Multi-label Recommendation of Web Services with the Combination of Deep Neural Networks

Yanglan Gan[1], Yang Xiang[2,3], Guobing Zou[2,3](✉), Huaikou Miao[2,4], and Bofeng Zhang[2](✉)

[1] School of Computer Science and Technology, Donghua University, Shanghai 201620, China
ylgan@dhu.edu.cn

[2] School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China
guobingzou@gmail.com, bfzhang@shu.edu.cn

[3] Shanghai Institute for Advanced Communication and Data Science, Shanghai University, Shanghai 200444, China

[4] Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai, China

Abstract. With the increasing number of web services on the Internet, how to effectively classify and recommend service labels has become a research issue. It plays an important role in web service organization and management. However, the deficiency of current approaches is that they either recommend only a single label for a web service or a set of independent labels without order ranking that is still difficult for service providers to publish their web services. In this paper, together with label embedding techniques, we propose a novel approach for service multi-label recommendation using deep neural networks. Unlike traditional approaches, the predicted service labels of our approach not only satisfy the demands of service multi-label recommendation, but also provide the importance with an ordered label ranking. The experiments are conducted to validate the effectiveness on a large-scale dataset from ProgrammableWeb, involving 13,869 real-world Web services. The experimental results demonstrate that our approach for multi-label recommendation of web services outperforms the competing approaches in terms of multiple evaluation metrics.

Keywords: Web service · Multi-label recommendation · Label embedding · Deep neural networks

1 Introduction

With the rapid advancement of web technology and the increasing demands on service-oriented applications, more and more software vendors publish their applications on the Internet as web services. Web services are platform-independent, modular, loosely coupled and self-describing distributed software components. As of 2019, the world's largest online web service registration platform, ProgrammableWeb, has registered 20,230 APIs and 7,937 mashup services. Those services significantly accelerate

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2019
Published by Springer Nature Switzerland AG 2019. All Rights Reserved

machine-to-machine interactions and promote the development of service-oriented applications. They can be published, discovered and selected [1], automatically composed [2, 3], scheduled [4], recommended [5, 6] and invoked over the Internet.

As the increasing number of web services published on the Internet and their diverse functionalities across different application domains, there are always hundreds of service categories in an online RESTful service repository. This makes it difficult to effectively organize and manage web services in a manual manner. As a result, it tends to be a labor-intensive challenging task for service providers to search and find one or multiple appropriate categories from registered ones, when publishing their API services on a service management platform [7]. Therefore, how to design a novel approach for service providers and help them effectively and automatically recommend appropriate service labels have become a challenging research topic.

In recent years, correlative research efforts have been posed to support automated organization and management of web services. Machine learning methods have been adopted for web service classification and recommendation. In the traditional approaches [7–15] for service classification and recommendation, only a single service label is recommended for a web service. However, in service-oriented software system applications, a web service often holds cross-domain characteristics. Thus, a service provider is required to choose multiple labels for a web service when it is published to a service management platform. Moreover, unlike the traditional multi-label classification problem, service multi-label recommendation aims at recommending a sorted sequence of service labels, instead of a set of independent ones. For example, each web service in the ProgrammableWeb has a sorted sequence of service domain labels with different priorities. Therefore, as the number of services increases dramatically, how to recommend ordered multiple labels for web services is an urgent research issue to be solved.

To handle above research issue, we proposed a novel approach for service multi-label recommendation using deep neural networks. First, a convolutional neural network [16] is applied to extract service general and sequence features. Then, by using the relationship among service labels, a label embedding model is proposed to generate label feature representation of a web service. Finally, together with label embedding and attention mechanism [17], a Gated Recurrent Unit (GRU) deep neural network is used to recommend a sorted sequence of service labels. Consequently, the predicted service labels not only satisfy the demands of service multi-label recommendation, but also provide the importance with an ordered label ranking.

To test the performance of service multi-label recommendation, extensive experiments are conducted on a large-scale real-world dataset from ProgrammableWeb, involving 13,869 real-world web services with 474 service labels. We compare our approach with the state-of-the-art three existing machine learning-based approaches on service multi-label recommendation. The experimental results demonstrate that our approach can outperform those competing approaches in terms of multiple evaluation metrics. The main contributions of this paper are summarized as follows.

- We propose a novel service multi-label recommendation framework with the combination of deep neural networks, where convolutional and recurrent neural networks are combined together to predict an ordered sequence of service labels, instead of a set of independent ones without label ranking.

- We propose a novel model for service label embedding. With the consideration of the associations among different service labels, a label embedding model is trained to support more accurate generation of service multi-label recommendation.
- We design and implement a prototype system and conduct extensive experiments on a real-world dataset from ProgrammableWeb. The experimental results demonstrate that our approach for multi-label recommendation of web services is superior to existing completing approaches.

The remainder of this paper is organized as follows. Section 2 presents the overall framework. Section 3 presents the details of our approach for service multi-label recommendation. Section 4 shows extensive experiments and analyzes the results. Section 5 reviews the related work. Finally, Sect. 6 concludes the paper.

2 Framework of the Approach

To recommend multiple labels for a Web service, we propose an approach using deep neural networks. It mainly consists of two components, including service feature extraction and service label generation. Figure 1 shows the overall framework of service multi-label recommendation.

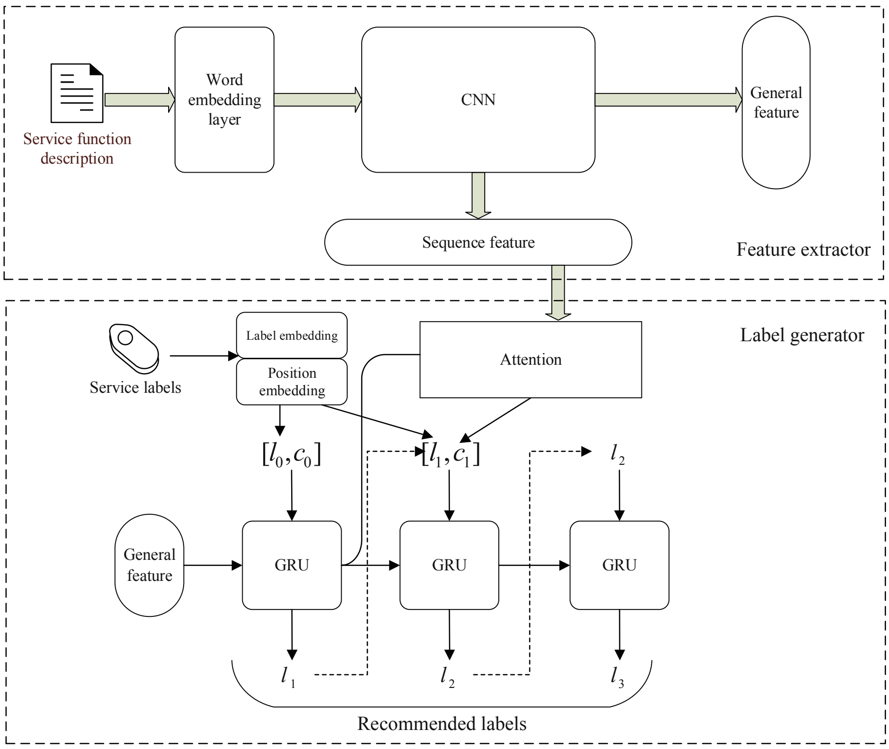

Fig. 1. The framework of multi-label recommendation of web services

In the stage of extracting service features, taking the functional description of web services as inputs, it is converted into a matrix representation through the layer of word embedding. Convolutional neural network (CNN) is then used to extract service general and sequence features respectively, where service general feature is a feature vector to reflect the whole feature of service functional description and sequence features correspond to each service description word.

In the stage of service multi-label recommendation, the extracted service features are taken to the label generator that is a recurrent neural network based on GRU. Specifically, service general feature is used as the initialization of the hidden feature of GRU model. Service labels and their positions are converted to a vector by the label embedding model. Then, the attention layer converts service sequence features to a vector. By merging service feature vector and label vector as the input of the GRU unit, it outputs the label probability of a web service. Through the iterative process, a sequence of sorted service labels is generated recommended.

3 Service Multi-label Recommendation

3.1 Service Feature Extraction

In order to extract service features, convolutional neural network (CNN) is applied to transform functionality description of a web service into a low-dimensional feature vector. Two kinds of service features are extracted based on convolutional neural network, including service general and sequence features. Service general features aim to describe the overall functionality of a web service, while service sequence features convey location information for each service description word. The process of extracting service features is shown in Fig. 2.

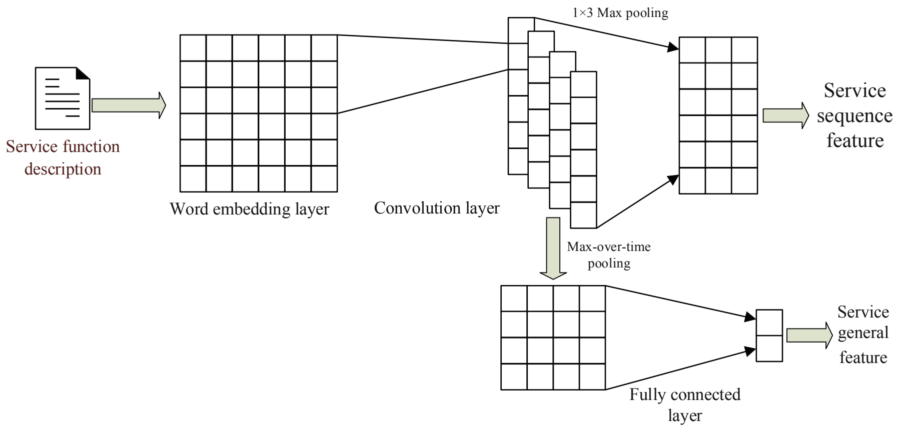

Fig. 2. Service general and sequence features extraction by convolutional neural network

In the word embedding layer, it takes a service function description as an input, where each word is converted into a vector of length t. By using a matrix $\boldsymbol{e}$ with $n \times t$

dimension, where each row represents a representation of a word, the embedded feature of a service description word is expressed as

$$We(w_i) = onehot(w_i)^T \boldsymbol{e} \tag{1}$$

Where $onehot(w_i)$ is an n dimension vector, and the value of the i-th is 1 and the rest of the values are 0. $We(w_i)$ is the embedded representation of w_i. $\boldsymbol{e}$ is initialized by the trained word embedding model.

Definition 1 (Service Embedding Representation). Given a service function description D_s, word embedding representation of a web service s is denoted as $e(D_s) = (We(w_1), We(w_2), \ldots, We(w_m))^{\mathrm{T}}$, where, $w_1, w_2, \ldots, w_m \in D_s$ and $e(D_s)$ is a matrix with the dimension $m \times t$.

In the convolutional layer, three different scales of convolutional kernels with the representation of parameter matrix W are used for one-dimensional convolution, which is expressed as

$$Y_{conv} = e(D_s) * W \tag{2}$$

In the convolutional operation, the value of each point in the convolution result vector is calculated by

$$y_i = \sigma(\sum_{k=1}^{m} w_k^T e_{i-k+1}) \tag{3}$$

Where y_i is value of the i-th point on the convolution result Y_{conv}, w_k is the parameters of the k-th line in the convolutional kernel matrix W, and e_k is the k-th line of the embedding matrix of service function description. σ is a nonlinear function that determines activation degree of each neural unit. For faster convergence, an activation function ReLU is used to improve the model learning efficiency.

When extracting service features, maximum pooling is applied to generate sequence features with the same length as the input sequence. For service general features extraction, max-over-time pooling is used based on a fully connected layer.

3.2 Service Label Embedding

To improve of the accuracy of service multi-label recommendation, service labels are embedded from the idea of Word2vec. Embodying service label semantics and label relationships, a service label is converted into an embedding vector. The training process of service label embedding is illustrated in Fig. 3, which is divided into pre-training stage and actual training stage.

In the pre-training stage, a Word2vec model is first trained through Wikipedia corpus, and then service function descriptions from web service library are selected as training data to boost the accuracy of Word2vec model. In the actual training stage, all of the service labels extracted from web service library and their relationships are taken into account to learn service label embedding model, when trained based on the generated Word2vec model.

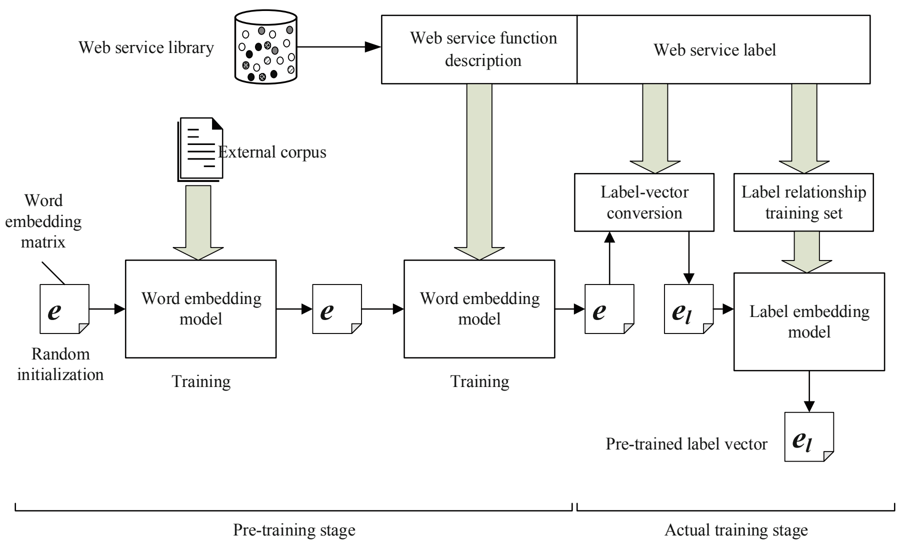

Fig. 3. The two-stage training process of service label embedding

Based on the pre-trained word embedding model, the pre-trained service label vector can be obtained. When a service label contains only one word, its vector can be directly obtained by pre-trained word embedding model. However, if a service label that is composed of multiple words, its corresponding vector is equal to the sum of the vectors of all the words in the label. For each position of the word embedding vector, it is divided by the number of words in the label. It is expressed as

$$\boldsymbol{e}(l_i) = \frac{1}{|Words(l_i)|} \sum_{w_j \in Words(l_i)} \boldsymbol{e}'(w_j) \tag{4}$$

Where $Words(l_i)$ represents the list of words contained in the label l_i, $\boldsymbol{e}$ is the pre-trained service label vector with multiple words, and $\boldsymbol{e}'$ is the pre-trained word embedding vector.

To learn a label embedding model, a training set based on the relationship between web services and their corresponding labels is generated. Assume that each web service in the training set has multiple labels. The process of training label embedding aims at applying the model to predict correlative service labels. Accordingly, a set of labels pairs are generated as training set of label embedding. Given a web service s_i that has j number of service labels $l_1, l_2, \ldots, l_j$, j group of service labels can be generated, including $(l_1|l_2, \ldots, l_j), (l_2|l_1, l_3 \ldots, l_j), \ldots, (l_j|l_1, \ldots, l_{j-1})$. As a result, each service label group $(l_x|l_1, \ldots, l_{x-1}, l_{x+1}, \ldots, l_{j-1})(1 \le x \le j)$, where a set of service label pairs $(l_x, l_1), (l_x, l_2), \ldots, (l_x, l_j)$ are generated. For example, suppose that there is a web service with labels on news, web, and mapping, six pairs of training data are generated, as shown in Table 1.

Table 1. A motivating example of six pairs of training data generated by a web service

Input label	Predicted label
News	Web
News	Mapping
Web	News
Web	Mapping
Mapping	News
Mapping	Web

After the above two steps, we construct a Word2vec model that is a simple fully connected network for service label embedding. Specifically, a vector representation of one of the service labels is used to predict the vector representation of the other service label. The objective function of the label embedding training process is

$$L = \sum_{w_i \in \mathbb{D}} \sum_{w_j \in Context(w)} \log P(w_i | w_j) \tag{5}$$

$$P(w_i | w_j) = \frac{\exp(\boldsymbol{e}'(w_i)^T \boldsymbol{e}(w_j))}{\sum_{w' \in \mathbb{V}} \exp(\boldsymbol{e}'(w')^T \boldsymbol{e}(w_j))} \tag{6}$$

Where L represents the objective function, $\mathbb{D}$ and $\mathbb{V}$ represent the entire corpus and dictionary. $\boldsymbol{e}$ and $\boldsymbol{e}'$ represent a word vector matrix to be trained and a word vector matrix to be generated as output, respectively. $\boldsymbol{e}$ and $\boldsymbol{e}'$ are updated using the stochastic gradient descent method to maximize the objective function.

After completing the training, we discard $\boldsymbol{e}'$ and use $\boldsymbol{e}$ as the finally pre-trained label embedding model. It converts a service label into a vector representation that contains both label semantics and inter-label associations. Thus, taking label embedding vectors as inputs for service label generation can improve the accuracy of service multi-label recommendation.

3.3 Service Label Sequence Generation

Based on the service feature extraction and service label embedding, a sorted sequence of service labels can be generated by service label generator, as illustrated in Fig. 4. It initializes the hidden feature by using the service general feature and GRU unit outputs the first label and the hidden state. They are then used as inputs to generate a subsequent sequence of service labels. The generation process is finished until it reaches the convergence condition. The service label generator consists of five correlative layers for multi-label recommendation of web services.

(1) Service Label Embedding Layer. When service label sequence generator predicts the *t*-th label, it receives the t-1 label that is converted into a label vector through service label embedding layer. It is constructed the same as the word embedding layer. Here, the weight matrix directly adopts the pre-trained label embedding model.

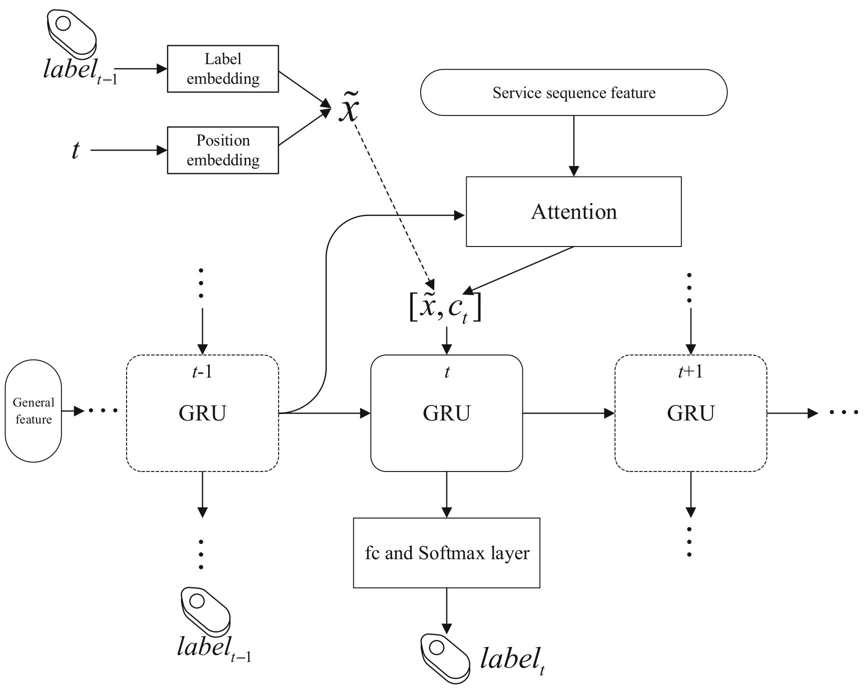

Fig. 4. The model of service label sequence generation with recurrent neural networks

(2) Position Embedding Layer. Together with service label embedding, we take into account position embedding to enable recurrent neural network to better capture location information to promote the multi-label recommendation. Position embedding is a technique for vectorizing position numbers in a sequence, which is defined as

$$\begin{cases} PE_{2i}(p) = \sin(p/10000^{2i/d}) \\ PE_{2i+1}(p) = \cos(p/10000^{2i/d}) \end{cases} \tag{7}$$

Where PE_i is the value of the i-th element of the position vector PE, d is the dimension of the position embedding vector, and p is the number of the current position. Here, it takes the same dimension as the label embedding vector. The connection of label embedding vector and position embedding vector is fed into the next GRU unit that is expressed as

$$\tilde{x} = [\boldsymbol{e}(label), PE(t)] \tag{8}$$

(3) Attention Layer. In order to solve the problem that a single feature vector lacks of enough information, the attention mechanism is applied to calculate the attention weight vector by the hidden state h_t and service sequence feature Z at the previous moment. After connecting with $\tilde{x}$, it is used as the input of the GRU layer.

$$x = [Attention(h_t, Z), \tilde{x}] \tag{9}$$

The attention layer uses the attention mechanism to generate a vector that activates the input sequence portion. The guided model focuses only on a portion of the input sequence, which is express as

$$u_i^t = v^T \tanh(W_1 h_t + W_2 Z_i) \tag{10}$$

Where u_i^t is the current attention score calculated by the input sequence and feature vector at the current time step t of the recurrent neural network, W_1, W_2 and v are internally adjustable weight matrices of the attention model, h_t is the hidden layer feature vector, Z_i is the output of the input sequence processed by the feature extractor at each time step. Since u_i^t is used as a weight, it is normalized using softmax so that the sum of the weights equals to 1.

$$a_i^t = \mathrm{softmax}(u_i^t) \tag{11}$$

Where a_i^t is the attention weight vector at the current time step t. After Z_i and a_i^t are weighted and summed, they are used as input to the recurrent neural network as

$$c_t = \sum_{i=1}^{|Z|} a_i^t Z_i \tag{12}$$

Where c_t is used as the input to our GRU unit at time step t, and h_t is the hidden state input of the recurrent neural network.

(4) GRU and Fully Connected with the Softmax Layer. In the structure of the recurrent neural network, GRU is used in the layer and its operation is subjected to service multi-label recommendation through the fully connected with softmax layer.

3.4 Model Training

As the service multi-label recommendation is an end-to-end learning model, the two crucial stages including service feature extraction and service label generation need to be combined together for training. Here, the loss function of the model for the t-th service label is evaluated by calculating a cross entropy as

$$J_t = -\frac{1}{n}\sum_{i=1}^{n} \left(y_i^t \log Model(x)_i^t + (1 - y_i^t)\log(1 - Model(x)_i^t)\right) \tag{13}$$

Where n is the dimension of the output service label and $Model(x)$ is the predicted label of the model for input x. y is the real service label and t is the t-th service label predicted by the model. The total loss for one piece of data is the average of the loss predicted by the label at each location, which is expressed as

$$J = \frac{1}{m}\sum_{t=1}^{m} J_t \tag{14}$$

Upon the preset maximum number of training, the loss function J is calculated by comparing with the real output service labels. Here, stochastic gradient descent method is used to iteratively optimize the objective loss function that can be minimized by backpropagation and updating the parameters in the model.

4 Experiments

4.1 Experimental Data Set

We have designed and implemented a prototype system and conducted the extensive experiments to validate the effectiveness of our proposed approach for service multi-label recommendation. All the experiments are run on Linux Operating System and carried out on a platform with an NVIDIA GTX1080Ti*2 GPU, Intel(R) Xeon(R) Gold 6132*2 CPU and 192 GB RAM.

The data set used in the experiments was collected from a web service management platform ProgrammableWeb[1], the world's largest online API and mashup service repository. As of 2019, there are 20,230 web APIs, 7,937 mashups, 545 web services development frameworks, 1,698 development libraries, and 14,325 SDKs. The Web APIs included in ProgrammableWeb are web services actually used in the real world applications. In the experiments, we have crawled API services, including service name, function description and labels that can be visualized and downloaded from our lab[2]. After the preprocessing, we obtained a collection of 13,869 API services. The statistics of experimental data set is shown in Table 2.

Table 2. The statistics of experimental data set crawled from ProgrammableWeb

Data set item	Value
Number of API services	13,869
Total number of labels	474
Total number of main labels	375
Average number of service labels	3.16
Minimum number of service labels	1
Maximum number of service labels	69
Total number of words of API services	932,450

[1] https://www.programmableweb.com/

[2] http://dmis.shu.edu.cn/ProgrammableWebData

In the experimental data set, API services have different number of labels. The distribution of the number of service labels is shown in Fig. 5. The majority of API services correspond to the number of labels ranging from 1 to 5, while the number of API services with 10 or more labels is 61, accounting for 0.44% of the total number of API services.

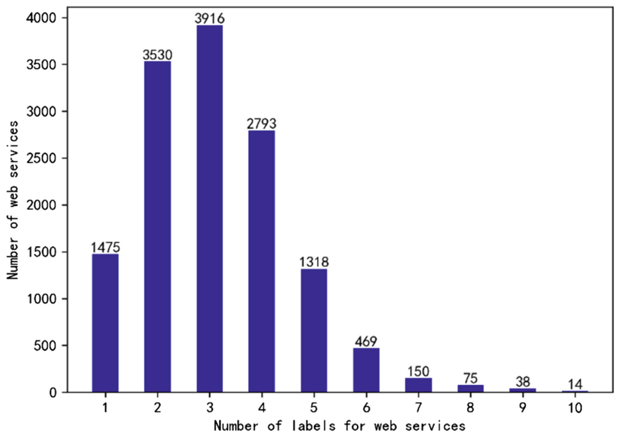

Fig. 5. Distribution of the number of service labels in experimental data set

In terms of the domain of service labels, each of which consists of the number of API services. The distribution of the number of API services within each service label is illustrated in Fig. 6. From the experimental data set, we can find that Tools, Financial, eCommerce, Messaging, Enterprise, Social, Payments, Mapping, Government, and Science correspond to the most number of API services.

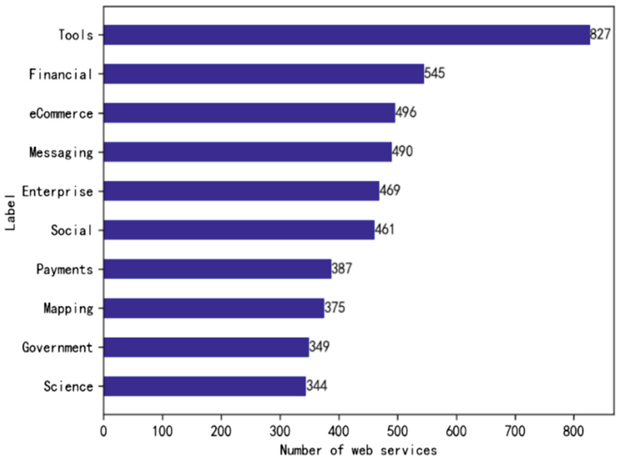

Fig. 6. Distribution of the number of API services within each service label

4.2 Competitive Methods

In order to show the feasibility and effectiveness of our approach, we carried out experiments and compared with three competing approaches for multi-label recommendation of web services, including multi-label Bayesian classifier (ML-Bayes), multi-label SVM classifier (ML-SVM), and convolutional neural network multi-label classifier (CNN-ML). Our self-developed and the comparative methods both have used the pre-trained word embedding model to convert service function description to vector and matrix representation. The three comparative service multi-label recommendation methods are described as below.

- **ML-Bayes.** It is a service multi-label recommendation method based on Bayesian classifier. It converts a multi-label recommendation problem into a set of naive Bayes classifiers of one-vs-other binary classification problem. Each classifier determines whether a web service to be classified belongs to the current label or other labels, which are combined to generate the final multi-label recommendation.
- **ML-SVM.** It is a service multi-label recommendation method based on Support Vector Machine (SVM) classifier. It converts a service multi-label recommendation problem into a set of one-vs-other two-category problem by using SVM classification. All of the classification results are integrated to obtain the final multiple service labels.
- **CNN-ML.** It is a service multi-label recommendation method based on convolutional neural network. We use Text-CNN deep learning model to extract service features. Followed by the fully connected layer of Text-CNN, the Sigmoid layer normalizes the output probability between 0 and 1. The output of each value of the output layer indicates the probability that a label is marked. By choosing the predefined number of recommended results, it generates a series of service labels.

4.3 Evaluation Metrics

In order to measure the accuracy on service multi-label recommendation among different approaches, Recall and MAP (Mean Average Precision) are used as the evaluation metrics. In addition, NDCG (Normalized Discounted Cumulative Gain) value is u is also provided as a ranking reference for evaluating the performance of service multi-label recommendation. Each evaluation indicator is described as below.

(1) Recall. It refers to the ratio of the correct number of predicted service labels in the multi-label recommendation list to the total number of labels of web service itself, which is defined as

$$Recall = \frac{|\{labels\} \cap \{predictedLabels\}|}{|\{labels\}|} \tag{15}$$

Where *labels* is the real set of labels of a service itself, and *predictedLabels* is a set of predicted labels recommended by a method. For example, given a web service s and its corresponding labels are $\{l_1, l_2, l_3\}$, if the recommended service labels are $\{l_1, l_3, l_5\}$, then the recall of multi-label recommendation is $\frac{|\{l_1, l_3\}|}{|\{l_1, l_2, l_3\}|} = \frac{2}{3} \approx 0.667$.

Recall@n refers to the rate when the total number of labels of web service itself $|\{labels\}| = n$. That is, the first n service labels are used to calculate the recall rate. Here, recall measures the degree of the recommendation results that cover all of the original service labels.

(2) MAP. It is the expected average of the precision that is the ratio of the correct number of predicted service labels in the recommendation list, which is defined as

$$P = \frac{|\{labels\} \cap \{predictedLabels\}|}{|\{predictedLabels\}|} \tag{16}$$

In order to measure precision of the service multi-label recommendation more accurately, the value of the precision corresponds to the *Pr* function that is defined as the value of the recall. The *Pr* function reflects the change in precision when the recall rate changes from 0 to 1. Integrating *Pr* function in the 0-1 interval yields the expectation of multi-label recommendation precision. It is defined as

$$MAP@n = \int_0^1 Pr(r)dr = \frac{\sum_{k=1}^{n} P(k)rel(k)}{|\{labels\}|} \tag{17}$$

Where $P(k)$ is the precision of the first k service multi-label recommendation results, $P(k) = \frac{|\{labels\} \cap \{predictedLabels(k)\}|}{k}$, and $rel(k)$ indicates whether the k-th service label prediction is correct. That is, if it is correctly predicted, $rel(k)$ is set as 1, otherwise it is set as 0.

(3) NDCG. It reflects the impact of the service multi-label recommendation at each location in the overall recommendation. It is a location-sensitive evaluation metric, that is, for the recommended results at each location, the value is decremented accordingly. It is defined as

$$NDCG@n = \frac{DCG@n}{IDCG@n} \tag{18}$$

Where $DCG@n = \sum_{k=1}^{n} \frac{2^{rel(k)}-1}{\log_2(k+1)}$ and $IDCG@n$ represents the idealized DCG value. That is, DCG value is calculated according to the optimal ranking in the current recommendation.

4.4 Experimental Results and Analyses

To test the performance of our proposed approach, the extensive experiments on service multi-label recommendation are conducted among four competitive approaches, where GRU LabelsGen is used to represent our self-developed one. Table 3 shows the experimental results on three different evaluation metrics.

Table 3. Experimental results of service multi-label recommendation among four approaches

	MAP@5	Recall@5	NDCG@5
ML-Bayes	0.168	0.129	N/A
ML-SVM	0.209	0.113	N/A
CNN-ML	0.331	0.188	N/A
GRU LabelsGen	**0.762**	**0.591**	**0.432**

It can be seen from the experimental results that our proposed service multi-label recommendation approach is superior to the existing ones among the three evaluation metrics. Therefore, our approach with the combination of deep neural networks for multi-label recommendation of web services is effective in large-scale data set with a set number of service labels.

In the experiments, ML-Bayes and ML-SVM convert a label prediction task into training a large number of classifiers. Thus, they both have high time complexity as the number of service labels increases. Also, the performance on MAP@5 and Recall@5 of SVM-based service multi-label recommendation approach outperforms that of Bayes-based approach. Furthermore, the proposed approaches have better performance on MAP@5 compared to Recall@5 in MAP. The main reason is that the order of the results of service multi-label recommendation has no absolute impact on the MAP, although it is partially related to the order when predicting the main service labels can improve the accuracy of the experimental results. Therefore, our approach holds a higher MAP score by recommending a sorted sequence of service labels.

As for NDCG evaluation index, it can further test the correctness of the order of recommended service labels for our proposed approach. Since the competitive three service multi-label recommendation approaches cannot output an ordered sequence of service labels, the corresponding NDCG value cannot be calculated. In our proposed approach, NDCG represents the overall ranking accuracy of the predicted sequence of service labels. From the results, it indicates that the proposed approach can recommend a reasonable label sequence of web services.

In order to further test the parameter influence on the service multi-label recommendation, a set of experiments are carried out among four competitive approaches. The experimental results of parameter tuning are illustrated in Figs. 7 and 8.

Along with the changes of parameter for the number of recommended service labels, Figs. 7 and 8 illustrate the experiments results on MAP@n and Recall@n among four competitive approaches. In the experiments, the parameter of MAP@n and Recall@n ranges from 1 to 5 and the results are calculated with different values. Compared to the existing approaches, Fig. 7 shows that the experimental results of our proposed approach has better MAP@n along with the changes of n. Specifically, as n of MAP@n changes from 1 to 5, the experimental results of each approach have a certain decrease, and reach the best multi-label recommendation at MAP@1. Therefore, as the number of service labels to be evaluated increases, the predictive power of each multi-label recommendation approaches declines. However, our self-developed approach has the lowest decline compared with the other three approaches. From the results, it is observed that the proposed method is more suitable for service multi-label recommendation with short function text description.

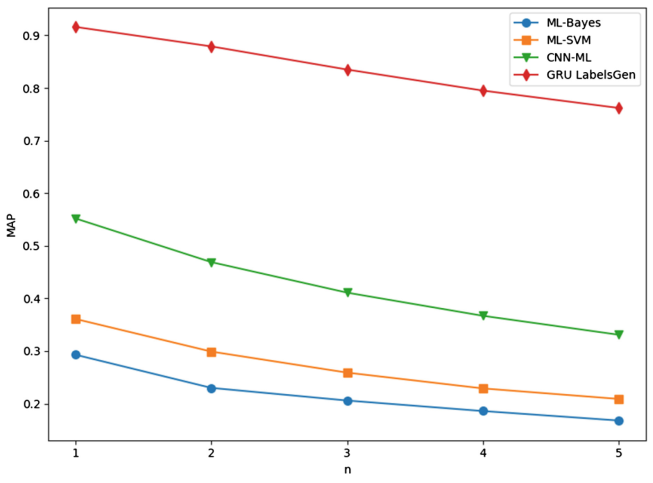

Fig. 7. The experimental results of MAP@n affected by the parameter n

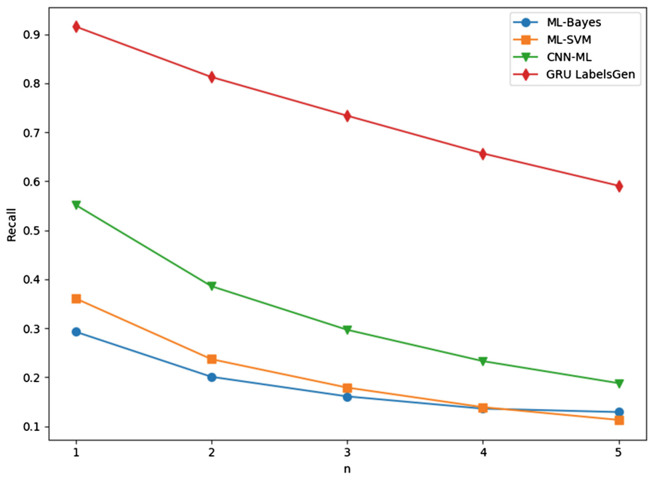

Fig. 8. The experimental results of Recall@n affected by the parameter n

Figure 8 compares the experimental results on Recall@n among four approaches. From the results, it is concluded that CNN-ML approach is better than the traditional two approaches without deep neural networks on Recall@1. However, along with the increase of parameter, the difference becomes smaller and smaller. It indicates that CNN-ML approach is more suitable for service single-label recommendation. The downward trend of our self-developed approach is lower than the other three approaches, indicating that the effectiveness of our proposed approach for multi-label recommendation is superior to the existing competitive approaches for recommending a sequence number of service labels.

5 Related Work

In recent years, correlative research has been investigated on web service classification and recommendation. Wang et al. took advantage of SVM to classify web services into corresponding categories [8]. During the process of service classification, a standard taxonomy, UNSPSC, was used to model the feature space of web services. Although SVM algorithm performs well in service classification, only a single classification method has been applied to recommend and classify web services. Based on the WSDL service description documents, Nisa et al. proposed a text mining approach [9] to automatically classify web services to specific domains and discover key concepts from their functionality descriptions. However, it is still subject to one classification method that still cannot achieve the best classification performance. To overcome above limitations of the conventional techniques for service classification, Qamar et al. focused on the classification of web services using a majority vote based classifier ensemble technique [10], where three heterogeneous classifiers Naïve Bayes, decision tree (J48), and Support Vector Machines are applied to vote for more effective classification of web services.

Recently, more sophisticated techniques are exploited to classify services. Since huge service classification taxonomies at multiple hierarchical levels, Syed et al. proposed a novel approach for web service classification by multi-layer perceptron neural network (MLPNN) [11]. A multi-layer perceptron optimized with tabu search for learning is proposed to automatically classify web services from multiple service categories. Lee et al. developed an IoT service classification and clustering system [12], which classifies the operation of an IoT service by their characteristics. Classic EM algorithm is used to cluster IoT services based on their classification in terms of their similarities. To reduce the human effort on labeling services, Liu et al. proposed a service classification approach by active learning algorithm [13], where LDA is applied to learn an optimum SVM model as service classifier. Based on [13], Shi et al. proposed a multi-label active learning approach [14] for web service tag recommendation, where active learning was considered to train a multi-label classifier with a correlation-aware learning strategy. More recently, Liang et al. presented a graph-based approach [15] to automatically assign tags to unlabeled API services by exploiting both graph structure information and semantic similarity.

Observed from the above investigations, we propose a novel approach for multi-label recommendation of web services with the combination of convolutional and recurrent deep neural networks. By considering the relationships of service labels, it achieves better service recommendation accuracy compared to the existing competitive approaches, where a sorted sequence of labels is recommended.

6 Conclusion

To effectively predict multiple labels of web services with order ranking, we proposed an approach for multi-label recommendation of web services using deep neural network. First, a convolutional neural network Text-CNN has been applied to extract general and sequence features of web services. Taking the associations of service labels, a label embedding model is then proposed to provide label feature representation. Finally, together with web services and their label embedding features, a recurrent neural network GRU is used to recommend a sorted sequence of service labels. Extensive experiments have been conducted on large-scale web service repository. The results demonstrate that our proposed approach outperforms the competitive ones for multiple labels recommendation of web services.

Acknowledgment. This work was partially supported by Shanghai Natural Science Foundation (No. 18ZR1414400, 17ZR1400200), National Key Research and Development Program of China (No. 2017YFC0907505), National Natural Science Foundation of China (No. 61772128, 61602109), and Shanghai Sailing Program (No. 16YF1400300).

References

1. Hwang, S., Hsu, C., Lee, C.: Service selection for web services with probabilistic QoS. IEEE Trans. Serv. Comput. **8**(3), 467–480 (2015)
2. Laleh, T., Paquet, J., Mokhov, S., Yan, Y.: Constraint adaptation in web service composition. In: IEEE International Conference on Services Computing, pp. 156–163 (2017)
3. Deng, S., Wu, H., Taheri, J., Zomaya, A., Wu, Z.: Cost performance driven service mashup: a developer perspective. IEEE Trans. Parallel Distrib. Syst. **27**(8), 2234–2247 (2016)
4. Li, W., Xia, Y., Zhou, M., Sun, X., Zhu, Q.: Fluctuation-aware and predictive workflow scheduling in cost-effective Infrastructure-as-a-Service clouds. IEEE Access **6**, 61488–61502 (2018)
5. Bai, B., Fan, Y., Tan, W., Zhang, J.: DLTSR: a deep learning framework for recommendation of long-tail Web services. IEEE Trans. Serv. Comput. (2017). https://doi.org/10.1109/tsc.2017.2681666
6. Shi, M., Tang, Y., Liu, J. Functional and contextual attention-based LSTM for service recommendation in mashup creation. IEEE Trans. Parallel Distrib. Syst. (2018). https://doi.org/10.1109/tpds.2018.2877363
7. Pang, S., Zou, G., Gan, Y., Niu, S., Zhang, B.: Augmenting probabilistic topic model for web service classification. Int. J. Web Serv. Res. **16**(1), 93–113 (2019)
8. Wang, H. B., Shi, Y. Q., Zhou, X., Bouguettaya, A.: Web service classification using support vector machine. In: IEEE International Conference on TOOLS with Artificial Intelligence, pp. 3–6 (2010)
9. Nisa, R., Qamar, U.: A text mining based approach for Web service classification. Inf. Syst. e-Business Manage. **13**(4), 751–768 (2015)
10. Qamar, U., Niza, R., Bashir, S., Khan, F.: A majority vote based classifier ensemble for Web service classification. Bus. Inf. Syst. Eng. **58**(4), 249–259 (2016)
11. Syed, A., Kumara, S.: Web service classification using multi-layer perceptron optimized with tabu search. In: IEEE International Advance Computing Conference, pp. 290–294 (2015)
12. Lee, D., Lee, H.: IoT service classification and clustering for integration of IoT service platforms. J. Supercomputing **74**(12), 6859–6875 (2018)
13. Liu, X.M., Agarwal, S., Ding, C., Yu, Q.: An LDA-SVM active learning framework for web service classification. In: IEEE International Conference on Web Services, pp. 49–56 (2016)
14. Shi, W.S., Liu, X.M., Yu, Q.: Correlation-aware multi-label active learning for web service tag recommendation. In: IEEE International Conference on Web Services, pp. 229–236 (2017)

15. Liang, T.T., Chen, L., Wu, J., Bouguettaya, A.: Exploiting heterogeneous information for tag recommendation in API management. In: IEEE International Conference on Web Services, pp. 436–443 (2016)
16. Gehring, J., Auli, M., Grangier, D., et al.: Convolutional sequence to sequence learning. In: International Conference on Machine Learning, pp. 1243–1252 (2017)
17. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)