Journal of Computers Vol. 34 No. 6, December 2023, pp. 63-74
doi: 10.53106/199115992023123406005

# Service Recommendation Method Based on Multi Model Fusion

Ting Yu*, Lihua Zhang, Hongbing Liu

School of Information Engineering, Jiaxing Nanhu University, Jiaxing, China
yuting@jxnhu.edu.cn

*Received 17 February 2023; Revised 2 August 2023; Accepted 23 August 2023*

**Abstract.** In recent years, the rapid development of service-oriented computing technology has increased the burden of choice for software developers when developing service-based applications. Existing Web service recommendation systems often face two challenges. First, developers are required to input keywords for service search, but due to their lack of knowledge in the relevant field, the keywords entered by the developers are usually freestyle, causing an inability to accurately locate services. Second, it is exceedingly difficult to extract services that meet the requirements due to the 99.8% sparseness of the application service interaction records. To address the above challenges, a framework for service recommendation through multi-model fusion (SRM) is proposed in this paper. Firstly, we employ graph neural network algorithms to deeply mine historical records, extract the features of applications and services, and calculate their preferences. Secondly, we use the BERT model to analyze text information and use the attention mechanism and fully connected neural networks to deeply mine the matching degree between candidate services and development requirements. The two models mentioned above are further merged to obtain the final service recommendation list. Extensive experiments on datasets demonstrate that SRM can significantly enhance the effectiveness of recommendations in service recommendation scenarios.

**Keywords:** service recommendation, multi model fusion, application, recommendation algorithm

## 1 Introduction

Due to the rapid development of the Internet, service recommendation technique has now become the norm for creating web-based applications. Developers can concentrate more on solving the specific development needs of applications instead of writing code from scratch, which substantially liberates developers' creativity. Nevertheless, choosing the best services for an application is quite difficult due to the continuous growth of services available in the Web application programming interface directory (like ProgrammableWeb). Therefore, one of the important technologies for creating new applications is still learning how to recommend related services.

Several service recommendation methods have been proposed by researchers during the last ten years [1-5]. These approaches can be broadly categorized into three groups: collaborative filtering (CF)-based service recommendation methods [6, 7], semantic-based service recommendation methods [8, 9], and graph-based service recommendation methods [10-12]. The methods based on collaborative filtering and semantic are still lacking in feature fusion and usage while creating the service recommendation model. In the event of sparse data, the CF-based method can benefit from the knowledge of related applications and can produce the specific results. However, it doesn't fully utilize the text information between services and applications. The semantic correlation between applications and services is estimated via description documents. Yet, if the text does not convey enough details, the service recommendation's results won't be satisfactory.

The plethora of potential services often makes it difficult for developers to choose, particularly for those who are not familiar with the services. For example, if a developer intends to complete an application with three functions {mapping, message, payment}, they first enter these three keywords and then use the keywords to search for a set of candidate services from the service repository. Finding the optimal combination out of so many services is known as the NP-hard problem, as numerous scholars have noted. A crucial issue in this scenario is figuring out how to recommend services to application developers. Therefore, although there has been a lot of research on this topic, there are still have the following challenges:

(1) Users (Developers) are not professionals in the service field and cannot accurately describe the functions of the required services. This leads to some semantic deviation between the development requirements provided by users and the service description.

* Corresponding Author

(2) There are many services in the service repository, and the proportion of the invoked services is very small. Most services are immersed in the repository, and the application service invocation record is very sparse, making it difficult to find services.

To address the above challenges, we proposed a new framework for service recommendation, which includes two components: semantic interaction model and historical interaction model. Considering that deep learning technology has made great progress in natural language processing, we employed the BERT model to extract the semantic features of description and tag information and converted them to text vectors, and we further use attention mechanisms to extract text features. It is difficult to find mathematical functions for the interaction between applications and services, graph neural network technology is employed in this framework to learn semantic interaction and historical interaction between applications and services. Finally, we integrate the features learned from semantic interaction model and historical interaction model to make more accurate service recommendations. The main contributions of this paper are summarized as follows:

(1) We propose a service recommendation framework, called SRM, which can recommend suitable candidate services to meet the development requirements.

(2) We propose a semantic interaction model to capture the semantic relation between applications and services. We employ the BERT model, attention mechanism and the deep Neural network to mine the inherent semantic connections between applications and services.

(3) We propose a historical interaction model to capture the interaction between applications and services. We use the LightGCN model to fully extract potential features of applications and services to obtain application preferences for the candidate services.

(4) The experimental results show that our service recommendation framework can effectively improve the service recommendation effect.

The remainder of this paper is organized as follows: Section 2 describes the related work. Section 3 introduce the service recommendation framework that we proposed. Section 4 presents the dataset, the experimental results, and the discussion. This paper is concluded in Section 5.

## 2 Related Work

Service recommendation aims to recommend suitable candidate services for developers. Here, we divide the related work into three parts according to its principle and emphasis: the semantic-based service recommendation method, the collaborative filtering-based service recommendation method, and the graph-based service recommendation method.

### 2.1 Semantic-based Service Recommendation Method

Given the development requirements of applications, the semantic-based method obtains the service list according to the text similarity between the service description and the development requirements of the applications. To extract the most common semantics from the service set, Naïm et al. [13] coupled probabilistic topic modeling with pattern mining, thoroughly examined the service description, and proposed a semantic extraction approach. Gu et al. [14] employed the topic model to build the semantic service package repository and proposed a service package recommendation model based on composite semantics. However, due to the small amount of data, large data noise, ignoring word order and other reasons, the performance of the topic model is not satisfactory. With the great success of pre training language models (PLMs) and deep learning models in natural language processing (NLP), some researchers began to use PLMs and deep learning methods to extract the semantic features. For example, inspired by successful experience of the BERT model in natural language processing, Qiu et al. [15] proposed a new U-BERT model based on pre training and fine-tuning, which uses comments from content rich domains to improve the recommendation for domains with insufficient content. Shi et al. [16] extracted three types of word relations to build a word map based on natural language and built a new text content representation method through the word map. However, this method is only applicable to news recommendation, and its generalization and adaptability are weak in other scenarios. The above literature mainly focuses on the functionalities of services, without considering how important the invocation history records are for service recommendations. Moreover, methods based on text similarity or topic similarity usually require mining the text description features of the application and service, service description summary files are either missing or very limited. The lack

of text descriptions makes it difficult to obtain good recommendation results.

### 2.2 Collaborative Filtering-based Service Recommendation Method

Collaborative Filtering (CF)-based methods complete service recommendation task by taking account to the historical data of the other users and services, in addition to the target user and the target service [17-19]. For example, Yao et al. [20] proposed a probability matrix factorization method with implicit correlation regularization, which allows users to identify the most suitable service in composite tasks and enhance the diversity of service recommendations. Although the matrix decomposition method can share the experience of similar applications and can achieve certain results in the case of sparse data, it does not pay enough attention to the development requirements themselves. To overcome the above problems, Nguyen et al. [21] proposed the attention probability matrix factorization model and injected the attention score and development requirement context similarity into the matrix factorization structure for training to obtain the service recommendation list. Historical invocation records and related auxiliary information between applications and services also play a key role in service recommendation. Ma et al. [22] employed recurrent neural networks to develop the service recommendation system, which relies on the powerful representation learning function provided by deep learning and extract hidden structures and features from various types of interactions between applications and services. The method based on collaborative filtering can achieve certain results in the case of sparse data, but it does not pay enough attention to the development requirements and service description. Previous CF-based approaches have used connections between applications and services directly or indirectly to promote services, but they have ignored or failed to investigate the high-order connectivity between applications and services. By investigating the higher-order connectivity from the application-service bipartite graph, our work closes this gap.

### 2.3 Graph-based Service Recommendation Method

Since graph topologies can clearly explain the complex interactions between entities, the graph-based service recommendation system has attracted interest as a new research area. The graph-based service recommendation method projects applications and services into a potential shared space and uses potential feature vectors to represent applications and services. For example, Xiong et al. [23] applied natural language processing and graph embedding technology to recommend services for mashup developers. They learned from the two patterns of mashups, services, and their relationships to obtain features for service recommendation. Besides, Wang et al. [24] proposed an unsupervised service recommendation method based on the depth of a random walk of the knowledge graph. This method uses the relationships between services and mashups to build a knowledge graph. It uses the Skip Gram model to achieve the implicit embedding of each node, calculate the correlation between mashup nodes and service nodes, and finally obtain the service recommendation list. Moreover, a deep knowledge-aware service recommendation approach, DKWSR, was proposed by Dang et al. [25] that learned entity knowledge embedding and text embedding using the knowledge graph representation learning method TransH and Word2vec, respectively. To learn the probability of users calling potential services, the obtained user vector representation and service vector representation were coupled and transmitted to a DNN neural network.

At present, there aren't many literatures on graph neural networks currently being used for service recommendation. However, graph neural networks have been used in recommendation systems with great success and have served as a useful model for service recommendation. For example, Jiang [26] proposed a machine learning method based on Bayesian personalized sorting (called HeteLearn) to learn the weight of links in a HIN (heterogeneous information network). To simulate users' preferences for personalized recommendations, they proposed a generalized random walk model with a restart on HINs. Besides, Neural Graph Collaborative Filtering (NGCF), developed by Wang et al. [27], is a brand-new recommendation system that explicitly encodes the collaborative signal in the form of high order connectivity by embedding propagation. They used the inner product to determine the user's choice for the target item after acquiring the final user and item embedding. The above work proves that graph-based service recommendation technology can help develop more efficient service recommendation models. In view of the above problems, this paper establishes the service domain graph to efficiently address the issue of data sparsity. At the same time, SRM model not only improves the accuracy of service recommendation by obtaining high-order information, but also makes the recommendation interpretable by fully understand develop requirements.

# 3 Recommendation Method based on Multi Model Fusion

Table 1 shows the definition of symbols that were used in this paper.

**Table 1.** The definition of symboles

| Symbol | Meaning |
|---|---|
| $a$ | An application |
| $A$ | Application set |
| $s$ | A service |
| $S$ | Service set |
| $vb_m$ | Description feature vector of the application |
| $vb_{mt}$ | Category feature vector of the application |
| $vb_s$ | Description feature vector of the service |
| $vb_{st}$ | Category feature vector of the service |
| $vs_m$ | Final text feature vector of the application |
| $vs_s$ | Final text feature vector of the service |
| $e_a$ | Structure feature vector of the application |
| $e_s$ | Structure feature vector of the service |

## 3.1 Framework

The structure of SRM is shown in Fig. 1. It contains three components: a semantic interaction model, a historical interaction model, and a fusion model. In semantic interaction model, we use the BERT model extracts the text feature representation of applications, services, and their tags. Then we use the attention mechanism to determine the importance of the description information and tag information to obtain the final text feature representation of the applications and services. Finally, we use the MLP model to further mine the relationship between the applications and services. The output layer outputs the probability of developers selecting candidate services based on text information. In historical interaction model, we employ LightGCN model to extract the structure features of applications and services. To further obtain the final service list, we fusion the above two models.

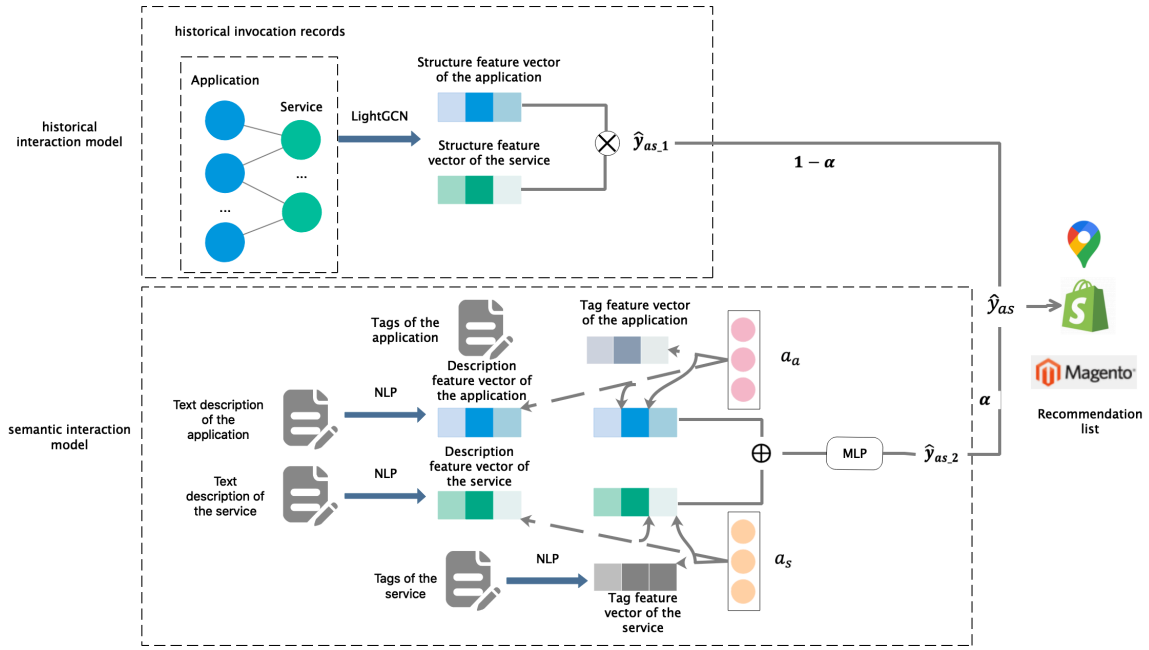



**Fig. 1.** The framework of SRM

### 3.2 Semantic Interaction Model

Compared to other metadata, text information specifications for applications and services are more detailed. In semantic-based recommendation methods, text modeling and feature representation are crucial. Developers will initially evaluate if the chosen service's text information can satisfy the requirements of the intended application when determining the service. To capture the interaction between them from the viewpoint of text information, we propose the semantic interaction model.

The BERT model [28] is used in the semantic interaction model to extract the textual description and tag features of applications and services. It is necessary to preprocess the text before utilizing deep learning technology to extract features. Text filtering, abbreviation substitution, and word form recovery are adopted to preserve crucial information in text.

With the BERT model, we convert the text requirements of the application, the tags of the application, the text of the service, and the tags of the service into text feature vectors, which are expressed as $vb_m$, $vb_{mt}$, $vb_s$ and $vb_{st}$.

Considering that the application's text requirements and the tags have different importance on the overall text feature generation, which means that the representation of development requirements and tag features should be assigned different weights. Here, the attention mechanism was used to integrate the representation of development requirements and tags. Assuming the features of development requirements and tags can be expressed as $M = [vb_m, vb_{mt}]$, through the attention mechanism, the final feature can be represented as $vs_m$. Here, we make $Key = Value = M = q$, Give attention distribution $a_i = softmax(s(key_i, q)) = softmax(s(M_i, q))$, Where $s(M_i, q)$ is the attention scoring mechanism of $M_i$ , the scaling point product model is used here, and the process can be represented as follows:

$$s(M_i, q) = \frac{M_i^T q}{\sqrt{d}}. \tag{1}$$

*Key*, *Value*, $q$ are the three specified inputs of the attention mechanism, where $M_i \in M$, where i = 0 or 1, $M_i$ is the description feature or tag feature of the application to be created, i.e., $M_0$ is the text feature of the application to be created (i.e., $vb_m$), $M_1$ is the tag feature of the application to be created (i.e., $vb_{mt}$). $a_i$ refers to the attention distribution of the text features or tag features of the application to be created, that is, $a_0$ refers to the attention distribution of the text features of the application to be created, and $a_1$ refers to the attention distribution of the tag features of the application to be created. $M_i^T$ is the transposition of $M_i$. $\sqrt{d}$ represents the scaling factor, it is the length of the text feature (that is, the length of the application text description feature vector $vb_m$). Finally, the final text feature vector $vs_m$ of the application is calculated, and the process can be represented as follows:

$$vs_m = att(q, m) = \sum_{i=0}^{1} a_i M_i. \tag{2}$$

Assuming the features of the service description and tags can be expressed as $S = [vb_s, vb_{st}]$, through the attention mechanism, the final feature of service can be represented as $vs_s$ . The calculation process is the same as that of the final text feature vector $vs_m$ of the application.

Given the feature vector $vs_m$ of the application and the feature vector $vs_s$ of the service, we connect them, the process is as follows:

$$vs_{ms} = vs_m \oplus vs_s. \tag{3}$$

We employ the MLP model to capture the text connection between applications and services. In addition, we choose the parameter correction linear unit (PReLU) as the activation function, because it can improve the model fitting with almost zero additional computing costs and little risk of over fitting. The learning process is as follows:

$$\hat{y}_{as\_2} = \delta(W_z(...\delta(W_1(vs_{ms})) + b_1)... + b_z). \tag{4}$$

Where $W_z$ and $b_z$ is the weight and bias vector. The advantage of the MLP model is that it can learn the inter-

action characteristics at different levels of abstraction. As the number of layers increases, the receptive field of each neuron becomes larger than that of the previous layer, so it can provide global semantics (global interaction) and abstract details, which is difficult to do in shallow and linear operations.

### 3.3 Historical Interaction Model

In addition to the text information, the historical invocation records between existing applications and services also help to obtain good service recommendation results. Through the historical invocation records between applications and services, we built an application-service bipartite graph, which can be expressed as $G = (V, E)$, $V$ represents node set, $V = A \bigcup S$, where $A = \{a_1, a_2, ..., a_p\}$ represents the application set, $S = \{s_1, s_2, ..., s_q\}$ represents the service set, $E$ represents the edge set, and $E = \{e_{h,j} = 1\}_{h,j=1}^{l} = E_{m,s}$. If an invocation occurs between application $a$ and service $s$, they are connected in $G$, thus forming $E_{m,s}$ edge set. The representation of existing applications and services can be obtained through LightGCN [29], which is a graph embedding method. Compared with traditional matrix factorization-based methods (such as probability matrix factorization and singular value factorization), LightGCN can capture more nonlinear relationships between applications and services. Specifically, we conduct information dissemination on the application service diagram. The information dissemination of layer $k$ can be expressed as follows:

$$e_a^{(k)} = \sum_{s \in N_a} \frac{1}{\sqrt{|N_a|}\sqrt{|N_s|}} e_s^{(k-1)}. \tag{5}$$

$$e_s^{(k)} = \sum_{a \in N_s} \frac{1}{\sqrt{|N_s|}\sqrt{|N_a|}} e_a^{(k-1)}. \tag{6}$$

Where $e_a^{(k)}$ represents that the k-th layer's information is propagated to application $a$, $e_s^{(k)}$ represents the k-th layer's information is propagated to service $s$, $a$ is one of an application in application set $\{a_1, a_2, ..., a_p\}$, $s$ is one of a service invoked by application $a$, $N_a$ is the neighbor of application $a$, $N_s$ is the neighbor of the service $s$. We remove the self-connection from the application service network graph and remove the nonlinear transformation from the information propagation function. The start point and end point of the self-connection node are the same. Nonlinear transformation refers to nonlinear activation function, which cancels feature transformation of application feature information and service feature information. Connect the features of all K layers to combine the information received from neighbors at different depths. The final application structure features and service structure features can be expressed as follows:

$$e_a = \sum_{k=0}^{K} e_a^{(k)}. \tag{7}$$

$$e_s = \sum_{k=0}^{K} e_s^{(k)}. \tag{8}$$

To obtain the recommended final list, we use the inner product to calculate, as shown below:

$$\hat{y}_{as\_1} = e_a^T e_s. \tag{9}$$

The traditional Bayesian personalized ranking (BPR) loss is used as the loss function of SRM, which can be expressed as follows:

$$L^{BPR} = \sum_{(a,s,s') \in Q} -In\sigma(y_{as_2} - y_{as'_2}). \tag{10}$$

Where $Q = \{(a, s, s') \mid a \in A, s, s' \in S, x_{as} = 1, x_{as'} = 0\}$, $\sigma(\cdot)$ is the sigmoid function. $y_{as_2}$ represents the similarity between the application $a$ and the service $s$, $y_{as'_2}$ represents the similarity between the application $a$ and the service $s'$. $x_{as}$ represents the service subset that invoked by the application $a$, $x_{as'}$ represents that the application $a$ and service $s'$ have not interacted before. Any service in service set $S$ that has not been invoked by application $a$ can be regarded as $s'$.

### 3.4 Model Fusion

To improve the accuracy of recommendations, we use a linear framework to combine the semantic interaction model and the historical interaction model, which can be expressed as follows:

$$y_{as} = \alpha \hat{y}_{as\_2} + (1-\alpha) y_{as\_1}. \tag{11}$$

Where $\alpha$ represents the importance of semantic interaction model. Each service has a corresponding $y_{as}$ value, if the corresponding $y_{as}$ value is higher, the service $s$ is more likely to be recommended to the application $a$. According to the final $y_{as}$ value of different services, the recommended list can be obtained.

## 4 Results and Discussion

In this section, we evaluated SRM performance on real-world datasets. Specifically, the experiment aims to answer the following three research questions:

**RQ1:** Is SRM better than other comparison methods in service recommendation tasks?

**RQ2:** How do the parameters in SRM affect the recommendation results?

**RQ3:** Can the semantic interaction model and historical interaction model in the SRM method capture information effectively?

### 4.1 Dataset

We have crawled 22,016 services and 6438 applications from ProgrammableWeb, the world's largest online Web service repository. Applications and services without function descriptions, services without invoked, and applications with fewer than three component services are deleted from the original dataset. The final experimental dataset contains 6347 applications and 1623 services. The sparsity of the application service invocation matrix is 99.87%. We randomly divide the whole data set into five parts, use one of them as the test set, and combine the other four parts as the training set. We run the algorithm five times, and finally take the average value of the five times as the result.

### 4.2 Evaluation Metric

In this paper, we choose two indicators, namely, precision and recall, to measure the service recommendation method, which are defined as follows:

$$\mathrm{Pr}\,ecision = \frac{|\,top_a(k) \cap test_a\,|}{k}. \tag{12}$$

$$\mathrm{Re}\,call = \frac{|\,top_m(k) \cap test_m\,|}{|\,test_m\,|}. \tag{13}$$

$top_a^{(k)}$ represents the first $k$ services recommended to application $a$. $test_a$ represents the service invoked by application $a$ in the test set.

### 4.3 Baseline

To evaluate the performance of the proposed SRM method, the following methods are selected for comparative experiments:

(1) Pop: The Pop model calculates the number of times each service is used in the application and sorts them according to their popularity, then recommend popular services for each application.

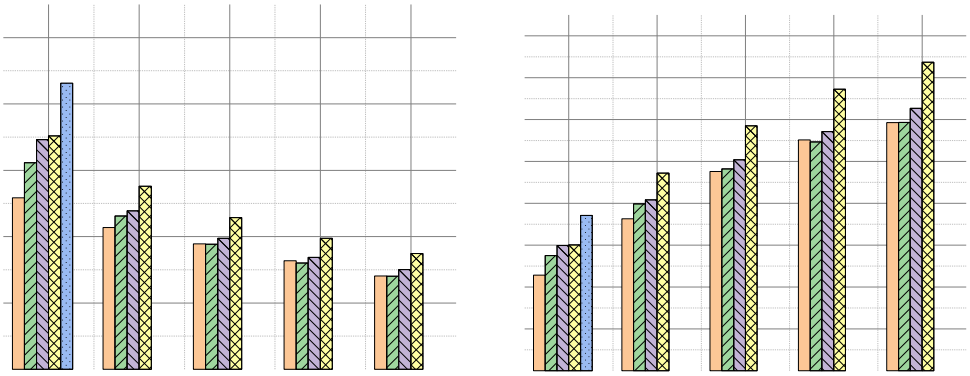
(2) NAIS [30]: The NAIS model combines the attention mechanism with the neural network to improve the prediction accuracy.

(3) APR [31]: The APR model combines BPR model with confrontation training to improve the robustness of the model.

(5) JCA [32]: The JCA model introduces a joint learning paradigm with paired loss, so that the automatic encoder model can capture the correlation between applications and services.

(6) NGCF [27]: NGCF explicitly constructs an application service bipartite graph to model higher-order connectivity and obtain more expressive representation of application and service characteristics.

### 4.4 Performance (RQ1)



(a) Precision (b) Recall

**Fig. 2.** SRM Experimental results of SRM model and comparison methods

In this section, SRM is compared with the comparison methods, the performance is shown in Fig. 2. Pop model has the worst performance among the comparison methods. Pop model simply counts the number of service invocation numbers and recommends the most popular service. The recommendation performance of JCA model is slightly better than that of Pop model. JCA deploys two separates classical autoencoders jointly optimized only by a single hinge loss function, and it does not capture the uncertainty of latent representations. The recommendation performance of NGCF model and NAIS model is better than that of JCA. The NGCF model learns the embedding representation of applications and services in the graph structure, so that the model can express high-dimensional features, while explicitly putting collaborative filtering signals into the embedding process. However, only multi-order neighbor information without semantic relations is used in information aggregation. NAIS model combines attention mechanism with neural network to recommend services. The output probability is the inner product of the application representation and the service representation. Obviously, this model is not ideal in the scenario of sparse application-service interactions because many unobserved applications in training set are represented by the average of all service embeddings. APR model introduces confrontation training in service recommendation scenarios, and its recommendation performance is only inferior to SRM. APR combines BPR with adversarial training methods to make the recommendation model more robust. From Fig. 2, we can see that the performance of the SRM model in this dataset is consistently better than other baselines. SRM improves over the strongest baselines w.r.t. Precision@1 by 13.4%; Recall@1 by 15.4%. By fully considering textual and historical invocation information, SRM can explore high order connectivity in an explicit way. Compared with the above baseline methods, we employ the BERT model to extract the semantic information features. At the

same time, we have introduced an attention mechanism in text information propagation, so that the information propagation between nodes has a designated weight, rather than a fixed weight. Meanwhile, we use LightGCN model for feature extraction from historical invocation records, which is easier to train and has better generalization ability compared to other neural network models such as APR and NGCF. The results indicate that our improvements have played a positive role in achieving better recommendations.

### 4.5 Parameter Analysis (RQ2)

(1) Influence of dimension $d$ on experimental results

Fig. 3 shows the impact of different dimension on the model performance. When the dimension reaches 80, the precision and recall reach the highest value. With the increase of dimension $d$, the precision and recall gradually decrease. The results on the change dimension show that the more potential features, the more relevant shared information can be extracted. Similarly, too few potential features (for example, $d$ =10) limit the ability of the model to extract relevant information, resulting in poor performance. However, too many potential features will lead to over fitting, which will reduce the performance of the model. As shown in Fig. 3, when $d$ exceeds 80, the model performance will decline.

(2) Influence of $\alpha$ on experimental results

$\alpha$ indicates the importance of text features for service recommendation. In this paper, we set the parameter to (0,1) and the step size to 0.1 to study the importance of text features that learn from the text information. Fig. 4 depicts precision@1 at different values. As Fig. 4 indicates, text features play an importance role in driving optimal performance. Specifically, the best setting for a dataset is $\alpha$ = 0.2.

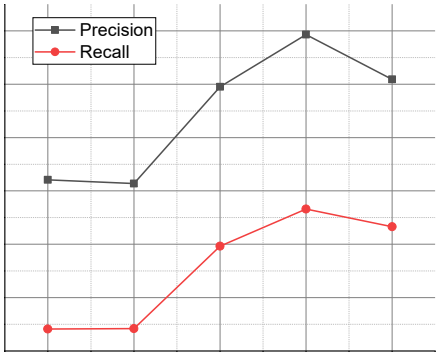



**Fig. 3.** Influence of dimension $d$ on experimental results

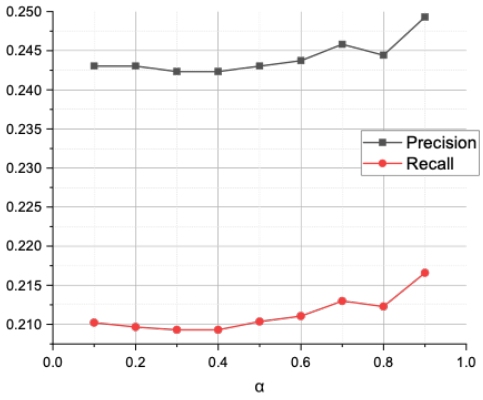



**Fig. 4.** Influence of $\alpha$ on experimental results

### 4.6 Ablation Experiments (RQ3)

The SRM model uses text description of applications and services, tag information, and historical invocation records to recommend services. To consider the impact of the above information on recommendation performance, we compared our model with its variants. Fig. 5 shows how these factors affect the performance of the SRM model.

(1) SRM-description: text information of applications and services is not considered in the SRM model.

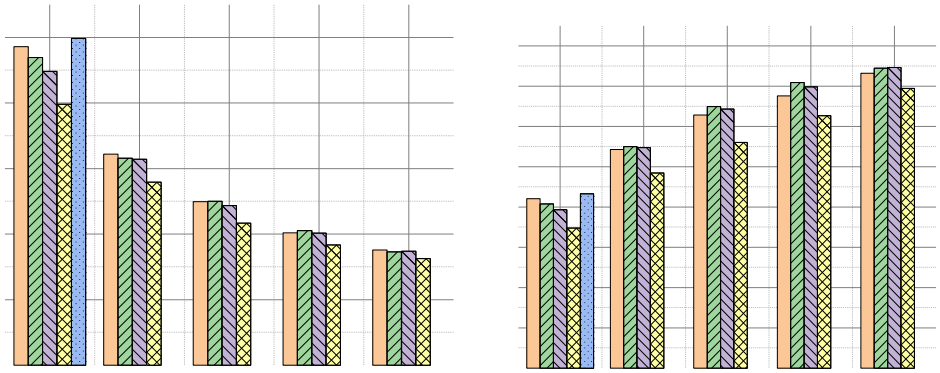
(2) SRM-records: historical invocation records information is not considered in the SRM model.

(3) SRM-attention: attention mechanism is not considered in SRM model.

(4) SRM-tag: tag information is not considered in SRM model.

Fig. 5 reveals the following contents: (1) SRM has the best recommended performance, which can prove the effectiveness of comprehensive consideration of these factors. (2) Compared with SRM, SRM-description has better recommendation performance, which indicates that historical invocation records information is more important than text information. (3) Compared with SRM-attention, SRM have better recommendation perfor-

mance, which indicates that the attention mechanism in the model plays a positive role in recommendation performance. (4) The recommendation performance of SRM-tag is the worst, which indicates that the role of tags should be fully considered in the recommendation process.



(a) Precision (b) Recall

**Fig. 5.** Experimental results of SRM and its variants

## 5 Conclusion

In this paper, we propose a service recommendation method called SRM based on multi model fusion. we first use the text information and tag information of applications and services to build a semantic interaction model, and then obtain the application's preference for services based on the semantic model. Furthermore, the collaboration information of the application is extracted through the historical interaction model, and the structural information is modeled through the graph neural network model. Finally, the semantic interaction model and historical interaction model are effectively integrated. The experimental results show that the SRM model is superior to several comparison service recommendation methods. However, the current AI technology cannot fully implement service recommendation. Next, we plan to incorporate the quality of service (QoS) attributes into the service recommendation process, and plan to use the proposed methods to solve the long tail problem to a certain extent.

## 6 Acknowledgement

The authors wish to thank all the participants in the evaluation process for their help. This work was supported by the Scientific Research Foundation of Zhejiang Provincial Education Department (No. Y202351988), the Cooperative Education Project of Production and Education Foundation of Jiaxing Nanhu University (No. 202002254017) and the Jiaxing Science and Technology Bureau of China under Grant (No. 2023AD11036).

## References

[1] Z. Zheng, H. Ma, M.R. Lyu, I. King, QoS-aware web service recommendation by collaborative filtering, IEEE Transactions on services computing 4(2)(2011) 140-152.

[2] G. Kang, M. Tang, J. Liu, X. Liu, B. Cao, Diversifying web service recommendation results via exploring service usage history, IEEE Transactions on Services Computing 9(4)(2016) 566-579.

[3] B.S. Balaji, N.K. Karthikeyan, R.S.R. Kumar, Fuzzy service conceptual ontology system for cloud service recommendation, Computers & Electrical Engineering 69(2018) 435-446.

[4] L. Wang, X. Zhang, R. Wang, C. Yan, H. Kou, L. Qi, Diversified service recommendation with high accuracy and efficiency, Knowledge-Based Systems 204(2020) 106196.
[5] J. Liu, M. Tang, Z. Zheng, X. Liu, S. Lyu, Location-aware and personalized collaborative filtering for web service recommendation, IEEE Transactions on Services Computing 9(5)(2016) 686-699.
[6] T. Liang, L. Chen, J. Wu, H. Dong, A. Bouguettaya, Meta-path based service recommendation in heterogeneous information networks, in: Proc. International Conference on Service-oriented Computing 2016, 2016.
[7] F. Xie, J. Wang, R. Xiong, N. Zhang, Y. Ma, K. He, An integrated service recommendation approach for service-based system development, Expert Systems with Applications 123(2019) 178-194.
[8] W. Gao, L. Chen, J. Wu, A. Bouguettaya. Joint modeling users, services, mashups, and topics for service recommendation, in: Proc. 2016 IEEE International Conference on Web Services (ICWS) 2016, 2016.
[9] L. Duan, T. Gao, W. Ni, W. Wang, A hybrid intelligent service recommendation by latent semantics and explicit ratings, International Journal of Intelligent Systems 36(12) (2021) 7867-7894.
[10] L. Qi, X. Zhang, W. Dou, C. Hu, C. Yang, J. Chen, A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment, Future Generation Computer Systems 88(2018) 636-643.
[11] T. Liang, X. Sheng, L. Zhou, Y. Li, H. Gao, Y. Yin, L. Chen, Mobile app recommendation via heterogeneous graph neural network in edge computing, Applied Soft Computing 103(2021) 107162.
[12] H. Mezni, D. Benslimane, L. Bellatreche, Context-aware service recommendation based on knowledge graph embedding, IEEE Transactions on Knowledge and Data Engineering 34(11)(2022) 5225-5238.
[13] Q. Gu, J. Cao, Y. Liu, CSBR: A Compositional Semantics-based Service Bundle Recommendation Approach for Mashup Development, IEEE Transactions on Services Computing 15(6)(2022) 3170-3183.
[14] K. Zhou, W. Zhao, S. Bian, Y. Zhou, J. Wen, J. Yu, Improving conversational recommender systems via knowledge graph based semantic fusion, in: Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining 2020, 2020.
[15] Z. Qiu, X. Wu, J. Gao, W. Fan, U-BERT: Pre-training user representations for improved recommendation, Proceedings of the AAAI Conference on Artificial Intelligence 35(5)(2021) 4320-4327.
[16] S. Shi, W. Ma, Z. Wang, M. Zhang, K. Fang, J. Xu, Y. Liu, S. Ma, WG4Rec: Modeling Textual Content with Word Graph for News Recommendation, in: Proc. of the 30th ACM International Conference on Information & Knowledge Management 2021, 2021.
[17] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, T. Zhang, Collaborative web service quality prediction via exploiting matrix factorization and network map, IEEE Transactions on Network and Service Management 13(1)(2016) 126-137.
[18] Z. Zheng, X. Li, M. Tang, F. Xie, M.R. Lyu, Web service QoS prediction via collaborative filtering: A survey, IEEE Transactions on Services Computing 15(4)(2022) 2455-2472.
[19] W. Liang, S. Xie, J. Cai, J. Xu, Y. Hu, Y. Xu, M. Qiu, Deep neural network security collaborative filtering scheme for service recommendation in intelligent cyber–physical systems, IEEE Internet of Things Journal 9(22)(2022) 22123-22132.
[20] L. Yao, X. Wang, Q. Sheng, B. Benatallah, C. Huang, Mashup recommendation by regularizing matrix factorization with API co-invocations, IEEE Transactions on Services Computing 14(2)(2021) 502-515.
[21] M. Nguyen, J. Yu, T. Nguyen, Y. Han, Attentional matrix factorization with context and co-invocation for service recommendation, Expert Systems with Applications 186(2021) 115698.
[22] Y. Ma, X. Geng, J. Wang, A deep neural network with multiplex interactions for cold-start service recommendation, IEEE Transactions on Engineering Management 68(1)(2021) 105-119.
[23] W. Xiong, Z. Wu, B. Li, B. Hang, Automating mashup service recommendation via semantic and structural features, Mathematical Problems in Engineering 2020(2020) 1-10.
[24] X. Wang, X. Liu, J. Liu, X. Chen, H. Wu, A novel knowledge graph embedding based api recommendation method for mashup development, World Wide Web 24(3)(2021) 869-894.
[25] D. Dang, C. Chen, H. Li, R. Yan, Z. Guo, X. Wang, Deep knowledge-aware framework for web service recommendation, The Journal of Supercomputing 77(12)(2021) 14280-14304.
[26] Z. Jiang, H. Liu, B. Fu, Z. Wu, T. Zhang, Recommendation in heterogeneous information networks based on generalized random walk model and bayesian personalized ranking, in: Proc. of the Eleventh ACM International Conference on Web Search and Data Mining 2018, 2018.
[27] X. Wang, X. He, M. Wang, F. Feng, T. Chua, Neural graph collaborative filtering, in: Proc. of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval 2019, 2019.
[28] J. Devlin, M. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding. <https://arxiv.org/abs/1810.04805>, 2018 (accessed 11.10.2018).
[29] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in: Proc. of the 43rd International ACM SIGIR conference on research and development in Information Retrieval 2020, 2020.
[30] X. He, Z. He, J. Song, Z. Liu, Y. Jiang, T.S. Chua, Nais: Neural attentive item similarity model for recommendation, IEEE Transactions on Knowledge and Data Engineering 30(12)(2018) 2354-2366.
[31] X. He, Z. He, X. Du, T. Chua, Adversarial personalized ranking for recommendation, in: Proc. The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval 2018, 2018.

[32] Z. Zhu, J. Wang, J. Caverlee, Improving top-k recommendation via joint collaborative autoencoders, in: Proc. The World Wide Web Conference 2019, 2019.