An API Recommendation Method Based on Beneficial Interaction

Siyuan Wang, Buqing Cao(✉), Xiang Xie, Lulu Zhang, Guosheng Kang, and Jianxun Liu

School of Computer Science and Engineering and Hunan Key Laboratory of Service Computing and New Software Service Technology, Hunan University of Science and Technology, Xiangtan, Hunan, China
buqingcao@gmail.com

Abstract. With the wide application of Mashup technology, it has become one of the hot and challenging problems in the field of service computing that how to recommend the API to developers to satisfy their Mashup requirements. The existing service recommendation methods based on Graph Neural Network (GNN) usually construct feature interaction graph by the interactions of service features, and regard it as the input of GNN to achieve service prediction and recommendation. In fact, there are some distinctions in the interactions between service features, and the importance of interactions is also different. To address this problem, this paper proposes an API recommendation method based on beneficial feature interaction, which can distinguish and extract beneficial feature interaction pairs from a large number of service feature interaction relationships. Firstly, feature extraction of Mashup requirements and API services is performed, and the correlation between API services is calculated based on the label and description document of the API services and used as a basis for recommending API services to Mashup requirements. Secondly, edge prediction component is used to extract beneficial feature pairs from input features of Mashup requirements and API services to generate beneficial feature interaction diagram between features. Thirdly, the beneficial feature interaction diagram is used as input of the graph neural network to predict and generate the API services set of recommendations for the Mashup requirements. Finally, the experiment on ProgrammableWeb dataset shows that the AUC of the proposed method has increased 20%, 24%, 27%, 13% and 21% respectively than that of AFM, NFM, DeepFM, FLEN and DCN, which means the proposed method improves the accuracy and quality of service recommendation.

Keywords: Recommendation · Beneficial feature interaction · L0-Predictin · GNN

1 Introduction

An API is an application programming interface that performs a specific function and allows software developers to call the API according to their requirements. Due to the

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 58–72, 2022.
https://doi.org/10.1007/978-3-031-24383-7_4

diversity and complexity of requirements, software developers often need to look up and call the API many times, which undoubtedly increases the time cost and complexity of software development. Mashups help software developers to achieve the functionality they need more easily and quickly by combining multiple APIs with different functionalities. The number of APIs has increased rapidly in recent years. As of 2021, for example, there were more than 24,073 APIs available on the Programmable Web platform. It is a challenge to quickly and effectively find the API services from such a large-scale service collection that meet the needs of Mashup for developer users.

To solve above problems, some researchers exploit service recommendation technology to improve the accuracy of service discovery. There is no doubt that how to recommend appropriate APIs to Mashup developers in the process of Mashup construction or update has become a hot topic in the field of service computing. In recent years, with the rapid development of neural network, some researchers begin to use graph neural network technology to carry on the recommendation process. For example, Zheng et al. [1] proposed a kind of deep cooperative neural network model, two parallel neural network models are used to learn the hidden features of users and items, and then an interaction layer is constructed on the two neural networks to predict the users' scores of items. Liu et al. [2] studied the problem of behavior prediction in location-based social networks, by using recurrent neural networks to grasp the dependence of sequential behavior, and to predict the behavior of the next moment based on the user's historical behavior sequence. When using graph neural network to make recommendation, because of the subjectivity of constructing input graph, graph neural network may not achieve the best effect, thus affecting the performance of recommendation. To solve this problem, this paper proposes a module to generate input graph automatically, so as to improve the accuracy and applicability of graph neural network.

According to our survey, there is no clear standard definition of how Mashups and APIs interact with each other in the existing research on Mashup-oriented API recommendations, whereas in previous research on graph neural networks, feature interactions are usually constructed based on relevant experience or personal judgment. This paper holds that the interaction between features can be divided into beneficial feature interaction and unbeneficial feature interaction. When extracting the beneficial feature interaction correctly, it is beneficial to improve the training accuracy of graph neural networks, when extracting the unbeneficial feature interaction, the feature relation graph based on this will adversely affect the training accuracy of the graph neural network. For example, in Mashups and APIs data sets, Mashup_category and API_category are a pair of beneficial features, while Mashup_Name and API_Name are a pair of unbeneficial features. Obviously, it is reasonable to recommend the relevant type of API to a Mashup based on its type, while it is not reasonable to recommend an API based on the name of the Mashup and API. In order to improve the accuracy of the recommendation model, this paper proposes an API recommendation method based on the beneficial feature interaction pair, which uses the edge prediction component to extract the beneficial feature interaction and removes the unbeneficial feature interaction. Specifically, the main contributions of this paper are as follows:

- Construct interactive graph of beneficial features exploiting edge prediction components. By using an edge prediction component L0, the beneficial feature interactions

between the input features of Mashup requirement and API services are extracted to progressively generate a graph structure suitable for graph neural network inputs and improve the accuracy of graph neural network training.
- Propose a method based on graph neural network to predict and recommend API services. The beneficial feature interaction graph is used as the input of the graph neural network, and the node vector representation of the graph neural network is updated to predict and get the recommendation set of the Mashup requirement- oriented APIs.
- Perform the comparative experiments based on the data crawled from ProgrammableWeb platform. The experimental results show that this proposed method has better recommendation quality than other state-of-art methods.

2 Related Work

The related work consists of two parts, i.e., traditional Web service recommendation method and deep learning-based Web service recommendation method. The traditional Web service recommendation methods usually explore features from users and items, and design recommendation system according to the similarity calculated by these features. Deep learning-based Web service recommendation methods exploit deep learning model and technology, such as LSTM, GCN, FNN, to devise recommendation system.

2.1 Traditional Web Service Recommendation Method

The traditional Web service recommendation methods mainly explore the functional feature, non-functional feature and hybrid feature for recommendation. Some researchers have developed the recommendation method by using a document topic generation model to improve the accuracy of service similarity matching [3, 4]. The document topic generation model trains Web service description documents to obtain the distribution of potential topics that represent the semantic information of their functions, and from this, the similarity between Mashup requirements and API description documents can be calculated. Web service recommendation considering non-functional features usually exploits users' activity and historical interaction information to learn users' preferences and uses collaborative filtering, matrix factorization and other techniques to predict missing QoS values, achieving the goal of recommending high-quality Web services. Collaborative filtering-based Web services recommendation [5–7]. Yao et al. [7] incorporated users, topics, and service- related latent features into service discovery and recommendation. QoS-based Web service recommendation methods [8–10] are more likely to predict missing QoS values and then recommend high-quality services to users. Rendle et al. [11, 12] devised a factorization model to alleviate the data sparsity problem. The model can handle any length and any number of eigenvectors. This means that the feature information fused by the model is multi-dimensional, which can reduce the influence of sparse data on recommendation performance. In the process of Web service recommendation, the combination of service functional characteristics and nonfunctional characteristics can comprehensively describe the association relationship between services. Therefore, Web service recommendation based on hybrid features has become more and more popular [13–18]. Lu et al. [13] raised a novel Web service

recommendation method based on the functional characteristics and QoS information. Cao et al. [14] explored a two-layer topic model, which aggregates the text information and network information of services to enhance the effect of service clustering, and then improve the recommendation accuracy of Web APIs. Gao et al. [15] first developed an API recommendation method based on diverse learning, and then proposed a service recommendation method by integrating the user's preference, service, Mashup, and topic [16].

2.2 Deep Learning-Based Web Service Recommendation

With the rapid development of deep learning technology, Web service recommendation based on deep learning has become a hot topic in recommendation system. In recent years, the use of GNNs for service recommendation has become a new direction. Zheng et al. [8] designed a deep cooperative neural network model, which uses two parallel neural network models to learn the hidden features of users and items, then an interaction layer is built on the two neural networks to predict the user's score on the items. Liu et al. [18] studied the problem of behavior prediction in location-based social networks, by using recurrent neural networks to capture the dependence of sequential behavior and predict the behavior of the next moment based on the user ' s historical behavior sequence. He et al. [19] designed the common framework of neural CF (NCF) based on the classical deep learning model. Sun et al. [20] developed a recommendation system based on long and short-term network which collects user preferences. In recent years, some studies focus on combination of neural network and knowledge graph for recommendation. Zhu et al. [21] developed a knowledge-aware attentional reasoning network for recommendation. Consequently, there are many studies has been conducted about deep learning-based Web service recommendation. Some representative studies of them include Web service recommendation based on deep neural network, graph neural network, knowledge graph and some other model. In the study of this paper, we believe that the structure of input graph has an influence on the accuracy of graph neural network. So, we use an edge prediction module to generate input graph automatically to make the graph neural network achieve more better recommendation quality.

3 Method

The framework of the proposed approach is shown in Fig 1, which consists of four components, i.e., data pre-processing, API similarity computation, L0-SIGN model construction, and API recommendation.

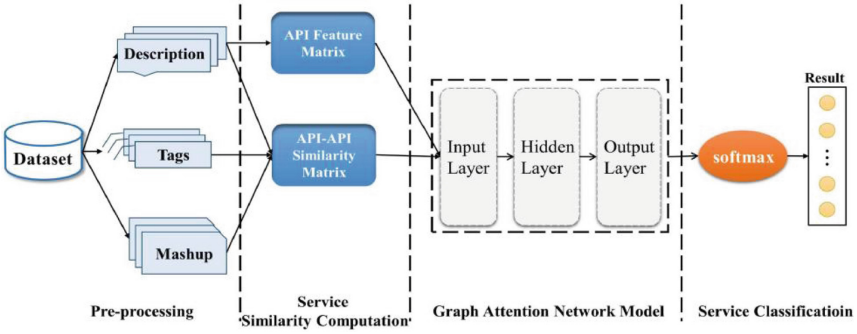

Fig. 1. The framework of service classification

3.1 Pre-processing

We use a dataset on mashups and APIs crawled from the ProgrammableWeb platform, and perform data pre-processing as follows:

Text normalization. Uniform case formatting of all words in the dataset, and remove all punctuation from the API description text.

Eliminate stop words. Remove function words and stop words from the API description text, leaving only the meaningful words for later processing.

Word stemming. Remove inflections or suffixes from the same word in the API description text. The same word may have different forms of expression because of its tense or part of speech. For example, the words "manage", "managed", and "management" appear in the description text, but all of their roots are "manage". We only extract their stems that can improve their computational accuracy in subsequent similarity calculations.

Invalid message culling. In the dataset, there are some APIs lack of function annotation and description document, which makes these APIs hard to calculate their similarity with other APIs, and also leads to the missing of input features. Therefore, we regard these APIs as invalid APIs information and cull them.

3.2 API Similarity Calculation

An API usually includes two important attributes, i.e., function annotation and description text. They can be used as a measure of API similarity. Specifically, (1) With API function annotation, each API adds labels according to its functions, and the similarity between APIs can be calculated through the labels of APIs. If two APIs share more tags, we determine that the two APIs have a higher similarity; (2) With API description text, each API adds a description text according to its function. The description text usually introduces the functions realized by the API in detail. We can measure the similarity of the corresponding API by comparing the similarity of the description text. The more similar the description texts are, the higher the similarity of the corresponding API. In order to express the similarity between APIs as accurately as possible, we designed a weighted similarity based on tag similarity and description similarity to express the similarity between APIs.

3.2.1 Functional Annotation Similarity of API

Tag refers to some descriptive words specified by users for various resources on the Internet, such as web services, music, video, and so on. Similar to keywords, tags also reflect the description and generalization of resources. However, keywords usually come from the resource content itself. They are keywords extracted from resources and can summarize and describe the content of resources to a certain extent. Tags are descriptive words specified by users for resources according to their understanding. They are unconstrained and subjective. Therefore, the tag may come from the content of the resource, or it may be a general word that does not appear in the resource.

The tags of API services usually summarize the functional attributes of services. For this reason, when two services have a shared annotation relationship, they can be considered to have a certain degree of similarity in functionality. In the API dataset, most APIs have at least three tags, so we use the Jaccard similarity coefficient to calculate the tag similarity between each two APIs. For example, the Jaccard similarity coefficient of service i and service j is shown in formula (1):

$$\mathrm{Jac}(a_i, a_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|} \tag{1}$$

where T_i, T_j are the tags owned by service a_i and a_j, $|T_i \cap T_j|$ represents the number of common tags owned by two services, $|T_i \cup T_j|$ represents the union of the number of tags owned by two services respectively.

3.2.2 Description Text Similarity of API

In data preprocessing, we make all cases in the description documents uniform, and remove the stop words in advance, while only meaningful word information is retained, to calculate the text-similarity of description documents between APIs more accurately. After completing the preprocessing, we use the doc2vec model to convert the API description text into the corresponding vector, and then use the cosine similarity to calculate the description text similarity between APIs, which can be denoted as below:

$$\mathrm{Dsim}(a_i, a_j) = \frac{v(a_i) * v(a_j)}{|v(a_i)| * |v(a_j)|} \tag{2}$$

where $\mathrm{Dsim}(a_i, a_j)$ represents the text-similarity of the description document of a_i and a_j, and $v(a_n)$ represents the generation vector of the description text of the *n*-th API.

3.2.3 Overall Similarity of API

By integrating the functional annotation similarity and description text similarity between APIs, the overall similarity between APIs will be obtained as:

$$\mathrm{sim}(a_i, a_j) = w_1 * \mathrm{Jac}(a_i, a_j) + w_2 * \mathrm{Dsim}(a_i, a_j) \tag{3}$$

where $\mathrm{sim}(a_i, a_j)$ represents the overall similarity between a_i and a_j, w_1, $w_2 \in (0,1)$,$w_1 + w_2 = 1$. To ensure the accuracy of overall similarity calculation between APIs, this paper will set different weight combinations for experiments to determine their optimal values.

3.3 L_0-sign Model Construction

The core idea of the L_0-sign model is to eliminate the unnecessary information between input features by exploiting information bottleneck theory, to retain only the beneficial interaction between features. In the L_0-sign model, firstly, an edge prediction component is used to automatically extract useful feature interaction pairs, and based on this, an appropriate feature interaction graph is generated as the input of the sign module. Then, the sign module updates the feature vector representation based on the structure of the feature interaction graph. Finally, a softmax function is used to generate a binary value to represent the recommendation relationship of the API [22].

3.3.1 Information Bottleneck Theory

Information bottleneck theory states that when facing a problem, people will try to use the least information to complete it [23]. The information bottleneck theory holds that the neural network will squeeze the information out of a bottleneck, to remove the input data containing noise, and retain only the characteristic data most related to the prediction target.

Information bottleneck theory aims to extract the most relevant information that input random variables X contains output variables Y by considering a trade-off between the accuracy and complexity of the process. The relevant part of X over Y denotes S. Information bottleneck theory can be represented as:

$$\min(\mathrm{I}(\mathrm{X};\mathrm{S}) - \beta \mathrm{I}(\mathrm{S};\mathrm{Y})) \tag{4}$$

where I is a function to denote mutual information between two variables and β is a weight factor.

A neural network often contains multiple functional layers with different functions. For instance, a simple convolutional neural network often includes a convolution layer, pooling layer, and full connection layer. When the neural network is trained with specific input x and label y, the training process can be understood as a process of weight combination. In the training process, the neural network constantly adjusts the weight of each functional layer, so that the information mostly related to the input x and the corresponding label y can be retained. According to the information bottleneck constraint, through multiple iterative training processes, the neural network can gradually squeeze out the information related to input x but not related to prediction y, to retain only the information related to input x and prediction value y, so that the neural network can complete the prediction function [24].

3.3.2 Edge Prediction Module

According to the constraints of information bottleneck theory, this paper chooses to use a neural network to construct an edge extraction module L0, to remove the feature interaction including noise that will lead to the decline of prediction effect, that is, useless feature interaction. At the same time, a model based on matrix factorization is used to extract useful feature interaction pairs. The module will determine the edge extraction according to the final prediction accuracy of the GNN. For example, if <

Mashup_Name, API_Name > is a pair of useless feature interactions. When inputting the set of feature interactions, the final test accuracy of GNN will decline, and the model effect will be affected. Therefore, the module will automatically eliminate < *Mashup_Name, API_Name* >. To achieve this, in the L0 edge prediction module, we first represent each feature as a *k*-dimensional vector, and then judge the interaction relationship of each group of features to determine whether it belongs to a beneficial feature interaction relationship or useless feature interaction relationship. For a set of feature interaction relations *P*, an edge prediction function based on matrix factorization is used to judge their interaction relation, i.e., $Z_{ij} = f_{ep}(v_i, v_j)$, where $(v_i, v_j) \in P$. The prediction function takes two b-dimensional vectors as inputs and outputs a binary value to indicate whether the feature interaction relationship belongs to beneficial feature interaction relationship. The input vectors in the prediction function is $v_i = O_i * W$, where O_i is the embedding vector of node *i*, and is a matrix parameter. During the process of model training, we reduce the number of useless feature interaction pairs through L0 edge prediction module, so as to generate an effective feature relationship graph for graph neural network as input to improve its accuracy.

We leverage activation regularization to link model with Information bottleneck theory to ensure the success of the interaction detection, which is shown as the below formula (4):

$$R(\theta, \omega) = \frac{1}{N}\sum_{n=1}^{N}(L(F_{LS}(G_n(X_n); \omega, \theta), y_n)) + \lambda_1 \sum_{i,j \in X_n}(\pi_n)_{ij} + \lambda_2 z_n \quad (5)$$

where θ and ω are the parameters in model, π_n is the probability of *(en)ij* being 1 (which means feature *i* and *j* is beneficial interaction feature pairs), λ *1* and $\lambda 2$ are the weight factors, and *L* corresponds to a loss function which minimizes loss and adjusts θ, ω.

3.3.3 SIGN Module

In the L0 edge prediction module, this paper takes the features of Mashup and API as the input, and extracts the beneficial feature interaction pairs as the input of sign module. In the sign module, firstly, each node is represented as a d-dimensional node embedding vector. The interaction of each pair of beneficial features is analyzed. The edge analysis function $h(u_i, u_j)$, is used to analyze node i and node j, where $= u_i = x_i * v_i$, v_i is the *d*-dimensional embedded vector of node i. The analysis result of the function is defined as, and the node embedding vector is updated with the result. Then, for each node, the embedded vector value of the node is updated by aggregating all the analysis result values *z* of its neighbor node. Finally, a linear function is used to generate a binary value as the final output of the sign module according to the finally updated node embedding vector, namely:

$$Y = f_{LS}(G(X_n, \varnothing); \theta, \omega) = f_S(G_n(X_n, F_{ep}(X_n; \omega); \theta)) \quad (6)$$

where, Y indicates the final prediction result of SIGN and it is calculated by a softmax function. We use a softmax function to calculate the possibility of each category, so as to complete the classification. When its value is 1, it indicates that the API is recommended to the Mashup, and when its value is 0, it is the opposite. X_n represents all input feature

nodes of Mashup and API, ω and θ is the relevant parameters of the model. f_{LS} means L0-SIGN which is able to extract edges and it should perform as well as the SIGN whose input is an appropriate graph which represented by f_S.

3.3.4 API Recommendation

In the process of Mashup-oriented APIs recommendation, the features of Mashup and API will be used as input, and a large number of feature data and interactive relationships are used to train the L0-SIGN model, so that the model can accurately predict the results of API recommendation. The specific process is as follows:

Firstly, Mashups are classified according to their characteristics, and the number of Mashups under each category is counted. On top of this, the number of Mashups under each category is sorted, and the Top-*k* Mashup categories are selected as Mashup classification data.

Secondly, according to the calling relationship between Mashup and API and the similarity between APIs, the recommendation relationship between Mashup and API is jointly mined and considered. According to experience, the API called by each Mashup is selected and the API with the strongest correlation is recommended, and the negative samples in other APIs are randomly generated, to build experimental data with recommendation relationship and train the L0-SIGN model.

Then, in the training process of the L0-SIGN model, the Mashup and API features are used as input, and the recommendation relationship is constructed as labels to train the L0 edge prediction module, so that the L0 edge prediction module can correctly extract the beneficial interaction between features and generate the beneficial feature interaction graph.

Finally, the beneficial feature interaction graph is used as the input of the SIGN module. In the SIGN module, the GNN iteratively updates the vector embedding of each feature according to the graph structure relationship. The softmax function is used to output a binary value y', which represents the recommendation relationship between the Mashup and the API, to achieve the API recommendation for Mashup requirements.

4 Experiment

4.1 Dataset Description and Experimental Setup

We crawl 17,783 Web APIs from ProgrammableWeb platform as the dataset source of service classification. For each Web API, its information includes service name, description text, category, tags and other information. Because the experimental dataset is too large, the top 10, 15, 20, 25 and 30 categories with the largest number of Web APIs are selected as the experimental dataset. The distribution of the top 30 categories with the largest number is shown in Table 1. During training, we randomly reorganized the experimental data, and then 60% of the dataset is selected as the training set, 20% as the verification set and 20% as the test set. Adam [26] method is used as the optimizer of the model. The learning rate is equal to 0.005, the batch size is 1, the number of attention heads is 8, and service similarity threshold is set to 0.8.

Table 1. Top 30 categories order by number

Category	Number	Category	Number	Category	Number
Tools	850	Telephony	338	Games	240
Financial	758	Reference	308	Photos	228
Messaging	601	Security	305	Music	221
eCommerce	546	Search	301	Stocks	200
Payments	526	Email	291	Cloud	195
Social	501	Video	289	Data	187
Enterprise	472	Travel	284	Bitcoin	173
Mapping	437	Education	275	Other	165
Government	369	Transportation	259	Project Management	165
Science	368	Advertising	254	Weather	164

4.2 Baselines

AFM [25]**.** An improved model designed by Jun Xiao [25] based on the deficiency that FM uses the same weight for different cross features. AFM model uses an attention network to better learn the feature interaction relationship, to improve the accuracy of the model prediction.

NFM [26]**:** A model presented by Xiangnan He by improving the FM model based on the intersection problem of sparse features. Its main feature is that after the embedding layer, a Bi-interaction operation is innovatively proposed to cross-process the features, to reduce the network complexity and accelerate the network training process.

DeepFM[27]**:** A model combining a deep neural network and factor decomposition machine. It uses the deep neural network part to do the high-order combination between features, exploits FM to do the low-order combination between features and combines the two methods in parallel..

FLEN [28]**:** A Spatio-temporal and efficient large-scale prediction model that uses field information to alleviate gradient coupling. It applies the field-wise Biinteraction pooling technology to solve the dilemma of time complexity and space complexity in the large-scale application of feature field information. At the same time, a method dice-factor to alleviate the gradient coupling problem is proposed.

DCN [29]**:** A model for Ad-click prediction proposed by Google and Stanford University in 2017 [29]. It is very efficient in learning specific order combinations, does not need feature engineering, and the additional complexity introduced is quite small.

4.3 Expriment Results and Analysis

4.3.1 API Recommendation Performance

Experimental results of API recommendation under different Mashup categories (i.e., 5, 10, 15, 20) are shown in Table 2. The best results are marked in bold. We draw Figure 2 clearly shows an improvement in the recommendation performance of the proposed method compared with the baseline methods under different mashup categories. The overall experimental results show that the proposed method is superior to other baseline methods in terms of ACC and AUC, and generally, AUC improvements are better than ACC.

The improvement in ACC is the most obvious in the case of 20 categories, and the AUC is improved when the number of categories is 15. Compared with AFM, NFM, DeepFM, FLEN, and DCN, the proposed method improves ACC by 12%, 19%, 21%, 9%, and 16%, respectively. Similarly, our proposed method improves AUC by 20%, 24%, 27%, 13%, and 21% on these same baselines. We attribute the performance of our proposed method to the fact that it considers the interaction relationship between features, which makes it easy to deeply mine the implicit information between features, resulting in better recommendation performance. The proposed method uses the beneficial interaction relationship between features to construct the feature interaction graph so that the neural network can deeply mine the implicit relationship between features. Doing so significantly improves the recommendation performance.

Table 2. Experimental results of different recommendation methods

Method	Number of Categories							
	5		10		15		20	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
AFM	0.85	0.90	0.90	0.92	0.84	0.81	0.83	0.88
NFM	0.86	0.82	0.86	0.82	0.82	0.78	0.78	0.85
DeepFM	0.85	0.84	0.87	0.85	0.79	0.82	0.77	0.77
FLEN	0.87	0.89	0.88	0.89	0.85	0.86	0.85	0.87
DCN	0.85	0.82	0.84	0.83	0.82	0.82	0.80	0.86
L0-SIGN	**0.95**	**0.99**	**0.94**	**0.98**	**0.93**	**0.97**	**0.93**	**0.98**

As shown in Table 2, when the number of Mashup categories is 5, the recommendation performance of the proposed method is the best. Most baseline methods can achieve the best recommendation effect when the number of Mashup categories is 10. The increase shown in the Fig. 2. Mashup categories, in turn, increases the number of individual services it contains so that the proposed method can mine more implicit information and improve its recommendation performance. However, continuously increasing the number of Mashup categories further increases the interference between Mashups, resulting in the decline in the recommendation performance of the proposed method.

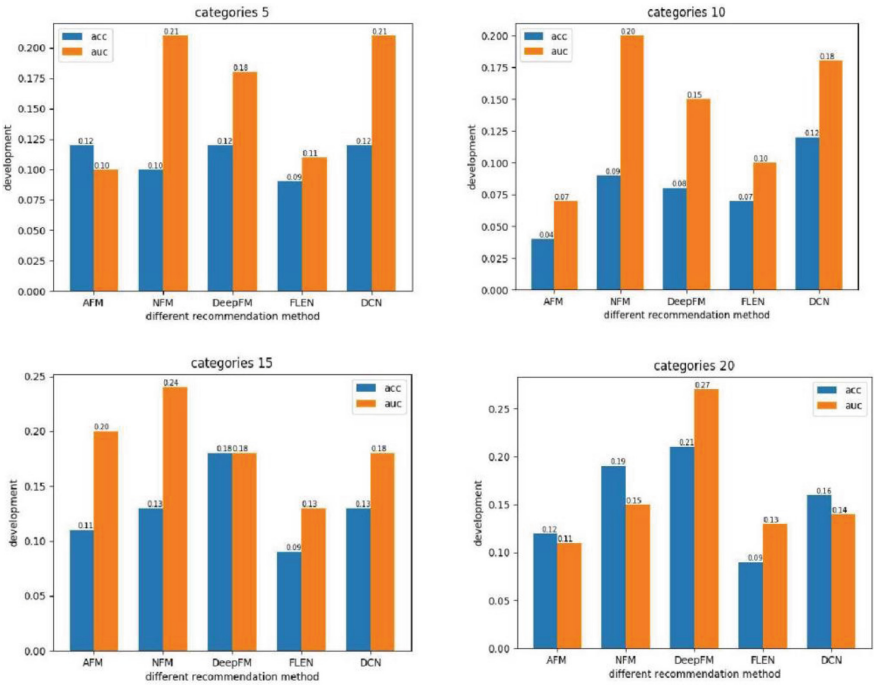

Fig. 2. The improvement of the recommendation performance of the proposed method compared with the baseline methods

Therefore, the recommendation performance of most baseline methods shows an overall trend of rising first and then declining, among which that of DeepFM decreases the fastest. The proposed method in this paper can eliminate the noise between features, and so reduce the decline of recommendation performance caused by the increase of Mashup categories to a certain extent.

4.3.2 Parameter Analysis

In this section, we analyze the influences of the weight coefficient w_1, w_2 in the overall similarity calculation between APIs and hidden layers of the model on recommendation performance in the proposed method (Table 3).

Influence of Different Weight Coefficient on API Recommendation. The different weight combinations in the calculation process of overall similarity indicate the proportion of functional annotation similarity and description text similarity. It can be seen from Table 3 that with the increase of the proportion of functional annotation similarity, the recommendation performance first increases and then decreases. When the proportion of functional annotation similarity is 0.7, the recommendation performance reaches the best. This is because the functional annotation or tag attribute of API can better represent its implied information than the description text, when the functional annotation accounts for a larger proportion, it can achieve a more recommendation effect. In addition, more

Table 3. Weight coefficient

W1: W2	ACC	AUC
0.1:0.9	0.90	0.96
0.3:0.7	0.95	0.98
0.5:0.5	0.95	0.99
0.7:0.3	0.96	0.99
0.9:0.1	0.95	0.98

Table 4. Hidden layers

Hidden Layer	ACC	AUC
8	0.89	0.96
16	0.95	0.99
32	0.94	0.99
64	0.92	0.97
128	0.91	0.96

importantly, the overall recommendation performance is relatively excellent and stable, which indicates that different weight coefficient of overall similarity calculation has a small impact on the proposed method.

Influence of Different Hidden Layers of the Model on API Recommendation. It can be seen from Table 4 that with the increase of the number of hidden layers in the model, its recommendation performance first increases and then decreases. When the number of hidden layers is 16, the proposed model and method can obtain the best recommendation performance, while when the number of hidden layers is 8, its recommendation performance is poor. Meanwhile, when the number of hidden layers is too large, the complexity of the model increases, making the model prone to over-fitting and resulting in a downward trend in the recommendation performance.

5 Conclusion and Future Work

Focusing on the problems existing in the service recommendation method using GNN, this paper proposes an API recommendation method based on the extraction of beneficial feature interaction pairs. Compared with the state-of-art recommendation methods based on GNN, this method firstly uses the edge prediction module to capture the interaction relationship between features. Then objectively constructs edges for beneficial feature interaction pairs and finally builds the input graph structure of GNN. The experimental results verify the effectiveness and advantage of the proposed method. In future work, we will consider exploring other neural networks such as graph convolutional networks, to further improve the performance of Web API recommendations.

Acknowledgment. Our work is supported by the National Natural Science Foundation of China (No. 61873316, 61872139, 61832014, and 61702181), the National Key R&D Program of China (No.2018YFB1402800), Hunan Provincial Natural Science Foundation of China under grant No. 2021JJ30274, and the Educational Commission of Hunan Province of China (No.20B244). Buqing Cao is the corresponding author of this paper.

References

1. Zheng, L., Noroozi, V., Philip, S.Y.: Joint deep modeling of users and items using reviews for recommendation. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pp. 425–434 (2017)
2. Liu, Q., Wu, S., Wang, L., et al.: Predicting the next location: a recurrent model with spatial and temporal contexts. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 194–200 (2016)
3. Fan, W., Ma, Y., Li, Q., et al.: Graph neural networks for social recommendation. The World Wide Web Conference, pp. 417–426 (2019)
4. Cao, B., Liu, J., Tang, M., et al.: Mashup service recommendation based on usage history and service network. Int. J. Web Services Res. (IJWSR) **10**(4), 82–101 (2013)
5. Klusch, M., Fries, B., Sycara, K.: OWLS-MX: a hybrid semantic web service matchmaker for OWL-S services. Web Semantics Science Services & Agents on the World Wide Web **7**(2), 121–133 (2009)
6. Xu, S., Raahemi, B.: A semantic-based service discovery framework for collaborative environments. Int. J. Simulation Modelling (IJSIMM) **15**(1), 83–96 (2016)
7. Yao, L., Sheng, Q.: Unified collaborative and content-based web service recommendation. IEEE Trans. Serv. Comput. **8**(3), 453–466 (2015)
8. Zheng, Z., Ma, H., Michael, R., et al.: Collaborative web service QoS prediction via neighborhood integrated matrix factorization. IEEE Trans. Serv. Comput. **6**(3), 289–299 (2013)
9. Chen, X., Zheng, Z., Yu, Q., et al.: Web service recommendation via exploiting location and QoS information. IEEE Trans. Parallel and Distributed Syst. **25**(7), 1913–1924 (2014)
10. Wang, X., Zhu, J., Zheng, Z., et al.: A spatial-temporal QoS prediction approach for time-aware web service recommendation. ACM Trans. Web **10**(1), 1–25 (2016)
11. Rendle, S.: Factorization machines. In: 2010 IEEE International Conference on Data Mining. Dec. 13- 17, pp. 995–1000 (2010)
12. Rendle, S.: Factorization machines with LibFM. ACM Trans. Intelligent Syst. Technol. (TIST) **3**(3), 1–22 (2012)
13. Lu, A.: Web service reputation evaluation model based on QoS and user recommendation. Yanshan University, pp. 18–26 (2010)
14. Cao, B., Liu, X., Rahman, M., et al.: Integrated content and network-based service clustering and web APIs recommendation for mashup development. IEEE Trans. Serv. Comput. **13**(1), 99–113 (2017)
15. Gao, W., Chen, L., Wu, J., et al.: Manifold-learning based API recommendation for mashup creation. In: 2015 IEEE International Conference on Web Services, June. 27-July. 2, pp. 432–439 (2015)
16. Gao, W., Chen, J.W., et al.: Joint modeling users, services, mashups, and topics for service recommendation. In: 2016 IEEE International Conference on Web Services (ICWS), June. 27-July. 2, pp. 260–267 (2016)

17. Xia, B., Fan, Y., Tan, W., et al.: Category-aware API clustering and distributed recommendation for automatic mashup creation. IEEE Trans. Serv. Comput. **8**(5), 674–687 (2015)
18. Liu, X., Fulia, I.: Incorporating user, topic, and service-related latent factors into web service recommendation. In: 2015 IEEE International Conference on Web Services (ICWS), June 27-July 2, pp. 185–192 (2015)
19. He, X.N., Liao, L.Z., Zhang, H.W., et al.: Neural collaborative filtering. In: Proceedings 26th International Conference on World Wide Web, pp. 173–182 (2017)
20. Sun, K., Qian, T., Chen, T., et al.: Where to go next: modeling long- and short-term user preferences for point-of-interest recommendation. In: National Conference on Artificial Intelligence Association for the Advancement of Artificial Intelligence (2020)
21. Zhu, Q., Zhou, X., Wu, J., et al.: A knowledge-aware attentional reasoning network for recommendation. In: National Conference on Artificial Intelligence Association for the Advancement of Artificial Intelligence (2020)
22. Su, Y., Zhang, R., Erfani, S., Xu, Z.: Detecting beneficial feature interactions for recommender systems. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI) (2021)
23. Tishby, N., Pereira, F., Bialek, W.: The Information Bottleneck Method. arXiv preprint physics/0004057
24. Louizos, C., Welling, M., Kingma, D.: Learning Sparse Neural Networks through L_0 Regularization. arXiv preprint/1712 01312
25. Xiao, J., Ye, H., He, X., et al.: Attentional factorization machines: learning the weight of feature interactions via attention networks. In: proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI), pp. 3120–3125 (2017)
26. He, X., Chua, T.: Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, Aug, pp. 355–364 (2017)
27. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. arXiv preprint arXiv:1703 04247
28. Chen, W., Zhan, L., Ci, Y., et al.: FLEN: Leveraging Field for Scalable CTR Prediction. arXiv preprint arXiv:1911 04690
29. Wang, R., Fu, B., Fu, G., et al.: Deep&Cross Network for Ad Click Predictions. In: Proceedings of the ADKDD'17, August, pp. 1–7 (2017)