarity to calculate the similarity between candidate web services and historical web services and we also forecast the ranking scores of candidate web services. At last, a diversification method is used to generate the top-K recommended list for users. And through a case study, we show that DivMTID is an effective, accurate, and diversified web service recommendation method.

1. Introduction

In recent years, web services have developed rapidly and are playing an increasingly important role in E-commerce and virtual reality applications. With the increasing of Internet web services’ numbers, people have more access to Internet information anytime and anywhere. However, people need to deal with a large amount of information resources, which makes it difficult for people to quickly find valuable services which they are interested in. In other words, the selection process is complicated in the age of big data [1–4]. Therefore, precise recommendation of web services is the key issue in service computing. As we all know, the recommender system has been widely used in many applications, such as https://Amazon.com, https://TiVo.com, and https://Netflix.com [5]. And web service recommendation is a process of actively identifying suitable web services and recommending them to users. The most common method is traditional collaborative filtering [6].

As we all know, collaborative filtering usually explores users’ preferences basing on users’ historical usage records and then recommends the most appropriate service items to users automatically [7]. However, this method mainly focuses on improving the accuracy of recommendation, which may lead to the redundancy of services in a limited list of top-K recommendations. Worse, the recommendation results may reduce users’ satisfaction and are not conducive to exploring users’ potential preferences for other services. For example, it is assumed that there is a certain service category with similar or related functions that match the interests of users and has better quality of services than other categories of services. Ordinary service recommendation
methods may only recommend this category of services to users in the final recommended list, but from users’ points of view, recommendation services with similar functions are redundant, and this phenomenon is called overfitting. Accordingly, the recommender system should also pay attention to the diversity of service recommendations while ensuring a high accuracy of recommendation results. In this manner, other categories of services that users may be interested in can be included in the top-K recommended list [3, 8].

Fortunately, diversification methods can not only avoid redundancy but also expand the range of users’ choices, which is beneficial to avoid the uncertainty in the prediction of users’ preferences [9]. However, there is a trade-off between accuracy and diversity [10] because high accuracy may often be obtained by safely recommending users the most popular and appropriate items, which can clearly lead to the reduction of diversity. And on the contrary, higher diversity can be achieved by trying to uncover and recommend highly idiosyncratic or personalized items with less data for each user, which will be more difficult to predict. And it may lead to the decrease of recommendation accuracy. Therefore, it is crucial for recommender systems to provide an optimal list of recommendations that takes into account both accuracy and diversity and to keep a balance between them [11–14]. This is also the main research direction of this paper. The main contributions of this paper are listed below:

(i) A new web service recommendation method which pays attention to both accuracy and diversity is proposed

(ii) Providing users with the list of top-K service recommendations, our method improves the disadvantages of traditional service recommendation methods and effectively solves the problem of overfitting

(iii) Our method weighs well the double indicators of accuracy and diversity in order to achieve the best recommendation effect and improve users’ satisfaction

The remainder of this paper is organized as follows. Section 2 describes a scenario of web service recommendation, and based on that, the main motivation and research content of this paper are further described. Section 3 presents the framework and specific steps of the proposed web service recommendation method (named DivMTID). Section 4 introduces a case study, where a specific case is solved by DivMTID. Section 5 summarizes this paper, draws conclusions, and expounds future work.

2. Research Scenario and Motivation

In this section, the research scenario and motivation of this paper are described. All the work we have done is based on the research scenario and motivation.

2.1. Research Scenario. Here, we use Figure 1 to describe the research scenario in this paper. Suppose that a website has many different types of modules (entertainment, military, sports, life, finance, cars, games, films, shopping, etc.), and there are many different web services under each module. Assume that there are M web services used by a user under all modules, and they are recorded as WS1, WS2, . . . , WS M. For each module, they are recorded as WSu1, WSu2, . . . , WSu x (x is a variable). Meanwhile, there are N candidate web services recorded as WS1, WS2, . . . , WSN in the set of candidate services. And each web service is described by the Web Service Description Language (which is called the WSDL document). In order to describe it exhaustively, the symbols mentioned in this paper and their meanings are shown in Table 1.

2.2. Motivation. In this subsection, we utilize the example in Figure 2 to demonstrate the motivation of our proposal. It is assumed that the recommender system intends to recommend a list of web services to a user. In this condition, to recommend appropriate web services to the user, the similarity between historical web services and candidate web services should be calculated first. And then the system generates the top-K recommended list to the user. However, in the process of similarity calculation and recommendation calculation, we will face the following challenges:

When calculating the similarity between historical web services and candidate web services, it is necessary to establish the relationship between historical records and the candidate service set. However, an effective method to predict the relative score of candidate service objects and filter the candidate web services is needed.

As the diversity of the recommended list is frequently neglected, the web services in the list may be similar to each other, which may lead to overfitting and failure to explore users’ potential preferences and finally reduce the users’ satisfaction.

Considering the above issues, a novel web service recommendation method named DivMTID is proposed, which will achieve the accuracy and diversity of recommendation results, and it will be presented in detail in the following sections.

3. A Diversified Service Recommendation Method Based on TF-IDF

Under the research scenario of Section 2, this paper proposes a new web service recommendation method named DivMTID, which is based on the TF-IDF algorithm. It utilizes cosine similarity and combines WSDL documents to calculate the ranking score of each candidate service and then uses the diversity algorithm to select the best web services from candidate services to set the top-K service recommended list. Meanwhile, it takes into account the accuracy and diversity of recommendation results. Table 2 lists the basic framework of DivMTID, which includes four steps.

3.1. Step 1: Explore Users’ Preferences Approximately. In step 1, we first make an approximate positioning of users’
preferences according to users’ historical score records. In order to give more effectively personalized service recommendations, we need to figure out what users like and why they like it. In other words, using more effective preference representation methods may make recommendation algorithms exhibit higher performance. In most service recommendation methods, a user’s score on web service can only represent the user’s opinion on a service, but the user’s preferences cannot be fully determined by a score record. However, a user’s historical score records can be used to make an approximate positioning of the user’s preferences. We can use the rating scores of web services to establish correlations with metadata and break the common limitation of expressing preferences with only one score.

For example, under the scenario described in Section 2, if a user rated 5 for all the web services under the module of military and rated 2 for all the web services under the module of finance, then the recommender system should infer that the user prefers the military module and should recommend more candidate web services about the military than finance.

We can establish the correlation between history scores and the information of the metadata module in equation (1), which utilizes score records for web services to calculate a user’s preference degree for each module.

\[ M_j = \frac{\sum_{i=1}^{n_{\text{service-rated}}} \left( r_i \times n_{\text{service-used}} \right)}{n_{\text{service-used}}} \]  

In equation (1), \( M_j \) represents the degree of a user’s preference for module \( j \). \( r_i \) represents a user’s historical rating scores for the used web services. \( n_{\text{service-used}} \) represents the number of web services which rated \( r_i \) under the metadata module \( j \), and \( n_{\text{service-used}} \) represents the number of all the used web services by the user under the metadata module \( j \).

We can calculate the user’s preference degree for the modules in equation (1) and make an approximate positioning of the user’s preference. A threshold “\( a \)” is set here, and the module with a calculated result greater than “\( a \)” is defined as the user’s preference module. For example, in the scenario of Section 2, we set a threshold 3. After calculation, if the modules with a result greater than 3 are military, finance, cars, and shopping, then the top-\( K \) recommended list should mainly consist of web services under these modules, which means that the modules below the threshold are automatically filtered out. At last, we put all the web services belonging to the preference modules together to form a set \( P \). The above is the content of step 1, its pseudocode can be described by Algorithm 1.
3.2. Step 2: Calculate TF-IDF Weight Vectors of Web Services.

The task of step 1 in DivMTID is to determine users’ preferences, filtering out the web services under all modules with low history rating scores. It saves a lot of time for the subsequent recommendation algorithm to run. However, step 1 cannot exactly determine what kind of services users like, what characteristics the web services with high scores have, and how to select the best web services from so many candidate services. Step 2 is designed to solve these problems. It is assumed that step 1 filtered out \( L \) web services together.

As is mentioned, each web service in set \( P \) has a corresponding WSDL document, the same as candidate services. Then, all meaningful words in the WSDL documents of all services can form a corpus. After that, a well-known TF-IDF algorithm [8, 15] is used to assess the importance of words in the corpus for each web service. The importance is proportional to the number of times that words appear in the document and inversely proportional to the frequency of words appearing in the corpus. The explanation is as follows.

\[
\text{tf(tj, WSDLi) =} \frac{\text{freq}(tj, \text{WSDLi})}{|\text{WSDLi}|}. \tag{2}
\]

\( t_j \) represents the \( j \)-th word in the corpus and \( \text{WSDL}_i \) represents the WSDL document of the \( i \)-th web service. \( \text{Freq}(t_j, \text{WSDL}_i) \) is the frequency of a word appearing in a WSDL document. It can be described in

\[
\text{tf(tj, WSDLi) =} \frac{\text{freq}(tj, \text{WSDLi})}{|\text{WSDLi}|}. \tag{2}
\]
words that appear in the WSDL document, and \(||\text{WSDL}_i|||\) represents the number of words that appear in the WSDL document. So we can also get the equation

\[||\text{WSDL}_i||| = \sum_i f(t_j, \text{WSDL}_i).\]

idf represents the inverse document frequency. It is expressed by the ratio of the total number of all WSDL documents and the number of documents containing the word. We can calculate the logarithm of the quotient in

\[\text{idf}(t_j, \text{WSDL}_i) = \log_2 \left( \frac{|\text{WSDL}_i|}{|\{\text{WSDL}_j : t_j \in \text{WSDL}_j\}|} \right). \quad (3)\]

|WSDL| represents the total number of WSDL documents. And \(|\{\text{WSDL}_j : t_j \in \text{WSDL}_j\}|\) represents the total number of documents containing word \(t_j\).

We use TF-IDF to assess the importance of words in a corpus for a web service. If a word appears with high frequency in a WSDL document of a web service and appears with low frequency in other WSDL documents of services, then we suppose that the word has a high importance and representativeness for this web service, which can be used to classify and distinguish different services.

Since WSDL documents are generally short, this paper chooses to give higher weight to the idf value to normalize the inherent bias with

\[\omega = \text{tf}(t_j, \text{WSDL}_i) \ast \text{idf}(t_j, \text{WSDL}_i). \quad (4)\]

The common way to implement TF-IDF is to give the same weight to word frequency and the inverse document frequency. However, this paper gives higher weight to idf in order not only to standardize the inherent deviation of the tf measurement in short documents but also to better exclude the common words that frequently appear in web services in the corpus [16]. In this way, it can improve the classification and differentiation ability among web services and so improve the accuracy of a user’s preferences. \(\omega\) represents the calculation result. It is the TF-IDF weight of word \(t_j\) to web services, which means the importance of word \(t_j\) for web services. Utilizing all the words in the corpus, we calculate the TF-IDF weight of a web service by equation (4) to form the weight vector of a certain web service. We candidate the TF-IDF weight vectors of all web services in the set P, denoted as \(\omega_i, i = 1, 2, \cdots, u(M - L)\). Similarly, for all candidate web services, their TF-IDF weight vectors are also calculated and denoted as \(\omega_j, j = 1, 2, \cdots, N\). The above is
of each candidate web service (de-

calculation. The value of

accuracy and is intuitive enough to describe the similarity
to dimension volume; (2) cosine similarity has higher

different services is twofold: (1) cosine similarity is not limited
to dimension volume; (2) cosine similarity has higher

accuracy and is intuitive enough to describe the similarity
calculation. The value of CosSim_{ij} is calculated in

\[
\text{CosSim}_{ij} = \cos \left( \omega_i, \omega_j \right) = \frac{\omega_i \cdot \omega_j}{|\omega_i| \times |\omega_j|}.
\]  

(5)

In equation (5), |\omega_i| and |\omega_j| is the Euclidean length of
the weight vector \omega_i and \omega_j. Besides, \omega_i \cdot \omega_j is their dot
product. Cosine similarity can be used to effectively evaluate the
similarity degree between two vectors, so we can also evaluate the
similarity between two web services. After that, we calculate
CosSim_{ij} of candidate web services by combining each
candidate web service and every web service in set P to get
their value of cosine similarity in order.

We can get the similarity between the candidate web
services and a user’s history web services according to the
value of CosSim_{ij}, so that we can calculate the ranking score
of each candidate web service (defined as Score_j) in

\[
\text{Score}_j = \lambda \sum_{i=1}^{M-L} r_i \times \text{CosSim}_{ij}.
\]  

(6)

In equation (6), \lambda is the parameter and r_i is users’ rating
on history web services. The aim of multiplying users’ rating
and the value of CosSim_{ij} is to give CosSim_{ij} a different weight.
After that, we carry on the accumulation, and we can obtain the
ranking score of each candidate service. At last, we sort
the score and set a threshold “b.” All the candidate web
services with a ranking score greater than “b” form a set Y.
And the web services in the top-K recommended list are
selected from this set. The above is the content of step 3; its
 pseudocode can be described by Algorithm 3.

3.4. Step 4: Create a Diversified Web Service Recommended List.
The purpose of setting threshold “b” is to ensure the
accuracy of the top-K recommended list, which is usually
recommended to the user by selecting the first K services
from high value to low value according to Score_j. Although
it ensures the high accuracy of the recommendation results,
it leads to the decrease of the diversity. Besides, it may cause
the problem of overfitting, which is not conducive to exploring
the potential preferences of users [18–21]. Therefore, we
need a method which can balance accuracy and diversity.

Input:
Y: set Y.
K: the length of recommended list
CosSim_{ij}: the similarity between service i and service j.

Output:
a diversified web service recommended list
1. f = |Y|/|f denotes the number of web services in the set Y
2. Sort(Y)
3. Create indexes for f web services
4. for i = 1 to CK/fdo/K < f
5. Form a list with K web services according to different
index numbers
6. Calculate list-diversity according to equation (7)
7. end for
8. return the list with the highest list-diversity value

Algorithm 4: Create a diversified web service recommended list.

<table>
<thead>
<tr>
<th>Table 3: The user’s history rating records.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entertainmen</td>
</tr>
<tr>
<td>Web_u1</td>
</tr>
<tr>
<td>Web_u2</td>
</tr>
<tr>
<td>Web_u3</td>
</tr>
<tr>
<td>Web_u4</td>
</tr>
<tr>
<td>Web_u5</td>
</tr>
<tr>
<td>Life</td>
</tr>
<tr>
<td>Web_u1</td>
</tr>
<tr>
<td>Web_u2</td>
</tr>
<tr>
<td>Web_u3</td>
</tr>
<tr>
<td>Web_u4</td>
</tr>
<tr>
<td>Web_u5</td>
</tr>
<tr>
<td>Games</td>
</tr>
<tr>
<td>Web_u1</td>
</tr>
<tr>
<td>Web_u2</td>
</tr>
<tr>
<td>Web_u3</td>
</tr>
<tr>
<td>Web_u4</td>
</tr>
<tr>
<td>Web_u5</td>
</tr>
</tbody>
</table>

The content of step 2; its pseudocode can be described by
Algorithm 2.
Step 4 provides a solution to how to make the recommendations more diverse while ensuring a high accuracy at the same time.

First, we set up an index of all candidate web services in the set \( Y \) and select \( K \) services according to different index numbers to form multiple recommended lists. Then, we define the diversity of web services in recommended lists as the list-diversity and each recommended list’s list-diversity is calculated in equation (7). Finally, we select the recommended list with the highest list-diversity value as the top-\( K \) recommended list to recommend to users.

\[
\text{List-diversity} = 1 - \frac{2}{N(N - 1)} \sum_{i,j \in Y, i \neq j} \cosSim_{ij}. \tag{7}
\]

The list-diversity means the average dissimilarity between each pair of web services in a recommended list. In equation (7), \( Y \) represents the set \( Y \) and \( N = |Y| \). \( \cosSim_{ij} \) represents the similarity of every two candidate web services in a list. The above is the content of step 4, its pseudocode can be described by Algorithm 4 (set the length of recommended list is \( K \)).

4. Case Study

In order to introduce the specific steps of DivMTID, and also to further illustrate the effectiveness of DivMTID, a case study is provided in this section.

Suppose that there are nine existing modules including entertainment, military, sports, life, finance, cars, games, films, and shopping. We assume that there are five different web services under each module and there are ten candidate web services. A user rated the web services he has used (rating values between 1 and 5, no rating value is recorded as null which equals to 0). Table 3 is the user’s history rating records.

Now, our work is providing the user with a top-\( K \) web service recommended list. We set the threshold “\( a \)” to 3.

4.1. Step 1: Explore Users’ Preferences Approximately. We use equation (1) to calculate the user’s preference degree for each module and make an approximate positioning of the user’s preference. After the calculation, we get the preference degree values \( M_j \), and the results are shown in Table 4.

Because we have set the threshold “\( a \)” to 3, the modules containing sports, life, and films whose \( M_j \) greater than 3 are the user’s approximate preference modules. The web services under these three modules form a set \( P \).

4.2. Step 2: Calculate TF-IDF Weight Vectors of Web Services. After approximately exploring the user’s preferences, we calculate the weight vectors of web services utilizing the WSDL documents of all services in the set \( P \) and the WSDL documents of all candidate services. Table 5 shows the WSDL documents of all web services in the set \( P \), and Table 6 shows the WSDL documents of all candidate services.

A corpus containing all meaningful words from the WSDL documents of all services in the set \( P \) and the WSDL documents of all candidate services is made (shooting, gymnastics, diving, marriage, cooking, Ang Lee, Hollywood, action movie, video, article, picture, long, short, fast, and...
slow). Then, we calculate the weight vector of each web service according to equation (4).

The sports module:
\[
\omega_{q1} = (1.35, 0, 0, 0, 0, 0, 0, 0, 0.54, 0, 0, 0.44, 0, 0.175),
\omega_{q2} = (1.35, 0, 0, 0, 0, 0, 0, 0, 0.54, 0, 0, 0.175, 0),
\omega_{q3} = (0, 4.68, 0, 0, 0, 0, 0, 0, 0.169, 0, 0, 0),
\omega_{q4} = (2.69, 0, 0, 0, 0, 0, 0, 0, 0.54, 0, 0.44, 0, 0),
\omega_{q5} = (0, 0.234, 0, 0, 0, 0, 0, 0, 0.54, 0, 0.44, 0, 0.175).
\]

The life module:
\[
\omega_{q1} = (0, 0, 0, 0, 4.68, 0, 0, 0, 0, 0.54, 0, 0.44, 0, 0),
\omega_{q2} = (0, 0, 0, 0, 4.68, 0, 0, 0, 0, 0.169, 0, 0, 0),
\omega_{q3} = (0, 0, 0, 1.75, 0, 0, 0, 0.54, 0, 0, 0, 1.35, 0.175),
\omega_{q4} = (0, 0, 0, 0, 5.24, 0, 0, 0, 0.84, 0, 0, 0),
\omega_{q5} = (0, 0, 0, 5.24, 0, 0, 0, 0.54, 0, 0, 0, 0).
\]

The films module:
\[
\omega_{q1} = (0, 0, 0, 0, 7.01, 0, 0, 0.54, 0, 0, 0, 0, 0),
\omega_{q2} = (0, 0, 0, 0, 2.31, 0, 0, 0.72, 0.58, 0, 0, 0),
\omega_{q3} = (0, 0, 0, 0, 2.34, 0, 0, 0.54, 0, 0.44, 0, 1.75),
\omega_{q4} = (0, 0, 0, 0, 6.64, 0, 0, 1.69, 0, 0, 0),
\omega_{q5} = (0, 0, 0, 0, 1.75, 0.54, 0, 0, 0, 0, 1.35, 0.175).
\]

The candidate services:
\[
\omega_i = (0, 0, 0, 0, 3.09, 0, 0, 0.72, 0.58, 0, 0),
\omega_i = (0, 0, 0, 4.66, 0, 0, 0, 0, 1.12, 0, 0),
\omega_i = (1.35, 0, 0, 0, 0, 0, 0, 0.54, 0, 0, 1.35, 0),
\omega_i = (0, 0, 0, 2.34, 0, 0, 0, 0.54, 0, 0.44, 0, 1.75),
\omega_i = (0, 0, 0, 14.36, 0, 0, 0, 0, 0, 1.12, 0, 0),
\omega_i = (0, 0, 0, 3.11, 0, 0, 0, 0, 0, 0.72, 0.58, 0, 0),
\omega_i = (0, 0, 0, 0, 0, 3.5, 0, 0, 0, 0, 1.69, 0, 0),
\omega_i = (0, 0, 0, 0, 0, 1.75, 0.54, 0, 0, 0, 1.35, 0),
\omega_i = (0, 0, 0, 0, 0, 6.64, 0, 1.09, 0, 0, 0),
\omega_i = (2.69, 0, 0, 0, 0, 0, 0, 0.54, 0, 0.44, 0, 0).
\]

4.3. Step 3: Predict the Ranking Scores of Candidate Services. According to equation (5), the cosine similarity of the TF-IDF weight vectors is calculated sequentially for each candidate web service with each historically used web service in the set P, and the CosSim_{ij} value of each candidate service is obtained. Then, the ranking score of each candidate web service is calculated by equation (6), and it is shown in Table 7.

<table>
<thead>
<tr>
<th>Web</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web3, Web8, Web4</td>
<td>15.685</td>
</tr>
<tr>
<td>Web2, Web8, Web10</td>
<td>13.166</td>
</tr>
<tr>
<td>Web8, Web4, Web1</td>
<td>11.253</td>
</tr>
<tr>
<td>Web2, Web8, Web2</td>
<td>9.834</td>
</tr>
<tr>
<td>Web1</td>
<td>8.311</td>
</tr>
</tbody>
</table>

4.4. Step 4: Create a Diversified Web Service Recommended List. Suppose the value of K is 3. Then, we need to build a diversified recommended list containing 3 web services for the user. Step 4 establishes an index of all candidate web services in the set Y, and three web services are selected according to different index numbers to form multiple recommended lists. The list-diversity of each recommended list is calculated by equation (7). Finally, the recommended list with the highest list-diversity value is selected as the top-3 recommended list recommended to the user. The results are shown in Table 8.

<table>
<thead>
<tr>
<th>Recommended list</th>
<th>List-diversity</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web3, Web8, Web4</td>
<td>0.930</td>
<td>10</td>
</tr>
<tr>
<td>Web2, Web8, Web10</td>
<td>0.938</td>
<td>8</td>
</tr>
<tr>
<td>Web8, Web4, Web1</td>
<td>0.938</td>
<td>8</td>
</tr>
<tr>
<td>Web2, Web8, Web2</td>
<td>0.996</td>
<td>4</td>
</tr>
<tr>
<td>Web3, Web8, Web1</td>
<td>0.993</td>
<td>7</td>
</tr>
<tr>
<td>Web2, Web8, Web1</td>
<td>1.000</td>
<td>1</td>
</tr>
<tr>
<td>Web8, Web4, Web1</td>
<td>0.994</td>
<td>6</td>
</tr>
<tr>
<td>Web2, Web8, Web1</td>
<td>1.000</td>
<td>1</td>
</tr>
<tr>
<td>Web4, Web8, Web1</td>
<td>0.997</td>
<td>3</td>
</tr>
</tbody>
</table>

5. Conclusions and Future Work
This paper presents a new web service recommendation method called DivMTID. This method first uses users’ history ratings about web services to approximately explore users’ preferences. Second, it uses the TF-IDF algorithm to calculate the weight vectors of each web service. Third, it uses the cosine similarity to calculate the similarity between
candidate web services and historical services in order to estimate the ranking scores of candidate services. Finally, list-diversity is used to generate the top-K recommended list. DivMTID takes the accuracy and diversity index of web service recommendation into account and achieves high diversity of recommendation results while ensuring high accuracy. It comprehensively balances the influence of accuracy and diversity on recommendation results, avoiding the appearance of recommendation redundancy and solving the problem of overfitting. DivMTID is an effective, accurate, and diverse service recommendation method, which is worth popularizing and using.

However, the specific influence of this method in many aspects of the recommender system is not measured. Therefore, in the future work, we will do more experiments about this method’s influence on each index of the recommender system.

In addition, we will take the time and space factors into consideration to improve the algorithm from many aspects, such as privacy [22–25]. We will also further improve the performance and effectiveness of the algorithm [26–28] by combining some new approaches such as Blockchain and Edge Computing [29–32].

Data Availability
Our study does not need any data set. And all the data used to support the findings of this study are included within the article.

Conflicts of Interest
The authors declare that they have no conflicts of interest.

Acknowledgments
This work was supported in part by the National Natural Science Foundation of China (No. 61872219), the Natural Science Foundation of Shandong Province (ZR2019MF001), and the Open Project of the State Key Laboratory of Novel Software Technology (No. KFKT2020B08).

References


