symmetry

Article

An Efficient Knowledge-Graph-Based Web Service Recommendation Algorithm

Zhiying Cao, Xinghao Qiao *, Shuo Jiang and Xiuguo Zhang *

School of Information Science and Technology, Dalian Maritime University, Dalian 116033, China; czysophy@dlmu.edu.cn (Z.C.); jsgg@dlmu.edu.cn (S.J.);
* Correspondence: qxh@dlmu.edu.cn (X.Q.); Zhangxg@dlmu.edu.cn (X.Z.)

Received: 14 February 2019; Accepted: 14 March 2019; Published: 18 March 2019

Abstract: Using semantic information can help to accurately find suitable services from a variety of available (different semantics) services, and the semantic information of Web services can be described in detail in a Web service knowledge graph. In this paper, a Web service recommendation algorithm based on knowledge graph representation learning (kg-WSR) is proposed. The algorithm embeds the entities and relationships of the knowledge graph into the low-dimensional vector space. By calculating the distance between service entities in low-dimensional space, the relationship information of services which is not considered in recommendation approaches using a collaborative filtering algorithm is incorporated into the recommendation algorithm to enhance the accurateness of the result. The experimental results show that this algorithm can not only effectively improve the accuracy rate, recall rate, and coverage rate of recommendation but also solve the cold start problem to some extent.

Keywords: Web service; Web services relationships; knowledge graph; representation learning; recommender systems

1. Introduction

The recommender system plays an important role in a personalized services research field. By mining the relationship between items or users, users can discover the items they may be interested in from a large number of different items.

With the rapid development of technologies such as cloud computing, mobile computing, service computing, the Internet, and big data, the emergence of a large number of Web services has caused users to face information overload problems. Finding relevant services among a large number of choices has become an important issue in the field of service computing and has been attempted to be solved by service recommendation [1,2].

Currently, most mainstream service recommendation algorithms are generally based on collaborative filtering (CF). Sreenath and Singh [3], Karta [4], and others adopted user-based CF to make recommendations. Herlocker et al. [5] proposed a K-nearest-neighbor model based on CF to improve recommendation accuracy, which is currently the most popular collaborative recommendation system model.

Quality of service (QoS) is a key factor in evaluating a service. Because we cannot obtain QoS data intuitively, many Web service recommendation systems use collaborative filtering algorithms to mine QoS attribute information, find users who are similar to the current requester, and predict the QoS values of unused services for the current requester. Shao et al. [6] proposed a user-based CF algorithm to predict service QoS values. Zheng et al. [7] combined user-based CF and item-based CF to improve recommendation accuracy.

The above Web service recommendation methods have the cold start and data sparse problems and do not consider relationships between services in the recommendation process. The service relationship is an important factor in the process of users utilizing a service; therefore, considering the service relationship in the recommendation process can give users more accurate recommendation results.

A Web service recommendation algorithm based on knowledge graph representation learning (kg-WSR) is proposed in this paper to solve the abovementioned problems. It fully considers the semantic relationship between Web services and has a good solution to the cold start problem. It uses knowledge graph representation learning to get the low-dimensional vector representation of service and user nodes, calculates the distance between services used by the user to obtain user classification, and sorts the candidate services according to the user's QoS preference to obtain a better recommendation set.

2. Related Works

Research on service recommendation has been published in the proceedings of numerous conferences and journals. Most of these methods are based on collaborative filtering, which provides recommendations through user similarity or service similarity. So, we discuss in this section several typical Web service recommendation algorithms based on collaborative filtering. Recently, many algorithms predict the quality of Web services by adding geographic location, text evaluation, and other factors to improve the recommendation effect. Thus, we also introduce here some related algorithms. In order to add semantic information to the recommendation process, in this study, we used knowledge graph representation learning to obtain the low-dimensional vectors which contain the semantic information of the corresponding entity; several typical representation learning algorithms are discussed below.

2.1. Research on Knowledge Representation Learning

Mikolov et al. [8] proposed the word2vec word representation learning model and toolkit in 2013. Through this model, they found that the word vector space has translation invariance. After that, representation learning has gained widespread attention.

The main methods of knowledge representation learning include the distance model [9], single-layer neural network [10], tensor neural network model [10,11], matrix decomposition model [12], and translation model [13–17]. Due to the huge performance improvement of the translation model, through which not only the training parameters are greatly reduced but also the accuracy is improved, the translation model has become the representative model of knowledge representation learning.

The TransE algorithm proposed by Bordes et al. [14] is the most representative algorithm of the translation model. It treats relationships in the knowledge base as a certain kind of translation vector between entities. For each triple (h, r, t), TransE uses the relation vector R as the translation between the head entity vector H and the tail entity vector T. We can also regard R as a translation from H to T. TransE implements the digitization of the entity, so knowledge representation learning can quickly calculate the semantic relevance of the entities.

The TransE algorithm has good results for one-to-one relationships between entities, but it is less effective for the one-to-many, many-to-one, and many-to-many relationships [18]. The important reason is that TransE represents both entities and relationships in the same plane, and the interaction between entities and relationships cannot be clarified, which in turn makes it impossible to distinguish complex relationships.

In order to deal with the complex relationship between entities, Wang et al. [19] proposed the TransH algorithm, which establishes a relationship-oriented hyperplane model. For a relation r, a relation-specific translation vector l_r is placed on the relationship-specific hyperplane l_{nr} (normal vector) instead of being represented in the same space as the entity. TransH can solve the problems faced by TransE when complex relationships occur.

2.2. Research on Web Service Recommendation

Zheng et al. [7,19] proposed a recommendation system based on a user feedback collaborative filtering algorithm (WSRec algorithm), which uses a collaborative filtering algorithm to predict QoS values and a linear combination of user-based and item-based collaborative filtering to improve the accuracy of QoS prediction. However, there are limitations in the recommendation system caused by the use of collaborative filtering algorithms; that is, sparse data seriously affects the accuracy of QoS prediction.

Sreenath and Singh [3], Karta [4], and others used user-based CF to recommend services, but these only consider user similarity and the recommendation is not comprehensive.

Ma et al. [20] predicted QoS values by tensor decomposition, marked services by a user preference learning method, and implemented service recommendation using service scores.

Herlocker et al. [5] proposed a K-nearest-neighbor (KNN) model based on collaborative filtering, namely, the UPCC algorithm, which is presently the most popular collaborative recommendation algorithm. However, it has serious data sparseness and cold start problems. For new users, new services cannot be given a reasonable recommendation.

Salakhutdinov et al. [21] proposed a product-based KNN model called the IPCC algorithm. This type of KNN model assumes that a user's present and past interests are similar; that is, if a service is similar to a service that a user liked in the past, then this service is now very likely to be liked by him. The breadth of recommendations for such algorithms is very low, for it is not possible to recommend other types of services that users may like but only to recommend a certain type of service.

Jhaveri et al. [22] proposed a composite model for two-way Web service recommendation using QoS and quality of experience (QoE). The model analyzes the positive or negative aspects of the evaluation text and combines the user's emotional score with the satisfaction score to improve the accuracy of the recommendation.

Papadakis et al. [23] proposed a recommendation algorithm based on the Vivaldi algorithm (SCoR), which maps users and items to nodes on the coordinate axes and calculates the distance of the nodes. The algorithm can solve the cold start problem to a certain extent.

Liu et al. [24] proposed a QoS prediction algorithm that considers the location and geographic factors affecting QoS in the recommendation algorithm to improve prediction accuracy.

Nilashi et al. [25] proposed a collaborative filtering recommendation algorithm based on ontology and dimensionality reduction. However, the establishment of ontology is complex, and there are very few existing Web service ontologies.

3. Knowledge-Graph-Based Web Service Recommendation Algorithm

In order to solve the cold start problem of traditional service recommendation and the problem of semantic associations between services not being taken into consideration, we developed the kg-WSR algorithm. In this algorithm, firstly, the entity relationships of a Web service knowledge graph are analyzed. There are two types of entity nodes in the Web service knowledge graph: Web services and users. The relationship between different services is judged by the input and output of the service; users and services are linked by rating. Secondly, the entities in the knowledge graph are embedded into a low-dimensional space by using the TransH algorithm. Then, the similarities between users are calculated to identify the neighbors and the nearest K users for a user, and the distances between services are calculated to find semantically similar services for each service. The smaller the distance, the greater the degree of association. For the user receiving the recommendation, the highly rated services of his neighbors are defined as set US, the semantically similar services of services that he marked highly form the recommended set RS, and his QoS preference is calculated according to the user rating vector and service QoS matrix. According to preference, the US and RS are sorted into ordered sequences L and K, respectively. A preliminary recommendation set is generated by integrating L and K in the proportion of p:q, and the final ranking is performed to create the final recommendation set C. The values of p and q are adjusted according to whether the user receiving the recommendation is a positive user or a negative user.

The main steps of the proposed method are shown in Figure 1.

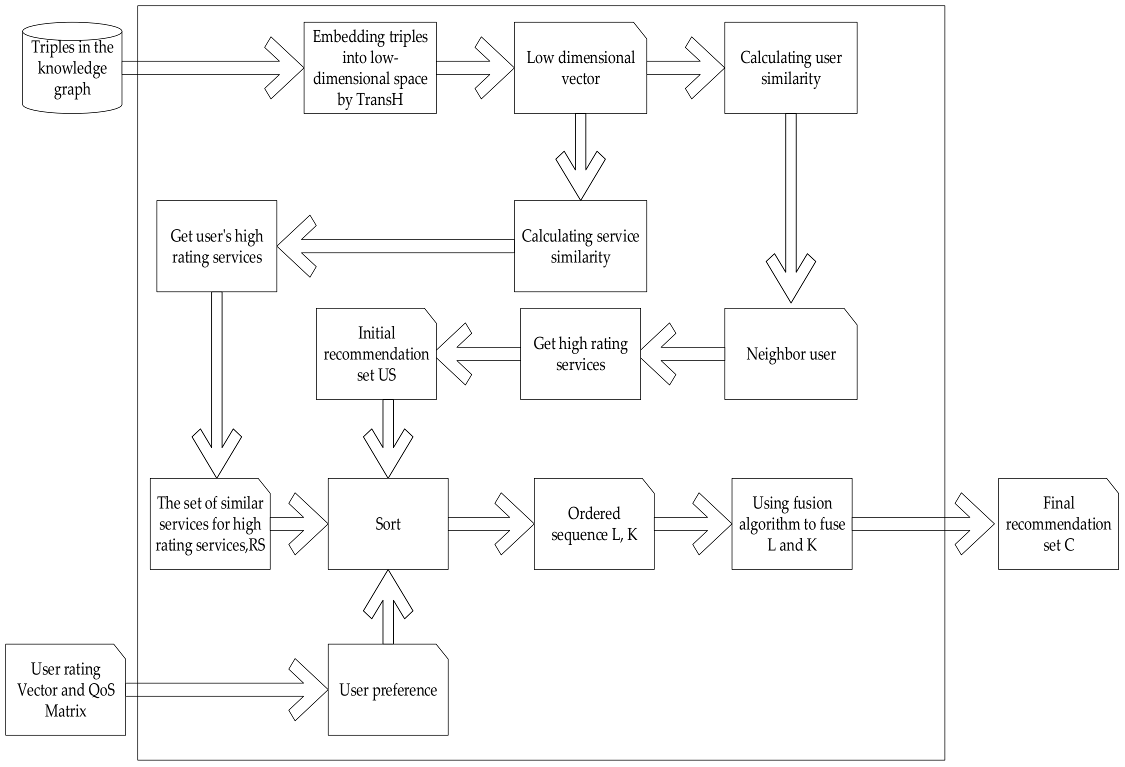
Triples in the knowledge graph
Embedding triples into low-dimensional space by TransH
Low dimensional vector
Calculating user similarity
Get user's high rating services
Calculating service similarity
Initial recommendation set US
Get high rating services
Neighbor user
The set of similar services for high rating services,RS
Sort
Ordered sequence L, K
Using fusion algorithm to fuse L and K
Final recommendation set C
User rating Vector and QoS Matrix
User preference

Figure 1. The main steps of the Web service recommendation algorithm based on knowledge graph representation learning (kg-WSR).

3.1. *Web Service Knowledge Graph Representation Learning*

3.1.1. Analyzing Service Relations

The service entities in the Web service knowledge graph are related through service relationships, so determining the service relationship is a crucial step in the Web service knowledge graph. From the perspective of a Web Services Description Language (WSDL) document, a Web service can be abstracted into a conceptual model consisting of three layers of components: service, operation, and parameter. The definition of service relationships is also divided into three levels, namely, parameter, operation, and service levels. This paper considers service-level relationships. Service-level relationships focus on describing interactions and constraints between services. On the Web, multiple services are functionally equivalent or similar, and there must be competition between them. On the other hand, the service process determines the existence of collaboration between services, which is the basis for control flow in composite services and business processes [26]. Therefore, according to the competition and collaboration relationship, there are five categories of service relationships, as shown in Table 1.

As for the relationship between users and services, they are linked by rating. There are five rating levels corresponding to this relationship: extremely dissatisfied, dissatisfied, medium satisfaction, satisfied, and very satisfied.

The relationship between services and the relationship between users and services are illustrated in the Web service knowledge graph shown in Figure 2.

Table 1. Service relationship.

Types	Categories	Description
Competitive relations	Exact	$\forall op_{ik} \in ws_i,\ \exists op_{jm} \in ws_j \Rightarrow op_{ik} = op_{jm}$
	plugin	$\forall op_{jk} \in ws_j,\ \exists op_{im} \in ws_i \Rightarrow op_{jk} = op_{im}$
	subsume	$\forall op_{ik} \in ws_i,\ \exists op_{jm} \in ws_j \Rightarrow op_{ik} = op_{jm}$
		$\forall op_{jk} \in ws_j,\ \exists op_{im} \in ws_i \Rightarrow op_{jk} = op_{im}$
Collaborative relations	Rely-total	$\exists op_{im} \in ws_i,\ \exists op_{jn} \in ws_j \Rightarrow op_{im}^{out} \supseteq op_{jn}^{in}$
	Rely-part	$\exists op_{im} \in ws_i,\ \exists op_{jn} \in ws_j \Rightarrow op_{im}^{out} \subset op_{jn}^{in}$

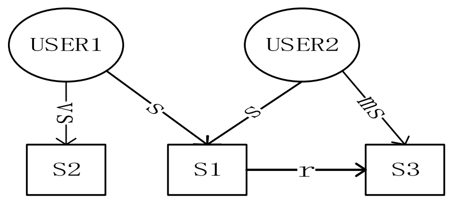

Figure 2. Simplified diagram of the knowledge graph.

3.1.2. Embedding Entities in Low-Dimensional Space

Based on the knowledge graph, the TransE algorithm is a representative algorithm in representation learning. It embeds the entities and relationships in the knowledge graph into the low-dimensional vector space.

For a triple (*h, r, t*) in knowledge graph G, let l_h, l_r, and l_t represent vectors corresponding to *h*, *r*, and *t*, respectively. TranE expects l_h, l_r, l_t to satisfy Equation (1):

$$l_h + l_r \approx l_t. \tag{1}$$

The score function is $d(l_h + l_r, l_t) = ||l_h + l_r - l_t||^2$, where $||\bullet||^2$ is the 2 norm of $\bullet$, the Euclidean distance.

Training the loss function is shown in Equation (2):

$$L = \sum_{(h,r,t)\in K} \sum_{(h\prime,r,t\prime)\in K\prime} [(\lambda + d(l_h + l_r, l_t) - d(l_{h'} + r, l_{t'}))]_+ \tag{2}$$

$$K\prime = \{(h\prime, r, t)/h\prime \in E\} \cup \{(h, r, t\prime)/t\prime \in E\}$$

where *E* is a set of entities in the knowledge graph, (*h, r, t*) is the correct triple from the training set, K' is a set of error triples, and λ is the separation distance between the score of the correct triple and the score of the wrong triple, which is generally it is set to 1. The error triple is not randomly generated. In order to select a representative error triple, TransE randomly replaces one of the header and tail entities of each triple in *K* with other entities to get K'.

$[x]_+$ represents the hinge loss function, described in Equation (3):

$$[x]_+ = \begin{cases} 0, x \le 0 \\ x, x > 0 \end{cases}. \tag{3}$$

TransE adopts the principle of maximum distance and trains by enlarging distances between the right and wrong samples. The loss function is optimized by a stochastic gradient descent algorithm. The resulting vector space has the following characteristics: (1) the head node vector plus the corresponding triplet relationship vector is approximately equal to the triplet tail node vector; (2) the closer the semantic of services in the knowledge graph, the smaller the entity distance [27]. The vector representation of the TransE algorithm is shown in Figure 3.

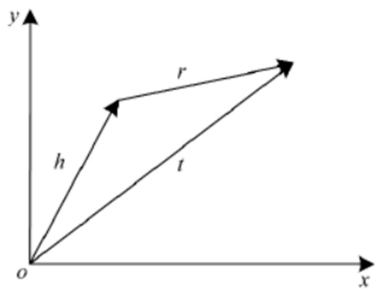

Figure 3. Vector representation of the TransE algorithm.

TransH establishes a relationship-oriented hyperplane model in order to optimize TransE's weak ability to handle complex relationships. For a relation *r,* the vector l_r corresponding to r is placed on the relationship-specific hyperplane lnr (normal vector) instead of being represented in the same space as the entity representation. For a triple (*h, r, t*), h and t are first mapped to $l_{hr},\ l_{tr}$ on the hyperplane l_{nr}. Equations (5) and (6) are used to link $\mathrm{l_{hr}}$, $\mathrm{l_{tr}}$, and $\mathrm{l_r}$ on the hyperplane. The score function is described in Equation (4):

$$d(l_{hr}+l_r,l_{tr})=||l_{hr}+l_r-l_{tr}||_2, \tag{4}$$

which can provide

$$l_{hr}=l_h-{l_{nr}}^T*l_h*l_{nr} \tag{5}$$

$$l_{tr}=l_t-{l_{nr}}^T*l_t*l_{nr}. \tag{6}$$

The above formula can be used in the scoring function after the trial of the hyperplane model:

$$d(l_{hr}+l_r,l_{tr})=\left|\left|(l_h-{l_{nr}}^T*l_h*l_{nr})+l_r-(l_t-{l_{nr}}^T*l_t*l_{nr})\right|\right|_2. \tag{7}$$

Consider the constraints of the vector:

$$\forall h,t\in E,||l_h||_2\le 1,||l_t||_2\le 1 \tag{8}$$

$$\forall r\in R,\left|{l_{nr}}^T*l_r\right|/\left|\left|l_r\right|\right|_2\le \varepsilon \tag{9}$$

$$\forall r\in R,||l_r||_2=1. \tag{10}$$

The final loss function is

$$\begin{aligned}L=\sum_{(h,r,t)\in K}\ \sum_{(h\prime,r,t\prime)\in K\prime}&[(\lambda+d(l_h+l_r,l_t)-d(l_{h'}+r,l_{t'}))]_++\\ &c\sum_{e\in E}||e||^2+c\sum_{r\in R}[\tfrac{({l_{nr}}^T*r)^2}{||l_r||^2}-\varepsilon^2]\end{aligned} \tag{11}$$

The second and third terms of the loss function are representations of the constraints. The training principle is similar to TransE.

Different entities can be distinguished after using the hyperplane model to make the knowledge representation more accurate. The vector representation of the TransH algorithm is shown in Figure 4.

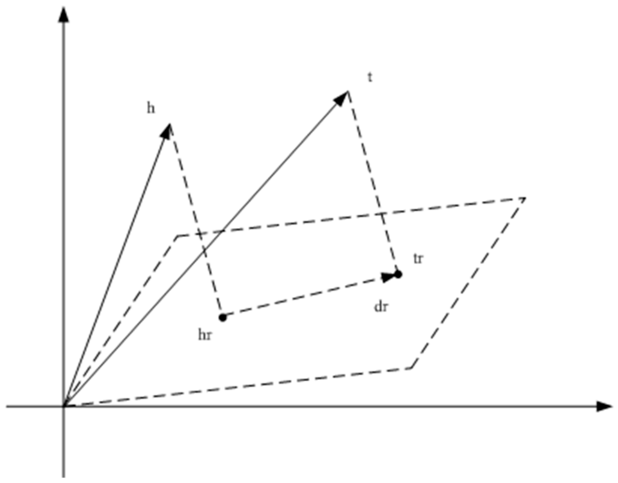

Figure 4. Vector representation of the TransH algorithm.

Since TransH calculates the loss function by Euclidean distance, the Euclidean distance is used to measure the similarity of the entities. The entity similarity is defined as Equation (12):

$$sim(A,B) = \frac{1}{||A-B||+1} \tag{12}$$

where A and B are two entity vectors in the knowledge graph. It can be seen from Equation (12) that the closer sim(A, B) is to 1, the higher the correlation between the entity vectors A and B is; that is, the closer the two entities are in the knowledge graph, the smaller sim(A, B) is and the more alienated the two entities are in the knowledge graph.

The similarities of users are calculated by using Equation (12) and sorted. The 10 users closest to u_m form the neighbor set U of user u_m.

3.1.3. Obtaining the Initial Recommendation Set

For the triple in the knowledge graph, the head vector plus the relationship vector is approximately equal to the tail vector. For each u_m's neighbor user u($u \in U$), Equations (13) and (14) are used to find his highly rated services, and the highly rated services of neighbor users of u_m form a set denoted as US:

$$||u + r_{satisfaction} - s\Big|\Big| < \gamma \tag{13}$$

$$||u + r_{verysatisfaction} - s\Big|\Big| < \gamma. \tag{14}$$

The value of γ represents the maximum distance between a service and the user's highly rated service. The smaller γ is, the more the service satisfies the conditions of the user's highly rated service. γ is determined by experiment, and s represents u's vector corresponding to the highly rated service. $r_{\text{satisfaction}}$ represents the vector corresponding to satisfaction, and $r_{\text{verysatisfaction}}$ represents the vector corresponding to very satisfied.

For the user u_m, his high evaluation services are revealed by Equations (13) and (14), the similarities between each highly rated service and other services are calculated, and the top five services closest to each highly rated service are added to the set RS as initial recommendation services.

US and RS are the initial recommendation service sets.

3.2. User QoS Preference Calculation and User Behavior Classification

3.2.1. QoS Preference Calculation

QoS is an important factor to be considered for recommendations. It includes multidimensional quality of service factors such as response time, reliability, availability, throughput, and so on. Different users have different acceptance levels for different QoS attributes. For example, some users are more concerned with reliability than response time, while others are more concerned with response time than reliability. Therefore, calculating the user's preference for different QoS attributes through user behavior can rank the recommendation services more reasonably.

Suppose that u_m has awarded ratings to N Web services and there are K QoS properties for each Web service.

Let $R_m = (r_1, r_2, \ldots, r_N)^T$ denote the rating vector of u_m, where r_n denotes the rating awarded by u_m for $service_n$.

Let $Q_m = (q_{n,k})_{N*K}$ denote the QoS matrix of u_m, where $q_{n,k}$ denotes the value of the k-th QoS property of service n which is observed from u_m.

Let $P_m = (p_1, p_2, \ldots, p_K)^T$ denote the preference vector of u_m, where pk denotes u_m's preference for k-th QoS attributes [21].

The relationship between R_m, Q_m, and P_m is expressed in Equation (15):

$$R_m = Q_m \bullet P_m. \tag{15}$$

The purpose is to calculate P_m. Because the number of QoS attributes is small, the noise in the QoS and rating data might provide a wrong solution when using linear algebra. So, the loss function is defined as shown in Equation (16):

$$F = ||R_m - Q_m \bullet P_m||_2 + \alpha ||P_m||^2 \tag{16}$$

where $\alpha ||P_m||^2$ is the regularization term to avoid overfitting. P_m can be solved by a gradient descent algorithm as follows:

(1) Randomly initialize the values of the parameters in P_m.

(2) Use Equation (17) to compute the partial derivative of F for pk and Equation (18) to update the parameters:

$$\frac{\partial F}{\partial P_k} = \sum_{n=1}^{N} [2\alpha P_k - 2r_n q_{n,k} + 2q_{n,k} \sum_{k=1}^{K} q_{n,k} p_k] \tag{17}$$

$$p_k = p_k - \tau \frac{\partial F}{\partial p_k} \tag{18}$$

where τ is the iteration step size for calculating P_m.

(3) If $F > \xi$, back to 2; if $F \leq \xi$, terminate the computation and P_m is determined.

3.2.2. User Behavior Classification

The usage habits of users are different. Some users like to use services that have been used or related to them. We define these users as negative users. Some users are open to and willing to try new services. We define these users as positive users. In the knowledge graph, the distance between entities represents the semantic similarity of the entities. So, we can judge the degree of association between the two services by calculating the distance of the Web service entities in the graph. Therefore, the category of a user can be effectively determined by the average distance between all the services used by him.

Let $S_m = (s_1, s_2, \ldots s_n)$ denote the set of highly rated services that u_m has used. According to the above definition and the distance of the entity vector in the knowledge graph, the user behavior is judged by Equation (19):

$$\overline{d} = \frac{1}{n}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}||s_i - s_j|| \tag{19}$$

where $\overline{d}$ represents the average distance of the Web services used by the user in the knowledge graph, which is used to determine whether it is a positive or negative user. The algorithm sets the ratio of positive to negative users as 1:1. $\overline{d}$ for each user is calculated and sorted ascendingly. Users with a $\overline{d}$ in the top 50% are negative users, and the others are positive users.

3.3. Recommendation Set Fusion

By Equations (12)–(14), $\mathrm{u_m}$'s neighbor users' highly rated services set US and the highly rated services' association services set RS of $\mathrm{u_m}$ are obtained. For $\mathrm{u_m}$, his predictive scores of services in US and RS are obtained by multiplying the calculated user preferences $\mathrm{p_m}$ with the predicted QoS attribute values of services. The average value of QoS evaluation of his neighbor users represents the QoS prediction value of $\mathrm{u_m}$ for services. The product of user preference and predictive QoS is added to the predictive score of the service. Web services in US and RS are sorted by predictive scores to get L and K. The services in US are far away from the services used by the user to receive a recommendation, so the positive user prefers to be recommended more services in US than in RS. The services in RS are closer to the services used by the user to receive a recommendation, so the negative user prefers to be recommended more services in RS than in US. The final recommendation set C should integrate L and K in the proportion p:q, and p and q are determined by experiment. Assuming that the length of L is l and the length of K is k, the length of C denoted as n can be calculated by Equation (20):

$$n = \begin{cases} [\frac{kp}{p+q}], & u \text{ is positive user} \\ [\frac{lq}{p+q}], & u \text{ is negative user} \end{cases} . \tag{20}$$

The fusion algorithm is shown as follow:

Algorithm 1: Recommendation set fusion algorithm

Input: Neighbor user service ordered set L; associated service for high-scoring services of recommended users ordered set K; user category NU (negative user), PU (positive user)
Output: Final recommendation set C

```
1. If(user is PU)
2. For $L_i \in \{L_{l-n}, L_{l-n+1}, \ldots, L_l\}$ do:
3. For $K_j \in \{K_1, K_2, \ldots, K_n\}$ do:
4. $L_i = K_j$
5. End do
6. End do
7. C = L
8. Else
9. For $K_i \in \{K_{k-n}, K_{k-n+1}, \ldots, K_k\}$ do:
10. For $L_j \in \{L_1, L_2, \ldots, L_n\}$ do:
11. $K_i = L_j$
12. End do
13. End do
14. C = K
```

3.4. Pseudocode of kg-WSR

The pseudocode for kg-WSR is shown as follows:

Algorithm 2: kg-WSR

```
Input: Triples in knowledge map; user $u_m$; $u_m$'s rating vector; $u_m$'s service QoS matrix; user set $U_F$; service set $S_F$

Output: Service recommendation set C of user $u_m$
1. Computing user preference $P_m$ by gradient descent
2. Using TransH to obtain the low-dimensional vector expression of entities and relationships
3. For each user u in $U_F$ do:
4. Computing the similarity between u and $u_m$
5. Sort similarities
6. End for
7. Ten users with the highest similarity as the neighbor set U of $u_m$
8. For each user u in U do:
9. For each service s in $S_F$ do:
10. Computing $||u+r_{satisfaction}-s||$ and $||u+r_{verysatisfaction}-s||$
11. If $||u+r_{satisfaction}-s|| < \gamma$ or $||u+r_{verysatisfaction}-s|| < \gamma$ then:
12. s added to set US
13. End if
14. End for
15. For each service s in $S_F$ do:
16. Computing $||u_m+r_{satisfaction}-s||$ and $||u_m+r_{verysatisfaction}-s||$
17. If $||u_m+r_{satisfaction}-s|| < \gamma$ or $||u_m+r_{verysatisfaction}-s|| < \gamma$ then:
18. s is highly rated service for user $u_m$
19. End if
20. End for
21. For each $u_m$'s highly rated service do:
22. For each service s in $S_F$ do:
23. Computing service similarity
24. End for
25. Sort similarities
26. The three services with the highest similarity are added to the set RS
27. End for
28. Sorting US and RS into ordered sequences L and K, respectively, using user preference $p_m$
29. Computing the average distance between highly rated services of $u_m$ and determining user classification
30. Use the fusion algorithm to form the final recommendation set C
```

4. Experiment and Evaluation

We conducted experiments to test the proposed algorithm and evaluated its recommendation effect in the environment we provided. The experimental results show that the proposed algorithm improved the accuracy rate, recall rate, and coverage rate of recommendation results.

4.1. Experimental Data

In order to obtain real Web services, 873 travel-related and map-related Web services were captured on multiple open platforms through Web crawlers. Travel-related services included hotel services, ticketing services, attraction services, etc. Relationships between services were determined by input–output keyword matching and the relationships defined in Table 1. At the same time, the rating information of 345 users and the evaluation of the two QoS attributes of response time and reliability were obtained. These data were stored in the Web service knowledge graph.

4.2. Evaluation Standard

For the recommendation results, the precision rate (P), recall rate (R), and coverage rate (C) indicators were used to evaluate this algorithm. The precision rate represents the recommended accuracy; that is, how many recommended services are correct. The recall rate reflects the proportion of items recommended by the recommended system occupying items that users really like [1]. Services used by fewer than 10 users were defined as new services. Coverage rate refers to the rate of new services in recommendation results. This was used to judge the degree of solving the cold start problem. The indicators were defined as:

$$P = \frac{|L_t \cap L_r|}{L_r} \tag{21}$$

$$R = \frac{|L_t \cap L_r|}{L_t} \tag{22}$$

$$C = \frac{|C_n|}{|S_n|} \tag{23}$$

where L_t represents the list of services actually used by the user, L_r is the recommended list of services, C_n represents new services in the recommendation list, and S_n represents new services in the knowledge graph.

4.3. Experimental Comparison and Verification

The low-dimensional vector representation of services and users in the knowledge graph were obtained by the TransH algorithm. Assuming the user had five highly rated services, through many experiments, we obtained γ = 0.15. After calculating the average distance of services used by each user, we found that when the parameter in Equation (19) $\overline{d} \leq 0.25$, the user was a negative user; when $\overline{d} > 0.25$, the user was a positive user.

4.3.1. Embedded Dimension Determination

Knowledge graph representation learning is to embed entities into low-dimensional vector space, so it will have different effects for different embedding dimensions. The experimental results of 100–500 dimensions are shown in Table 2.

Table 2. P, R values corresponding to different embedding dimensions.

Dimension	P	R
100	0.580	0.435
200	0.587	0.441
300	0.582	0.437
400	0.578	0.434
500	0.576	0.432

It can be seen from Table 2 that the recommended effect was best with 200 dimensions.

4.3.2. Fusion Ratio Determination

The proportional fusion experiment was performed with the learning embedding dimension of 200 and the number of recommended set C was 30. Table 3 shows the P and R for each ratio when the user classification was not performed. Table 4 shows the P and R for each ratio when the user classification was performed.

From Tables 3 and 4, it can be seen that the performance of P and R values after user classification was better than that before classification. The optimal fusion ratio was 6:4 for positive users and 4:6 for negative users.

Table 3. P, R values for each proportion when there was no user classification.

Fusion Ratio (p:q)	P	R
0:10	0.465	0.347
1:9	0.497	0.376
2:8	0.520	0.388
3:7	0.544	0.410
4:6	0.567	0.420
5:5	0.542	0.409
6:4	0.527	0.396
7:3	0.487	0.368
8:2	0.461	0.349
9:1	0.430	0.316
10:0	0.415	0.310

Table 4. P, R values for each proportion when there was user classification.

NU Fusion Ratio	PU Fusion Ratio	P	R
0:10	10:0	0.482	0.360
1:9	9:1	0.508	0.383
2:8	8:2	0.530	0.395
3:7	7:3	0.570	0.429
4:6	6:4	0.587	0.441
5:5	5:5	0.546	0.406

4.3.3. Performance Comparison

Experiments were performed with an optimal dimension of 200 and an optimal ratio of 6:4, 4:6. The following algorithms were selected to be compared with the kg-WSR:

SCoR: This algorithm puts the entity into the coordinate system to calculate the Euclidean distance for recommendation. This algorithm has a good effect on the cold start problem.

UPCC: Research on the neighbor model, KNN model.

IPCC: Research on product-based KNN model.

WSRec: Improved collaborative filtering prediction algorithm and integrated information on similar users and services.

Pearson-CF: Collaborative filtering service recommendation algorithm based on Pearson coefficient.

Bi-WSR: Integrated recommendation model with integrated QoS and QoE.

The recommended results for the comparison algorithms were mainly affected by the QoS matrix data density (D). Therefore, the experiment set three different densities of 10%, 20%, and 30%. Figure 5 shows the accuracy rate comparison with kg-WSR.

Figure 6 shows the comparison of recall rates.

In order to evaluate the cold start problem, different numbers of new services were injected into the experimental data and the algorithm coverage rates were compared. Figure 7 shows the comparison of algorithm new service coverage rates.

It can be seen from Figures 5–7 that the kg-WSR algorithm improved the accuracy, recall, and coverage rates of recommendation results by using the service association information and user information in the knowledge graph. Also, the influence of data density on the algorithm was small. The coverage rate of the new services was obviously higher than that of the other algorithms. Thus, this algorithm better solves the cold start problem.

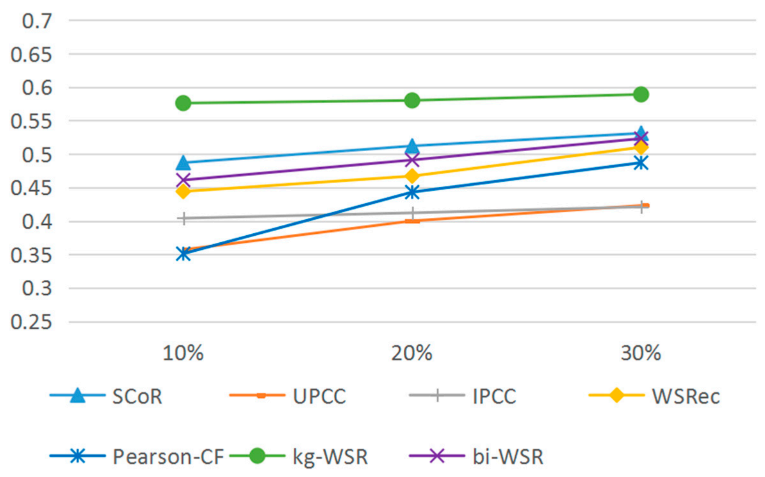

Figure 5. Comparison of algorithm accuracy rates.

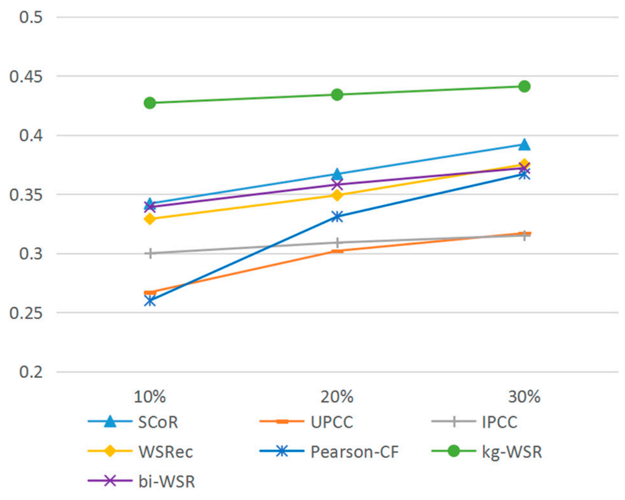

Figure 6. Comparison of algorithm recall rates.

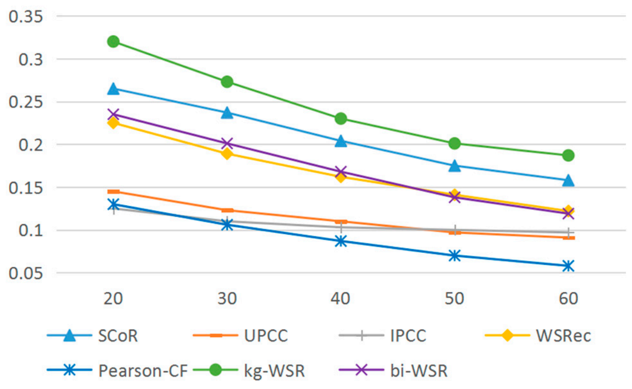

Figure 7. Comparison of algorithm new service coverage rates.

4.3.4. Algorithm Efficiency

Figure 8 shows the running time of each algorithm, which was the average of that of 10 repeated experiments.

It can be seen from Figure 8 that the kg-WSR algorithm proposed in this paper was more computationally intensive because factors such as user preference and user classification were involved. The running time was slightly higher than that of WSRec and SCoR while lower than that of bi-WSR.

The running time was higher than UPCC, IPCC, and Pearson-CF because they only use similarity for recommendations.

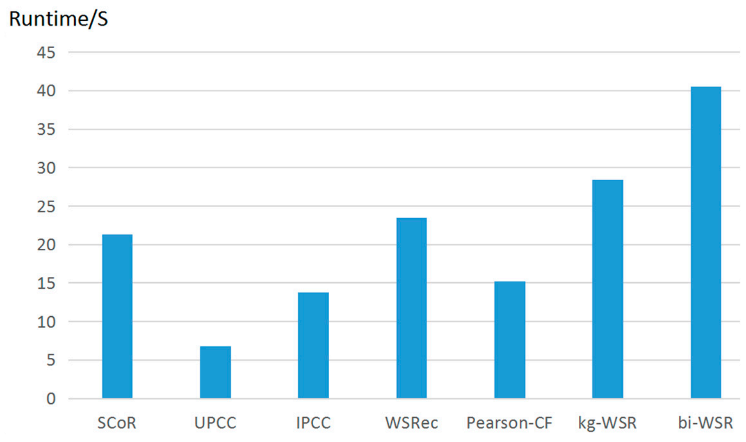

Figure 8. Algorithm running time comparison.

5. Conclusions

This paper proposed an efficient Web service recommendation algorithm based on knowledge graph representation learning. This algorithm not only takes into account the semantic information between services but also calculates users' QoS preferences and user classification. Therefore, the algorithm has a comprehensive recommendation effect. The algorithm embeds the entity into the low-dimensional space through knowledge graph representation learning (TransH), calculates similarities between entities to find neighbor users and associated services, and classifies users by the distance between services used.

Experiments conducted show that the kg-WSR improved the accuracy of the recommendation result and had a certain effect on solving the cold start problem. Although the time cost of the algorithm was higher than that of the other algorithms to which it was compared, it greatly improved the accuracy and recall rates and achieved a better recommendation effect.

Author Contributions: Author Contributions Writing—original draft, Z.C., X.Q., S.J., and X.Z.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 51679025, 51479021), the Fundamental Liaoning Provincial Natural Science Foundation of China (Grant No. 20170520196) and the Fundamental Research Funds for the Central Universities (Grant No.3132016308).

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

References

1. Zhang, X.; He, K.; Wang, J.; Liu, J. A Survey of Web Services Personalized Recommendations. *Comput. Eng. Sci.* **2013**, *35*, 132–140.
2. Yue, K.; Wang, X.; Zhou, A. Web Services Core Support Technology: A Review of Research. *J. Softw.* **2004**, *15*, 428–442.
3. Sreenath, R.M.; Singh, M.P. Agent-based service selection. *Web Semant. Sci. Serv. Agents World Wide Web* **2004**, *1*, 261–279. [CrossRef]
4. Karta, K. An Investigation on Personalized Collaborative Filtering for Web Service Selection. Honours Programme Thesis, University of Western Australia, Brisbane, Australia, 2005.
5. Herlocker, J.L.; Konstan, J.A.; Borchers, A.; Riedl, J. An algorithmic framework for performing collaborative filtering. In Proceedings of the International ACM Sigir Conference on Research & Development in Information Retrieval, Berkeley, CA, USA, 15–19 August 1999.

6. Shao, L.; Zhang, J.; Wei, Y.; Zhao, J.; Xie, B.; Mei, H. Personalized QoS Prediction for Web Services via Collaborative Filtering. In Proceedings of the IEEE International Conference on Web Services, Salt Lake City, UT, USA, 9–13 July 2007.
7. Zheng, Z.; Ma, H.; Lyu, M.R.; King, I. Wsrec: A collaborative filtering based web service recommender system. In Proceedings of the 7th IEEE International Conference on Web Services, Los Angeles, CA, USA, 6–10 July 2009; pp. 437–444.
8. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 3111–3119.
9. Bordes, A.; Weston, J.; Collobert, R.; Bengio, Y. Learning Structured Embedding of Knowledge Bases. In Proceedings of the 25th AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7–11 August 2011.
10. Socher, R.; Chen, D.; Manning, C.D.; Ng, A. Reasoning with Neural Tensor Networks for Knowledge Base Completion. In Proceedings of the International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 8–13 December 2013; Curran Associates Inc.: Red Hook, NY, USA, 2013.
11. Yang, B.; Yih, W.; He, X.; Gao, J.; Deng, L. Embedding entities and relations for learning and inference in knowledge bases. *arXiv*, 2014; arXiv:1412.6575.
12. Nickel, M.; Tresp, V.; Kriegel, H.P. A three-way model for collective learning on multi-relational data. In Proceedings of the International Conference on International Conference on Machine Learning, Bellevue, WA, USA, 2–28 June 2011.
13. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv*, 2013; arXiv:1301.3781.
14. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. In Proceedings of the NIPS, Cambridge, UK, 4–10 December 2013.
15. Ji, G.; He, S.; Xu, L.; Liu, K.; Zhao, J. Knowledge Graph Embedding via Dynamic Mapping Matrix. In Proceedings of the Meeting of the Association for Computational Linguistics & the International Joint Conference on Natural Language Processing, Beijing, China, 27–31 July 2015.
16. Xiao, H.; Huang, M.; Hao, Y.; Zhu, X. TransG: A Generative Mixture Model for Knowledge Graph Embedding. *arXiv*, 2016; arXiv:1509.05488.
17. He, S.; Liu, K.; Ji, G.; Zhao, J. Learning to represent knowledge graphs with gaussian embedding. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 18–23 October 2015; pp. 623–632.
18. Fang, Y.; Zhao, X.; Tan, Z.; Yang, S.; Xiao, W. An Improved Translation-Based Knowledge Mapping Representation Method. *J. Comput. Res. Dev.* **2018**, *55*, 139–150.
19. Wang, Z.; Zhang, J.; Feng, J.; Chen, Z. Knowledge Graph Embedding by Translating on Hyperplanes. In Proceedings of the Twenty-Eighth Aaai Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014.
20. You, M.A.; Xin, X.; Wang, S.; Li, J.; Sun, Q.; Yang, F. QoS Evaluation for Web Service Recommendation. *China Commun.* **2015**, *12*, 151–160. [CrossRef]
21. Salakhutdinov, R.; Mnih, A. Probabilistic Matrix Factorization. In Proceedings of the International Conference on Neural Information Processing Systems, Kitakyushu, Japan, 13–16 November 2007.
22. Jhaveri, S.; Soundalgekar, P.M.; George, K.; Kamath, S.S. A QoS and QoE based Integrated Model for Bidirectional Web Service Recommendation. In Proceedings of the Pacific Neighborhood Consortium Annual Conference and Joint Meetings, Tainan, Taiwan, 7–9 November 2018.
23. Papadakis, H.; Panagiotakis, C.; Fragopoulou, P. SCoR: A Synthetic Coordinate based Recommender system. *Expert Syst. Appl.* **2017**, *79*, 8–19. [CrossRef]
24. Liu, J.; Tang, M.; Zheng, Z.; Liu, X.F.; Lyu, S. Location-aware and personalized collaborative filtering for web service recommendation. *IEEE Trans. Serv. Comput.* **2016**, *9*, 686–699. [CrossRef]
25. Nilashi, M.; Ibrahim, O.; Bagherifard, K. A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. *Expert Syst. Appl.* **2018**, *92*, 507–520. [CrossRef]

26. Chen, S.; Feng, Z.; Wang, H. Service Relationship and Its Application in Service-Oriented Computing. *Chin. J. Comput.* **2010**, *33*, 2068–2083. [CrossRef]
27. Liu, Z.; Sun, M.; Lin, Y.; Xie, R. Progress in Knowledge Representation Learning. *J. Comput. Res. Dev.* **2016**, *53*, 247–261.