Expert Systems With Applications 180 (2021) 115070

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Co-attentive representation learning for web services classification

Bin Tang[a,b], Meng Yan[a,b,c,*], Neng Zhang[d], Ling Xu[a,b], Xiaohong Zhang[a,b], Haijun Ren[b]

[a] Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, Chongqing University, Chongqing 400044, China
[b] School of Big Data and Software Engineering, Chongqing University, Chongqing, China
[c] PengCheng Laboratory, China
[d] School of Software Engineering, Sun Yat-sen University, China

ARTICLE INFO

Keywords:
Service computing
Web service classification
Co-attentive representation

ABSTRACT

The rapid adoption of services-related technologies, such as cloud computing, has lead to the explosive growth of web services. Automated service classification that groups web services by similar functionality is a widely used technique to facilitate the management and discovery of web services within a large-scale repository. The existing service classification approaches primarily focus on learning the isolated representations of service features but ignored their internal semantic correlations. To address the aforementioned issue, we propose a novel deep neural network with the Co-Attentive Representation Learning (CARL-Net) mechanism for effectively classifying services by learning interdependent characteristics of service without feature engineering. Specifically, we propose a service data augmentation mechanism by extracting informative words from the service description using information gain theory. Such a mechanism can learn a correlation matrix among embedded augmented data and description, thereby obtaining their interdependent semantic correlation representations for service classification. We evaluate the effectiveness of CARL-Net by comprehensive experiments based on a real-world dataset collected from ProgrammableWeb, which includes 10,943 web services. Compared with seven web service classification baselines based on CNN, LSTM, Recurrent-CNN, C-LSTM, BLSTM, ServeNet and ServeNet-BERT, the CARL-Net can achieve an improvement of 5.66%–172.21% in the F-measure of web service classification.

1. Introduction

With the wide adoption of Service-Oriented Architecture (SOA), web services (e.g., Mashups) are becoming popular on the web platforms and mobile application markets. For example, the largest web API services repository, ProgrammableWeb,[1] has collected over 23,000 web API services as of April 1, 2020. Many famous IT enterprises have developed their service platforms, such as Microsoft's Azure Marketplace,[2] Amazon's Service Marketplace,[3] and Baidu's Cloud Market.[4] The drastically increasing number of services has dramatically increased the burden of selecting appropriate services from a large candidate set and efficiently managing the service repository. Service classification has been demonstrated to be an effective solution to tackle this challenge, which plays a crucial role in many tasks, such as services discovery or selection (Elgazzar, Hassan, & Martin, 2010; Xia, Chen, Bao, Wang, & Yang, 2011), services Mashup or composition (Skoutas, Sacharidis, Simitsis, & Sellis, 2010) and services recommendation (Xia et al., 2014; Zhu, Kang, Zheng, & Lyu, 2012).

There are many related works on service classification that have been done so far. Most of them mainly focus on using keywords to match keywords in other service descriptions or measuring semantics distance among different services. The results of classification will classify the services that have similar functionality into the same category. These keywords-based approaches rely on the quality of keywords in the

* Corresponding author at: Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, Chongqing University, Chongqing 400044, China.

E-mail addresses: bintang@cqu.edu.cn (B. Tang), mengy@cqu.edu.cn (M. Yan), zhangn279@mail.sysu.edu.cn (N. Zhang), xuling@cqu.edu.cn (L. Xu), xhongz@cqu.edu.cn (X. Zhang), jhren@cqu.edu.cn (H. Ren).

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

[2] https://azuremarketplace.microsoft.com.

[3] https://aws.amazon.com/marketplace/.

[4] https://market.baidu.com/

https://doi.org/10.1016/j.eswa.2021.115070
Received 7 December 2020; Received in revised form 5 March 2021; Accepted 16 April 2021
Available online 26 April 2021
0957-4174/© 2021 Elsevier Ltd. All rights reserved.

service descriptions, which are manually appointed by service providers. However, service providers have different knowledge about similar functionality services. It is challenging for providers to choose the best keywords for services, which will cause semantic gap problems between different providers. Such a gap will limit the low accuracy of service classification.

To address the limitation of the keywords-based approach, many researchers have proposed various semantic-based service classification approaches. They typically learn the service description probabilistic topics based on vector space models for measuring the similarity among services and classify services. For example, Li, Zhang, Huai, Guo, and Sun (2013) utilized Latent Dirichlet Allocation (LDA) to extract the latent factors of service description documents for service classification. However, LDA-based classification methods are unsuitable for modeling short and sparse web service descriptions.

Deep learning is widely applied in the field of machine learning research, e.g., text classification, service classification. Convolutional neural networks (CNN) have also become a hot topic in natural language processing in recent years (Kim, 2014). Researchers proposed many deep learning models (Wang, Huang, & Deng, 2018, 2016, 2016, 2015, 2018, 2019), which have excellent performance to capture various information semantics to improve classification accuracy. One of the representative approaches is CNN proposed by Kim (2014) that integrated CNN with pre-trained word vectors to classify text. And the other is Att-BLSTM proposed by Zhou et al. (2016), which performed classification using an attention mechanism.

Nevertheless, the aforementioned deep learning models have some issues for service classification. (1) They usually only extracted features from the service description for classifying web services. (2) They fail to solve the data-sparse and context-independent issues, e.g., they adopted the context-independent embedding model (e.g., GloVe and Word2Vec), which will cause the dataset sparse when including some long text descriptions. To tackle these problems, Yang et al. (2020) proposed a deep neural network for services classification called ServeNet-BERT. ServeNet-BERT embedded service name and service description into vector spaces by **Google BERT** (Bidirectional Encoder Representation from Transformers) model (Devlin, Chang, Lee, & Toutanova, 2018), learned their high-level features by neural networks and merged them into unified features for web service classification. Their experimental results show that ServeNet-BERT outperforms 10 machine learning methods (Yang et al., 2020), including LDA-SVM (Liu, Agarwal, Ding, & Yu, 2016) and C-LSTM (Kim & Cho, 2018). Despite the fact that the ServeNet-BERT has an advantage over deep learning models on service classification, we observe that it still has a limitation: ServeNet-BERT simply joints service name features and service description features instead of fully taking advantage of the latent semantic correlations among service name and service description.

To address the limitation of ServeNet-BERT and capture more service features for service classification, we propose a novel deep neural network based on the Co-Attentive Representation Learning (CARL-Net) mechanism and extract informative words from service descriptions for service classification. The prior studies' aim is to propose an advanced model to classify services by extracting and jointing more service features. They joint service features but ignored the latent semantic correlations, such as service name features and service description features. Instead, this paper proposes a data augmentation mechanism and a co-attentive representation learning mechanism. The data augmentation mechanism adds service informative words as the service augmented data features, and the co-attentive representation learning mechanism learns interdependent service features semantic representations.We performed experiments on a real-world dataset collected from ProgrammableWeb to evaluate the effectiveness of our approach CARL-Net. Experimental results show that CARL-Net outperforms ServeNet-BERT by 5.66% in terms of F-measure of service classification.

1.1. *Contributions*

The main contributions of this paper are summarized as follows:

- We are the first to propose a co-attentive representation learning approach for service classification (named CARL-Net). It learns interdependent characteristics between service augmented data and descriptions by leveraging a co-attentive representation learning mechanism.
- We propose a service data augmentation mechanism that extracts service informative words from descriptions as additional service features. Such features are merged with the service names to form the service augmented data features that are conducive to service classification.
- We validate CARL-Net on a real-world dataset collected from ProgrammableWeb. The results show that CARL-Net outperforms the state-of-the-art baselines for service classification. We open source our replication package,[5] including the dataset and the source code for follow-up studies.

The remainder of this paper is structured as follows. Section 2 introduces the architecture of CARL-Net in detail. Section 3 presents our experimental setup, followed by experimental results in Section 4 and discussion in Section 5. Section 6 surveys the related works, and Section 7 concludes this study and discusses future work.

2. Proposed Approach

This section illustrates the overall architecture and implementation procedure of our proposed approach.

2.1. *Architecture*

Fig. 1 shows the overall structure of our approach CARL-Net. CARL-Net takes services name, services informative words and services description as inputs. Generally, the whole process of our approach could be divided into four parts.

(a) **Service Informative Words Extraction.** The first part is extracting informative words from service descriptions by performing information gain theory.

(b) **Embedding and Feature Extraction.** The second part is embedding vector-matrices based on the Google BERT model and extracting features by utilizing the neural network, including services descriptions, informative words, and corresponding service names.

(c) **Co-Attentive Representation Learning.** All the service features are modeled in the first two parts intend to be used for a co-attentive learning mechanism in the third part. The goal of co-attentive learning is to learn interdependent correlations among service features.

(d) **Service Classification.** The co-attentive representation vectors of service features could be integrated based on the fully connected layer to classify service in the final part.

2.2. *Service informative words extraction*

Generally speaking, the web service developers usually provide a description document about this web service functionality. Fig. 2 illustrates an example of a service from ProgrammableWeb, called Google Earth Engine API.[6] It includes its name, category and description. According to information gain theory, we can obtain service informative words extracted from service descriptions. Then take it as one part of augmented data of the approach CARL-Net. Information gain measures the decrease of information entropy when a characteristic word is

[5] https://github.com/tangbin9527/Service-Classification.

[6] https://www.programmableweb.com/api/google-earth-engine.

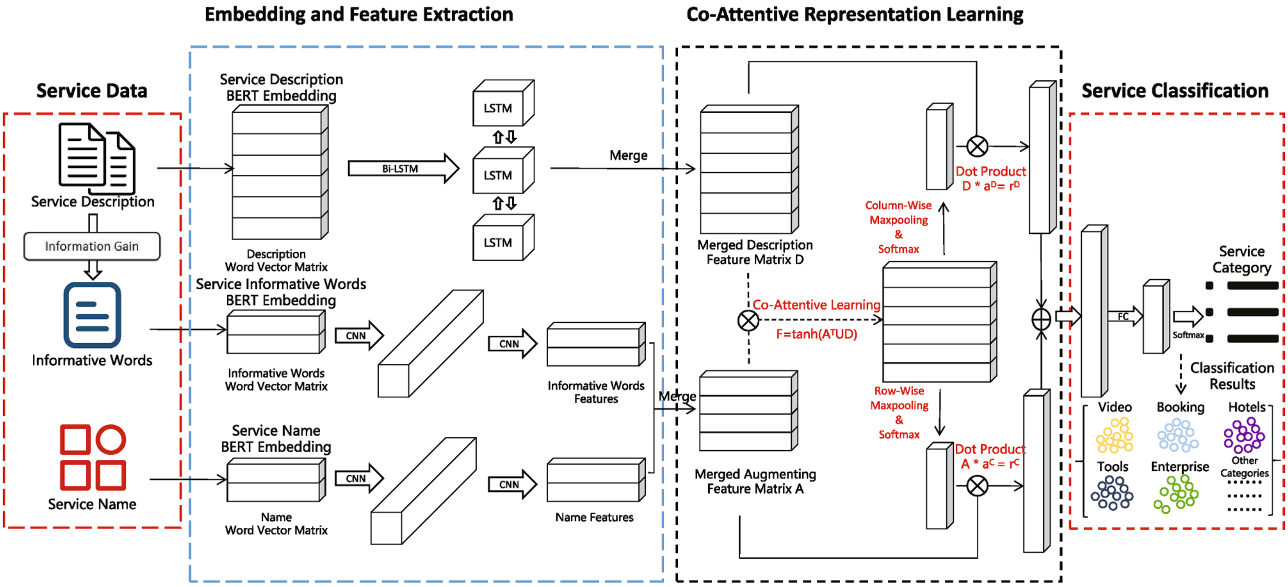
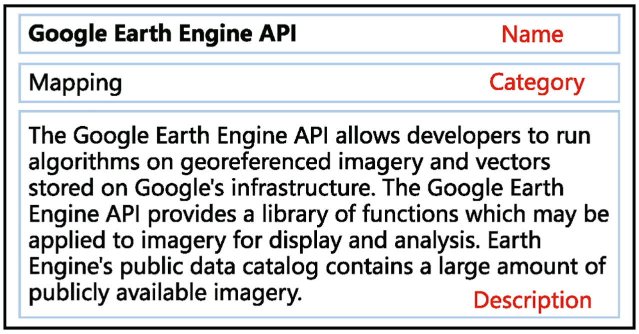


Fig. 1. The architecture of service classification with co-attentive representation learning. The notation ⊗ denotes the inner product and the notation ⊕ denotes the addition product.



Google Earth Engine API Name

Mapping Category

The Google Earth Engine API allows developers to run algorithms on georeferenced imagery and vectors stored on Google's infrastructure. The Google Earth Engine API provides a library of functions which may be applied to imagery for display and analysis. Earth Engine's public data catalog contains a large amount of publicly available imagery. Description

Fig. 2. An example of service.

present or absent in the text documents (Forman, 2003). The greater the information gain score of a word in the text document, the more potential characteristics carried by this word. We consider the service informative words that carry essential information about service that can be used for service classification.

As shown in the web service description in Fig. 2, some words, such as "a," "the," "or," which provide meaningless information so that couldn't be classifying web API "Google Earth Engine API" into category Mapping clearly. However, we notice other words, such as "earth," "engine," "display," that can predict the web API "Google Earth Engine API" be classified into Mapping naturally. Following Zhang, Liu, Cao, Xiao, and Wen (2018), after tokenizing services description, removing stop words according to English stop-words list and stemming, we can obtain information gain score for each word in the documents is calculated by formula (1):

$$IG(w) = -\sum_{k=1}^{N} P\left(C_k\right) logP\left(C_k\right) + P\left(w\right)\sum_{k=1}^{N} P\left(C_k|w\right) logP\left(C_k|w\right) + P\left(\overline{w}\right)\sum_{k=1}^{N} P\left(C_k|\overline{w}\right) logP\left(C_k|\overline{w}\right) \tag{1}$$

where N is the number of service categories, $P(C_k)$ is the probability of category C_k, $P(w)$ and $P(\overline{w})$ are the probabilities of presence and absence of word w respectively, $P(C_k|w)$ and $P(C_k|\overline{w})$ are respectively conditional probabilities of category C_k given presence and absence of word w. These terms can be calculated by the following formulas.

$$P\left(C_k\right) = \frac{num(C_k)}{\sum_{i=1}^{N} num\left(C_i\right)} \tag{2}$$

$$P\left(w\right) = \frac{\sum_{i=1}^{N} num\left(C_i^w\right)}{\sum_{i=1}^{N} num\left(C_i\right)} \tag{3}$$

$$P(\overline{w}) = 1 - P(w) \tag{4}$$

$$P\left(C_k|w\right) = \frac{num\left(C_k^w\right)}{\sum_{i=1}^{N} num\left(C_i^w\right)} \tag{5}$$

$$P\left(C_k|\overline{w}\right) = \frac{num\left(C_k\right) - num\left(C_k^w\right)}{\sum_{i=1}^{N}\left(num\left(C_i\right) - num\left(C_i^w\right)\right)} \tag{6}$$

where $num(C_k)$ is the number of services on the category C_k, $\sum_{i=1}^{N} num(C_i)$ is the total number of services on all categories, $num(C_k^w)$ is the number of services with the word w in their description on category C_k, $\sum_{i=1}^{N} num(C_i^w)$ is the total number of services with the word w in their description, $\sum_{i=1}^{N} num(C_i) - num(C_i^w)$ is the total number of services with the word w not in their description.

2.3. Embedding and feature extraction

In this section, we implement the word embedding based on the Google BERT model and feature extraction.

(1) Service description embedding and feature extraction: BERT model is a pre-trained natural language model that can embed a sentence into a k-dimensional vector or transform each word of a sentence into a k-dimensional vector according to the word's contexts. We can get two types of embedding output by performing the BERT model when

providing a sentence: (a) pool_output is a k-dimensional vector of a sentence and (b) sentence_output is an embedding vector matrix with the same order and length of a sentence. To integrate service features of the service name, service informative words, and service description into the deep neural network for further feature extraction, we embed the service name, service informative words, and service description into word vector matrices based on the BERT model.

The word sequence of service description is extracted by splitting it on the BERT tokenization. We can obtain a k-dimensional matrix of length n (n is the length of description after tokenizing) for the next step of feature extraction after the BERT embedding sequential word tokens of the service description. Let x_1 be the service description with arbitrary length, y_1 is the description embedding matrix, and the operation of obtaining sentence_output based on the BERT model is f_{bert_seq}. The representation of service description can be defined as:

$$y_1 = f_{bert\_seq}\left(x_1\right) \tag{7}$$

Considering the sequential features of the service description and the backpropagation does not work well in the deep neural network, we implement Bi-LSTM to extract service description features. The architecture of bidirectional LSTM in our CARL-Net for extracting service description features is shown in Fig. 3.

Let $a_i \in R^k$ is the k-dimensional word vector of the i-th word in the service description matrix y_1. The representation $h_i \in R^d$ is the hidden state of time step i, which can be obtained by the Bi-LSTM, and d is the number of each hidden state unit. Generally, in the bi-directional LSTM, the forward hidden state $\overrightarrow{h_i}$ is updated by preceding memory cell $\overrightarrow{c_{i-1}}$, previous hidden state $\overrightarrow{h_{i-1}}$ and input vector a_i. The back hidden state $\overleftarrow{h_i}$ is updated by next memory cell $\overleftarrow{c_{i+1}}$, later hidden state $\overleftarrow{h_{i+1}}$ and input vector a_i simultaneously. The hidden state can be formulated as follows:

$$\overrightarrow{h_i} = f\left(\overrightarrow{c_{i-1}}, \overrightarrow{h_{i-1}}, a_i\right) \tag{8}$$

$$\overleftarrow{h_i} = f\left(\overleftarrow{c_{i+1}}, \overleftarrow{h_{i+1}}, a_i\right) \tag{9}$$

where f is a non-linear activation function.

Therefore, the hidden state h_i of the i-th word in the service description is ultimately concatenated by forward LSTM and backward LSTM. The output feature D of the service description is concatenated by the all-time step hidden state.

$$h_i = \overrightarrow{h_i} \oplus \overleftarrow{h_i} \tag{10}$$

$$D = h_1 \oplus h_2 \oplus \ldots \oplus h_n \tag{11}$$

where n is the number of hidden state.

(2) Service informative words embedding and feature extraction: In this paper, we take informative words for the model inputs as augmented data for service classification. Considering the informative words are the keywords of service, we implement service informative words embedding in the same way as service description by using the BERT model. We assume that x_2 is the informative word of a service. Likewise, service informative words are eventually embedding into a matrix y_2 of k-dimensional informative words of length n (n is the length of informative words) based on the BERT model.

$$y_2 = f_{bert\_seq}\left(x_2\right) \tag{12}$$

The CARL-Net adopts the 2-D CNN for further feature extraction of service informative words. Let $e_i \in R^k$ is the word vector corresponding to the i-th word in the service informative words. We extend an extra dimension for the informative word matrix. A product operation extracts local features by applying a convolutional kernel $W_{I_1} \in R^{1\times p_1 \times q_1}$, and the number of convolutional kernel is **t**. Here, one represents the extra dimension, q_1 is the number of informative word vector dimensions, and p_1 is the number of informative words in a convolutional filter window. The local feature $c_{i,j}$ of informative words is generated by:

$$c_{i_1,j_1} = f\left(W_{I_1} {}^{*} e_{i_1:(i_1+p_1-1),j_1:(j_1+q_1-1)} + b\right) \tag{13}$$

where $b \in R$ is a bias term. We keep the number of informative words and the word vector dimension the same as the service informative word embedding matrix. Then, the same as preceding 2-D CNN, we use a filter $W_{I_2} \in R^{t\times p_2 \times q_2}$ and squeeze extra dimension, to get corresponding service informative words feature maps I.

$$c_{i_2,j_2} = f\left(W_{I_2} {}^{*} c_{i_2:(i_2+p_2-1),j_2:(j_2+q_2-1)} + b\right) \tag{14}$$

$$I = \left[c_{i,1}, c_{i,2}, \ldots, c_{i,k-q_2+1}\right] \quad \left(i = 1, 2, \ldots, n - p_2 + 1\right) \tag{15}$$

(3) Service name embedding and feature extraction: The service name is a brief but exhaustive summarization of the service's functionality, which means that the service name includes the abstract service semantic features for service classification. According to the above

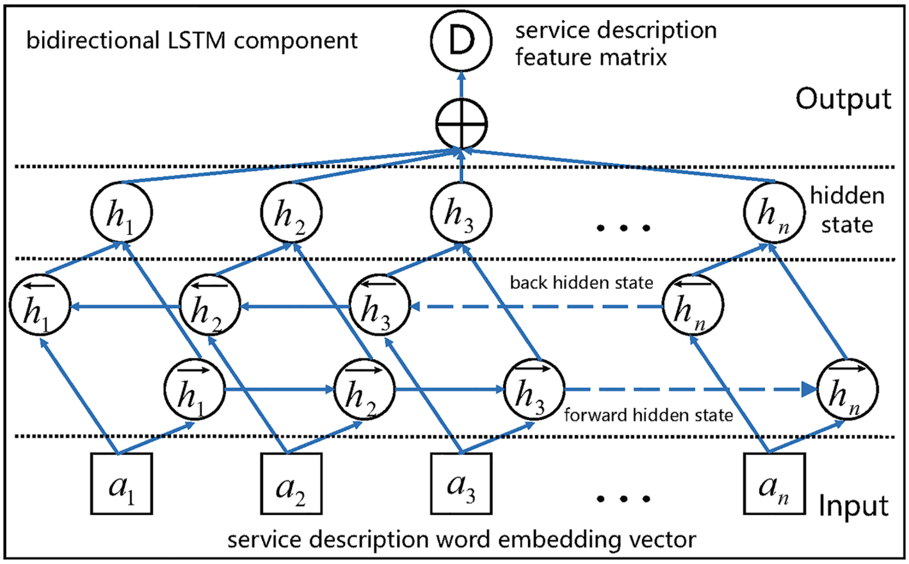


Fig. 3. The architecture of bidirectional LSTM for extracting service description features.

considerations, we obtain the sequential matrix embedding of the service name from BERT output. We assume that x_3 is the name of a service. The service name is finally embedded into a name matrix y_3. Let $s_i \in R^k$ is the k-dimensional word vector corresponding to the i-th word of the service name sequence. Same as the informative words extract feature, extracting service name local features by applying a convolutional kernel $W_{S_1} \in R^{1\times p_1\times q_1}$, the number of convolutional kernel is **t**, and $W_{S_2} \in R^{t\times p_2\times q_2}$, to get corresponding service name feature maps S.

$$y_3 = f_{bert\_seq}\left(x_3\right) \tag{16}$$

$$c_{i_1,j_1} = f\left(W_{S_1} * s_{i_1:(i_1+p_1-1),j_1:(j_1+q_1-1)} + b\right) \tag{17}$$

$$c_{i_2,j_2} = f\left(W_{S_2} * c_{i_2:(i_2+p_2-1),j_2:(j_2+q_2-1)} + b\right) \tag{18}$$

$$S = \left[c_{i,1}, c_{i,2}, \dots, c_{i,k-q_2+1}\right] \quad \left(i = 1, 2, \dots, n - p_2 + 1\right) \tag{19}$$

(4) Service Augmented Feature Fusion: Considering the service informative words and service name as the additional service information can be used to service classification. After embedding and extract service informative words and service name features to two matrices, we finally concatenate them into one matrix A as the augmented feature matrix of service:

$$A = I \oplus S \tag{20}$$

2.4. Co-Attentive Representation Learning

After embedding and feature extraction of service description and service augmented data, we can get two feature matrices $D \in R^{k\times p}$ and $A \in R^{k\times q}$ for service description and service augmented data. Here, p and q are the sizes of augmented data and description feature matrices, respectively. We compute the service feature correlation matrix $F \in R^{p\times q}$ by introducing a parameter matrix $U \in R^{d\times d}$, which can be learned by neural networks as follows:

$$F = tanh\left(A^T U D\right) \tag{21}$$

The service feature correlation matrix F has a co-attentive sight on service augmented data and service descriptions' feature semantic correlations. More specifically, the i-th row in F is the semantic correlations of each word in service description with the i-th service augmented data word. Meanwhile, the j-th column in F is the semantic correlation of each word in service augmented data with the j-th service description word.

Then, we implement a max-pooling operation to capture the most important semantic correlation vector representation between each word in augmented data and description. The service semantic vectors $g^A \in R^p$ and $g^D \in R^q$ represent service augmented data and description and the operation as follows:

$$\begin{aligned} g_i^A &= maxpooling\left[F_{i,1}, \dots, F_{i,q}\right] \\ g_i^D &= maxpooling\left[F_{1,i}, \dots, F_{p,i}\right] \end{aligned} \tag{22}$$

The service semantic vectors of service augmented data $g^A \in R^p$ and service description $g^D \in R^q$ are concatenated by:

$$\begin{aligned} g^A &= \left[g_1^A, \dots, g_p^A\right] \\ g^D &= \left[g_1^D, \dots, g_q^D\right] \end{aligned} \tag{23}$$

We obtain the service attention vectors of service augmented data $a^A \in R^p$ and service description $a^D \in R^q$ through conducting softmax function on the service semantic vectors $g^A \in R^p$ and $g^D \in R^q$:

$$\begin{aligned} a^A &= softmax(g^A) \\ a^D &= softmax(g^D) \end{aligned} \tag{24}$$

Finally, we generate service co-attentive representation vectors of service augmented data $r^A \in R^p$ and service description $r^D \in R^q$ by employing dot operation between the service attention vectors a^A, a^D and service feature matrices A, D, respectively.

$$\begin{aligned} r^A &= Aa^A \\ r^D &= Da^D \end{aligned} \tag{25}$$

2.5. Service classification

This part is the final classification task layer. We implement a fully-connected layer neural network f_{fc}, which adopts a softmax activation function to compute the probability of each category. The model inputs of this part are service co-attentive representation vectors $r^A \in R^p$ and $r^D \in R^q$ and outputs a prediction vector l of service classification. The l_i with the maximum value in the prediction vector is the most likely service category.

$$l = softmax\left(f_{fc}\left(r^A + r^D\right)\right) \tag{26}$$

After the above classification process, all web services are classified into different functional categories according to their interdependent semantic information.

3. Experimental setup

3.1. Dataset

The dataset used in our experiment is collected from the ProgrammableWeb site and contains 15,344 services with 401 categories. For each web service, we can obtain the meta-data of service's *API name, description, primary category*. Note that to avoid ambiguity, we define *API name* as service name, *description* as service description, *primary category* as the judgment standard of service classification. Fig. 2 illustrates an example service. We eliminate some categories which contain a small number of services. Counting and sort the number of services in each category. We choose the top 50 categories with the most services as the experimental dataset. The experimental dataset contains 10,943 web services. We adopt random selection by category to split each category's services into 80% of the training set and 20% of the testing set. Finally, we obtain 8,733 web services of the training set and 2,210 web services of the testing set. We train and test the CARL-Net and the baselines on the after preprocessed dataset.

3.2. Parameter settings

We choose the BERT model to embed each word of service descriptions, service informative words and service names into a 768-dimensional vector. The first convolutional layer has 32 filters with kernel size 3 × 3, and the second convolutional layer has one filter with kernel size 1 × 1. The hidden layer size of LSTM is 1024. We take the categorical cross-entropy as the loss function and optimize CARL-Net using the SGD algorithm with a learning rate of 0.01. The total epoch number is 50, with a batch size is 32. We adopt L2 regularization of 0.01 to regularize the weights of parameters. All the experiments are conducted on a server with one Nvidia Titan V GPU with 256 GB memory.

3.3. Evaluation metrics

To evaluate the classification performance of the proposed approach CARL-Net, we utilize several standard evaluation metrics. The details as following:

Top-k Accuracy, the prediction is correct when the target category label could be found in the top-k ranked list, which are k labels with the highest predict probability. Top-k Accuracy on the category c is:

$$A_{topk}\left(c\right) = \frac{Num_{topk\_correct}}{Num_c} \tag{27}$$

where $Num_{topk\_correct}$ is the number of correct predictions, Num_c is the number of the services in category c. A_{topk} accuracy can be calculated by:

$$A_{topk} = \frac{\sum_{c=1}^{N} A_{topk}\left(c\right)}{N} \tag{28}$$

where N is the number of the categories in the whole dataset. Following Yang et al. (2020), we evaluate A_{top1},A_{top5}.

In addition, we adopt the internationally accepted evaluation indicators: precision rate P, recall rate R, F-measure value.

$$P = \frac{A}{A+B} \tag{29}$$

$$R = \frac{A}{A+C} \tag{30}$$

$$F - measure = \frac{2PR}{P+R} \tag{31}$$

where **A** indicates that a certain type of service is correctly classified as the number of samples of the category, **B** represents that the service of other categories is recognized as the number of samples of the category, **C** indicates that a certain type of service is recognized as the number of samples of other categories.

3.4. Baseline models

To demonstrate the effectiveness of the proposed approach CARL-net, this research compares the following the most competitive models:

ServeNet (Yang, Ke, Wang, & Zhao, 2019**):** This approach stacks 2-D CNN and BLSTM deep neural networks to extract features from service descriptions for service classification. It embeds word vectors by Global Vectors for Word Representation and abstracts automatically low-level representations and high-level representations without feature engineering.

ServeNet-BERT (Yang et al., 2020**):** This approach is one of the state-of-the-art service classification models. The model combines the original ServeNet and introduces the service name as the input of the model. It embeds each word of the service name and service description vector by the BERT model, follows by the original ServeNet to extract service description features and takes the pool_output of the service name as the service name features. Then, ServeNet-BERT merges service name feature and description feature into unified features for service classification. We take it as our baseline model and re-ran the ServeNet-BERT by using the source code[7] shared on the GitHub.

Beyond the above-mentioned models, other models like CNN (Kim, 2014), LSTM (Johnson & Zhang, 2016), Recurrent-CNN (Liu, Qiu, & Huang, 2016), C-LSTM (Shi, Wang, & Li, 2019) and BLSTM (Zhang, Zheng, Hu, & Yang, 2015) have outstanding performance in text classification. Considering them were usually adopted as baselines when there is no suitable deep learning model for addressing service classification questions (Yang et al., 2019; Yang et al., 2020). In this paper, we also take them into approaches of comparison.

3.5. Research questions

RQ1: how effective is the proposed CARL-net in services classification?

We compared the proposed approach CARL-Net with seven models to prove whether the co-attentive representation learning is beneficial for service classification, including the recent service classification model ServeNet-BERT and six additional deep learning models with excellent performance in the text classification.

RQ2: How does the service name and service informative word affect the classification effectiveness?

This RQ investigated the impacts by augmented data merged by service name and informative words on model effectiveness. We run the proposed approach CARL-Net with individual features of the augmented data to analyze their effects on the classification tasks and compare whether using two parts together contributes to improve model performance.

RQ3: How does the BERT model affect the accuracy of classification?

BERT is the advanced model in the field of Natural Language Processing, which has excellent ability for word embedding. To analyze the impact of the BERT, we run our proposed approach CARL-Net under the traditional GloVe (Global vectors for word representation) model (Pennington, Socher, & Manning, 2014) embedded word vectors and compare the classification results to the previous results to research the effect of BERT.

RQ4: How does CARL-Net perform under different parameter settings and initial conditions?Our proposed approach CARL-Net contains two critical parameters: the number of informative words and learning rate; and two important initial conditions: Dropout and L2 regularization. This RQ investigates the impacts of different parameter settings and initial conditions on the service classification performance of CARL-Net.

4. Experimental results

This section presents and analyzes a series of experimental results to answer the four research questions, as referred to in Section 3.5.

4.1. RQ1: Model effectiveness

Table 1 compares the effectiveness of services classification between several deep learning methods. The higher A_{top1},A_{top5}, Precision, Recall, and F-measure value demonstrate the classification results better. As can be seen from the experimental results table, the ServeNet-BERT achieves the average A_{top1},A_{top5}, Precision, Recall and F-measure value are 0.681, 0.905, 0.668, 0.653, and 0.654, respectively. The average A_{top1},A_{top5}, Precision, Recall and F-measure value of our classification model CARL-Net are 0.715, 0.890, 0.703, 0.689, and 0.691, separately. In comparison, CARL-Net has an improvement of 4.99%, 5.24%, 5.51%, 5.66% over ServeNet-BERT in terms of A_{top1}, Precision, Recall, and F-measure. Our approach CARL-Net and ServeNet-BERT are comparable in terms of Top-5 Accuracy.

For other deep learning methods, LSTM can reach an F-measure value of 0.403 through long-term dependencies from past time steps to learn global text features, where CNN only learning local features and gets an F-measure value of 0.252. However, Recurrent-CNN (RNN + CNN) and C-LSTM (CNN + LSTM) get a higher F-measure value of 0.543 and 0.578 by stacking CNN and sequence model for learning text features. When learning bidirectional dependencies, BLSTM reaches an F-measure value of 0.587. Finally, ServeNet uses 2-D CNNs with BLSTM to

Table 1
Comparison of Classification Results.

Model	A_{top1}	A_{top5}	P	R	F-measure
CNN	0.295	0.569	0.296	0.240	0.252
LSTM	0.510	0.789	0.477	0.410	0.403
RCNN	0.597	0.857	0.608	0.529	0.543
CLSTM	0.612	0.851	0.595	0.585	0.578
BLSTM	0.619	0.853	0.636	0.576	0.587
ServeNet	0.631	0.874	0.631	0.602	0.608
ServeNet-BERT	0.681	**0.905**	0.668	0.653	0.654
CARL-Net	**0.715**	0.890	**0.703**	**0.689**	**0.691**

[7] https://github.com/yylonly/ServeNet.

learn both local and global features; it reaches an F-measure value of 0.608. The baseline ServeNet-BERT uses both information from service name and description with context-dependent word embedding, which reaches a higher F-measure value of 0.654 among all benchmarks. Moreover, CARL-Net integrates the above thought and co-attentive representation learning together. It reaches the highest F-measure value of 0.691 among all benchmarks. These results verify the effectiveness of CARL-Net in incorporating co-attentive representation learning into service classification.

Table 1 only denotes the average results on 50 categories of each model, and the more contrastive details about the F-measure value of all categories could be found in Table 4. We found CNN and LSTM have ineffective effects on certain categories, such as “Internet of Things,” “Other,” and “Storage” due to lacking data and are difficult to extract adequate features. However, our proposed approach CARL-Net has higher values because of the more robust ability to extract features.

The classification results of CARL-Net and ServeNet-BERT are shown in Fig. 4. We used the t-distributed stochastic neighbor embedding (t-SNE) algorithm (Van & Hinton, 2008) that mapped all classified high-dimensional data into a 2D space. The same color indicates the same category of services. The visualization results with clearer color boundaries indicate a better classification performance. We observe that some services with different categories in Fig. 4(b) are more scattered than those in Fig. 4(a). From this visualization, it is proved that CARL-Net achieves better classification results than ServeNet-BERT.

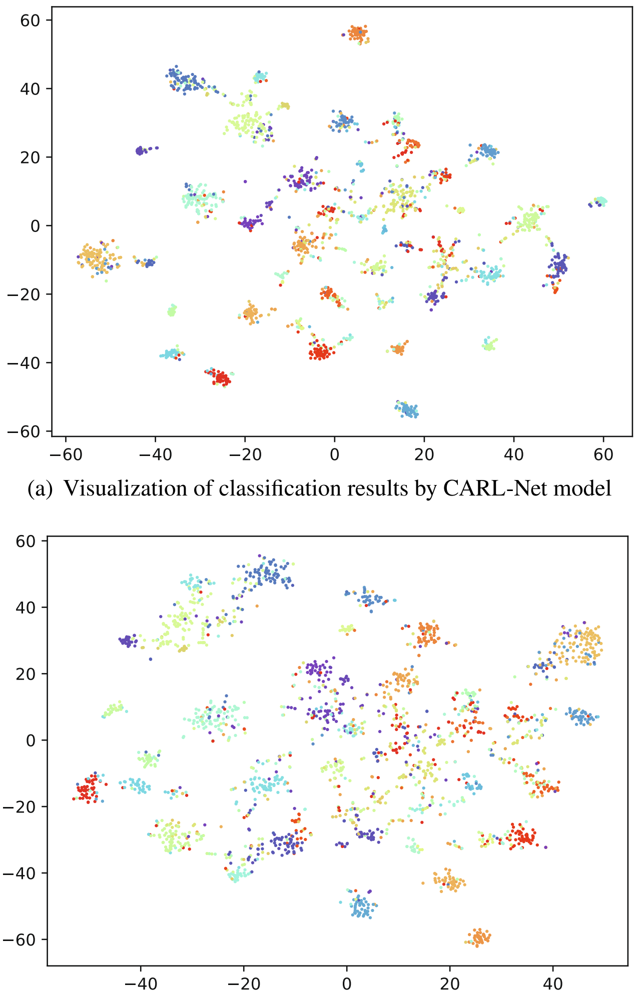
Compared with seven web service classification baselines, the proposed approach CARL-Net can achieve an improvement of 5.66%-172.21% in terms of F-measure for web service classification.

4.2. RQ2: Impact of augmented data features

CARL-Net uses augmented data as their inputs, including service name (N) and service informative words (I) extracted by service description documents. To research the relative importance of these two features, we run CARL-Net with individual features at a time. As shown in Table 2, we can observe that by only use one feature of augmented

Table 4
Comparison Results of F-measure Value on Each Category

Service Category	CNN	LSTM	RCNN	CLSTM	BLSTM	ServeNet	ServeNet-BERT	CARL-Net
eCommerce	0.248	0.689	0.679	0.704	0.732	0.715	0.734	**0.783**
Photos	0.234	0.506	0.554	0.694	0.615	0.677	0.615	**0.720**
Stocks	0.606	0.591	0.571	0.651	0.809	0.650	0.926	**0.931**
Chat	0.348	0.381	0.522	0.788	0.667	**0.896**	0.667	0.750
Telephony	0.143	0.451	0.651	0.620	0.626	0.613	0.703	**0.732**
Medical	0.182	0.556	**0.800**	0.633	0.727	0.700	0.615	0.579
Backend	0.095	0.069	0.114	0.364	0.286	0.383	0.533	**0.588**
Travel	0.371	0.719	0.758	0.716	0.771	0.766	0.857	**0.871**
Domains	0.452	0.000	0.827	0.812	0.827	0.788	0.909	**0.941**
Data	0.118	0.000	0.222	0.259	0.242	0.340	0.367	**0.468**
Internet of Things	0.000	0.000	0.322	0.375	0.278	0.526	0.667	**0.750**
Transportation	0.442	0.702	0.762	0.753	0.791	0.707	**0.800**	0.782
Government	0.414	0.631	0.687	0.710	**0.796**	0.773	0.785	0.769
Marketing	0.077	0.364	0.240	0.357	0.364	0.428	0.240	**0.600**
File Sharing	0.428	0.250	0.560	0.500	**0.606**	0.555	0.588	0.500
Enterprise	0.273	0.376	0.374	0.448	0.429	0.520	0.600	**0.642**
Cloud	0.276	0.494	0.545	0.591	0.613	0.627	0.606	**0.630**
Games	0.269	0.605	0.776	0.790	0.740	0.686	0.829	**0.867**
Financial	0.412	0.581	0.664	0.677	0.709	0.699	0.767	**0.796**
Weather	0.540	0.857	0.784	0.800	0.808	0.823	**0.877**	0.806
Payments	0.416	0.687	0.700	0.719	0.739	0.687	0.744	**0.759**
Science	0.484	0.743	0.708	0.718	0.781	0.785	**0.791**	0.772
Email	0.083	0.661	0.741	0.808	0.716	0.758	**0.916**	0.893
Project Management	0.170	0.059	0.522	0.586	0.553	0.667	**0.678**	0.630
Other	0.000	0.000	0.049	0.119	0.078	0.078	0.097	**0.226**
Tools	0.268	0.379	0.481	0.446	0.461	0.520	0.572	**0.586**
Database	0.093	0.000	0.244	0.311	0.325	0.310	0.339	**0.464**
Storage	0.000	0.100	0.400	0.631	0.516	**0.649**	0.520	0.628
Banking	0.307	0.000	0.647	0.489	0.628	0.650	0.722	**0.789**
Application Development	0.098	0.000	0.146	0.178	0.263	0.228	0.454	**0.536**
Real Estate	0.238	0.684	0.769	0.789	0.762	0.700	**0.826**	0.775
Bitcoin	0.231	0.476	0.571	0.449	0.577	0.654	0.710	**0.847**
Messaging	0.500	0.769	0.760	0.784	0.811	0.751	0.811	**0.824**
Media	0.069	0.076	0.100	0.240	0.296	0.385	0.536	**0.606**
Security	0.094	0.165	0.547	0.627	0.512	0.559	0.644	**0.691**
Analytics	0.102	0.000	0.322	0.227	0.454	0.444	0.522	**0.591**
Entertainment	0.095	0.100	0.154	0.228	**0.476**	0.312	0.470	0.474
Images	0.348	0.000	0.300	0.258	0.228	0.454	**0.437**	0.385
Video	0.410	0.673	0.796	0.816	0.763	0.766	0.826	**0.829**
Sports	0.642	0.750	0.835	0.864	0.873	0.905	0.903	**0.905**
Education	0.277	0.562	0.637	**0.774**	0.657	0.683	0.711	0.711
News Services	0.000	0.400	0.560	0.545	0.522	**0.571**	0.461	0.500
Search	0.187	0.178	0.387	0.467	0.407	0.400	0.538	**0.574**
Shipping	0.341	0.873	0.889	0.898	0.902	**0.943**	0.784	0.784
Music	0.410	0.706	0.709	0.779	0.762	0.753	**0.886**	0.857
Events	0.000	0.333	0.683	0.686	0.667	0.706	0.789	**0.829**
Reference	0.057	0.169	0.212	0.240	0.246	0.230	0.379	**0.387**
Social	0.243	0.479	0.546	**0.639**	0.538	0.571	0.507	0.635
Mapping	0.365	0.680	0.726	0.677	0.708	0.743	0.736	**0.813**
Advertising	0.160	0.602	0.609	0.701	0.681	0.667	0.698	**0.729**
Average	0.252	0.403	0.543	0.578	0.587	0.608	0.654	**0.691**



(a) Visualization of classification results by CARL-Net model

(b) Visualization of classification results by ServeNet-BERT model

Fig. 4. Visualization of classification results. Each point represents a service. The same color of a point represents the same category of the service.

Table 2
Effectiveness Comparison with Different Feature Settings.

Model	A_{top1}	A_{top5}	P	R	F-measure
CARL-Net(N)	0.684	**0.907**	0.672	0.656	0.657
CARL-Net(I)	0.691	0.903	0.690	0.672	0.671
CARL-Net(N + I)	**0.715**	0.890	**0.703**	**0.689**	**0.691**

data as model input, the classification performance has decreased. Specifically, the results drop over 4.53% and 5.17% in terms of A_{top1} and F-measure value when the model learns latent information from service name and service description. Meanwhile, the A_{top1} and F-measure of only co-attentive learning service informative words with service description decrease over 3.47% and 2.98% comparing with the performance of combining two augmented data features. We can observe that all features of augmented data are conducive to increase the model effectiveness and the informative words affect the performance most.

Both service name and service informative words make a contribution to the effectiveness of our approach CARL-Net. In terms of their importance comparison, informative words is more important than service name.

4.3. RQ3: Impact of BERT model

GloVe (Pennington et al., 2014) model can embedding each word of service description into a vector and take zero to pad all service descriptions in the dataset to a fixed length. In order to research the importance of the BERT model, we compared the impact of the BERT model embedded word vector and GloVe model embedded word vector on the classification performance. We perform a pre-trained GloVe model that embeds words into 50-dimensional, 100-dimensional, 200-dimensional, 300-dimensional vectors. Table 3 shows the experimental results, we can observe that the BERT model's performance significantly outperforms four dimensional word vectors embedded by the GloVe model in terms of Top-1/5 Accuracy, P, R, and F-measure. These results imply that the BERT model is a better choice for service classification. BERT model is better because it is context-aware to extract representation of word-level and sentence-level, then output high-level features.

For service classification, BERT model is a better choice for word embedding than GloVe.

4.4. RQ4: Impact of parameter settings

4.4.1. Impact of the number of informative words

In this part, we particularly evaluate the impact of the number of informative words on services classification. Informative words carry the pivotal information of service description, which is conducive to classify service. The number of informative words determines how much information in the description is used for model optimization. We increase the number of informative words from 2 to 10, and the step interval is 2 in this set of experiments. Then we calculate the values of our performance evaluation metrics. The experimental results of our proposed approach CARL-Net can be seen in Fig. 5. The comprehensive indicator F-measure is becoming better, when the number of informative words varies from 2 to 6. As those words carry more valuable information that can be extracted by the co-attentive layer. Meanwhile, the effectiveness is becoming worse when the number of informative words increases from 6 to 10. This is because more additional words that carry more noise resulted in indistinct classification. So, we consider the number 6 is the best choice for informative words of the CARL-Net.

4.4.2. Impact of learning rate

We use the SGD algorithm (Bottou, 2010) to optimize the model because the parameters of BERT have exceeded the limits of processors. SGD optimizer has the benefits of using lower computational complexity to get relatively good training results in large-scale learning problems (Bottou, Curtis, & Nocedal, 2018). In the process of model learning, the learning rate affects the speed of loss function minimization. Generally speaking, the speed of loss function minimization is positively correlated with the learning rate. However, the high learning rate is usually difficult to reach the optimal solution when a low learning rate often leads to a local optimal solution. To study the impact of the learning rate in our approach CARL-Net, we set up the maximum number of iterations for 50, and the number of informative words is 6. Fig. 6 shows the evaluation metrics changed when the learning rates were 0.05, 0.01, 0.005 and 0.001. From these pictures, we can observe that the 0.01 learning rate's performance is better than other learning rates by comparing Top-

Table 3
Experimental Results of Different Word Embedding Model

Model	A_{top1}	A_{top5}	P	R	F-measure
GloVe(dim = 50)	0.494	0.795	0.483	0.442	0.446
GloVe(dim = 100)	0.537	0.820	0.524	0.494	0.492
GloVe(dim = 200)	0.546	0.843	0.563	0.500	0.514
GloVe(dim = 300)	0.589	0.850	0.625	0.537	0.550
BERT (dim = 768)	**0.715**	**0.890**	**0.703**	**0.689**	**0.691**

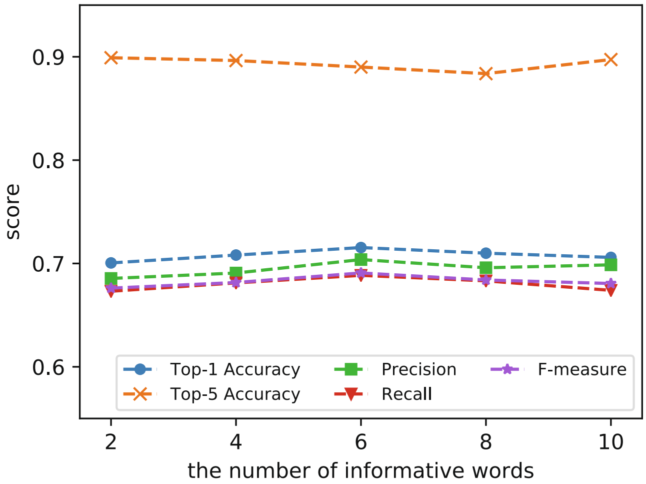


Fig. 5. The effectiveness of service classification with various number of informative words.

1 Accuracy and F-measure. Compared with the learning rate of 0.005 and 0.001, the learning rate of 0.01 can reach the optimal solution faster. When the learning rate is 0.001, the model's performance is considered to reach the local optimal solution because of the low learning rate. When the learning rate is 0.05, the result of the CARL-Net has sharp vibration and will spend more time to reach the optimal solution. So, we consider the best learning rate is 0.01 for the CARL-Net.

4.4.3. *Impact of initial conditions*

In this part, we run CARL-Net without Dropout (D) and L2 regularization (L) separately to survey the impact of these initial conditions on service classification. As shown in Table 5, the F-measure values of CARL-Net (D) and CARL-Net (L) are 68.6% and 67.5 %, respectively. In terms of most of the metrics (except A_{top5}), CARL-Net achieves better performance than the two variants. These results indicate that Dropout and L2 regularization algorithms are essential for CARL-Net.

For our approach CARL-Net, the best choice for the number of informative words is 6, the learning rate is 0.01, and the conditions with Dropout and L2 regularization, which are beneficial to service classification effectiveness.

5. Discussion

5.1. *Why does CARL-Net Work?*

This section discusses the advantage of our proposed approach CARL-Net with examples. Fig. 7 shows the services to will be categorized by CARL-Net vs. ServeNet-BERT. We found that ServeNet-BERT classifies Fig. 7(a) service into the "Analytics" category and Fig. 7(b) service into the "Tools" category after analysis classification results. We notice that the "Analytics" category includes services of the systematic computational analysis of data or statistics, the "Marketing" category contains services of the action or business of promoting and selling products, the "Tools" category involves services of carrying out a particular function, and services of storing data comprise the "Database" category. ServeNet-BERT joints features between service name and description is inadequate for classifying correctly. In contrast, CARL-Net can classes the services into the expected categories although Fig. 7(a) service description contains the keyword "analytics," that is because it exploits some classified function related informative words, such as "marketing," "customer" and etc. in Fig. 7(a). These results indicate that the co-attentive mechanism learning by taking informative words into service features surpasses the simply joint features.

Table 5
Effectiveness Comparison with Different Initial Conditions.

Model	A_{top1}	A_{top5}	P	R	F-measure
CARL-Net(D)	0.707	0.888	0.704	0.680	0.686
CARL-Net(L)	0.703	0.911	0.693	0.670	0.675
CARL-Net	0.715	0.890	0.703	0.689	0.691

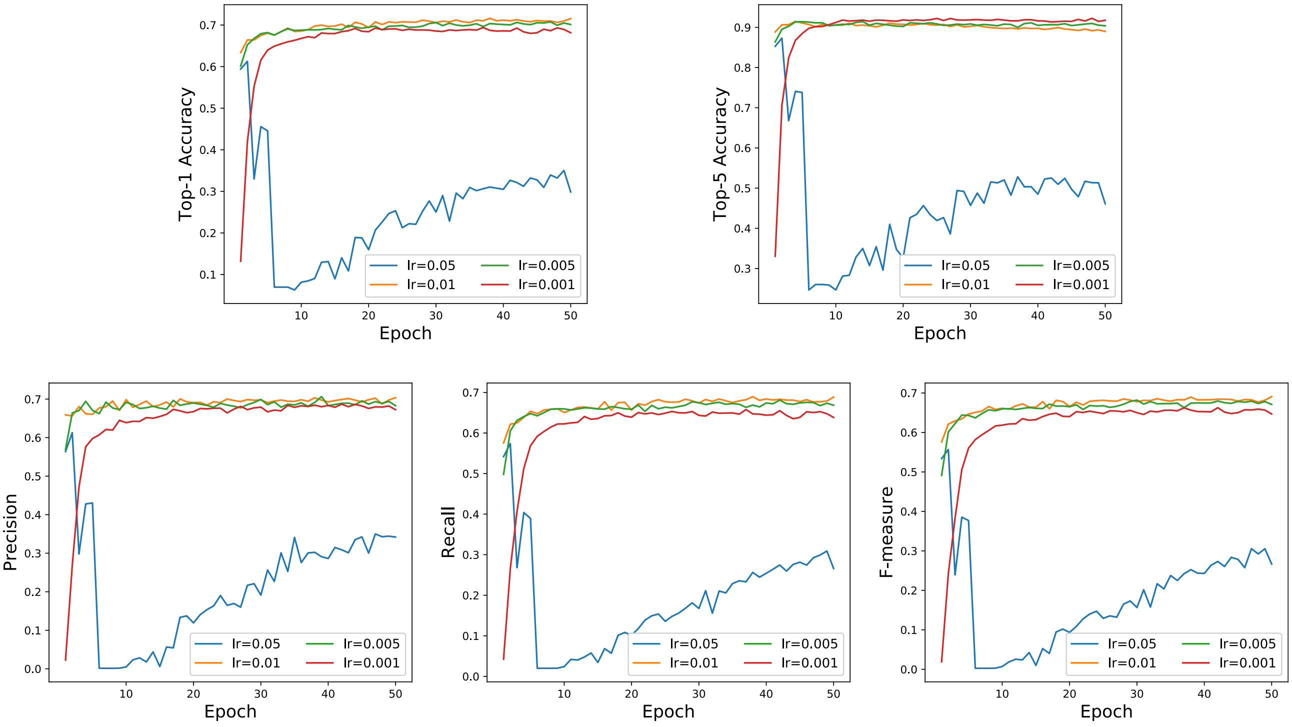
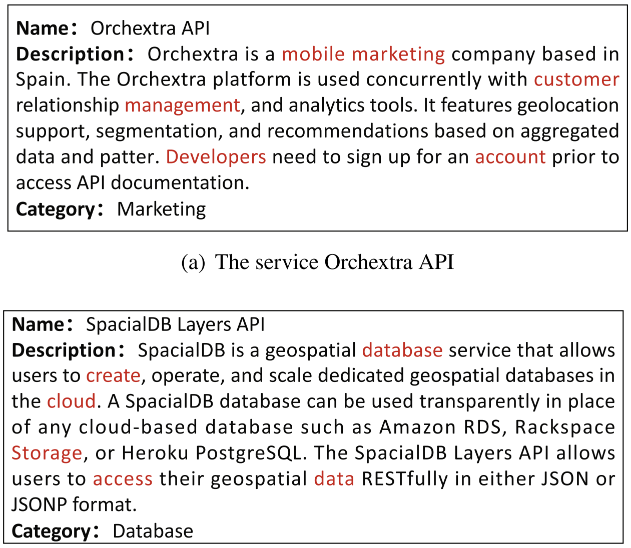


Fig. 6. The impact of learning rates.



Name： Orchextra API
Description： Orchextra is a mobile marketing company based in Spain. The Orchextra platform is used concurrently with customer relationship management, and analytics tools. It features geolocation support, segmentation, and recommendations based on aggregated data and patter. Developers need to sign up for an account prior to access API documentation.
Category： Marketing

(a) The service Orchextra API

Name： SpacialDB Layers API
Description： SpacialDB is a geospatial database service that allows users to create, operate, and scale dedicated geospatial databases in the cloud. A SpacialDB database can be used transparently in place of any cloud-based database such as Amazon RDS, Rackspace Storage, or Heroku PostgreSQL. The SpacialDB Layers API allows users to access their geospatial data RESTfully in either JSON or JSONP format.
Category： Database

(b) The service SpacialDB Layers API

Fig. 7. The examples of web service which are classified by ServeNet-BERT and CARL-Net.

5.2. Limitations

The experimental results demonstrate the effectiveness of our CARL-Net in service classification. However, there are some limitations of CARL-Net needed to be pointed out.

Limitations on Parameter Settings: The main difficulty of our approach CARL-Net is selecting optimal setting of the parameters when applying the approach. To ensure the experimental fairness, our approach CARL-Net's parameters setting follows the baseline ServeNet-BERT, and the experimental result present CARL-Net better than the ServeNet-BERT. However, the parameters setting may not be the optimal parameters for our approach CARL-Net. We plan to implement metaheuristic algorithms (Stojanovic, Nedic, Prsic, Dubonjic, & Djordjevic, 2016; Amato & Venticinque, 2016) to find optimal solutions of the CARL-Net in the future.

Limitations on Practical Applications: CARL-Net may not achieve good classification performance for some special cases of service categories and services, such as (1) the categories that contain only a small number of services; and (2) the services with too short descriptions to describe the service functionality. Meanwhile, there are some external disturbances and noises that will affect the classification performance of CARL-Net. For example, service providers may use special symbols or mathematical formulas to explain the functionality of services. However, CARL-Net may not understand their meanings. In our experiments, we remove the special symbols in the data preprocessing stage according to the basic Ascii code table. The influence of external disturbances and noises need to be thought of when applying CARL-Net in practical applications (Tao, Wang, Chen, Stojanovic, & Yang, 2020; Stojanovic & Prsic, 2020; Tao, Li, Paszke, Stojanovic, & Yang, 2021).

5.3. Time overhead

To evaluate the computational burden of CARL-Net, we compare the training and prediction time of CARL-Net and the advanced baseline ServeNet-BERT on the dataset. On average, ServeNet-BERT takes about 102.03 s for every epoch's optimization and 8.97 s to predict service categories in one epoch. CARL-Net takes 112.45 s for each epoch's training and 10.56 s to test each epoch. These results show that CARL-Net and ServeNet-BERT are comparable in terms of time cost.

6. Related work

As we know, web service classification as an effective approach to advance the performance of web service discovery has been proved Elgazzar et al. (2010). And a significant number of web service classification achievements have been published in recent years. Generally speaking, existing literature can be divided into two categories: conventional machine learning methods and deep learning methods.

Machine learning-based methods: There are some works classify service based on keywords in the Web Services Description Language (WSDL) document. Keyword-based web service discovery methods match keywords in query with service descriptions. Liu and Wong (2009) clustered services based on the four functional elements(contains service description text, service context, hostname, and service name), which collect from WSDL documents utilizing text mining techniques. Hao, Zhang, and Cao (2010) used the Term Frequency-Inverse Document Frequency (TF-IDF) to measure the similarity between services, discovery and rank web services automatically. Elshater, Elgazzar, and Martin (2015) built a KDtree index structure and generated TF-IDF model of the service corpus to improve Web discovery. It transforms user queries into corresponding IF-IDF vector using the IF-IDF model and retrieve relevant services by navigating KDtree.

These kinds of methods consider the semantic relevance of services by extracting some significant information from web services description. Such as Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet Allocation (LDA) topic model are discover implicit semantic corrections and identify latent functional factors among document clustering and text categorization (Lu, Mei, & Zhai, 2011). Liu and Fulia (2015) formed user-related, service-related, and topic-related latent factor models by considering historical usage data and service descriptions, which combined probabilistic topic model and matrix factorization (Zheng, Ma, Lyu, & King, 2013) to service discovery and recommendation. Aznag, Quafafou, and Jarir (2014) extracted topics from semantic service descriptions and grouped hierarchical clusters to search services based on Correlated Topic Model (CTM). Wu, Chen, Zheng, Lyu, and Wu (2014) and Chen, Wang, Yu, Zheng, and Wu (2013) clustered web services by integrating both WSDL documents and tags data, and Shi, Liu, Zhou, Tang, and Cao (2017) improved the performance of web services clustering by leveraging the high-quality word vectors through enhanced LDA. Liu et al. (2016) addressed the issues caused by service descriptions generate sparse term vectors and problems of vectors high dimension by incorporating LDA-based probabilistic topic models with Support Vector Machine (SVM) classifier to improve the effectiveness in service classification. He, Zhao, Huang, Fox, and Wang (2020) supported multi-label classification through utilizing the latent variable model to discover thing's latent relation strength and learning binary Support Vector Machine classifier for each label.

However, most existing topic model-based approaches mine the latent factor from WSDL documents (Liu & Fulia, 2015; Aznag et al., 2014; Liu et al., 2016), which is hard to acquire well-performance clustering accuracy because of the semantic sparsity limitation of short text in web service descriptions (Cao et al., 2016). Many researchers integrated external information widely to increase the service discovery performance to tackle the semantic sparsity problems. Kenter and De Rijke (2015) incorporated an arbitrary number of word embedding sets to predict the semantic similarity of short texts. In a similar way, the work (Jin, Liu, Zhao, Yu, & Yang, 2011) transferred learning from related long texts to clustering short text of service descriptions. Shi et al. (2017) and Shi, Liu, Cao, Wen, and Zhang (2018) incorporated prior knowledge to improve the clustering process under word scarcity problems.

Besides compute similarity between text and mine the latent topic relation, these are works based on a social network of services and users to recommend and discover services. In Cao, Liu, Tang, Zheng, and Wang (2013), Cao et al. recommended mashup services by considering users' interest from their Mashup service usage history and social network based on social relationships information among Mashup services, web APIs and tags. Xu, Cao, Hu, Wang, and Li (2013) presented a social-aware service recommend approach, which utilized multi-dimensional social relationships that are described by the matrix model among potential users, topics, mashups, and services. Liang, Chen, Wu, Xu, and Wu (2016) proposed a framework incorporating social media information that includes semantic similarity, popularity, activity and decay factor for effectively discovering the suitable services.

The most current researches based on machine learning methods of service discovery and classification mainly depend on the quality of feature engineering. Researchers take advantage of personal intelligence and prior knowledge to generate feature engineering, which the process is usually difficult and expensive. However, feature engineering can handle the problems and difficulties of extracting critical information from original data, and excellent feature engineering improves the performance of service discovery and classification (Bengio, Courville, & Vincent, 2013).

Deep learning-based methods: Deep learning (LeCun, Bengio, & Hinton, 2015) is a promising alternative to machine learning, and it can automatically abstract from the word-level characteristics to document-level representations from the original data without feature engineering. Due to its robust learning of deep features, it has been widely applied in the domain of natural language processing (Zhang, Zhao, & LeCun, 2015). Kim (2014); Gao, Li, and Huang, 2018 combined Convolutional Neural Network (CNN) with pre-trained word vectors for classification and achieve excellent results. Wang et al. (2018) present a model by incorporating multiple-scale feature attention with CNN for text classification. The Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNN) also have been widely applied in text processing, they improve the performance of learning and processing text classification problems. RNN (Liu et al., 2016) is suitable to process variable length text, but the range of context is limited due to the vanishing gradient problem. LSTM (Johnson & Zhang, 2016) embed variable text regions and tackle the troubles of vanishing gradient and exploding gradient in the training process. In addition, some methods combine deep learning with attention mechanisms to increase performance. Zhou et al. (2016) proposed the Attention-based Bidirectional-LSTM model to capture semantic information of sentences on classification tasks. Recurrent-CNN (Lai, Xu, Liu, & Zhao, 2015) and C-LSTM (Kim & Cho, 2018; Shi et al., 2019) can achieve better performance by stacking the conventional layer with the sequence model. These deep learning approaches have achieved a good performance of classification, and their experimental results demonstrate the validity of extract text features without feature engineering.

7. Conclusion

With the rapidly growing of services, discovering, compositing, and managing appropriate services in large repositories is becoming an imperative challenge. However, it is feasible that utilizing classification techniques to mitigate this issue. Recently, a deep neural network-based service classification model ServeNet-BERT is demonstrated to be advantageous over conventional machine learning methods, such as LDA-SVM and C-LSTM. Generally, ServeNet-BERT captures features from service names and service descriptions separately, then concatenates them into a unified feature used for service classification. However, ServeNet-BERT still hardly extracts the interdependent associations between name and description.

To address this issue, we propose an approach named CARL-Net that utilizes a co-attentive representation learning mechanism to learn interdependent relationships for service features. We extract informative words from descriptions by performing information gain theory and take it with the service name as the additional service features for service classification. CARL-Net can learning latent information between service augmented data and description. Experiments on 10,943 services with 50 categories show that CARL-Net outperforms ServeNet-BERT. Therefore, co-attentive representation learning with service informative words is conducive for service classification.

In the future, we plan to (i) apply metaheuristic algorithms to find optimal parameters; (ii) solve the external disturbances and noises in practical applications; (iii) explore more auxiliary information of service into the co-attentive neural network for learning more latent characteristics to promote classification performance, such as service factors from social media or users' preferences.

CRediT authorship contribution statement

Bin Tang: Methodology, Validation, Investigation, Writing - original draft. **Meng Yan:** Conceptualization, Formal analysis, Supervision, Writing - review & editing. **Neng Zhang:** Formal analysis. **Ling Xu:** Methodology. **Xiaohong Zhang:** Project administration. **Haijun Ren:** Supervision.

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.

Acknowledgements

This work was supported in part by the National Key Research and Development Project under Grant 2018YFB2101200, in part by the National Natural Science Foundation of China under Grant 62002034, in part by the Fundamental Research Funds for the Central Universities under Grants 2020CDCGRJ072 and 2020CDJQY-A021.

References

Amato, A., & Venticinque, S. (2016). Multiobjective optimization for brokering of multicloud service composition. *ACM Transactions on Internet Technology (TOIT), 16*, 1–20.

Aznag, M., Quafafou, M., & Jarir, Z. (2014). Leveraging formal concept analysis with topic correlation for service clustering and discovery. In *2014 IEEE International Conference on Web Services* (pp. 153–160).

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence, 35*, 1798–1828.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177–186). Springer.

Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. *Siam Review, 60*, 223–311.

Cao, B., Liu, J., Tang, M., Zheng, Z., & Wang, G. (2013). Mashup service recommendation based on user interest and social network. In *2013 IEEE 20th international conference on web services* (pp. 99–106). IEEE.

Cao, B., Liu, X., Li, B., Liu, J., Tang, M., Zhang, T., & Shi, M. (2016). Mashup service clustering based on an integration of service content and network via exploiting a two-level topic model. In *2016 IEEE International Conference on Web Services (ICWS)* (pp. 212–219).

Chen, L., Wang, Y., Yu, Q., Zheng, Z., & Wu, J. (2013). Wt-lda: User tagging augmented lda for web service clustering. In *International conference on service-oriented computing* (pp. 162–176). Springer.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv: 1810.04805.

Elgazzar, K., Hassan, A. E., & Martin, P. (2010). Clustering wsdl documents to bootstrap the discovery of web services. In *2010 IEEE International Conference on Web Services* (pp. 147–154). IEEE.

Elshater, Y., Elgazzar, K., & Martin, P. (2015). Godiscovery: Web service discovery made efficient. In *2015 IEEE International Conference on Web Services* (pp. 711–716). IEEE.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research, 3*, 1289–1305.

Gao, M., Li, T., & Huang, P. (2018). Text classification research based on improved word2vec and cnn. In *International Conference on Service-Oriented Computing* (pp. 126–135). Springer.

Hao, Y., Zhang, Y., & Cao, J. (2010). Web services discovery and rank: An information retrieval approach. *Future generation computer systems, 26*, 1053–1062.

He, H., Zhao, W., Huang, S., Fox, G. C., & Wang, Q. (2020). Research on the architecture and its implementation for instrumentation and measurement cloud. *IEEE Transactions on Services Computing, 13*, 944–957. https://doi.org/10.1109/TSC.2017.2723006

Jin, O., Liu, N. N., Zhao, K., Yu, Y., & Yang, Q. (2011). Transferring topical knowledge from auxiliary long texts for short text clustering. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 775–784).

Johnson, R., & Zhang, T. (2016). Supervised and semi-supervised text categorization using lstm for region embeddings. arXiv preprint arXiv:1602.02373.

Kenter, T., & De Rijke, M. (2015). Short text similarity with word embeddings. In *Proceedings of the 24th ACM international on conference on information and knowledge management* (pp. 1411–1420).

Kim, T.-Y., & Cho, S.-B. (2018). Web traffic anomaly detection using c-lstm neural networks. *Expert Systems with Applications, 106*, 66–76.

Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. *Twenty-ninth AAAI conference on artificial intelligence*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*, 436–444.

Li, C., Zhang, R., Huai, J., Guo, X., & Sun, H. (2013). A probabilistic approach for web service discovery. In *2013 IEEE International Conference on Services Computing* (pp. 49–56). IEEE.

Liang, T., Chen, L., Wu, J., Xu, G., & Wu, Z. (2016). Sms: A framework for service discovery by incorporating social media information. *IEEE Transactions on Services Computing.*

Liu, P., Qiu, X., & Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101.

Liu, W., & Wong, W. (2009). Web service clustering using text mining techniques. *International Journal of Agent-Oriented Software Engineering, 3*, 6–26.

Liu, X., Agarwal, S., Ding, C., & Yu, Q. (2016). An lda-svm active learning framework for web service classification. In *2016 IEEE International Conference on Web Services (ICWS)* (pp. 49–56). IEEE.

Liu, X., & Fulia, I. (2015). Incorporating user, topic, and service related latent factors into web service recommendation. In *2015 IEEE International Conference on Web Services* (pp. 185–192). IEEE.

Lu, Y., Mei, Q., & Zhai, C. (2011). Investigating task performance of probabilistic topic models: an empirical study of plsa and lda. *Information Retrieval, 14*, 178–203.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research, 9*.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543).

Shi, M., Liu, J., Cao, B., Wen, Y., & Zhang, X. (2018). A prior knowledge based approach to improving accuracy of web services clustering. In *2018 IEEE International Conference on Services Computing (SCC)* (pp. 1–8). IEEE.

Shi, M., Liu, J., Zhou, D., Tang, M., & Cao, B. (2017). We-lda: A word embeddings augmented lda model for web services clustering. In *2017 IEEE International Conference on Web Services (ICWS)* (pp. 9–16).

Shi, M., Wang, K., & Li, C. (2019). A c-lstm with word embedding model for news text classification. In *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)* (pp. 253–257). IEEE Computer Society.

Skoutas, D., Sacharidis, D., Simitsis, A., & Sellis, T. (2010). Ranking and clustering web services using multicriteria dominance relationships. *IEEE Transactions on Services Computing, 3*, 163–177.

Stojanovic, V., Nedic, N., Prsic, D., Dubonjic, L., & Djordjevic, V. (2016). Application of cuckoo search algorithm to constrained control problem of a parallel robot platform. *The International Journal of Advanced Manufacturing Technology, 87*, 2497–2507.

Stojanovic, V., & Prsic, D. (2020). Robust identification for fault detection in the presence of non-gaussian noises: Application to hydraulic servo drives. *Nonlinear Dynamics, 100*, 2299–2313.

Tao, H., Li, X., Paszke, W., Stojanovic, V., & Yang, H. (2021). Robust pd-type iterative learning control for discrete systems with multiple time-delays subjected to polytopic uncertainty and restricted frequency-domain. Multidimensional Systems and Signal Processing, (pp. 1–22).

Tao, H., Wang, P., Chen, Y., Stojanovic, V., & Yang, H. (2020). An unsupervised fault diagnosis method for rolling bearing using stft and generative neural networks. *Journal of the Franklin Institute, 357*, 7286–7307.

Wang, S., Huang, M., & Deng, Z. (2018). Densely connected cnn with multi-scale feature attention for text classification. In IJCAI (pp. 4468–4474).

Wu, J., Chen, L., Zheng, Z., Lyu, M. R., & Wu, Z. (2014). Clustering web services to facilitate service discovery. *Knowledge and information systems, 38*, 207–229.

Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J., & Wu, C. (2014). Category-aware api clustering and distributed recommendation for automatic mashup creation. *IEEE Transactions on Services Computing, 8*, 674–687.

Xia, Y., Chen, P., Bao, L., Wang, M., & Yang, J. (2011). A qos-aware web service selection algorithm based on clustering. In *2011 IEEE International Conference on Web Services* (pp. 428–435). IEEE.

Xu, W., Cao, J., Hu, L., Wang, J., & Li, M. (2013). A social-aware service recommendation approach for mashup creation. In *2013 IEEE 20th International Conference on Web Services* (pp. 107–114). IEEE.

Yang, Y., Ke, W., Wang, W., & Zhao, Y. (2019). Deep learning for web services classification. In *2019 IEEE International Conference on Web Services (ICWS)* (pp. 440–442). IEEE.

Yang, Y., Qamar, N., Liu, P., Grolinger, K., Wang, W., Li, Z., & Liao, Z. (2020). Servenet: A deep neural network for web services classification. In *2020 IEEE International Conference on Web Services (ICWS)* (pp. 168–175). IEEE.

Ye, H., Cao, B., Peng, Z., Chen, T., Wen, Y., & Liu, J. (2019). Web services classification based on wide & bi-lstm model. *IEEE Access, 7*, 43697–43706.

Zhang, S., Zheng, D., Hu, X., & Yang, M. (2015). Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia conference on language, information and computation* (pp. 73–78).

Zhang, X., Liu, J., Cao, B., Xiao, Q., & Wen, Y. (2018). Web service discovery based on information gain theory and bilstm with attention mechanism. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing* (pp. 643–658). Springer.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in neural information processing systems (pp. 649–657).

Zheng, Z., Ma, H., Lyu, M. R., & King, I. (2013). Collaborative web service qos prediction via neighborhood integrated matrix factorization. *IEEE Transactions on Services Computing, 6*, 289–299.

Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., & Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. In *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 207–212).

Zhu, J., Kang, Y., Zheng, Z., & Lyu, M. R. (2012). A clustering-based qos prediction approach for web service recommendation. In *2012 IEEE 15th international symposium on object/component/service-oriented real-time distributed computing workshops* (pp. 93–98). IEEE.