Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8822992, 10 pages
https://doi.org/10.1155/2020/8822992

Research Article

Service Recommendation with High Accuracy and Diversity

Shengqi Wu,[1] **Huaizhen Kou**,[1] **Chao Lv**,[2,3] **Wanli Huang**,[1] **Lianyong Qi**,[1,4] **and Hao Wang**[5]

[1]*School of Computer Science, Qufu Normal University, Rizhao, China*
[2]*China Telecom Smart Home Competence Center, China*
[3]*E-Surfing Smart Home Technology Co., Ltd., China*
[4]*State Key Laboratory for Novel Software Technology, Nanjing, China*
[5]*Department of Computer Science, Norwegian University of Science and Technology, Gjøvik, Norway*

Correspondence should be addressed to Wanli Huang; wanlih1983@126.com and Hao Wang; hawa@ntnu.no

Received 4 August 2020; Revised 24 October 2020; Accepted 25 November 2020; Published 17 December 2020

Academic Editor: Yaguang Lin

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 $\mathrm{WS}_{\mathrm{u}1}$, $\mathrm{WS}_{\mathrm{u}2}$,…, $\mathrm{WS}_{\mathrm{u}M}$. For each module, they are recorded as $\mathrm{WS}_{\mathrm{u}1}$, $\mathrm{WS}_{\mathrm{u}2}$,…, $\mathrm{WS}_{\mathrm{u}x}$ (x is a variable). Meanwhile, there are N candidate web services recorded as WS_1, WS_2,…, WS_N 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'

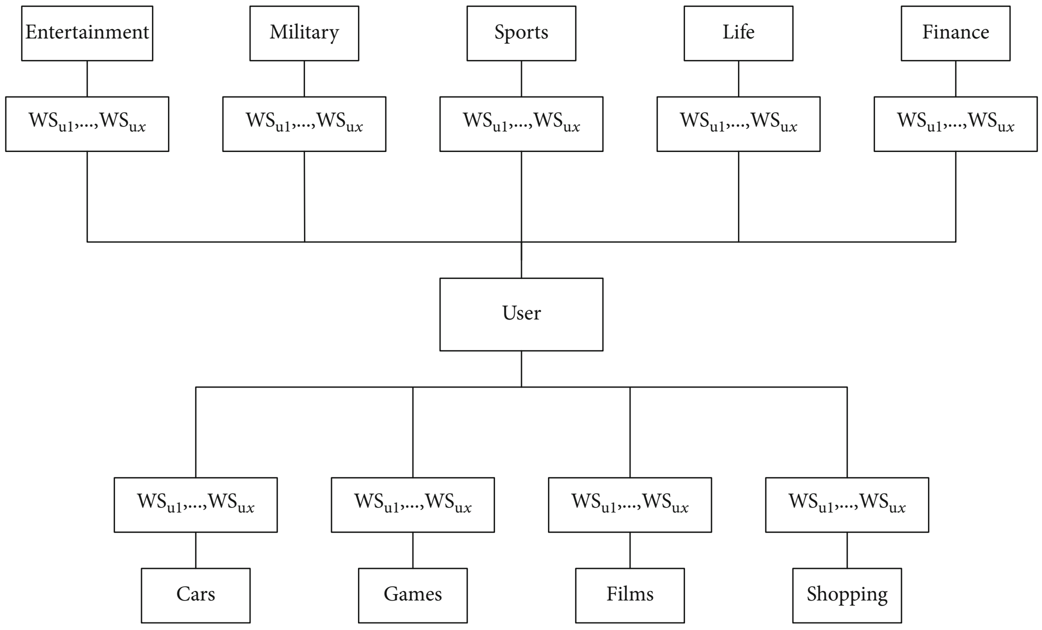

Figure 1: Research scenario.

Table 1: Symbols and their meanings.

Symbol	Meaning
WS_{ui}	Web service i used in a user's history
WS_j	Candidate web service j
$WSDL_i$	The WSDL documentation of web service i
M	The number of web services used in a user's history
N	The number of candidate web services
r_i	A user's rating of web service i used in history
M_j	Degree of a user's preference for module j
a, b	Threshold setting
t_j	The *j*-th word in the corpus
ω	The weight vector of web service
$CosSim_{i,j}$	The similarity level of web service i and web service j
$Score_j$	The predicted ranking score of candidate web service j

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.

$$M_j = \frac{\sum_{i=r_{\min}}^{r_{\max}} \left(r_i \times n_{r_{\text{service-rated}}}\right)}{n_{r_{\text{service-used}}}}. \tag{1}$$

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. $n_{r_{\text{service-rated}}}$ represents the number of web services which rated r_i under the metadata module j, and $n_{r_{\text{service-used}}}$ 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.

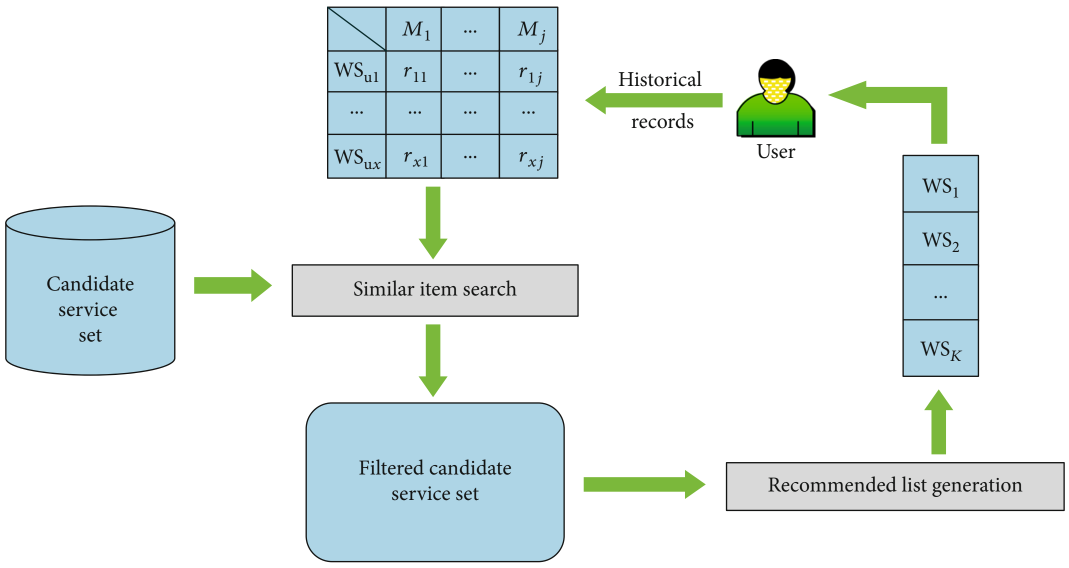

Figure 2: A motivating example.

Table 2: The basic framework of DivMTID.

Step 1: explore users' preferences approximately
By establishing the relationship between a user's history score records and the information of the metadata module, the preference degree of each module is calculated, and the user's preferences is approximately explored.
Step 2: calculate TF-IDF weight vectors of web services
Using the TF-IDF algorithm, the importance of words in the corpus to web services is calculated and finally represented by the TF-IDF weight vector in order to make a distinction among web services.
Step 3: predict the ranking scores of candidate services
The similarity between candidate web services and historically used web services is calculated by using cosine similarity, and the ranking score values of candidate web services are predicted.
Step 4: create a diversified web service recommended list
According to different index numbers, K different web services are selected to form multiple recommended lists. Then, it needs to calculate the list-diversity value of each list, and the list with the highest value becomes the web service recommended list that is finally recommended to the user.

```
Input:
  $\mathrm{WS}_{\mathrm{u}1}, \mathrm{WS}_{\mathrm{u}2}, \ldots, \mathrm{WS}_{\mathrm{u}M}$: web services used by a user.
  $r_1, r_2, \ldots, r_M$: the rating scores.
  $a$: the threshold.
Output:
  P: a set.
1.for $j = 1$ to $g$ do//assume there are $g$ modules
2.    $n_{r_{\text{service-used}}} = \mathrm{count}(\mathrm{WS}_{\mathrm{u}i})$
3.    for $r = r_{\min}$ to $r_{\max}$ do
4.        $n_{r_{\text{service-rated}}} = \mathrm{count}(\mathrm{WS}_{\mathrm{u}i})$
5.        $n_{r_{\text{service-rated}}} * r$
6.    end for
7.    Calculate $M_j$ according to equation (1)
8.    if $M_j \geq a$
9.        then add $\{\mathrm{WS}_{\mathrm{u}i} \mid \mathrm{WS}_{\mathrm{u}i} \in j\}$ to P
10.   end if
11.end for
12.return P
```

Algorithm 1: Explore users' preferences approximately.

3.2. Step 2: Calculate TF-IDF Weight Vectors of Web Services.

The task of step1 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.

tf represents the word frequency, indicating the frequency of a word appearing in a WSDL document. It can be described in

$$\mathrm{tf}\left(t_j, \mathrm{WSDL}_i\right) = \frac{\mathrm{freq}\left(t_j, \mathrm{WSDL}_i\right)}{\left|\mathrm{WSDL}_i\right|}. \tag{2}$$

t_j represents the j-th word in the corpus and WSDL_i represents the WSDL document of the i-th web service. Freq$(t_j$,

```
Input:
  WS_u1, WS_u2,..., WS_u(M-L): web services in set P.
  WS_1, WS_2,..., WS_N: candidate web services.
Output:
  $\omega_i$: weight vectors of services in set P.
  $\omega_j$: weight vectors of candidate services.
1. Count (|WSDL | )
2. for $i = \mathrm{u1}$ to $u(M - L)$ do
3.   for j = 1 to $n$ do //assume there are $n$ words in the corpus
4.     if $t_j \in \mathrm{WSDL}_i$
5.       then freq($t_j$, WSDL$_i$)
6.       Count |WSDL$_i$ |
7.       Count $|\{\mathrm{WSDL}_i : t_j \in \mathrm{WSDL}_i\}|$
8.       Calculate $\omega_i$ according to equation (4)
9.     end if
10.  end for
11. end for
12. $\omega_i = (\omega_1, \omega_2, \cdots, \omega_n)$
13. Calculate candidate services' TF-IDF weight vectors $\omega_j$
14. return $\omega_i$, $\omega_j$
```

Algorithm 2: Calculate TF-IDF weight vectors of web services.

$\mathrm{WSDL}_i)$ represents the number of times that t_j appears in the WSDL_i document, and $|\mathrm{WSDL}_i|$ represents the number of words that appear in the WSDL_i document. So we can also get the equation $|\mathrm{WSDL}_i| = \sum_j \mathrm{freq}(t_j, \mathrm{WSDL}_i)$.

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

$$\mathrm{idf}\left(t_j, \mathrm{WSDL}_i\right) = \log_2 \frac{|\mathrm{WSDL}|}{\left|\left\{\mathrm{WSDL}_i : t_j \in \mathrm{WSDL}_i\right\}\right|}. \quad (3)$$

$|\mathrm{WSDL}|$ represents the total number of WSDL documents. And $|\{\mathrm{WSDL}_i : t_j \in \mathrm{WSDL}_i\}|$ 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

$$\omega = \mathrm{tf}\left(t_j, \mathrm{WSDL}_i\right) * \mathrm{idf}^2\left(t_j, \mathrm{WSDL}_i\right). \quad (4)$$

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. ω 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 ω_i, $i = \mathrm{u1}, \mathrm{u2}, \cdots, \mathrm{u}(M - L)$. Similarly, for all candidate web services, their TF-IDF weight vectors are also calculated and denoted as ω_j, $j = 1, 2, \cdots, N$. The above is

```
Input:
  $\omega_i$,$\omega_j$: weight vectors of services.
  $r_i$: the rating scores.
  $b$: the threshold.
Output:
  $Y$: a set.
1. for $j = 1$ to $N$ do
2.   for $i = 1$ to $M - L$ do
3.     Calculate CosSim$_{i,j}$ according to equation (5)
4.     $r_i$ * CosSim$_{i,j}$
5.   end for
6.   Calculate Score$_j$ according to equation (6)
7.   if Score$_j > b$
8.     then add WS$_j$ to $Y$
9.   end if
10. end for
11. return $Y$
```

Algorithm 3: Predict the ranking scores of candidate services.

```
Input:
    Y: set Y.
    K: the length of recommended list
    CosSim_{i,j}: 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 j = 1 to $CK f$ do//$K < f$
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.

the content of step 2; its pseudocode can be described by Algorithm 2.

3.3. Step 3: Predict the Ranking Scores of Candidate Services. In order to evaluate the similarity between two web services, we use the TF-IDF weight vector of web services to calculate their cosine similarity [17] and define the similarity level between two web services as $\mathrm{CosSim}_{i,j}$. The reason that we choose cosine similarity to measure the distance between 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 $\mathrm{CosSim}_{i,j}$ is calculated in

$$\mathrm{CosSim}_{i,j} = \cos\left(\omega_i, \omega_j\right) = \frac{\omega_i \cdot \omega_j}{|\omega_i| \times |\omega_j|}. \quad (5)$$

In equation (5), $|\omega_i|$ and $|\omega_j|$ is the Euclidean length of the weight vector ω_i and ω_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 $\mathrm{CosSim}_{i,j}$ 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 $\mathrm{CosSim}_{i,j}$, so that we can calculate the ranking score of each candidate web service (defined as Score_j) in

$$\mathrm{Score}_j = \lambda \sum_{i=1}^{M\text{-}L} r_i \times \mathrm{CosSim}_{i,j}. \quad (6)$$

In equation (6), λ is the parameter and r_i is users' rating on history web services. The aim of multiplying users' rating and the value of$\mathrm{CosSim}_{i,j}$is to give$\mathrm{CosSim}_{i,j}$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.

Table 3: The user's history rating records.

	Entertainment	Military	Sports
Web_{u1}	Null	2	4
Web_{u2}	2	3	5
Web_{u3}	Null	1	3
Web_{u4}	3	2	5
Web_{u5}	4	1	3
	Life	Finance	Cars
Web_{u1}	4	1	Null
Web_{u2}	4	1	4
Web_{u3}	5	Null	3
Web_{u4}	3	5	2
Web_{u5}	3	Null	1
	Games	Films	Shopping
Web_{u1}	2	5	1
Web_{u2}	1	3	2
Web_{u3}	1	3	2
Web_{u4}	1	3	3
Web_{u5}	2	5	Null

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.

Table 4: The user's module preference degree.

	Entertainment	Military	Sports
M_j	1.8	1.8	4.0
	Life	Finance	Cars
M_j	3.8	1.4	2.0
	Games	Films	Shopping
M_j	1.4	3.8	1.6

Table 5: The WSDL documents of web services in set P.

	Sports	Life	Films
Web_{u1}	Shooting	Marriage	Ang Lee
	Video	Marriage	Ang Lee
	Long	Article	Ang Lee
	Slow	Long	Article
Web_{u2}	Shooting	Marriage	Hollywood
	Video		Article
	Short	Picture	Long
	Fast		
Web_{u3}	Gymnastics	Cooking	Ang Lee
		Video	Video
	Picture	Short	Long
		Fast	Slow
Web_{u4}	Shooting	Cooking	Action movie
	Shooting	Cooking	
	Article	Cooking	Picture
	Long	Picture	
Web_{u5}	Gymnastics	Cooking	Hollywood
	Video	Cooking	Video
	Long	Cooking	Short
	Slow	Article	Fast

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.

$$\text{List-diversity} = 1 - \frac{2}{N(N-1)} \sum_{i,j \in Y, i \neq j} \text{CosSim}_{i,j}. \qquad (7)$$

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 $N = |Y|$. $\text{CosSim}_{i,j}$ 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).

Table 6: The WSDL documents of candidate web services.

	Candidate services
Web_1	Ang Lee, article, long
Web_2	Cooking, cooking, picture
Web_3	Shooting, video, short, fast
Web_4	Marriage, video, long, slow
Web_5	Diving, diving, picture
Web_6	Gymnastics, article, long
Web_7	Hollywood, picture
Web_8	Hollywood, video, short, fast
Web_9	Action movie, article
Web_{10}	Shooting, shooting, article, long

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:

$$\begin{aligned}
\overrightarrow{\omega_{u1}} &= (1.35, 0, 0, 0, 0, 0, 0, 0, 0.54, 0, 0, 0.44, 0, 0, 1.75),\\
\overrightarrow{\omega_{u2}} &= (1.35, 0, 0, 0, 0, 0, 0, 0, 0.54, 0, 0, 0, 1.35, 1.35, 0),\\
\overrightarrow{\omega_{u3}} &= (0, 4.68, 0, 0, 0, 0, 0, 0, 0, 0, 1.69, 0, 0, 0, 0),\\
\overrightarrow{\omega_{u4}} &= (2.69, 0, 0, 0, 0, 0, 0, 0, 0, 0.54, 0, 0.44, 0, 0, 0),\\
\overrightarrow{\omega_{u5}} &= (0, 2.34, 0, 0, 0, 0, 0, 0, 0.54, 0, 0, 0.44, 0, 0, 1.75).
\end{aligned} \tag{8}$$

The life module:

$$\begin{aligned}
\overrightarrow{\omega_{u1}} &= (0, 0, 0, 4.68, 0, 0, 0, 0, 0, 0.54, 0, 0.44, 0, 0, 0),\\
\overrightarrow{\omega_{u2}} &= (0, 0, 0, 4.68, 0, 0, 0, 0, 0, 0, 1.69, 0, 0, 0, 0),\\
\overrightarrow{\omega_{u3}} &= (0, 0, 0, 0, 1.75, 0, 0, 0, 0.54, 0, 0, 0, 1.35, 1.35, 0),\\
\overrightarrow{\omega_{u4}} &= (0, 0, 0, 0, 5.24, 0, 0, 0, 0, 0, 0.84, 0, 0, 0, 0),\\
\overrightarrow{\omega_{u5}} &= (0, 0, 0, 0, 5.24, 0, 0, 0, 0, 0.54, 0, 0, 0, 0, 0),
\end{aligned} \tag{9}$$

The films module:

$$\begin{aligned}
\overrightarrow{\omega_{u1}} &= (0, 0, 0, 0, 0, 7.01, 0, 0, 0, 0.54, 0, 0, 0, 0, 0),\\
\overrightarrow{\omega_{u2}} &= (0, 0, 0, 0, 0, 0, 2.31, 0, 0, 0.72, 0, 0.58, 0, 0, 0),\\
\overrightarrow{\omega_{u3}} &= (0, 0, 0, 0, 0, 2.34, 0, 0, 0.54, 0, 0, 0.44, 0, 0, 1.75),\\
\overrightarrow{\omega_{u4}} &= (0, 0, 0, 0, 0, 0, 0, 6.64, 0, 0, 1.69, 0, 0, 0, 0),\\
\overrightarrow{\omega_{u5}} &= (0, 0, 0, 0, 0, 0, 1.75, 0, 0.54, 0, 0, 0, 1.35, 1.35, 0).
\end{aligned}$$

The candidate services:

$$\begin{aligned}
\overrightarrow{\omega_1} &= (0, 0, 0, 0, 0, 3.09, 0, 0, 0, 0.72, 0, 0.58, 0, 0, 0),\\
\overrightarrow{\omega_2} &= (0, 0, 0, 0, 4.66, 0, 0, 0, 0, 0, 1.12, 0, 0, 0, 0),\\
\overrightarrow{\omega_3} &= (1.35, 0, 0, 0, 0, 0, 0, 0, 0.54, 0, 0, 0, 1.35, 1.35, 0),\\
\overrightarrow{\omega_4} &= (0, 0, 0, 2.34, 0, 0, 0, 0, 0.54, 0, 0, 0.44, 0, 0, 1.75),\\
\overrightarrow{\omega_5} &= (0, 0, 14.36, 0, 0, 0, 0, 0, 0, 0, 1.12, 0, 0, 0, 0),\\
\overrightarrow{\omega_6} &= (0, 3.11, 0, 0, 0, 0, 0, 0, 0, 0.72, 0, 0.58, 0, 0, 0),\\
\overrightarrow{\omega_7} &= (0, 0, 0, 0, 0, 0, 3.5, 0, 0, 0, 1.69, 0, 0, 0, 0),\\
\overrightarrow{\omega_8} &= (0, 0, 0, 0, 0, 0, 1.75, 0, 0.54, 0, 0, 0, 1.35, 1.35, 0),\\
\overrightarrow{\omega_9} &= (0, 0, 0, 0, 0, 0, 0, 6.64, 0, 1.09, 0, 0, 0, 0, 0),\\
\overrightarrow{\omega_{10}} &= (2.69, 0, 0, 0, 0, 0, 0, 0, 0, 0.54, 0, 0.44, 0, 0, 0).
\end{aligned} \tag{10}$$

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 $\mathrm{CosSim}_{i,j}$ 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 7: The ranking scores of candidate web services.

	Web_3	Web_8	Web_4	Web_2	Web_1
Score_j	15.685	13.166	11.253	9.834	8.311
	Web_7	Web_6	Web_{10}	Web_9	Web_5
Score_j	7.052	6.234	5.801	3.347	0.275

Table 8: The list-diversity and the rank of recommended list.

Recommended list	List-diversity	Rank
Web_3, Web_8, Web_4	0.930	10
Web_3, Web_8, Web_2	0.938	8
Web_3, Web_8, Web_1	0.938	8
Web_3, Web_4, Web_2	0.996	4
Web_3, Web_4, Web_1	0.993	7
Web_3, Web_2, Web_1	1.000	1
Web_8, Web_4, Web_2	0.996	4
Web_8, Web_4, Web_1	0.994	6
Web_8, Web_2, Web_1	1.000	1
Web_4, Web_2, Web_1	0.997	3

We set the threshold "*b*" to 8 and make all candidate web services with a ranking score higher than 8 form a set Y. It is shown that the web services which are in set Y contain Web_3, Web_8, Web_4, Web_2, and Web_1.

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.

As shown in Table 8, we can see that there are two recommended lists ranked first. If two lists have the same ranking value that indicates the same diversity, we need to consider accuracy to further rank them. In other words, we need to compare the sum of every candidate service's ranking score through Step 3. And the list that has a higher ranking score sum of candidate services is preferred. As a consequence, we choose the list including Web_3, Web_2, and Web_1 as the top-3 web service recommended list.

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

[1] B. Alhijawi and Y. Kilani, "The recommender system: a survey," *International Journal of Advanced Intelligence Paradigms*, vol. 15, no. 3, p. 1, 2020.

[2] L. Qi, H. Xiang, W. Dou, C. Yang, Y. Qin, and X. Zhang, "Privacy-preserving distributed service recommendation based on locality-sensitive hashing," *IEEE International Conference on Web Services*, pp. 49–56, 2017.

[3] L. Qi, W. Dou, C. Hu, Y. Zhou, and J. Yu, "A context-aware service evaluation approach over big data for cloud applications," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 338–348, 2020.

[4] C. Zhou, A. Li, A. Hou et al., "Modeling methodology for early warning of chronic heart failure based on real medical big data," *Expert Systems with Applications*, vol. 151, article 113361, 2020.

[5] P. Pirasteh, D. Hwang, and J. J. Jung, "Exploiting matrix factorization to asymmetric user similarities in recommendation systems," *Knowledge-Based Systems*, vol. 83, no. 1, pp. 51–57, 2015.

[6] X. Wu, B. Cheng, and J. Chen, "Collaborative filtering service recommendation based on a novel similarity computation method," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 352–365, 2017.

[7] L. Qi, Q. He, F. Chen et al., "Finding all you need: web APIs recommendation in web of things through keywords search," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1063–1072, 2019.

[8] G. Kang, M. Tang, J. Liu, X. Liu, and B. Cao, "Diversifying web service recommendation results via exploring service usage history," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 566–579, 2016.

[9] J. Li, T. Cai, K. Deng, X. Wang, T. Sellis, and F. Xia, "Community-diversified influence maximization in social networks," *Information Systems*, vol. 92, article 101522, 2020.

[10] M. Kunaver and T. Požrl, "Diversity in recommender systems – a survey," *Knowledge-Based Systems*, vol. 123, pp. 154–162, 2017.

[11] A. Gogna and A. Majumdar, "Balancing accuracy and diversity in recommendations using matrix completion framework," *Knowledge-Based Systems*, vol. 125, pp. 83–95, 2017.

[12] T. Yu, J. Guo, W. Li, H. J. Wang, and L. Fan, "Recommendation with diversity: an adaptive trust-aware model," *Decision Support Systems*, vol. 123, article 113073, 2019.

[13] Y. Wang, Z. Cai, Z.-H. Zhan, B. Zhao, X. Tong, and L. Qi, "Walrasian equilibrium-based multiobjective optimization for task allocation in mobile crowdsourcing," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 4, pp. 1033–1046, 2020.

[14] Y. Wang, Z. Cai, Z.-H. Zhan, Y.-J. Gong, and X. Tong, "An optimization and auction-based incentive mechanism to maximize social welfare for mobile crowdsourcing," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 3, pp. 414–429, 2019.

[15] A. Guo and T. Yang, "Research and improvement of feature words weight based on TF-IDF algorithm," in *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, pp. 415–419, Chongqing, China, 2016.

[16] D. Kim, D. Seo, S. Cho, and P. Kang, "Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec," *Information Sciences*, vol. 477, pp. 15–29, 2019.

[17] Y. Liu, Q. Xu, and Z. Tang, "Research on text classification method based on PTF-IDF and cosine similarity," in *2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, pp. 205–208, Shanghai, China, 2019.

[18] L. Wang, X. Zhang, R. Wang, C. Yan, H. Kou, and L. Qi, "Diversified service recommendation with high accuracy and efficiency," *Knowledge-Based Systems*, vol. 204, article 106196, 2020.

[19] J. Moody and D. H. Glass, "A novel classification framework for evaluating individual and aggregate diversity in top-N recommendations," *ACM Transactions on Intelligent Systems and Technology*, vol. 7, no. 3, pp. 1–21, 2016.

[20] L. Wang, X. Zhang, T. Wang et al., "Diversified and scalable service recommendation with accuracy guarantee," *IEEE Transactions on Computational Social Systems*, pp. 1–12, 2020.

[21] Y. Zuo, M. Gong, J. Zeng, L. Ma, and L. Jiao, "Personalized recommendation based on evolutionary multi-objective optimization," *IEEE Computational Intelligence Magazine*, vol. 10, no. 1, pp. 52–62, 2015.

[22] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6492–6499, 2019.

[23] J. Wang, Z. Cai, and J. Yu, "Achieving personalized *k*-anonymity based content privacy for autonomous vehicles in CPS," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4242–4251, 2020.

[24] Y. Wang, Z. Cai, X. Tong, Y. Gao, and G. Yin, "Truthful incentive mechanism with location privacy-preserving for mobile crowdsourcing systems," *Computer Networks*, vol. 135, pp. 32–43, 2018.

[25] T. Liu, Y. Wang, Y. Li, X. Tong, L. Qi, and N. Jiang, "Privacy protection based on stream cipher for spatio-temporal data in IoT," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7928–7940, 2020.

[26] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Cost-effective app data distribution in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 31–44, 2020.

[27] Y. Wang, Q. He, D. Ye, and Y. Yang, "Formulating criticality-based cost-effective fault tolerance strategies for multi-tenant service-based systems," *IEEE Transactions on Software Engineering*, vol. 44, no. 3, pp. 291–307, 2018.

[28] L. Lin, T.-T. Goh, and D. Jin, "How textual quality of online reviews affect classification performance: a case of deep learning sentiment analysis," *Neural Computing and Applications, Springer London*, vol. 32, pp. 4387–4415, 2020.

[29] Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang, "Blockchain empowered arbitrable data auditing scheme for network storage as a service," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 289–300, 2020.

[30] Q. He, G. Cui, X. Zhang et al., "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2020.

[31] L. Yu, H. Shen, Z. Cai, L. Liu, and P. Calton, "Towards bandwidth guarantee for virtual clusters under demand uncertainty in multi-tenant clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, pp. 450–465, 2018.

[32] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4854–4866, 2019.