The Journal of Systems and Software 207 (2024) 111873

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Iterative framework based on multi-task learning for service recommendation ☆

Ting Yu [a,b], Dongjin Yu [a,*], Dongjing Wang [a], Quanxin Yang [a], Xueyou Hu [a]

[a] School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, 310018, China
[b] Jiaxing Nanhu University, Jiaxing, 314001, China

ARTICLE INFO

Dataset link: https://github.com/HduDBSI/ISRMM

Keywords:
Service recommendation
Application creation
Multi-task learning
Multi-model fusion

ABSTRACT

In recent years, service-oriented computing technology has developed rapidly, which, however, has increased the burden of selection for software developers when developing service-based systems. To solve this problem, people have proposed various methods to recommend services which are composed into an application. Nevertheless, most of the existing service recommendation methods cannot serve iterative scenarios, i.e., multiple request–response recommending rounds, which frequently occur in real application development. Moreover, they usually fail to utilize full features such as user requirements and service categories, leading to poor performance of service recommendation. To solve the above problems, we propose an iterative framework for service recommendation through multi-model fusion and multi-task learning, called ISRMM. More specifically, we design two models to capture the preferences of applications towards services, through the perspectives of user requirements and history interaction respectively. The output features of the above models are further fused to predict the next service that will be recommended. In addition, we add a tag judgment task to make our framework capable of multi-task learning, through which, the training signal information implied can be used as an inductive bias to improve service recommendation capabilities. Extensive experiments on real datasets show that ISRMM outperforms several state-of-the-art service recommendation methods in iterative service recommendation scenarios.

1. Introduction

With the rapid development of Internet, service orient computing has been frequently employed as the mainstream of paradigm when developing web-based applications. Compared with the need to write code from scratch, by leveraging SOTA technologies, developers do not prescribe to reinvent the wheel for each new application. Instead, they can focus on the goal of fulfilling the unique requirements of the application, which greatly frees up the creativity of the developer. Unfortunately, The rapid growth in the number of services available in web application programming interface catalogs poses a great challenge in choosing the right services for applications. Therefore, how to recommend appropriate component services to develop a new application remains as one for the key technologies.

In the past decade, many service recommendation (Zheng et al., 2010; Kang et al., 2015; Wang et al., 2020) methods have been proposed. They can be broadly divided into three categories: collaborative filtering (CF)-based service recommendation (Liang et al., 2016; Xie et al., 2019), semantic-based service recommendation (Gao et al., 2016; Duan et al., 2021), and graph-based service recommendation (Qi et al., 2018; Liang et al., 2021; Mezni et al., 2021). Given keyword-based application requirements, the semantic-based approaches make recommendations based on textual similarities between service descriptions and developer requirements, while the CF-based approaches draw on the history of similar applications or services to generate a recommendation list. As for the graph-based service recommendation approaches, they project applications and services into a shared latent space. In the real world, to compose applications based on appropriate component services is usually a complex decision process which cannot be fulfilled in a straightforward manner, but requires multiple request–response recommending rounds. However, the existing service recommendation approaches are primarily suitable for application development in non-iterative scenarios (Shi et al., 2019; Nguyen et al., 2020). In other words, they recommend all candidate services for a new application at a time, which is a single-round request–response recommendation. As shown in Fig. 1(a), given the requirement of "create an application that can send voice to family and friends", in the non-iterative

☆ Editor: Uwe Zdun.

* Corresponding author.
E-mail addresses: yuting@hdu.edu.cn (T. Yu), yudj@hdu.edu.cn (D. Yu).

https://doi.org/10.1016/j.jss.2023.111873
Received 8 November 2022; Received in revised form 1 October 2023; Accepted 4 October 2023
Available online 14 October 2023

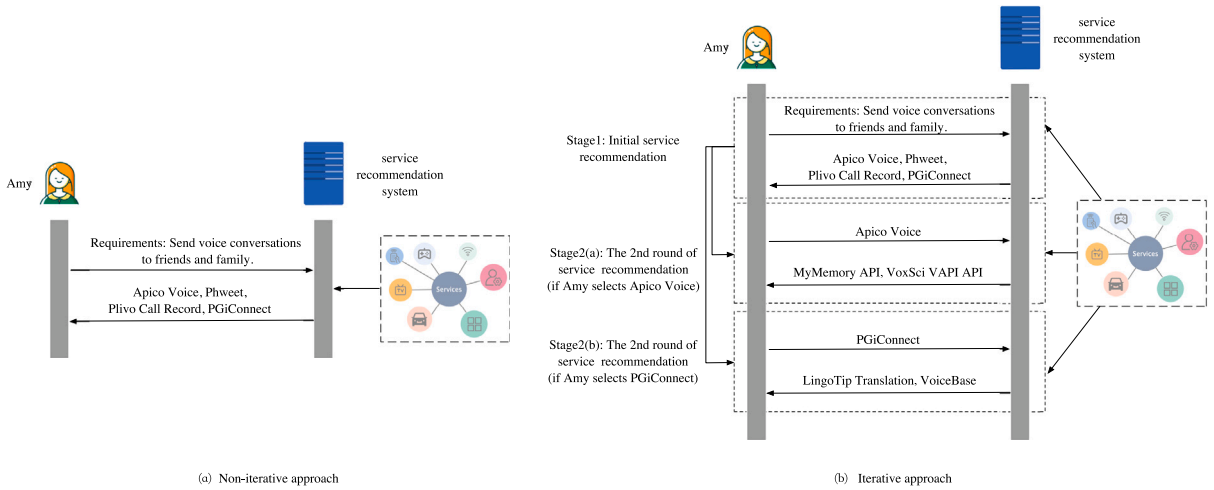


Fig. 1. Two different approaches for application creation. (a) Non-iterative application creation. (b) Iterative application creation.

scenario, the service recommendation system will probably generate a recommendation list that contains, for example, Apico Voice, Phweet, Plivo Call Record and PGiConnect. Here, Apico Voice provides voice sending and recording features, while Phweet makes conversations and conference calls possible between social networks. Besides, Plivo Call Record allows developers to record a live call at anytime during a call, and PGiConnect can send voice, fax, email, and SMS messages via networking. However, such recommendation results are not always satisfactory under many circumstances. For example, due to the limited semantics given in such a short piece of requirements, the recommendation system is prone to providing similar services with the same functionality, such as Apico Voice and PGiConnect, or not-well-matched services, such as Plivo Call Record. Besides, it may lack a translation service for users when the two sides of the conversation speak in different languages. In fact, it is very hard to exactly express what a developer actually wants only using a short list words. Instead, if being prompted in a step by step way, he (she) is more likely to show his (her) real intention clearly.

To address this problem, in this paper, we propose an Iterative Service Recommendation method based on Multi-model fusion and Multi-task learning (ISRMM), which includes two parts: the semantic interaction model and the history interaction model. Moreover, we integrate the features learned from the semantic interaction model and the history interaction model to make more accurate recommendations. At the same time, considering the importance of categories to services and applications, we add a tag judgment task to make our framework capable of multi-task learning. The main contributions of this paper can be summarized as follows:

1. Different from the traditional service recommendation scenario, we discuss service recommendation in interactive scenarios. We propose a recommendation framework for service-oriented interaction, called ISRMM, which can recommend suitable candidate services to meet the developer's requirements progressively.
2. We implement two separate models, which capture interactions of applications and services through the descriptions and the history invocations of them, respectively.
3. We propose an application-oriented multi-model fusion and multi-task learning model. The model combines the advantages of CF-based and semantic-based methods and realizes category prediction and service recommendation according to the descriptions and historical interaction records of applications and services.

The paper is structured as follows. Section 2 introduces the related work, Section 3 introduces research motivation. The proposed ISRMM framework is introduced in Section 4. Afterwards, Section 5 introduces the experimental results and analysis. Finally, we summarize this paper and put forward our future work.

2. Related works

The purpose of service recommendation is to recommend suitable candidate service subsets for developers according to application requirements, to reduce the burden of manual selection. Here, we divide the related work into two parts according to its principle and focus, namely service recommendation system and interactive recommendation system.

2.1. Service recommendation system

In recent years, different kinds of methods for service recommendation have emerged rapidly. Traditional information retrieval is a major method for selecting appropriate services based on the matching of requirements, but it is difficult to apply to large-scale services. As natural language processing technology has developed quickly, more advanced technologies have also been used to extract the semantic relationships between applications and services. Shi et al. (2018) employed a Long Short-Term Memory-(LSTM-) based method with a functional attention mechanism and a contextual attention mechanism to recommend services. Zhong et al. (2016) explored the description of both APIs and mashups with their historical interactions to reconstruct the profile for services which are used as auxiliary information for rating prediction. Gu et al. (2021) presented a compositional semantics-based model to recommend a set of complementary services for mashup creation. The main problem of the above methods is that a large amount of human involvement is required to extract valid features, especially for textual data. Compared with these works, we use a deep neural network as the modeling basis to build a complete end-to-end recommendation framework.

More complex models are used to extract features for service recommendation. Jiang et al. (2018) put forward a Bayesian Personalized Ranking based machine learning method (named HeteLearn), to learn the weights of links in a HIN (heterogeneous information network). To model user preferences for personalized recommendation, they proposed a generalized random walk with restart model on HINs. Chen et al. (2018) presented a neural CF recommender model that learns user preference on services in terms of their matching degree on explicitly-declared attribute preference and implicit preference mined from past invocations. Xiong et al. (2020) applied Natural Language Processing (NLP) and graph embedding techniques to recommend services for mashup developers. They extracted structural

semantics from a two-mode graph to describe mashups, services, and their relations. Wang et al. (2021) proposed an unsupervised service recommendation method based on deep random walks on a knowledge graph, which used the relationship between service and mashup to build a knowledge graph, and used the Skip-Gram model to implement an implicit embedding representation of each node and calculate the correlation between mashup nodes and service nodes to obtain a list of service recommendations. Wu et al. (2022) proposed a method of popularity-aware and diverse API composition recommendation. Taking not only compatibility but also popularity as the major optimization goal. These efforts have proved that graph-based service recommendation technology can help develop more efficient service recommendation models. However, there is little work on graph-based service recommendation in iterative scenarios.

2.2. Interactive recommendation system

In the field of service computing, the concept of interaction design rarely appears in service recommendation. Some approaches extract service composition patterns from the history of service interaction records or the neighborhood in networks. While others make recommendations based on user behavior history. Zhang et al. (2018) extracted dynamic user preference from time slice data by the time-series Latent Dirichlet Allocation (LDA) model and generated a service list based on the latest user preference and Quality of Service (QoS). Kwapong et al. (2019) composed a user's invocation preference at a timestamp as a combination of non-functional attributes such as QoS and functional features extracted from the Web Services Description Language (WSDL) file of the invoked service. However, the above method only focuses on the evolution mode of user selection behavior or preference and ignores the consistency between the selected service and application requirements. Zhou et al. (2018) utilized an attention-based pooling layer to integrate items, following which an MLP is adopted. However, it only considers the interaction between services and applications and fails to fully consider the information interaction between applications and services. Wu et al. (2021) proposed a neural framework based on multi-model fusion and multi-task learning. It exploits a semantic component to generate representations of requirements and introduces a feature interaction component to model the feature interaction between mashups and services. However, they use matrix factorization algorithm to extract feature vectors, which is not easy to capture high-order nonlinear relations. Moreover, they do not fully explore the textual description relations of services and applications. In this paper, we have explored these aspects and achieved positive results. This is the essential difference between our work and theirs.

3. Motivation

Fig. 1(b) gives the example of iterative service recommendation approach. As mentioned earlier, in Stage1, the service recommendation system might generate a recommendation list that contains Apico Voice, Phweet, Plivo Call Record and PGiConnect. In Stage 2(a), suppose Amy selects one service based on the recommendation list, namely, Apico Voice, which can perform the voice sending function. It should be noted that both PGiConnect and Phweet share the similar function with Apico Voice. As for Plivo Call Record, it allows developers to record a live call at anytime during a call. Because it does not have high adaptability to the given requirement, Amy definitely would not choose Plivo Call Record. In fact, it is not always suitable to obtain services only according to the given requirements. Unlike the single-round request–response recommendation, the iterative recommendation steps into the second stage after receiving the developer's initial feedback, and complements the recommendation list based on the services selected in the previous stage. As a result, those services related closely to the selected services will have a higher priority in the new recommendation list. Through analyzing the initial text requirements and the selected service (Apico Voice), the recommendation system finds that the translation service may be the potential one because the two sides of the conversation may speak in different languages. It therefore recommends MyMemory API in Stage 2(a), which offers the translation service. Moreover, in some cases where it is not convenient to play voice messages, it is necessary for the voice to be translated into text. Therefore, voice to text services are also needed to be recommended, such as VoxSci VAPI API. Here, the reason why MyMemory API and VoxSci VAPI API are recommended is that, through analysis of historical invoking records and other aspects, it has been found that MyMemory API and VoxSci VAPI API have a certain correlation with Apico Voice, such as the similar hosted regions.

Of course, Amy could select PGiConnect instead of Apico Voice, based on the recommendation list generated in Stage 1, as Stage 2(b) shows. PGiConnect also offers the voice sending feature. Similarly, through analyzing the initial text requirements and the selected service (PGiConnect), the service recommendation system finds that the voice translation service and voice to text service are highly needed to be provided. Actually, there are many services with similar functions in the service repository. By analyzing the initial text requirements, the selected service PGiConnect, the historical invoking records and other aspects, LingoTip Translation and VoiceBase are recommended to Amy. Here, LingoTip Translation provides the language translation function, while VoiceBase converts voice into text. As can be imagined, LingoTip Translation and VoiceBase have a certain correlation with PGiConnect, because their hosted countries are all coming from United States and the co-occurrence of these three is high in the past.

In many cases, when an application needs to be created, it is usually a challenging task to accurately describe what the application actually is. Instead, the iterative recommendation can effectively refine requirements and precisely capture the implicit intentions of developers through a step-by-step manner. All in all, a typical iterative service recommendation process consists of multiple stages. In the first stage, the service recommendation system analyzes the given requirements, then returns a list of candidate services, from which the developer selects one or more. The recommendation process will enter the next stage if the result does not fully satisfy the requirements. At this moment, the service recommendation system continues to generate a new service list according to the requirements and the previously selected services. This step will repeat until the developers no longer request further recommendation or no further recommendation can be provided.

4. Proposed approach

4.1. Preliminary

In this section, we introduce the service recommendation problem studied in this paper, and also present the definitions of key concepts. The main symbols and their explanation are given in Table 1.

Definition 1 (*Service*)**.** A service s represents the application programming interface provided by the web service provider, which makes it easier for developers to create complex functions. In the context of this work, we only use the name, description and tags of the service, but ignore its interface, although the latter is the core of a service. In other words, we simplify a service as a 3-tuple, i.e., $s = \langle s_n; s_{des}; (s_{t1}, s_{t2}, \ldots, s_{ti})\rangle$, where s_n, s_{des}, $(s_{t1}, s_{t2}, \ldots, s_{ti})$ represent the name, text description and tag sets of s, respectively. Services are generally available through REpresentational State Transfer (REST).

For example, the service Twitter can be denoted by Twitter=< Twitter REST API v2, "It includes functions to specify data fields, plus added objects including mentioned accounts in Tweets, tagged places, quoted and replied-to Tweets, attached media, poll results and pinned Tweets. It also offers public and private Tweet metrics, insights on topics, and improved conversation tracking"., (Social, Blogging, Tweets)>.

Table 1
Symbols used in this work.

Symbol	Description
S	service set
s	a service
s_n	the name of service s
$(s_{t1}, s_{t2}, \dots, s_{tl})$	the tags of service s
SC	the tag set of S
M	application set
m	an application
m_n	the name of application m
$(m_{t1}, m_{t2}, \dots, m_{tj})$	the tags of application m
MC	the tag set of M
Y	service ecosystem
$SS = \{s_0, s_1, \dots s_t\}$	the selected services for an application

Definition 2 (*Application*)**.** An application m is a combination of two or more services, making itself available as a composite service, formalized as a 4-tuple, i.e., $m =< m_n; m_{des}; (s_1, s_2 \dots s_n); (m_{t1}, m_{t2}, \dots, m_{tj}) >$, where m_n, m_{des}, $(m_{t1}, m_{t2}, \dots, m_{tj})$ represent the name, text description and tag sets of m and $(s_1, s_2, \dots, s_n)$ represents the service record invoked by m. Applications often mediate among a set of services provided by a heterogeneous set of providers.

For example, the application Stweet can be denoted by Stweet=< Stweet, "Stweet is a mix of street and twitt, offering a real new way to discover geolocalised twitts from Twitter on a Google Street View panorama"., (Google Maps, Twitter), (Blogging, Map-ping)>.

Definition 3 (*Service Ecosystem*)**.** The service ecosystem Y denotes a set of services and applications, or $Y = (M; S)$, where $M = \{m_1, m_2, \dots, m_p\}$ and $S = \{s_1, s_2, \dots, s_q\}$ denote the collection of services and applications in service ecosystem Y.

Definition 4 (*Application-Service Invocation Records*)**.** The invocation records R denote if an application once invokes a service. If application m_i invoked service s_j, $r_{m_i,s_j} = 1$, otherwise $r_{m_i,s_j} = 0$.

(Problem Definition) Given an application and the corresponding selected services, i.e., $m'_n, \{s_0, s_1, \dots, s_t\}$, its tags $(m'_{t1}, m'_{t2}, \dots, m'_{tj})$ and description m'_{des}, the problem to be addressed is to find the Top-K follow-up services that best match the preferences of the requirements of m' and the selected services $\{s_0, s_1, \dots, s_x\}$.

4.2. Framework

Fig. 2 shows the framework of ISRMM in detail, which consists of three parts: semantic interaction model, history interaction model and multi-model fusion. According to the application requirements entered by the developer, the semantic interaction model first extracts the text feature representation of the application, the selected services, and the candidate services, and then employs DNN to understand the interaction between them. The output layer predicts the possibility of choosing the candidate service as the component service in the next round. In particular, the semantic interaction model introduces application and service tag judgment as auxiliary tasks. The history interaction model is very similar to the semantic interaction model. The difference is that it utilizes historical invocation records between applications and services to extract interaction feature representation of the application, the selected services and the candidate service. Finally, the features obtained from the above two models are further fused to obtain a list of service recommendations.

4.3. Semantic interaction model

The description of applications and services contains more information than other metadata. Therefore, text modeling and representation technology are very important in semantic-based recommendation methods. When calculating the probability that a service will be selected, developers will first consider whether the description of the candidate service can meet the requirements of the target application and the selected services. Therefore, we design a semantic interaction model to capture the relationship between them. Fig. 3 shows the framework of the semantic interaction model in detail.

When the developer begins to develop an application m, its requirements are first entered. According to the historical invocation records, we obtain the invoked service repository, and further retrieve the text description information of the services. In the semantic interaction model, we employ Electra (Clark et al., 2020) to extract the text features of applications and services. Before feature extraction by deep learning technology, we need to preprocess the text. The text usually contains noisy information and cannot be used directly for model learning. To reserve the important information in the text, the following processes are required.

Text Filtering: Invalid words such as tags, punctuation marks and non-characters, are usually unmeaningful. We use regular expressions to filter them out. In addition, a stop word list is used to delete the stop words in the text.

Abbreviation Replacement: For example, 'don't' is an equivalent abbreviation for 'do not' in human eyes, while the machine however does not think so. Consequently, we replace such abbreviations with their complete spellings.

Word Form Restoration: Word Form Restoration is the process of converting a word to its base form through the WorldNet's built-in morph function. For example, the word "trees" is reduced to "tree", and the word "used" is converted to "use".

Through the Electra model, we can convert the requirements of application m, the description of the selected services and the candidate service into a shared latent space.

There usually exists a co-occurrence relationship between services. For example, after selecting Flickr, it is more likely to choose YouTube as the candidate service. In other words, the services usually occur in the same application to establish a "deep" and long-term relationship. Considering the efficiency of GRU (Hidasi et al., 2015) in modeling sequence dependencies, we adopt it to model the relationship between services. The GRU-Based layer in the framework of ISRMM is illustrated in Fig. 4.

Given the selected services record $SS = \{s_1, \dots, s_t\}$ of application m, and their the text feature vectors $\{v_{s_1}, v_{s_2}, \dots, v_{s_t}\}$, the GRU-based layer computes the currently hidden state vector hs^m_{t-L} conditioned on previous hidden state vector hs^m_{t-L-1} as below:

$$\begin{aligned} z^{(t-L)} &= \sigma(W_z * [hs^m_{t-L-1}, v_{s_{t-L}}]) \\ r^{(t-L)} &= \sigma(W_r * [hs^m_{t-L-1}, v_{s_{t-L}}]) \\ \tilde{hs}^m_{t-L} &= tanh(W_s * [hs^m_{t-L-1} \odot r^{(t-L)}, v_{s_{t-L}}]) \\ hs^m_{t-L} &= z^{(t-L)} \odot \tilde{hs}^m_{t-L} + (1 - z^{(t-L)}) \odot hs^m_{t-L-1} \end{aligned} \tag{1}$$

where W_z, W_r, W_s are learnable parameters, $\sigma(\cdot)$ is a logistic sigmoid function to do non-linear projection, and $\odot$ is the Hadamard product between two vectors. Besides, $z^{(t-L)}$ and $r^{(t-L)}$ denote update and reset gate respectively, which control how information flows through the sequence. Note that the dimensionalities of hs^m_{t-L-1} and $v_{s_{t-L}}$ are the same for the convenience of prediction. In this way, the predictor encodes the selected services of application m into a hidden vector, expressed as hs^m_t, which models the selected services preference representation of application m.

In general, the similarity between two vectors can be measured by element multiplication or element subtraction. Given the text feature vector v_m of application m and the text feature vector hs^m_t of all the selected services calculated from GRU-based layer, we concatenate v_m and hs^m_t, as Eq. (2) indicates.

$$v_{m,ss} = v_m \oplus hs^m_t \tag{2}$$

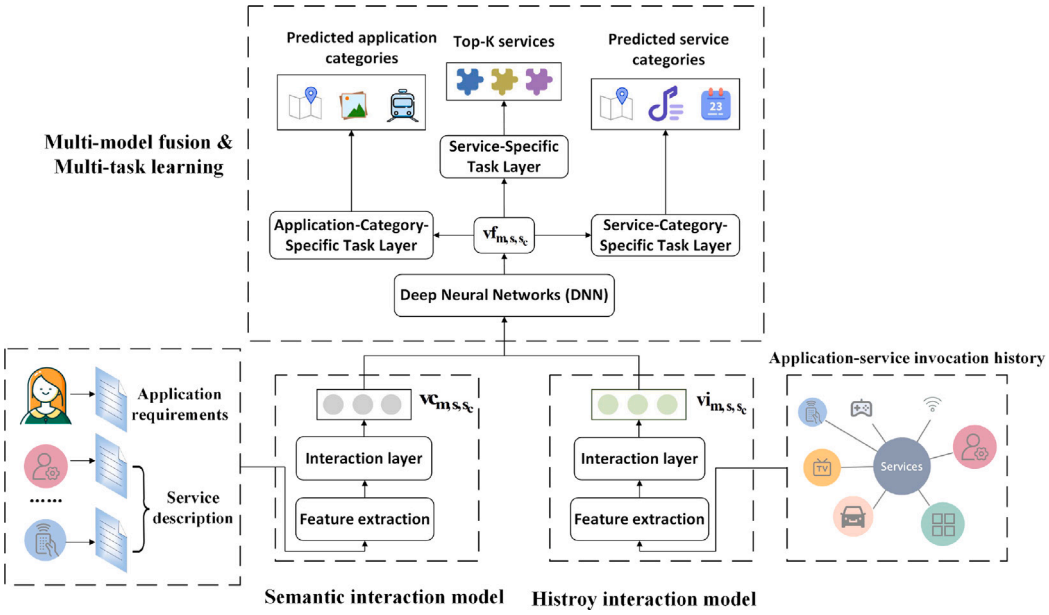


Fig. 2. The framework of ISRMM.

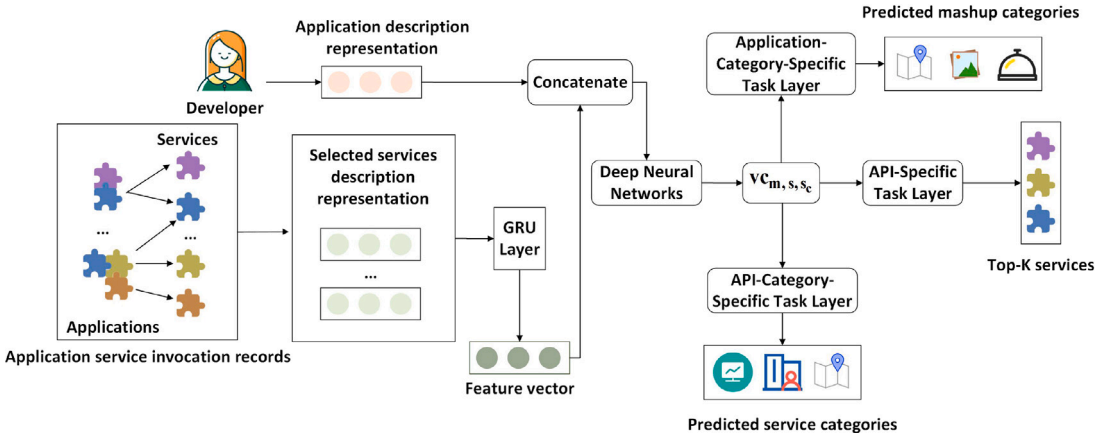


Fig. 3. The framework of semantic interaction model.

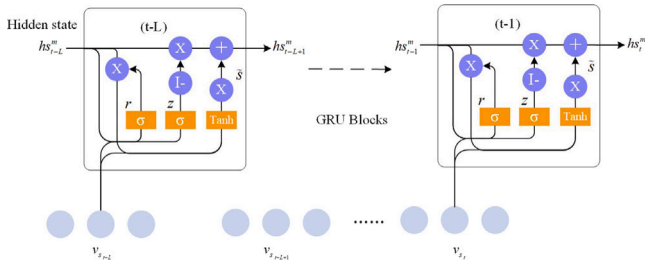


Fig. 4. GRU-based layer of ISRMM.

Suppose the text feature vector of the candidate service is expressed as v_{s_c}, We then utilize an DNN layer to capture interactions among m, the selected services, and the candidate service. Moreover, we select the parametric rectified linear unit (PReLU) as the activation function since it can help the model reduce the overfitting risk to some extent, with nearly zero extra computational cost (Qu et al., 2020). The learning process can be written as:

$$vc_{m,s,s_c} = \delta(W_z(\dots\delta(W_1(v_{m,ss} \oplus v_{s_c}) + b_1)\dots + b_z)) \tag{3}$$

where W_z and b_z represent weight and bias vector in the $z-th$ layer of DNN, δ is an activation function. Here, the advantage of DNN is that it can learn interactive features at different levels of abstraction. With the increase of the number of layers, the receptive field of each neuron becomes larger relative to the previous layer. In this way, it can provide global semantic (global interaction) and abstract details, which is difficult to do in a shallow layer and linear operation.

Multi-task learning is an inductive transfer method that makes full use of the specific domain information implicit in the training signals of multiple related tasks. We finally feed the learned interaction vector vc_{m,s,s_c} into a sigmoid function, whose outputs are $\hat{y}_c$, $\hat{y}_{m_c}$ and $\hat{y}_{s_c}$. Here, $\hat{y}_c$ represents the probability of service s_c as the next recommended service, $\hat{y}_{m_c}$ represents the probability that application m belongs to each application tag m_{ti}, and $\hat{y}_{s_c}$ represents the probability that service s belongs to each service tag s_{tj}. The process can be written as:

$$\hat{y}_c = sigmoid(W_c v_{m,s,s_c} + b_c) \tag{4}$$

$$\hat{y}_{m_c} = sigmoid(W_{m_c} v_{m,s,s_c} + b_{m_c}) \tag{5}$$

$$\hat{y}_{s_c} = sigmoid(W_{s_c} v_{m,s,s_c} + b_{s_c}) \tag{6}$$

where W_c, W_{m_c} and W_{s_c} are the weight matrix and b_c, b_{m_c} and b_{s_c} are the bias vectors in a specific task layer. A dropout layer is added to the DNN network layers to correct the mistakes from prior layers, reducing the overfitting risk in turn and making the proposed model more robust. The principle of dropout is to randomly drop certain connections, which can be understood as only considering incomplete inputs each time, preventing parameters from overly relying on training data and increasing the generalization ability of parameters on the dataset (Jiang et al., 2020). Here, we set dropout=0.5.

4.4. History interaction model

In addition to content information, the history invocation records between the existing applications and services also help to achieve good recommendation results. Through the history invocation records between applications and services, we build an application-service graph. The structural feature vectors of existing applications and services can be obtained through node-2vec (Grover and Leskovec, 2016), a typical graph embedding method. Compared with traditional matrix factorization-based methods (such as probability matrix factorization and singular value decomposition), node2vec can capture more nonlinear relationships between applications and services.

Furthermore, we input the structural feature vectors of the selected services into the GRU layer of the history interaction model to obtain a hidden feature vector, hh_t^m. The structural feature vector of the application m, the hidden feature vector hh_t^m and the structural feature vector of the candidate service are further fed to the DNN layer of the history interaction model to get final structural feature vector vi_{m,s,s_c}. By inputting vi_{m,s,s_c} into the multi-task learning layer, the score of the candidate service can be eventually predicted.

4.5. Multi-model fusion

The above two models learn two forms of interaction and obtain the preference of application m for candidate service s_c according to content information and history invocation records. This section will introduce how to integrate the two models.

As previously mentioned, the ISRMM model consists of the feature extraction, GRU layer, DNN layer, and multi-task layer. The model takes content information and historical invocation records as input and learns different types of interaction. As the final step, we will concatenate the two learned vectors and further input them into the DNN layer. This process can be expressed as:

$$vf_{m,s,s_c} = (W_i(vc_{m,s,s_c} \oplus vi_{m,s,s_c}) + b_i) \tag{7}$$

Moreover, the vector vf_{m,s,s_c} is fed into the multi-task learning layer to predict the score of the candidate service. It is worth mentioning that this hybrid model is easy to expand. When a new kind of information is available, we can add another underlying model to learn the new type of interaction, and then integrate it into the original model to further improve the performance.

4.6. Model training

Multi-task learning is an inductive transfer method, which makes full use of the domain-specific information hidden in the training signals of multiple related tasks. In the process of backpropagation, multi-task learning allows the features in the shared hidden layer dedicated to one task to be used by other tasks, to achieve the effect of implicit data enhancement and collaborative regularization, which is seriously conducive to the model training of the deep neural network. To make our model has the ability of multi-task learning, three specific task layers are introduced.

For the task of our proposed model, we transform it into a multi-label classification problem. Our goal is to train whether the service will be invoked in the application (main task) and each service or application is decomposed into multiple independent binary classification problems (auxiliary task). In order to train our model through the main task, we adopt Binary Cross Entropy Loss (BCELoss), which is defined as follows:

$$L_{ms}(\hat{y}_c, y_c) = -\frac{1}{|S|}\sum_{s\in S}(y_c[s]log\hat{y}_c[s])+ (1-y_c[s])log(1-\hat{y}_c[s]) \tag{8}$$

where $y_c[s]$ corresponds to the true value which indicates whether the service s matches the requirements of application m ($y_c[s] = 1$ if m invoked s; otherwise $y_c[s] = 0$), and $\hat{y}_c[s]$ is the predicted value corresponding to $y_c[s]$. After model training, the recommendation list of the candidate services is generated.

At the same time, we add a tag judgment task to make our framework have the ability of multi-task learning. Consider the importance of categories to services and applications. We also adopt BCEloss to optimize model parameters. The process is as follows:

$$L_{mc}(\hat{y}_{m_c}, y_{m_c}) = -\frac{1}{|MC|}\sum_{mc\in MC}(y_{m_c}[mc]log\hat{y}_{m_c}[mc])+ (1-y_{m_c}[mc])log(1-\hat{y}_{m_c}[mc]) \tag{9}$$

$$L_{sc}(\hat{y}_{s_c}, y_{s_c}) = -\frac{1}{|SC|}\sum_{sc\in SC}(y_{s_c}[sc]log\hat{y}_{s_c}[sc])+ (1-y_{s_c}[sc])log(1-\hat{y}_{s_c}[sc]) \tag{10}$$

By aggregating the loss functions of the three tasks and imposing regularization constraints, we obtain the objective function of our model as follows:

$$min\frac{1}{|M|}\sum_{m\in M}(L_{ms}+L_{mc}+L_{sc})+\lambda\|\theta'\|_2^2 \tag{11}$$

where θ' is the model parameters, and $\lambda\|\theta'\|_2^2$ is a L2 regularization term used to prevent the model from over-fitting.

The training algorithm of ISRMM is as follows.

5. Experiment

In this section, we will conduct a series of experiments to evaluate our proposed methods and answer the four research questions.

RQ1: How does ISRMM perform against the baseline models?

RQ2: Do the semantic interaction model and the history interaction model contribute to the performance of ISRMM?

RQ3: Does multi-task learning contribute to the performance of ISRMM?

RQ4: Does the efficiency of our model outperform other models?

RQ5: How do the parameters in ISRMM influence the recommendation results?

5.1. Experimental setups

All experiments were developed in Python and carried out on a personal PC with Intel Core i5 CPU with 2.4 GHz and 8 GB RAM, running the macOS High Sierra. We made the data and code publicly available at https://github.com/HduDBSI/ISRMM.

5.1.1. Dataset

We crawled 22016 services and 6438 applications from service ecosystem ProgrammableWeb, the world largest online Web service registry, on March, 2021. The applications and services without functional descriptions, the services that have not been invoked, and the application with fewer than three component services were removed from the original dataset. The final experimental dataset contains 1979 applications and 728 services. The schema of the dataset as shown in Table 2. The sparsity of the application-service invocation matrix is 99.6%. We randomly select 80% application-service pairs as the training set and the remaining 20% as testing set.

Algorithm 1 Algorithm of SRMG

Input: application set M, service set S
application service invocation records R
the selected services SS in application m
Output: preference for each candidate service

for epoch=1,...k **do**
  for each y in Y **do**
    Compute vc_{m,s,s_c} using Eqs. (3)
    Compute $\hat{y_c}$, $\widehat{y_{m_c}}$, $\widehat{y_{s_c}}$ using Eqs. (4)–(6)
    Update model parameter in Eqs. (8)–(10)
  end for
end for
for epoch=1,...n **do**
  for each y in Y **do**
    Compute vi_{m,s,s_c}
    Compute $\hat{y_c}$, $\widehat{y_{m_c}}$, $\widehat{y_{s_c}}$ using Eqs. (4)–(6)
    Update model parameter in Eqs. (8)–(10)
  end for
end for
for epoch=1,...x **do**
  Initialize the semantic interaction model and the history interaction model with learnable parameters and then freeze them
  Compute vf_{m,s,s_c} using Eq. (7)
  Compute $\hat{y_c}$, $\widehat{y_{m_c}}$, $\widehat{y_{s_c}}$ using Eqs. (4)–(6)
  Update model parameter in Eqs. (8)–(10)
end for
for each m in M **do**
  $SS \leftarrow$ selected services of application m
  for each candidate service s in S **do**
    Compute preference for each candidate service using Eq. (4)
  end for
end for

Table 2
The schema of the dataset.

Name	Description
m_n	the name of the application m
m_{des}	the description of the application m
m_c	the tags of the application m
s_n	the name of the service s
s_{des}	the description of the service s
s_c	the tags of the service s

5.1.2. Evaluation metrics

We evaluated different recommendation approaches using the five-fold cross-validation technique. In other words, the dataset was divided into five folds. For each time, one-fold was used for testing, and the other four for training. Then, we averaged the results of five folds and took them as the final one.

We use the *Precision*, *Recall*, $F1$, MAP, and $NGCD$ scores of Top-K recommendation to perform the performance evaluation, defined as follows.

Precision, signed as $Precision@k$, measures how many recommended services correspond to the true services in testing data for a given application, as shown in Eq. (12), where $top_m(k)$ represents the top k services recommended to application m, and $test_m$ represents the services actually invoked by application m in the test set.

$$Precision@k = \frac{|top_m(k) \cap test_m|}{k} \tag{12}$$

Recall, signed as $Recall@k$, measures how many true services in testing data have been recommended for a given application as shown in Eq. (13).

$$Recall@k = \frac{|top_m(k) \cap test_m|}{|test_m|} \tag{13}$$

$F1$, signed as $F1@k$, considers both recall and precision as shown in Eq. (14). Therefore, it is a common performance metrics used to evaluate the recommendation performance.

$$F1@k = 2\frac{precision@k \times recall@k}{precision@k + recall@k} \tag{14}$$

The Mean Average Precision (MAP) at top k services in the ranking list is defined as Eq. (15).

$$MAP@k = \frac{\sum_{i=1}^{k} Precision@i \times rel_i}{\sum_{i=1}^{k} rel_i}/n \tag{15}$$

where n represents the number of applications in the test set, rel_i refers to the graded relevance of the result ranked at position i. In this paper, we use the binary relevance, i.e., $rel_i = 1$ if the result is in the test set, and 0 otherwise.

Normalized Discounted Cumulative Gain ($NDCG$) is a measure of accuracy from information retrieval as shown in Eq. (17), DCG is accumulated from the top of the result list to the bottom, with the gain of each result discounted at lower ranks. $NDCG$ indicates the ideal DCG.

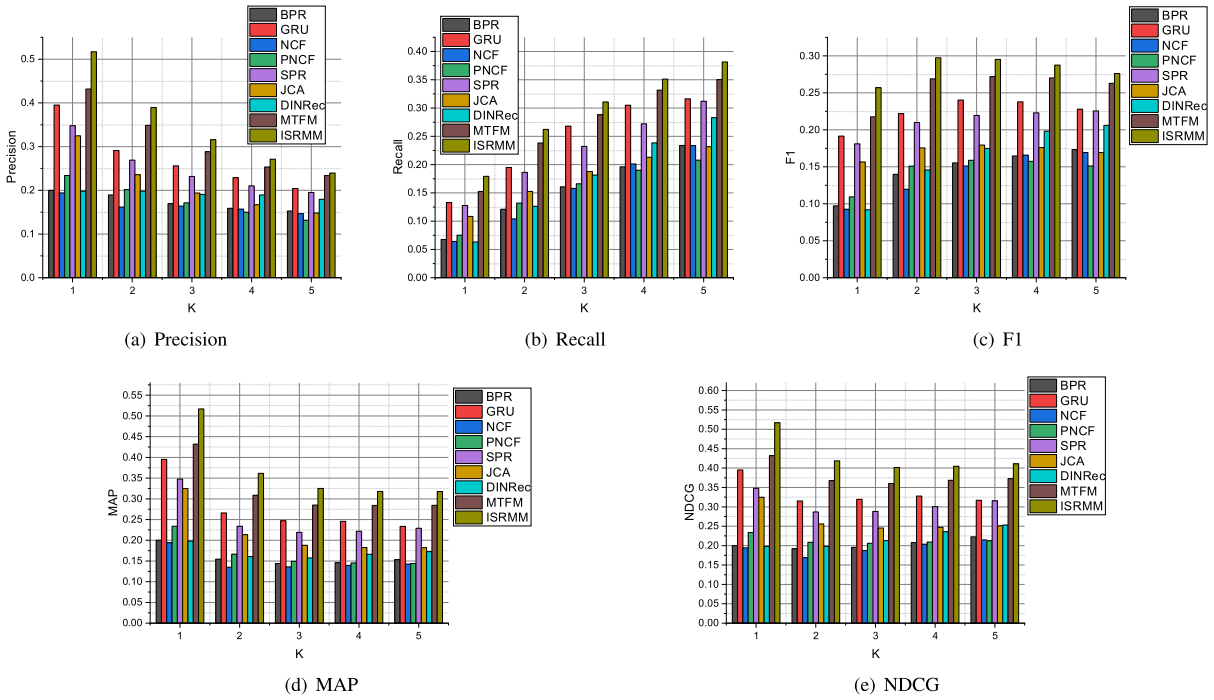
$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(1+i)} \tag{16}$$

$$NDCG@k = \frac{DCG@k}{IDCG@k} \tag{17}$$

5.1.3. Baseline methods

To emphasize the effectiveness of our approach, we selected eight methods for comparison. Currently, there are not many universal benchmarks for the recommendations of services, we have selected four general recommendation methods and four service recommendation methods.

- **BPR** (Rendle et al., 2012): BPR is a bayesian personalized ranking method for learning with implicit feedback. It adopts a pairwise loss function which assumes that an adopted service should be more weighted compared with an unadopted service.
- **GRU4Rec** (Hidasi et al., 2015): GRU4Rec employs GRU to model user sequences to generate recommendation results. It utilizes session-parallel mini-batch process and ranking-based loss functions to learn the model.
- **NCF** (He et al., 2017): NCF is a classic neural model in recommender systems. We initialize the embedding of application using our semantic component model and train the model through implicit feedback of the invocation between applications and services.
- **PNCF** (Chen et al., 2018): PNCF compresses all textual features of applications and services in an embedding layer, and then, uses MLP to model their interaction.
- **SPR** (Zhong et al., 2016): SPR uses descriptions and structures of applications to discover important lexical features of services and bridge the vocabulary gap between application developers and service providers. It jointly models application descriptions and invoked services, and uses Author Topic Model (ATM) to reconstruct service profiles.
- **JCA** (Zhu et al., 2019): JCA introduces a joint learning paradigm with a pair-wise loss such that the Auto-Encoder model captures the correlation between applications and services. This uses the pairwise loss proposed in JCA.
- **DINRec** (Zhou et al., 2018): DINRec applies a deep interest network (DIN) in CTR prediction to Web service recommendations. It exploits available features of the target mashup, selected and candidate services, and then learns their interaction in a well-designed network.
- **MTFM** (Wu et al., 2021): MTFM exploits a semantic component to generate representations of requirements and introduces a feature interaction component to model the feature interaction between applications and services.



(a) Precision (b) Recall (c) F1

(d) MAP (e) NDCG

Fig. 5. Performance comparison of different approaches (After selecting one service).

Table 3
Parameters used in this work.

	Parameter setting
GRU-based layer	unit number=50
DNN-based layer	unit numbers={64,32,16,8}
Adam algorithm	learning rate=0.0001
Electra model	hidden dimension=768
node2vec model	d=64, walk length=14, $p = 0.25$, q=4

5.1.4. Parameter settings

For the parameters of the ISRMM framework are given in Table 3.

For all the baseline approaches, we retained the default parameter settings mentioned in the original references.

5.2. Performance of ISRMM(RQ1)

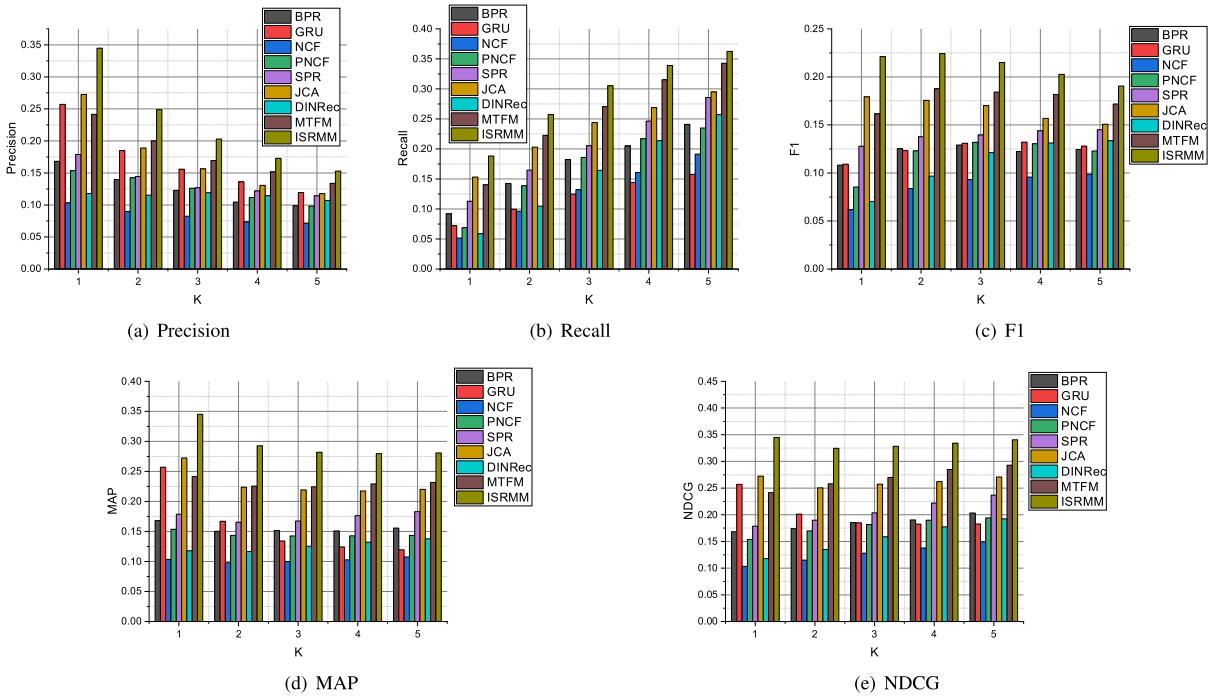
In this section, the performance of ISRMM is compared with the baseline methods, as shown in Figs. 5 and 6. In our experiment, we define two cases, one service or two services to be selected by the application. All results reported in this section pass t-test with p-value < 0.05, which means the improvements are significant. As can be found in Figs. 5 and 6, the experimental results perform similar trends on different cases. When there is only one selected service, for the baseline methods, the NCF model obtains the lowest values over all cases. NCF uses Electra to model the descriptions of applications and services, which requires relatively rich information. Unfortunately, many application documents in ProgrammableWeb dataset are short, which makes NCF unable to run stably. Besides, PNCF and DINRec perform slightly better than NCF, but they did not perform as well as we expected. Although BPR is simple, its performance is slightly better than NCF.

JCA performs slightly better than the above three models. With the increase in the number of selected services, JCA has shown better performance. SPR performs slightly better than JCA when there is only one selected service. Its hidden topic layer can overcome the disadvantages of the bag of words model. It semantically connects the requirements with the services even though the given documents may be not informative. Meanwhile, GRU4Rec has certain advantages in sequence modeling and has the function of long and short-term memory. The performance of MTFM is second best only to ISRMM. The possible reason is that, like ISRMM, MTFM utilizes description information and historical invocation records. When the developer selects only one service, ISRMM performs better in Precision, Recall, F1, MAP and NDCG than MTFM. The reason may lie in that the solely-selected service contributes little useful information or even becomes noise to the interactive learning process. When the number of selected services is increased to two, ISRMM achieves the best results for all given metrics. ISRMM takes full advantage of the descriptions and the invocation records between applications and services into service recommendations to reflect the relevant relationship between applications and services more effectively.

5.3. Ablation study (RQ2, RQ3)

Ablation study is widely used to detect the importance of components in machine learning models, especially in complex neural networks. For our ISRMM, We ablated four important components and conducted different approaches in this experiment.

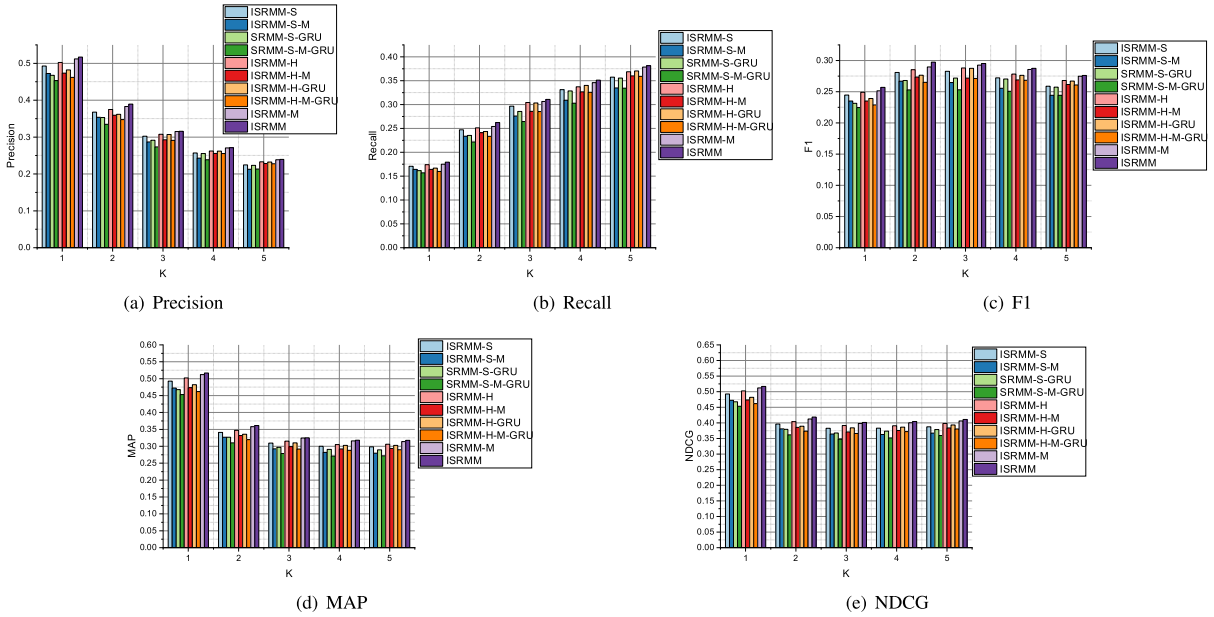
- **ISRMM-S**: the ISRMM framework without the semantic interaction model and only uses the historical interaction model to explore the relationship between applications and services.
- **ISRMM-S-M**: the ISRMM framework without the semantic interaction model and there is no multi-task learning layer in the historical interaction model.
- **ISRMM-S-GRU**: the ISRMM framework without the semantic interaction model and there is no GRU-based layer in the historical interaction model.
- **ISRMM-S-M-GRU**: the ISRMM framework without the semantic interaction model and there is neither GRU-based layer nor multi-task learning layer in the historical interaction model.
- **ISRMM-H**: the ISRMM framework without the historical interaction model and only uses the semantic interaction model to explore the relationship between applications and services.
- **ISRMM-H-M**: the ISRMM framework without the historical interaction model and there is no multi-task learning layer in the semantic interaction model.



(a) Precision (b) Recall (c) F1

(d) MAP (e) NDCG

Fig. 6. Performance comparison of different approaches (After selecting two services).



(a) Precision (b) Recall (c) F1

(d) MAP (e) NDCG

Fig. 7. Performance of ISRMM variants (After selecting one service).

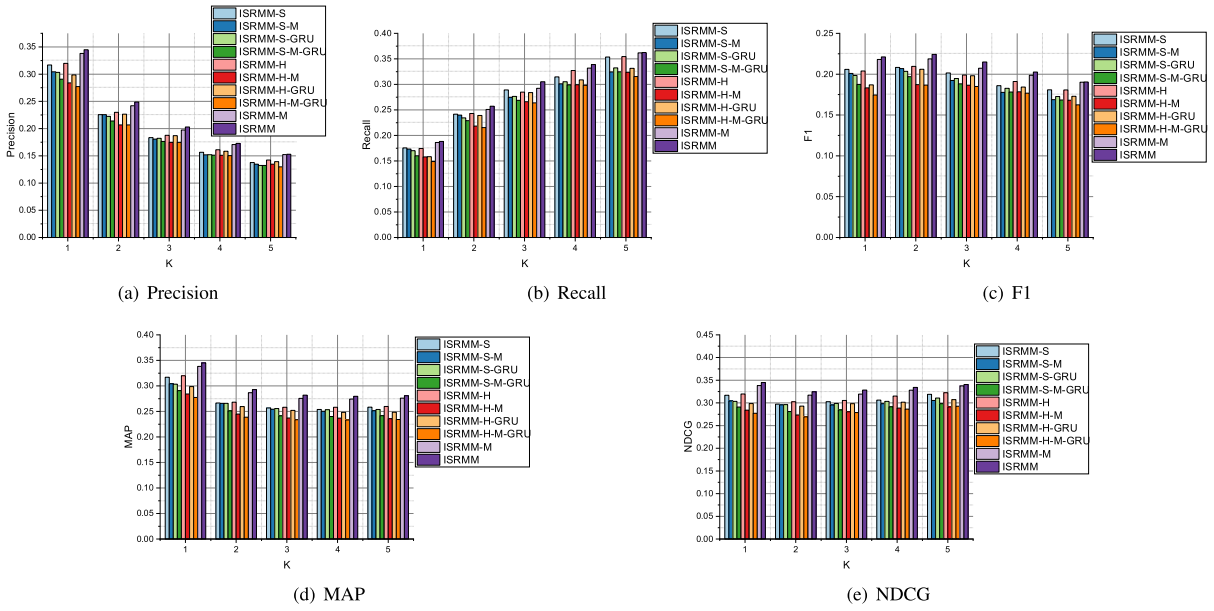
• **ISRMM-H-GRU**: the ISRMM framework without the historical interaction model and there is no GRU-based layer in the semantic interaction model.

• **ISRMM-H-M-GRU**: the ISRMM framework without the historical interaction model and there is neither GRU-based layer nor multi-task learning layer in the semantic interaction model.

• **ISRMM-M**: the ISRMM framework without the multi-task learning layer in the multi-model fusion model.

In order to verify the effectiveness of semantic interaction model and historical interaction model, we removed them from ISRMM and obtain ISRMM-S and ISRMM-H respectively. As can be found in Figs. 7 and 8, Integrating the semantic interaction model and historical interaction model can improve the overall performance of recommendations. ISRMM-H outperforms ISRMM-S, meaning that the factor of description text has a more significant impact than history invocation records. In order to verify the effect of the GRU layer on the recommended performance, we remove it in ISRMM-S and ISRMM-H respectively, and get the other two ablation models ISRMM-S-GRU and ISRMM-H-GRU. Similar to the above observations, ISRMM-S performs better than ISRMM-S-GRU, and ISRMM-H perform better than ISRMM-H-GRU, indicating that the GRU layer plays a positive role in the service recommendations.

In order to verify the impact of multi-tasking on recommendation, we remove it from the ISRMM, ISRMM-S, and ISRMM-H and obtain



(a) Precision (b) Recall (c) F1

(d) MAP (e) NDCG

Fig. 8. Performance of ISRMM variants (After selecting two services).

another three ablation models, namely ISRMM-M, ISRMM-S-M and ISRMM-H-M respectively. As can be found in Figs. 7 and 8, ISRMM-M performs best among the three ablation models, but not as well as ISRMM. ISRMM-S has higher values than ISRMM-S-M, and ISRMM-H also has higher values than ISRMM-H-M, indicating that multi-task learning factor plays important effect in our proposed approach. Our proposed ISRMM achieves the best. Because it integrates the advantage of all factors (i.e., the semantic interaction model, the history interaction model, the multi-task learning, and multi-model fusion) into the recommendation process.

From the experimental results, we can conclude that, the semantic interaction model plays the greatest role in our proposed framework. Besides, the history interaction model has the second most important effect. Finally, the GRU-based layer and multi-task learning also play a key role in interactive service recommendation.

5.4. Running time comparison (RQ4)

Fig. 9 depicts the running time (measured in seconds) of our proposed method compared to baseline methods. As we can see, the speed up of our ISRMM over MTFM is significant. MTFM requires 324 s for training, while ISRMM only takes 84 s. thereby indicating that the ISRMM can better model the complex relationship between applications and services compared with MTFM. However, although the efficiency of ISRMM is slightly worse than SPR and JCA, the performance of ISRMM is obviously better. Taking a trade-off between effectiveness and efficiency, our proposed ISRMM still has the best performance. Thus, we can conclude that the empirical running time of our method is relatively acceptable compared to the related deep learning-based method.

5.5. Parameters analysis (RQ5)

Impact of hidden layers. The hidden layer size is a hyper-parameter. It is an important factor affecting the recommendation performance of ISRMM. We test the proposed framework with 1–4 hidden layers used to learn the application-service interaction in varying sizes from 256 to 32 hidden units. Results were recorded for the different measures@2, as demonstrated in Fig. 10. we find that three hidden layers result in the best performance for our dataset.

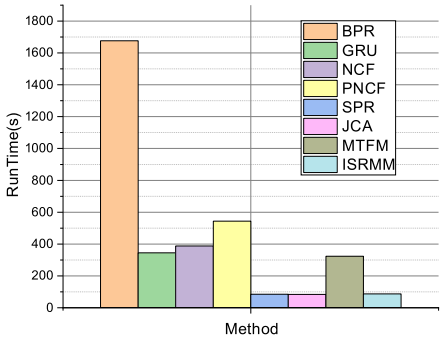
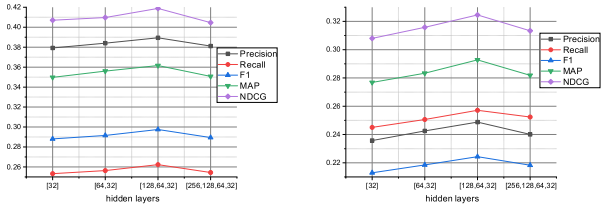


Fig. 9. The running time of the ISRMM compared with the baseline methods.



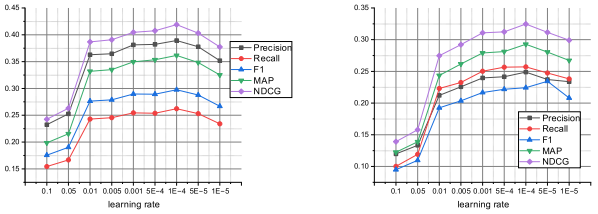
(a) After selecting one service (b) After selecting two services

Fig. 10. Impact of hidden layers.

Impact of learning rate. Since choosing the suitable value of this parameter has a direct effect on the recommendation accuracy. If it is set too large, the training may make the result exceed the optimal value and not converge. If it is set too small, the optimization process will take a long time due to the too slow descent rate. Therefore, choosing an appropriate learning rate is pivotal. Given the training set, we set the number of iterations as 300. We trained the ISRMM framework on the dataset instances using different learning rates (between 0.00001 and 0.1). Results were recorded for the different measures@2, as depicted in Fig. 11. The best performance under all evaluation metrics is achieved at learning rate=0.0001. The observation indicates that a relatively small learning rate is good for enhancing the recommendation results.

Table 4
Case study on service recommendation for DoSocial (where hit services are ranked in bold font).

Target application (Selected service)	Methods	Top3 recommendation services		
DoSocial (Facebook)	BPR	YouTube	Google Maps	**Twitter**
	GRU	Google Maps	YouTube	**Twitter**
	NCF	Amazon Product Advertising	Google Maps	**Twitter**
	PNCF	**Twitter**	eBay	Flickr
	SPR	**Twitter**	Flickr	Google Maps
	JCA	Google Maps	Flickr	Google Plus
	DINRec	**Twitter**	YouTube	Google Maps
	MTFM	Flickr	**LinkedIn**	Google Maps
	Ours	**Twitter**	**LinkedIn**	Feedly
	GroundTruth	**Google Base, Twitter, LinkedIn**		



(a) After selecting one service (b) After selecting two services

Fig. 11. Impact of learning rate.

5.6. Case study

Here, to be more illustrative, we select one existing application, *DoSocial* as a case study. With *DoSocial*, one can streamline her/his social media marketing campaigns and connect his/her social accounts to schedule and automate postings to them. By analyzing the above text requirements, the developer choose the *Facebook* service first. Table 4 shows the Top-3 recommendation results obtained by various methods after selecting *Facebook*. It also presents the services *DoSocial* actually calls (ground truth), i.e., *Google Base*, *Twitter*, and *LinkedIn*. As shown in the table, PNCF, SPR, and DINRec successfully put *Twitter* in the first place, but fail to recommend the rest two (*Google Base*, and *LinkedIn*) as the Top-3 candidates. Although BPR, GRU, and NCF recommend *Twitter* as a candidate, it was selected at the third place. JCA hits no service of ground truth among its Top-3 recommendation. As for MTFM, it correctly recommends a service different from others, i.e., *LinkedIn*. However, it fails to hit other services of ground truth. As Table 4 indicates, our method outperforms all others, with two hit services in the front positions. Intuitively, *Facebook*, *Twitter*, and *LinkedIn* are all popular online social platforms that our method recommends, and *Feedly* is a well-known service to manage content on relevant websites. The services we recommend meet the purpose of *DoSocial*. On the other hand, although *Google Maps*, *Flickr*, and *YouTube* are also popular services, they do not fit well the purpose of *DoSocial*. From the perspective of this requirement, the *YouTube* service may not be very suitable for recommendation because it allows users to integrate their program with YouTube. Furthermore, although YouTube is a social platform, it focuses on video sharing rather than posting. By comparison, services such as *Twitter* and *LinkedIn* may be more in line with the requirements. Besides, *Google Base*, as a service of database, can store text, images and web pages, while *Feedly* aggregates a user's favorite sites into a streamlined summary. *Google Base* and *Feedly* both offer the way to organize contents. On this point, they share some similarities to a certain extent. Here, the given requirements of "postings to them" may be consistent with the function of "store text, images, and web pages" which *Google Base* can provide. However, *Feedly* provides a more perfect match by only offering the feature of blog follow-ups. In other words, to replace *Google Base* with *Feedly* is a reasonable choice.

6. Conclusion and future work

This paper focuses on the online interactive service recommendation scenario for application development. We propose an interactive service recommendation framework based on deep learning to solve the problem of multiple rounds of service recommendation. The framework first extracts the collaboration information of the application through the history interaction model and models the description information through the semantic interaction model. Based on the fused features, we use tag judgment as an auxiliary task to improve the performance of service recommendations. The framework can understand the interaction between target application, the selected services, and the candidate service. The experimental results show that the framework is superior to some SOTA methods in different interactive development scenarios.

This paper only studies online interactive service recommendations from the perspective of function. In the future, we plan to incorporate QoS attributes into the interactive recommendation process. Secondly, a more powerful neural network model can be used in the semantic interaction model to enhance the effect of description modeling. Finally, we plan to extent the proposed approach to solve the long tail problem.

CRediT authorship contribution statement

Ting Yu: Conceptualization, Methodology, Writing – original draft. **Dongjin Yu:** Writing – review & editing, Supervision. **Dongjing Wang:** Supervision. **Quanxin Yang:** Investigation. **Xueyou Hu:** Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data and code are publicly available at https://github.com/HduDBSI/ISRMM.

Acknowledgments

This work is supported by National Natural Science Foundation of China (No. 62372145), Zhejiang Key Research and Development Program of China (No. 2023C03200), Jiaxing Science and Technology Bureau of China (No. 2023 AD11036) and Scientific Research Foundation of Zhejiang Provincial Education Department, China (No. Y202250319).

References

Chen, L., Zheng, A., Feng, Y., Xie, F., Zheng, Z., 2018. Software service recommendation base on collaborative filtering neural network model. In: International Conference on Service-Oriented Computing. Springer, pp. 388–403.

Clark, K., Luong, M.-T., Le, Q.V., Manning, C.D., 2020. Electra: Pre-training text encoders as discriminators rather than generators. arXiv preprint arXiv:2003.10555.

Duan, L., Gao, T., Ni, W., Wang, W., 2021. A hybrid intelligent service recommendation by latent semantics and explicit ratings. Int. J. Intell. Syst. 36 (12), 7867–7894.

Gao, W., Chen, L., Wu, J., Bouguettaya, A., 2016. Joint modeling users, services, mashups, and topics for service recommendation. In: 2016 Ieee International Conference on Web Services (Icws). IEEE, pp. 260–267.

Grover, A., Leskovec, J., 2016. node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 855–864.

Gu, Q., Cao, J., Liu, Y., 2021. Csbr: A compositional semantics-based service bundle recommendation approach for mashup development. IEEE Trans. Serv. Comput.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S., 2017. Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web. pp. 173–182.

Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D., 2015. Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939.

Jiang, Z., Liu, H., Fu, B., Wu, Z., Zhang, T., 2018. Recommendation in heterogeneous information networks based on generalized random walk model and bayesian personalized ranking. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. pp. 288–296.

Jiang, D., Qi, G., Hu, G., Mazur, N., Zhu, Z., Wang, D., 2020. A residual neural network based method for the classification of tobacco cultivation regions using near-infrared spectroscopy sensors. Infrared Phys. Technol. 111, 103494.

Kang, G., Tang, M., Liu, J., Liu, X., Cao, B., 2015. Diversifying web service recommendation results via exploring service usage history. IEEE Trans. Serv. Comput. 9 (4), 566–579.

Kwapong, B.A., Anarfi, R., Fletcher, K.K., 2019. Personalized service recommendation based on user dynamic preferences. In: Services Computing–SCC 2019: 16th International Conference, Held As Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 16. Springer, pp. 77–91.

Liang, T., Chen, L., Wu, J., Dong, H., Bouguettaya, A., 2016. Meta-path based service recommendation in heterogeneous information networks. In: Service-Oriented Computing: 14th International Conference, ICSOC 2016, Banff, AB, Canada, October 10–13, 2016, Proceedings 14. Springer, pp. 371–386.

Liang, T., Sheng, X., Zhou, L., Li, Y., Gao, H., Yin, Y., Chen, L., 2021. Mobile app recommendation via heterogeneous graph neural network in edge computing. Appl. Soft Comput. 103, 107162.

Mezni, H., Benslimane, D., Bellatreche, L., 2021. Context-aware service recommendation based on knowledge graph embedding. IEEE Trans. Knowl. Data Eng. 34 (11), 5225–5238.

Nguyen, M., Yu, J., Bai, Q., Yongchareon, S., Han, Y., 2020. Attentional matrix factorization with document-context awareness and implicit API relationship for service recommendation. In: Proceedings of the Australasian Computer Science Week Multiconference. pp. 1–10.

Qi, L., Zhang, X., Dou, W., Hu, C., Yang, C., Chen, J., 2018. A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment. Future Gener. Comput. Syst. 88, 636–643.

Qu, Q., Wei, S., Liu, S., Liang, J., Shi, J., 2020. JRNet: Jamming recognition networks for radar compound suppression jamming signals. IEEE Trans. Veh. Technol. 69 (12), 15035–15045.

Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L., 2012. BPR: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618.

Shi, M., Liu, J., et al., 2018. Functional and contextual attention-based LSTM for service recommendation in mashup creation. IEEE Trans. Parallel Distrib. Syst. 30 (5), 1077–1090.

Shi, M., Tang, Y., Liu, J., 2019. TA-BLSTM: tag attention-based bidirectional long short-term memory for service recommendation in mashup creation. In: 2019 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.

Wang, X., Liu, X., Liu, J., Chen, X., Wu, H., 2021. A novel knowledge graph embedding based API recommendation method for Mashup development. World Wide Web 24 (3), 869–894.

Wang, L., Zhang, X., Wang, R., Yan, C., Kou, H., Qi, L., 2020. Diversified service recommendation with high accuracy and efficiency. Knowl.-Based Syst. 204, 106196.

Wu, H., Duan, Y., Yue, K., Zhang, L., 2021. Mashup-oriented web API recommendation via multi-model fusion and multi-task learning. IEEE Trans. Serv. Comput. 15 (6), 3330–3343.

Wu, S., Shen, S., Xu, X., Chen, Y., Zhou, X., Liu, D., Xue, X., Qi, L., 2022. Popularity-aware and diverse web APIs recommendation based on correlation graph. IEEE Trans. Comput. Soc. Syst. 10 (2), 771–782.

Xie, F., Wang, J., Xiong, R., Zhang, N., Ma, Y., He, K., 2019. An integrated service recommendation approach for service-based system development. Expert Syst. Appl. 123, 178–194.

Xiong, W., Wu, Z., Li, B., Hang, B., 2020. Automating Mashup service recommendation via semantic and structural features. Math. Probl. Eng. 2020, 1–10.

Zhang, Y., Qian, Y., Wang, Y., 2018. A recommendation algorithm based on dynamic user preference and service quality. In: 2018 IEEE International Conference on Web Services (ICWS). IEEE, pp. 91–98.

Zheng, Z., Ma, H., Lyu, M.R., King, I., 2010. Qos-aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput. 4 (2), 140–152.

Zhong, Y., Fan, Y., Tan, W., Zhang, J., 2016. Web service recommendation with reconstructed profile from mashup descriptions. IEEE Trans. Autom. Sci. Eng. 15 (2), 468–478.

Zhou, G., Zhu, X., Song, C., Fan, Y., Zhu, H., Ma, X., Yan, Y., Jin, J., Li, H., Gai, K., 2018. Deep interest network for click-through rate prediction. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1059–1068.

Zhu, Z., Wang, J., Caverlee, J., 2019. Improving top-k recommendation via jointcollaborative autoencoders. In: The World Wide Web Conference. pp. 3483–3482.

Ting Yu received the B.S. and M.S. degrees in software engineering from Chongqing University, Chongqing, China, in 2012 and 2015, respectively. She is currently a Ph.D student in College of Computer Science of Hangzhou Dianzi University. Her current research interests include recommender systems, machine learning, and data mining.

Dr. Dongjin Yu is currently a professor from Hangzhou Dianzi University, China. His research efforts include service computing, machine learning and intelligent software engineering. He is the director of Big Data Institute, and the director of Computer Software Institute of Hangzhou Dianzi University. He is a senior member of IEEE, and a senior member of China Computer Federation (CCF). He also serves as the executive members of the technical committees of Service Computing CCF and Software Engineering CCF.

Dongjing Wang received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 2012 and 2018, respectively. He was co-trained at the University of Technology Sydney, Ultimo, NSW, Australia, for one year. He is currently a Lecturer with Hangzhou Dianzi University, Hangzhou. His current research interests include recommender systems, machine learning, and business process management.

Quanxin Yang received his B.S. degree in the Internet of Things Engineering from Xuchang University, Henan, China, in 2017. He is currently pursuing the Ph.D. degree in Computer Science and Technology with Hangzhou Dianzi University, Hangzhou, Zhejiang, China. His current research interests include image processing and intelligent software engineering.

Xueyou Hu has participated in numerous projects related with data mining and software engineering. His current research interests include service recommendation, intelligent software engineering, and big data.