Future Generation Computer Systems 130 (2022) 231–240

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Multi-information fusion based few-shot Web service classification

Yongqiang Liu, Bing Li *, Jian Wang *, Duantengchuan Li, Yutao Ma

School of Computer Science, Wuhan University, Wuhan, China

ARTICLE INFO

Article history:
Received 17 June 2021
Received in revised form 16 November 2021
Accepted 24 December 2021
Available online 3 January 2022

Keywords:
Web service classification
Few-shot learning
Deep learning
Long-tail distribution

ABSTRACT

Automatic Web service classification becomes an essential topic in the services computing field. The distribution of Web services over various categories usually follows the long-tail distribution, suggesting that many categories (i.e., the tail categories) contain very limited services. An empirical experiment shows that the classification performance of tail categories is much worse than that of head categories due to the limited training samples. Existing works on Web service classification usually ignore this problem. Towards this issue, we propose a few-shot Web service classification approach called MIF-FWSC (multi-information fusion based few-shot Web service classification), which exploits both the knowledge learned from head categories and the information contained by category names to improve the classification of tail categories. Experiments show that our proposed approach for few-shot Web service classification achieves state-of-the-art accuracy on two real-world Web service datasets.

1. Introduction

A Web service is a software component that can be published, located, and accessed by following standard Web protocols [1]. With the rapid development of services computing and cloud computing, more and more enterprises are encapsulating their computing capabilities and applications as Web services. Thus, a vast number of Web services have been released on the Internet. Currently, many Web service registries or marketplaces, e.g., the PW[1] (ProgrammableWeb) and Amazon Web service marketplace,[2] have been created to publish and manage publicly available Web services. Each service is typically described in a service registry with its name, description, and one or more categories like financial, mapping, transportation, and so on. The category information is crucial for the discovery and reuse of Web services [2]. Because manually assigning Web services categories is time-consuming and error-prone, automatic Web service classification has become an essential topic in the services computing field.

In recent years, lots of Web service classification approaches [3–8] have been proposed, which leverage various machine learning and deep learning techniques in service classification. Classical machine learning-based service classification methods firstly obtain vectorized representations of textual service descriptions using BOW (Bag of Words), TF-IDF (Term Frequency-Inverse Document Frequency), or topic models, and then train classification models like Naive Bayes [5], K-Nearest Neighbor (KNN), Logistics Regression (LR) and Support Vector Machines (SVM) [6] based on these vectors. Many deep learning models, like CNN (Convolutional Neural Networks) and LSTM (Long Short Term Memory), are also leveraged to perform feature extraction and service classification using end-to-end mechanisms. These approaches usually require large-scale labeled datasets in model training to achieve ideal performances and can hardly learn from limited samples [8,9]. However, the category distribution of Web services usually follows the long-tail effect, which suggests that most Web services are accounted for by a small number of categories (also known as head categories), and most categories (also known as tail categories) contain very limited Web services. As a typical example, in a dataset with over 20,000 Web services and over 400 categories crawled from PW on Jan 10, 2020, over 200 categories have less than 50 Web services, and 66 categories include less than ten services, which account for 44.31% and 13.66%, respectively. As shown in Fig. 1, the number of services belonging to a category has decreased rapidly from head categories to tail categories.

We conducted an empirical experiment to analyze the limitation of existing classification models in classifying tail categories mentioned above. Here we only illustrate observations on representative approaches: KNN, SVM, LR, and TextCNN, covering both traditional machine learning and deep learning. More descriptions of the experimental dataset and settings are introduced in Section 5. In PW, each Web service may belong to multiple categories. We adopt Mean Average Precision (MAP), a widely-used metric in multi-label classification, to evaluate the classification performance. In each category, 80% of all Web services are randomly selected for training and the remainder are used for

* Corresponding authors.
E-mail addresses: bingli@whu.edu.cn (B. Li), jianwang@whu.edu.cn (J. Wang).

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

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

https://doi.org/10.1016/j.future.2021.12.020
0167-739X/

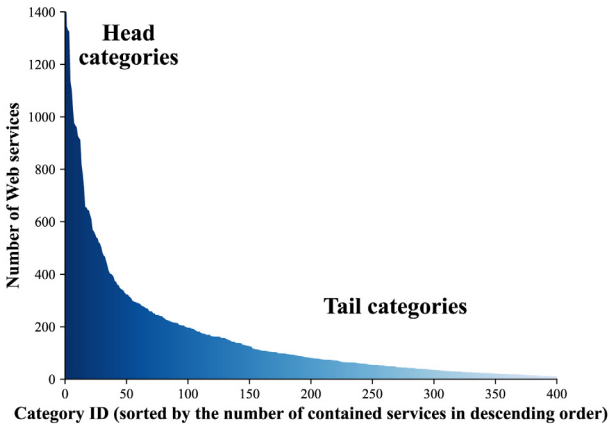


Fig. 1. The category distribution of Web services in PW. As the color changes from dark to light, the category contains fewer and fewer services, which means that the categories change from head categories to tail categories.

Table 1
Mean Average Precision (MAP) among different category ID range (category IDs are sorted by the number of contained services in descending order).

Model	Set of Category IDs				
	1–100	101–200	201–300	301–400	>400
KNN	0.4596	0.3512	0.3153	0.2975	0.2245
SVM	0.5443	0.4398	0.3646	0.3446	0.2881
LR	0.5718	0.4651	0.4028	0.3950	0.3199
CNN	0.6026	0.4898	0.3835	0.3512	0.2813

testing. In Table 1, since category IDs are sorted by the number of contained services in descending order, the categories in the ID set "1–100" have more training samples than those in "101–200" on average. We can observe that with the decrease of the number of training samples, the classification performance will decrease accordingly. The four classification models show the same trend. Therefore, the classification performance is positively correlated with the number of training services in categories no matter which classification method is adopted. In other words, existing classification models can hardly achieve ideal classification performance with only a few training samples.

Towards this issue, we focus our attention on the classification task of categories containing only a few Web services, which will significantly affect the overall classification performance. However, it was ignored in previous research work. We view this task as a few-shot text classification problem. Few-shot learning (FSL) [10,11] has recently become a hot topic in many machine learning tasks like image classification and sentiment classification. Similar to existing progress in FSL, we also attempt to utilize the knowledge learned from the head categories to improve the classification performance of the tail categories. Moreover, considering that Web service descriptions are usually characterized by short-length and low-information density [12], our goal is to extract the key and domain-relevant information from service descriptions. The contributions of this paper are summarized as follows:

- We find that the distribution of Web services over different categories usually follows the long-tail distribution. We conduct experiments to show that the classification performance of tail categories is much worse than head categories. This phenomenon was neglected in previous research work.
- A few-shot Web service classification method called MIF-FWSC (multi-information fusion based few-shot Web service classification) based on meta-learning framework is proposed to classify the categories with only a few Web services. MIF-FWSC can find key words in the service description and exploit the knowledge in head categories to make the model more robust across different categories. It also incorporates the information contained in the category names to enhance the classification ability.
- A series of experiments are conducted on two real-world datasets to demonstrate the effectiveness of our proposed model. The results show that MIF-FWSC achieves state-of-the-art performance for few-shot Web service classification.

The remainder of the paper is organized as follows: Section 2 introduces related work on Web service classification and the few-shot text classification. Some preliminaries about the few-shot classification and meta-learning framework are described in Section 3. Section 4 provides the details of MIF-FWSC, and Section 5 reports the experiment results. Finally, concluding remarks are described in Section 6.

2. Related work

Web service classification has been widely investigated in services computing, and a lot of works have been proposed in this area using various machine learning and deep learning techniques. Existing Web service classification approaches have not considered the classification scenarios of categories with only a few Web service samples. Few-shot text classification recently becomes a problem worth exploring in machine learning, and we will introduce the related work in this area.

2.1. Web service classification

Quite a few Web service classification works based on conventional machine learning methods have been proposed in recent years. These works typically extract features from Web service descriptions, and then construct classifiers based on the extracted features. For example, Liu et al. [5] presented a semantic Web service classification method based on Naive Bayes. The work [6,7] compared several machine learning methods such as SVM, KNN, and decision tree for Web service classification on seven categories. According to their reports, SVM performs the best on their datasets. Qamar et al. [3] employed the ensemble of Naive Bayes, Decision Tree (J48), and SVM for the classification of Web services, which outperforms every single classifier over the average accuracy on a publicly available dataset.

With the rapid development of deep learning in text classification, many works attempted to leverage deep learning in service classification. Yang et al. [4] introduced a stacked deep neural network named ServeNet by integrating CNN and LSTM, which can automatically extract low-level representations from service descriptions without any feature engineering. The work [8] refined ServeNet by adding service names as input and changing the embedding method from fixed embeddings to context-aware embeddings.

A significant limitation of service classification approaches based on conventional machine learning and deep learning is that they need sufficient training samples to achieve ideal classification performance, as discussed in Section 1. However, many service categories in real-world registries contain only a few samples, making it challenging to train an accurate model. Therefore, many existing service classification approaches (e.g., [8]) removed the one-shot, few-shot, small size categories and kept only large size categories for service classification experiments, which suggests that these approaches can only be applied in datasets whose categories all have sufficient samples.

2.2. Few-shot text classification

Few-shot text classification aims to construct a text classifier for the categories with only a few samples. The classical classification methods like SVM, Multi-Layer Perceptron (MLP), and the transformer cannot directly achieve ideal classification performance on this problem since there are too few labeled samples in each category, while these methods all require a large number of labeled samples for model convergence. Recently, meta-learning [13,14] has become the mainstream framework for few-shot classification. It extracts certain transferable knowledge from a set of auxiliary tasks to help solve the target few-shot classification problem. The work [15] proposed prototypical networks, which learn a non-linear mapping of the input into an embedding space and represent each class by using the mean of the samples' embedding vectors belonging to it. An embedded query point can be classified by finding the nearest class prototype. The work [16] argued that in prototypical networks, the key information may be lost by simply averaging representations of samples belonging to a class. They built an induction module using the capsule network [17] to get representations for each class and a relation module to compare representations between samples and classes. Victor et al. [18] fed the embeddings of samples into a graph neural network and then got the prediction labels through trainable adjacency. Bao et al. [19] used distributional information like TF-IDF to get weights for words to build transferable lexical representations for texts. They then adopted the ridge regression to fit these representations with their labels. They also conducted experiments based on BERT [20] to build contextualized representations for texts and found that BERT performs not well on the keyword-based few-shot text classification problem.

To the best of our knowledge, the few-shot Web service classification problem has not been investigated in the existing literature. This paper aims to leverage few-shot text classification techniques in Web service classification. Since the description texts of Web services are usually characterized by short-length and low-information density [12], it is particularly important to capture the key information carried by one or several keywords in the description text. The work [19] has shown that distributional signatures like TF-IDF are effective in finding keywords of text for few-shot text classification. We further improve the way of calculating word weights to better fit for Web service classification. Moreover, to tackle the inaccurate class representation caused by insufficient labeled Web services, we use an attention mechanism to incorporate the information contained in the category names, which can enhance the classification ability.

3. Preliminaries

3.1. Few-shot classification

Classical machine learning approaches have made a huge success in data-intensive tasks, but their performance declined significantly when there are only a limited number of training samples. Few-shot learning (FSL) [10,11] is proposed to address this problem. Few-shot classification is a specific application of FSL, which learns classifiers given only a few labeled examples in each class. The few-shot classification has been applied in many areas, such as image classification, sentiment classification, and object recognition. Generally speaking, the few-shot classification problem needs two parameters N and K, where N denotes the number of categories, and K denotes the number of labeled samples contained in each category. Note that K is usually a small value, typically between one and five. This is also called an N-way K-shot problem.

3.2. Meta-learning framework

Meta-learning [13,14] is a mainstream framework to deal with few-shot learning problems. It aims to learn prior knowledge on a large-scale of samples with the same distribution as the few labeled samples. Formally, suppose that we have a training set S_{train} and a test set S_{test}, S_{train} and S_{test} share the same data distribution. Let C_{train} and C_{test} denote the categories of S_{train} and S_{test}, respectively. $C_{train} \cap C_{test} = \emptyset$. Each category in C_{test} contains only a few labeled samples. The ultimate objective of meta-learning is to train a representation model that can transfer samples from raw inputs to informative vectors, as well as a classification model that can classify samples based on these vectors. Typically, the representation model is only trained on the training set, while the classification model can be trained on the test set besides the training set.

4. Proposed model

As discussed above, since the category distribution of Web services usually follows the long-tail effect, most categories have only a few services. The problem to be addressed in this paper is how to classify categories with a few labeled Web services, also known as the few-shot Web service classification problem.

We adopt the meta-learning framework to address this problem, aiming to transfer the knowledge learned from categories with large-scale samples to categories with a few samples. This paper proposes a few-shot Web service classification approach called MIF-FWSC (multi-information fusion based few-shot Web service classification), which exploits both the knowledge learned from head categories and the information contained by category names to improve the classification of tail categories.

Since MIF-FWSC adopts an episode-based mechanism for model training, validation and test, which is different from traditional supervised learning, we first introduce the episode-based mechanism of MIF-FWSC. Then we describe the proposed classification model in detail.

4.1. Episode-based mechanism of MIF-FWSC

MIF-FWSC follows the meta-learning framework by using the episode-based mechanism for training, validation and test procedures. It is important to describe the episode-based mechanism and these procedures in detail for the differences between them and traditional supervised learning. Since the validation procedure is very similar to the test procedure, we only introduce the test procedure below. The only difference between the validation procedure and the test procedure is that the former is carried out after every training procedure among the validation set, while the latter is carried out after the whole training procedure among the test dataset. Note that the symbols defined in Section 3.2 will be applied here.

The model training process consists of multiple episodes, as introduced in [21]. The procedure of each episode is defined as follows. Given parameters N and K (in the N-way K-shot problem), we first select a subset containing N categories from C_{train}, as shown in Fig. 2. Then we randomly choose K Web services from each selected category and these $N \times K$ samples make up a support set, as indicated in Fig. 3. A subset of the remaining training samples is then chosen as a query set to evaluate the model's performance after training on the support set. The sample number of each category in the query set is Q, which is also a predefined parameter. In this way, the training strategy is called an episode-based mechanism. Since the procedure of selecting categories, a support set and a query set for an episode is independent of other episodes, the model can learn transferable and generalized knowledge across different episodes.

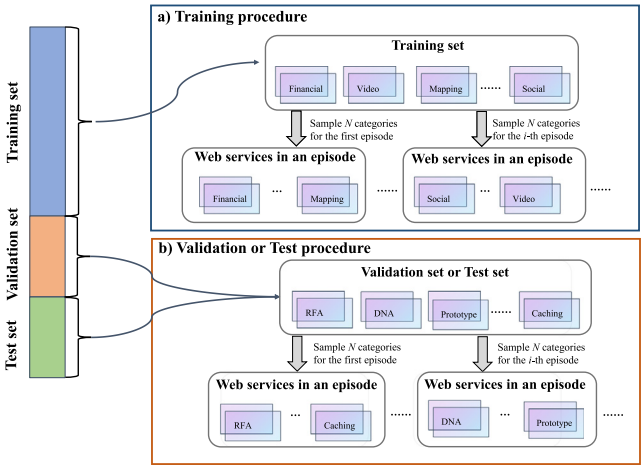


Fig. 2. Episode-based training, validation, and test procedures.

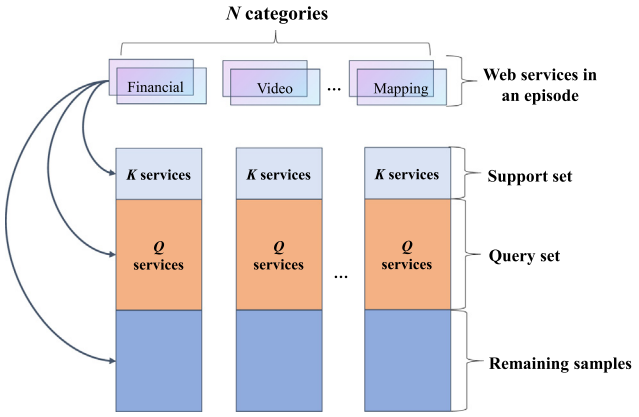


Fig. 3. Sampling a support set and a query set in an episode.

After training the model, we apply the same episode-based mechanism in the test procedure. Specifically, we also first sample N categories from C_{test} in each test episode. Afterward, both the support set and the query set are sampled from these N categories like the training procedure. The final test performance of the model is defined as the average performance on the query set across all testing episodes. Figs. 2 and 3 show the episode-based training, validation, and test procedure, as well as the process of selecting a support set and a query set in an episode, respectively.

4.2. Details of MIF-FWSC

As mentioned before, a typical meta-learning framework for few-shot classification usually consists of two parts: a representation model and a classification model. Our model is no exception. Traditional feature extractors like CNN and RNN always perform poorly when used as the representation part for textual few-shot classification. Bao et al. [19] have shown that word distributional signatures calculated on both the whole dataset and the dataset of each episode are essential in offering transferable classification knowledge across different categories. In addition to this, we also found that the category names of the Web services in each episode can help improve classification performance. Inspired by these ideas, we build our few-shot service classification model named MIF-FWSC. As shown in Fig. 4, MIF-FWSC consists of three components:

- **Word attention generator**, which generates a corresponding attention score for each word in the service description by combining the attention scores calculated among the whole training set and the specific episode using BiLSTM;
- **Description vector generator**, which produces a lexical representation vector for the description based on the generated word attentions and word vectors. For descriptions in the support set, MIF-FWSC also uses the word vector of the category name to augment the description vector;
- **Classification component**, which trains a classifier on the support set. The classifier we adopt is a modified ridge regression due to its efficient and effective, according to [19, 22].

Note that the first two components that form the representation part are shared by all episodes, while the last that forms the classification part is independent among different episodes. We introduce these three components in detail in the subsequent subsections.

4.2.1. Word attention generator

The word attention generator aims to assess the attention score of each word in a sentence of a service description. Under the episode-based mechanism, the only labeled samples are the Web services in the support set of each episode. The distributional signatures among the episode samples and the whole dataset can both contribute to the attention score of a word [19]. Thus, we calculate the local attention among the episode Web services and the global attention among the remaining Web services which constitute the sample pool. The scope of the sample pool has a slight difference in the training stage and the test stage. To ensure that only unlabeled Web services are included in a sample pool, all Web services in the training set except those in the support set of the current episode are contained in the sample pool in the training stage; while the sample pool refers to all Web services in the training set. Finally, the overall attention of a word can be calculated by combining these two attention scores. Next, we describe in detail how to calculate them.

The local attention scores of a specific episode need to be discriminative for the classification task of this episode. Intuitively, the more similar a word is to the category names of the episode, the more important it is in the episode classification task. We, therefore, calculate the cosine similarities between word w and each category name in the episode. Then, we choose the maximum similarity as la_w:

$$la_w = \max_{cat \in \{cat_1, \ldots, cat_N\}} f_{ebd}(cat) \cdot f_{ebd}(w), \tag{1}$$

where N is the category number of the support set, $\{cat_1, \ldots, cat_N\}$ is the category names in this episode, $f_{ebd}(\cdot)$ is a word embedding function that maps a word to a dense word vector.

Since words appearing frequently are unlikely to be informative, an intuitive strategy is to reduce the weights of frequent words and improve the weights of rare words. We choose the Inverse Word Frequencies [23] to calculate the global attention ga_w of word w in a service description.

$$ga_w = \log \frac{\sum_{k=1}^{m} n_{w,k}}{n_w}, \tag{2}$$

where n_w is the occurrence frequency of w in the sample pool, m is the vocabulary size of the sample pool and $n_{w,k}$ is the occurrence frequency of the kth word in the vocabulary.

To leverage the complementary information of global attention scores and local attention scores, we first concatenate them together and then employ a bi-directional LSTM to transfer the two scores of each word to a hidden state $\mathbf{h}$.

$$(\mathbf{h}_1, \ldots, \mathbf{h}_i, \ldots, \mathbf{h}_l) = BiLSTM([ga_1, la_1], \ldots, [ga_i, la_i], \ldots, [ga_l, la_l]), \tag{3}$$

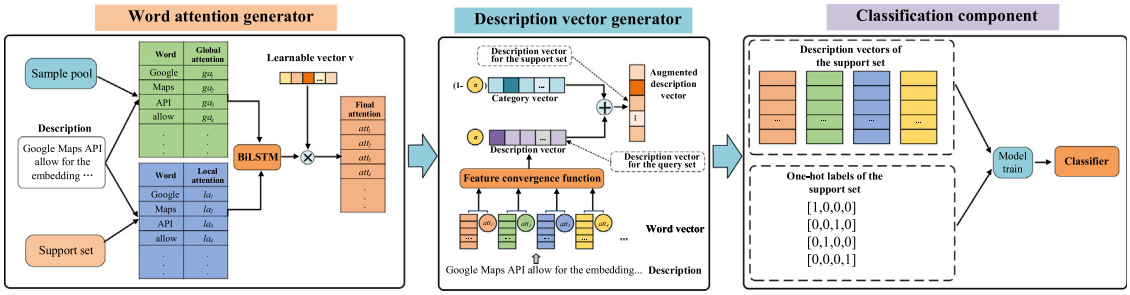


Fig. 4. The framework of MIF-FWSC.

where ga_i, la_i and $\mathbf{h}_i$ are the global attention score, the local attention score and the hidden state of the ith word of the description respectively. l denotes the length of the description and $[\cdot]$ means the combination operation of vectors.

Afterward, we apply the dot product between $\mathbf{h}$ and a learnable vector $\mathbf{v}$ to calculate a score for each word. Finally, we use the Softmax function to normalize these scores and output the final attention score att_w for each word w in a service description.

$$att_i = \frac{e^{\mathbf{h}_i \cdot \mathbf{v}}}{\sum_{k=1}^{l} e^{\mathbf{h}_k \cdot \mathbf{v}}}, \tag{4}$$

where att_i is the final attention score for the ith word of the description.

4.2.2. Description vector generator

After obtaining the attention scores of all words in a description, we can construct a lexical representation for the description. First, the embedding vector of each word is multiplied by its attention score. Then the feature convergence function will convert these weighted word vectors to a description vector. The vector of a service description is defined as:

$$\begin{aligned}\mathbf{vec}_{des} = \mathscr{F}(&f_{ebd}(word_1) \cdot att_1, \ldots, \\ &f_{ebd}(word_i) \cdot att_i, \ldots, f_{ebd}(word_l) \cdot att_l),\end{aligned} \tag{5}$$

where des denotes the description, l denotes the length of des, $word_i$ is the ith word of des, $f_{ebd}(\cdot)$ is a word embedding function that maps a word to a dense word vector, and att_i is the final attention score of the ith word of des, and $\mathscr{F}$ is the feature convergence function which aggregates all weighted word vectors in des into a description vector.

For the description in the support set whose real category is known, we also use its category name to augment the description vector. The vector of the category name is the average of the vectors of all the words in the description. For example, the vector of category "word processing" is the average of the vectors corresponding to "word" and "processing". Then we use a self-attention mechanism to combine the vector of a service description $\mathbf{vec}_{des}$ and the vector of label category $\mathbf{vec}_{cat}$:

$$\mathbf{vec}_{aug} = \alpha \cdot \mathbf{vec}_{des} + (1 - \alpha) \cdot \mathbf{vec}_{cat}, \tag{6}$$

where α is the weight term obtained by self-attention mechanism, and $\mathbf{vec}_{aug}$ is the augmented vector for the description in the support set. This augmentation will provide a more accurate representation of Web service for the following classification component. Note that this augmentation is only designed for the support set since the real categories are only known for the Web services in the support set.

4.2.3. Classification component

For an N-way K-shot problem, the support set can be vectorized as a matrix: $\mathbf{X}_s \in \mathbb{R}^{(N\times K)\times E}$, where E denotes the dimension of the sentence vector. Each row of $\mathbf{X}_s$ is a lexical representation of a service description of the support set. $\mathbf{Y}_s \in \mathbb{R}^{(N\times K)\times N}$ is a set of one-hot categories of the support set. The classification component needs to fit $\mathbf{X}_s$ with the target $\mathbf{Y}_s$. According to [22], the ridge regression minimizing squared loss for one-hot categories works well with this classification problem:

$$L(\mathbf{W}) = \text{argmin}_{\mathbf{W}} \|\mathbf{X}_s\mathbf{W} - \mathbf{Y}_s\|^2 + \lambda\|\mathbf{W}\|^2, \tag{7}$$

where $\mathbf{W} \in \mathbb{R}^{E\times N}$ is the weight matrix for ridge regression and λ is a positive parameter controlling the complexity of $\mathbf{W}$. We can obtain $\mathbf{W}$ by:

$$\mathbf{W} = \mathbf{X}_s^T \left(\mathbf{X}_s\mathbf{X}_s^T + \lambda\mathbf{I}\right)^{-1}\mathbf{Y}_s, \tag{8}$$

where $\mathbf{I}$ is an identity matrix. Note that λ is a hyper-parameter for the ridge regressor but can be learned in the whole training procedure.

Following the definitions of the support set, we define $\mathbf{X}_q \in \mathbb{R}^{(Q\times K)\times E}$ and $\mathbf{Y}_q \in \mathbb{R}^{(Q\times K)\times N}$ as the representation of the query set and the one-hot categories of the query set, respectively. Then the predicted probabilities $\hat{\mathbf{Y}} \in \mathbb{R}^{(Q\times K)\times N}$ can be calculated by:

$$\hat{\mathbf{Y}} = \mathbf{X}_q\mathbf{W}. \tag{9}$$

$\hat{\mathbf{Y}}_{ij}$, an element in row i and column j of $\hat{\mathbf{Y}}$, represents the probability before normalization for the ith Web service in the query set belonging to category j. To normalize these probabilities, we apply the Softmax function over $\hat{\mathbf{Y}}$ to get $\mathbf{P}_{ij}$, which indicates the predicted probability that the ith Web service in the query set belongs to category j:

$$\mathbf{P}_{ij} = \frac{e^{\hat{\mathbf{Y}}_{ij}}}{\sum_k e^{\hat{\mathbf{Y}}_{ik}}}. \tag{10}$$

Then we can predict the category of the ith Web service according to Eq. (11):

$$\hat{y}_i = \text{argmax}_{j\in[0,N)}\,\mathbf{P}_{ij}. \tag{11}$$

We can compute the cross-entropy between $\mathbf{P}$ and $\mathbf{Y}_q$ during the training procedure:

$$CE\left(\mathbf{Y}_q, \mathbf{P}\right) = -\frac{1}{Q\times K}\sum_{i=1}^{Q\times K}\sum_{j=1}^{N}\mathbf{Y}_{ij}\log\mathbf{P}_{ij}. \tag{12}$$

Since both $\mathbf{X}_s$ and $\mathbf{X}_q$ depend on the word attention generator and the description vector generator, $CE(\mathbf{Y}_q, \mathbf{P})$ can provide supervision for them. For the test procedure, we use Eq. (11) to get the predicted categories for Web services in the query set.

5. Experiments

We performed a series of experiments using two real-world Web service datasets to evaluate the proposed MIF-FWSC. All experiments were carried out on a workstation with Intel Core 8 at 3.50 GHz, GeForce GTX 2080, and 32 GB memory, running the

Table 2
An example Web service in ProgrammableWeb.

Attribute	Value
Name	BitStamp HTTP
Categories	Financial, Bitcoin, Currency, Marketplace
Description	BitStamp is an online exchange for bitcoins. Online consumers and traders can use it as a global marketplace to buy and sell BitCoins…"

Table 3
An example Web service in Amazon Web service marketplace.

Attribute	Value
Name	GluonCV YOLOv3 Object Detector
Categories	Computer Vision, Image
Description	"Given an input image, this will return object coordinates and category predictions…"

Ubuntu operating system. We make our code and data publicly available for further study.[3] The experiments were designed to answer the following research questions:

- **RQ1:** Does MIF-FWSC outperform the baselines on the few-shot Web service classification task?
- **RQ2:** Does the meta-learning framework adopted in our model perform better than traditional supervised learning on the few-shot Web service classification task?
- **RQ3:** Is each part of MIF-FWSC necessary to achieve ideal classification performance?
- **RQ4:** How does the feature convergence function of MIF-FWSC affect the classification performance?

5.1. Dataset and evaluation metrics

ProgrammableWeb (PW) and Amazon Web service marketplace (AWS) are by far the two largest online Web service registries, where new Web services and new service categories are continuously registered. Each Web service in PW and AWS belongs to one or more service categories, and lots of categories contain only a few Web services. In the dataset crawled from PW on Jan 10, 2020, over 200 categories have less than 50 Web services, and 66 categories include less than ten services, which account for 44.31% and 13.66%, respectively, of the total 483 categories. And in the dataset crawled from AWS on Aug 7, 2021, over 31 categories have less than 50 Web services, and 10 categories include less than ten services, which account for 25.41% and 8.2%, respectively, of the total 122 categories. Table 2 and Table 3 show example Web services in PW and AWS, respectively.

To ensure that there are enough services to construct the support set and query set, we discard categories with less than six Web services since we attempt to experiment with 1-shot and 5-shot problems. The training set and the test set are split according to the service categories. We need to guarantee sufficient tail categories (those containing a relatively small number of services) can be leveraged in the validation and test sets, which is necessary to make a comprehensive experimental evaluation. The categories with more than 30 Web services are selected as the training set for PW and the threshold is 150 for AWS. The rest categories are equally divided into the validation set and the test set. Since many Web services belong to more than one category, they may occur in both the training set and the validation set (or the test set). To avoid overlapping, we remove Web services in the training set that also belong to the validation set or the test set. Table 4 and Table 5 show the statistics of the constructed PW and AWS datasets, respectively.

[3] https://github.com/ssea-lab/FSC4WebService.

Table 4
Statistics of the constructed PW dataset for few-shot Web service classification.

Statistics	Value
Number of categories in the training set	145
Number of Web services in the training set	7,698
Number of categories in the validation set	60
Number of Web services in the validation set	1,034
Number of categories in the test set	59
Number of Web services in the test set	1,016

Table 5
Statistics of the constructed AWS dataset for few-shot Web service classification.

Statistics	Value
Number of categories in the training set	51
Number of Web services in the training set	21,013
Number of categories in the validation set	23
Number of Web services in the validation set	1,191
Number of categories in the test set	22
Number of Web services in the test set	1,127

Many Web services belong to more than one category. Since we adopt the episode-based mechanism, which only samples a few categories in an episode, and we have removed the Web services belonging to both the training set and the test set, it is very rare for a Web service with more than one category occurring in an episode. Even if there are exceptions in a few episodes, we will remove the Web services belonging to more than one category in these episodes.

To evaluate the performance of the proposed model, we report the average accuracy on the test set across all test episodes:

$$Accuracy = \frac{1}{M}\sum_{i} \mathrm{I}\left(y_i, \hat{y}_i\right), \tag{13}$$

where M is the number of Web services used in the test procedure, y_i is the real label of the ith service in the whole query service list used in all test episodes and $\hat{y}_i$ is the corresponding predicted label. $\mathrm{I}(\cdot)$ is the indicator function that returns 1 when y_i equals to $\hat{y}_i$, and 0 otherwise. Since the average accuracy will fluctuate due to the episode mechanism, the standard deviation among all test episodes is also reported.

5.2. Competing approaches

Since a typical meta-learning framework usually consists of a representation part and a classification part, we compare our model with different combinations of representation and classification models.

Representation model. We evaluate three representation models. META denotes the representation part in our model introduced in Sections 4.2.1 and 4.2.2. CNN denotes the TextCNN, which applies one-dimension convolution kernels on the embedding matrix of input words and then gets the representation of the words by max-pooling. AttnBiLSTM denotes a bidirectional LSTM encoder with an attention mechanism. We also experiment with pre-trained BERT embeddings (Devlin et al. [20]) as the representation model.

Classification model. Ridge regressor (RR) [22] is introduced in Section 4.2.3. ROUTING [17] computes a prototype for each class through dynamic routing over the support set. It uses a neural tensor layer [24] to predict the relation between each query example and the class prototypes. Similar to ROUTING, the prototypical networks (PROTO) [15] also compute a prototype representing each class. However, it just uses the centroid of the samples belonging to the class in the support set as the prototype. It then optimizes the embedding space by minimizing

the Euclidean distance between the prototype and the samples of this class in the query set. GNN [18] constructs a graph on the combination of both the support set and the query set, and then gets the prediction labels through trainable adjacency.

Due to the limitation of GPU, we only experiment with the combination with PROTO and RR for BERT. For the classification model using GNN, we only conduct experiments on the PW dataset. It should be noted that these missing experiments will not affect our final observation and analysis of the experimental results. MIF-FWSC denotes our approach, and the combination of META and RR is a variant of our approach without category name augmentation. Since some part of our method is adapted from [19] (referred to as BAO), we also report the performance of their model in experiments.

5.3. Parameter settings

We choose pre-trained fastText embeddings [25] as word vectors for all approaches except BERT. The embedding dimension is 300 for pre-trained fastText embeddings and 768 for pre-trained BERT embeddings. We set the hidden unit of the bi-directional LSTM and the learnable vector of the attention generation component in both MIF-FWSC and BAO to 50. We use the summation function as the feature convergence function in the description vector generator of MIF-FWSC. When using CNN as the representation model, we use one-dimensional convolution kernels with sizes of 3, 4, and 5, and the number of kernels is set to 50 for each size. When using AttnBiLSTM as the representation model, the hidden unit is set to 128. We choose Adam [26] as the optimizer and set the learning rate to 0.001.

During the training stage, we sample 100 training episodes per epoch. We also sample 100 validation episodes after each training epoch. An early stop mechanism is applied by stopping the training procedure when the validation accuracy does not improve for 20 epochs. After training, we sample 1000 episodes on the test set and calculate the average accuracy and corresponding standard deviation among all episodes.

5.4. Accuracy comparison (RQ1)

Table 6 shows the results of different approaches in settings of 5-way 1-shot, 10-way 1-shot, 5-way 5-shot, and 10-way 5-shot on PW. Table 7 shows the same results on AWS. As can be seen, our approach, with or without category name augmentation, exhibits improvements over all competing methods across all settings. The representation model (META) adopted in our approach achieves better results than other representation models when using the same classification model. The classification model (RR) adopted in our approach performs better than other classification models when using the same representation model. Compared with BAO, our approach performs much better, even without category name augmentation (META+RR). Specifically, it exceeds BAO on PW by 11.11%, 12.48%, 2.92%, and 3.01% in settings of 5-way 1-shot, 10-way 1-shot, 5-way 5-shot, and 10-way 5-shot, respectively. On AWS, the improvement is 4.54%, 4.54%, 1.18%, and 1.96%, respectively. These improvements are mainly caused by the different word attention calculation ways between their model and ours. To calculate local attention scores of words, BAO first represents each service with the average word embedding. Then they use the ridge regression to generate a weight matrix $\mathbf{W} \in \mathbb{R}^{E\times N}$ with vectors of Web services and their corresponding labels in the support set. The local attention of a word is obtained by multiplying its embedding vector by $\mathbf{W}$ and picking the maximum of all dimensions of the result. We modify the calculation process by choosing the maximum cosine similarities between the word and each category name in the episode, as shown in Eq. (1). After adding category name augmentation, it brings an additional 6.14%, 7.92%, 0.53% and 1.27% improvement on PW and 11.96%, 11.73%, 0.83% and 1.51% improvement on AWS, respectively.

In general, MIF-FWSC exceeds BAO on PW by 17.25%, 20.40%, 3.54%, and 4.28% in settings of 5-way 1-shot, 10-way 1-shot, 5-way 5-shot, and 10-way 5-shot, respectively. And on AWS, the improvement is 16.50%, 18.27%, 2.01%, and 3.47%, respectively. These results show the considerable advantage of MIF-FWSC over all other baselines for the few-shot Web services classification.

We can also observe from the results that when the shot number increases from 1 to 5, the improvement ratios of our approach become smaller whether or not using category name augmentation. The reason for the decline of the improvements is analyzed as follows. Since the improvement of our approach mainly comes from the use of category names of each episode and each category contains very few labeled Web services in an episode, it is inaccurate to only use their description to represent the category. The use of category names will correct the representation of Web services. When the number of labeled Web services increases, the improvement effect of this correction will be reduced.

5.5. Meta-learning vs. Supervised learning (RQ2)

The meta-learning framework adopted in this paper aims to learn transferable knowledge from training data and make predictions for unseen classes. We show this advantage by comparing the performances of meta-learning and traditional supervised learning. For supervised learning approaches, we follow the episode-based mechanism but remove the training procedure of the meta-learning framework. In other words, we train models using the support set of the test procedure and test them using the query set of the test procedure. For example, for 5-way 5-shot, we train a five-classification model that only uses 25 labeled Web services (that is, five labeled Web services for one class) and then test this model on the query set of this episode. We select some traditional machine learning methods, including KNN, SVM, and LR, for comparison. Note that we use TF-IDF for feature representation to reduce overfitting since there are too few samples to learn high-quality embedding vectors. The number of classes is set to 5 and the number of training samples is increased from 1 to 50 for supervised learning to observe the relationship between the accuracy and the number of training samples. The episode numbers of both meta-learning and supervised learning are 1000.

As shown in Fig. 5, SVM achieves the best performance in the three supervised learning algorithms. The performance of LR is similar to SVM. However, the accuracy of KNN, SVM, and LR is much lower than that of 5-shot under the meta-learning framework even when the number of training samples is increased to 50. For the PW dataset, the accuracy of these supervised learning approaches is even much lower than that of 1-shot under the meta-learning framework. These results show that the meta-learning framework can learn transferable knowledge from other categories to improve classification performance.

5.6. Ablation study (RQ3)

As mentioned above, our representation model consists of several components like the local attention generator and the global attention generator. We perform ablation studies to show the necessity of each component. Since the representation model has a more complex structure, we mainly consider different variants of the representation model. To verify whether the global attention and the local attention contribute to the classification performances, we conduct experiments by constructing the word attention generator using only the global attention or only the

Table 6
Accuracy comparison of different approaches on PW. Rep denotes representation model and Cls denotes classification model. The best values are marked by bold font. $\Delta\%$ denotes the improvement of MIF-FWSC over BAO.

Method		5-way 1-shot	10-way 1-shot	5-way 5-shot	10-way 5-shot
Rep	Cls				
CNN	ROUTING	0.4348 ± 0.1124	0.2807 ± 0.0714	0.5182 ± 0.1071	0.3578 ± 0.0672
AttnBiLSTM	ROUTING	0.5459 ± 0.1325	0.4438 ± 0.0862	0.7301 ± 0.1002	0.5890 ± 0.0678
META	ROUTING	0.5528 ± 0.1592	0.4708 ± 0.0961	0.7643 ± 0.1069	0.6841 ± 0.0691
BERT	PROTO	0.5133 ± 0.1232	0.3866 ± 0.0785	0.7216 ± 0.1036	0.6129 ± 0.0810
CNN	PROTO	0.3881 ± 0.1082	0.2843 ± 0.0814	0.5645 ± 0.1080	0.4223 ± 0.0878
AttnBiLSTM	PROTO	0.4932 ± 0.1206	0.3273 ± 0.0871	0.6489 ± 0.1136	0.4384 ± 0.0834
META	PROTO	0.6069 ± 0.1316	0.4916 ± 0.0970	0.8063 ± 0.0789	0.6336 ± 0.0739
CNN	GNN	0.4820 ± 0.1075	0.3292 ± 0.0679	0.5403 ± 0.1224	0.4822 ± 0.0875
AttnBiLSTM	GNN	0.5492 ± 0.1224	0.4038 ± 0.0864	0.7424 ± 0.0971	0.5663 ± 0.1017
META	GNN	0.6772 ± 0.1400	0.5752 ± 0.0984	0.8247 ± 0.0931	0.7499 ± 0.0805
BERT	RR	0.4829 ± 0.1075	0.3779 ± 0.0747	0.7424 ± 0.0967	0.6561 ± 0.0782
CNN	RR	0.5089 ± 0.0143	0.3505 ± 0.0856	0.6541 ± 0.0970	0.4622 ± 0.0901
AttnBiLSTM	RR	0.5900 ± 0.1313	0.4140 ± 0.1052	0.7943 ± 0.0871	0.6496 ± 0.0887
META	RR	0.8061 ± 0.1117	0.7306 ± 0.0877	0.9040 ± 0.0543	0.8433 ± 0.0501
BAO		0.6950 ± 0.1304	0.6058 ± 0.0972	0.8748 ± 0.0698	0.8132 ± 0.0591
MIF-FWSC		**0.8675 ± 0.0696**	**0.8098 ± 0.0586**	**0.9093 ± 0.0505**	**0.8560 ± 0.0408**
$\Delta\%$		**17.25%**	**20.40%**	**3.45%**	**4.28%**

Table 7
Accuracy comparison of different approaches on AWS. Rep denotes representation model and Cls denotes classification model. The best values are marked by bold font. $\Delta\%$ denotes the improvement of MIF-FWSC over BAO.

Method		5-way 1-shot	10-way 1-shot	5-way 5-shot	10-way 5-shot
Rep	Cls				
CNN	ROUTING	0.4330 ± 0.1234	0.3061 ± 0.0683	0.5233 ± 0.1121	0.3329 ± 0.0803
AttnBiLSTM	ROUTING	0.4790 ± 0.1189	0.3120 ± 0.0742	0.5832 ± 0.1166	0.4049 ± 0.0742
META	ROUTING	0.5762 ± 0.1369	0.4039 ± 0.0842	0.7215 ± 0.1163	0.6071 ± 0.0801
BERT	PROTO	0.4643 ± 0.1125	0.3404 ± 0.0681	0.6425 ± 0.1013	0.5203 ± 0.0702
CNN	PROTO	0.4843 ± 0.1401	0.2938 ± 0.0707	0.5819 ± 0.0958	0.4526 ± 0.0677
AttnBiLSTM	PROTO	0.4636 ± 0.1457	0.3197 ± 0.0767	0.5734 ± 0.1199	0.4476 ± 0.0767
META	PROTO	0.5617 ± 0.1455	0.3984 ± 0.0936	0.7737 ± 0.1032	0.6715 ± 0.0832
BERT	RR	0.4637 ± 0.1080	0.3427 ± 0.0703	0.7378 ± 0.1028	0.6381 ± 0.0741
CNN	RR	0.4836 ± 0.1281	0.3412 ± 0.0670	0.7214 ± 0.1018	0.6071 ± 0.0722
AttnBiLSTM	RR	0.5162 ± 0.1286	0.3745 ± 0.0827	0.6895 ± 0.1143	0.6031 ± 0.0769
META	RR	0.5881 ± 0.1369	0.4903 ± 0.0804	0.8219 ± 0.0994	0.7384 ± 0.0778
BAO		0.5427 ± 0.1290	0.4449 ± 0.0827	0.8101 ± 0.1000	0.7188 ± 0.0740
MIF-FWSC		**0.7077 ± 0.1073**	**0.6276 ± 0.0681**	**0.8306 ± 0.0919**	**0.7535 ± 0.0745**
$\Delta\%$		**16.50%**	**18.27%**	**2.01%**	**3.47%**

* Since GNN needs to construct a graph on the combination of support set and query set, and the size of query set of the AWS dataset is too large to run on GPU, we did not conduct experiments on the AWS dataset using GNN.

Table 8
Accuracy comparison in the ablation study on PW. G and L denote the global attention and the local attention, respectively.

Method	5-way 1-shot	10-way 1-shot	5-way 5-shot	10-way 5-shot
G	0.7959 ± 0.0904	0.7460 ± 0.0641	0.8667 ± 0.0669	0.8139 ± 0.0486
L	0.7434 ± 0.1088	0.6869 ± 0.0645	0.7827 ± 0.0806	0.7007 ± 0.0659
G+MLP	0.8039 ± 0.0845	0.7482 ± 0.0607	0.8779 ± 0.0601	0.8183 ± 0.0479
L+MLP	0.7651 ± 0.0991	0.7163 ± 0.0636	0.8504 ± 0.0722	0.7906 ± 0.0537
G+L+MLP	0.8105 ± 0.0884	0.7527 ± 0.0620	0.8842 ± 0.0605	0.8317 ± 0.0470
G+BiLSTM	0.8148 ± 0.0837	0.7654 ± 0.0602	0.8775 ± 0.0605	0.8194 ± 0.0505
L+BiLSTM	0.7939 ± 0.0901	0.7335 ± 0.0638	0.8582 ± 0.0740	0.7971 ± 0.0542
G+L+BiLSTM	**0.8675 ± 0.0696**	**0.8098 ± 0.0586**	**0.9093 ± 0.0505**	**0.8560 ± 0.0408**

local attention. To analyze the impact of BiLSTM, we carry out experiments by replacing BiLSTM with a fully connected network with one hidden layer (represented in MLP). Since both BiLSTM and MLP make a certain transformation to the original attention scores of words (including the global attention, the local attention, and the combination of them), we also attempt to use the original attention scores directly by removing the deep learning model.

As shown in Tables 8 and 9, we observe that:

- Both the global attention and the local attention of a word contribute to the classification performance, although the global attention has a considerable impact.
- BiLSTM is essential to classification performance since it captures some sequential information between the word attentions of service descriptions.
- In most cases, a transformation to the original attentions of words using deep learning models will improve the classification performance.

5.7. Feature convergence function (RQ4)

In MIF-FWSC, the feature convergence function converts the weighted word matrix to a description vector. To study the effect of different feature convergence functions, we carry out experiments and count the accuracy using different feature convergence functions under settings of different ways and different shots.

Table 9
Accuracy comparison in the ablation study on AWS. G and L denote the global attention and the local attention, respectively.

Method	5-way 1-shot	10-way 1-shot	5-way 5-shot	10-way 5-shot
G	0.6874 ± 0.1231	0.6051 ± 0.0623	0.8245 ± 0.0959	0.7443 ± 0.0639
L	0.6763 ± 0.1201	0.5832 ± 0.0649	0.7786 ± 0.1003	0.6839 ± 0.0724
G+MLP	0.6913 ± 0.1168	0.6158 ± 0.0648	0.8272 ± 0.0986	0.7445 ± 0.0680
L+MLP	0.6857 ± 0.1216	0.6064 ± 0.0657	0.8125 ± 0.1035	0.7335 ± 0.0709
G+L+MLP	0.7038 ± 0.1087	0.6191 ± 0.0638	0.8282 ± 0.0962	0.7477 ± 0.0700
G+BiLSTM	0.6963 ± 0.1199	0.6243 ± 0.0663	0.8302 ± 0.0989	0.7442 ± 0.0718
L+BiLSTM	0.6931 ± 0.1212	0.6049 ± 0.0639	0.8242 ± 0.0994	0.7436 ± 0.0695
G+L+BiLSTM	**0.7077** ± **0.1073**	**0.6276** ± **0.0681**	**0.8306** ± **0.0919**	**0.7535** ± **0.0745**

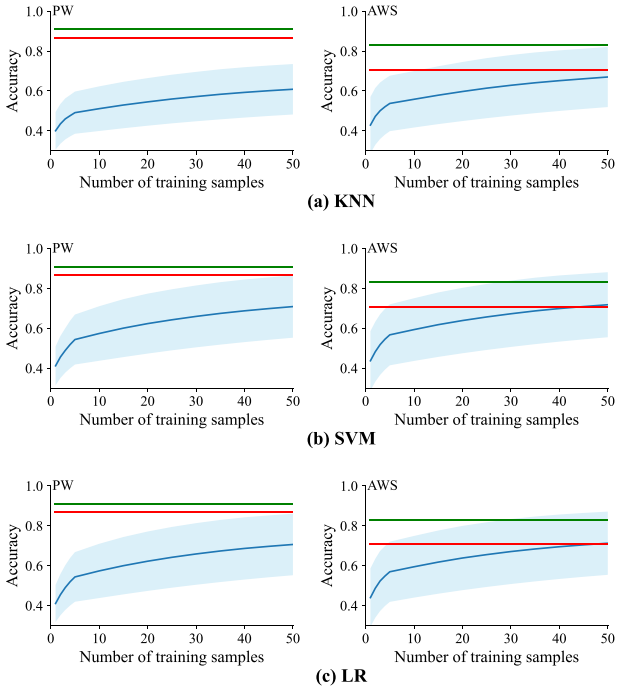


Fig. 5. Accuracy of KNN, SVM, LR, and our model. The blue line indicates the supervised models' accuracy, with standard deviation shaded. The red line is the accuracy of our model in the setting of 5-way 1-shot and the green line is the accuracy of our model in the setting of 5-way 5-shot.

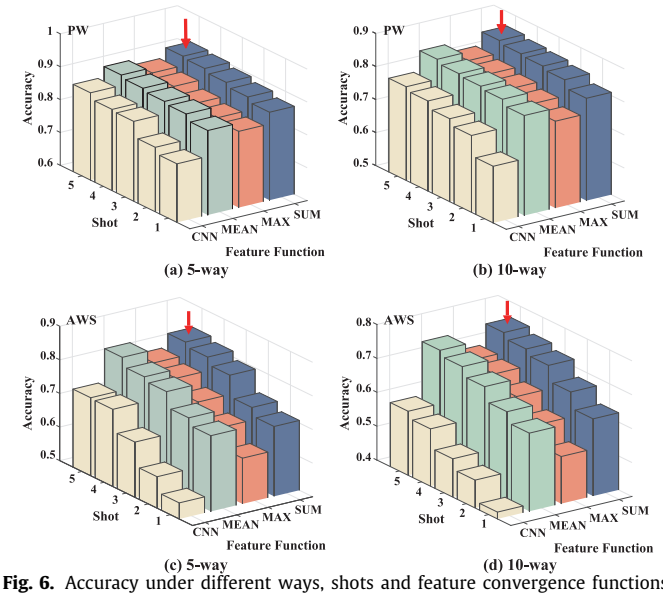


Fig. 6. Accuracy under different ways, shots and feature convergence functions. The red arrow points to the highest accuracy.

Specifically, we choose summation function (SUM), maximum function (MAX), average function (MEAN), and one-dimension convolution function (CNN) as feature convergence functions. Specifically, SUM calculates the sum of all weighted word vectors in the matrix, MAX keeps the maximum value of each dimension in all vectors, MEAN calculates the average value of all vectors, and CNN applies one-dimension convolution kernels on the weighted word matrix and then gets the representation of the description by max-pooling. For the SUM, MAX and MEAN, there are no weight parameters. For the CNN, the weight parameters are the parameters of all one-dimension convolution kernels. The size of one-dimensional convolution kernels of CNN is set to 1, 2 and 3, and the number of kernels is set to 100 for each size.

As shown in Fig. 6, SUM performs the best, the performance of MEAN is similar to SUM, followed by MAX, and CNN performs the worst under different ways and shots setting. In the few-shot service classification scenario, only limited samples are involved in each episode of the training procedure. Since CNN contains more parameters when compared to the other three functions, it is hard to learn ideal settings of parameters from the limited samples. SUM and MEAN can retain more information contained by the weighted word matrix than MAX and CNN, and they do not have complex parameters, so they perform better than others.

6. Conclusion and future work

Web service classification is crucial for service reuse. It is a time-consuming task to manually assign categories for a Web service description, especially when many new Web services are released. Existing work on service classification neglect the long-tail distribution of Web services categories, which leads to a poor classification performance on the tail categories containing only a few services. This paper treats the task as a few-shot text classification problem and designs a deep learning model named MIF-FWSC to tackle it. A series of competing experiments are conducted on two real-world datasets crawled from PW and AWS, which shows that our model achieves state-of-the-art accuracy on the few-shot Web service classification. Comparisons with traditional supervised learning also show the advantages of our model in learning transferable knowledge from training data and making predictions for unseen classes. An ablation study shows that each component of our model is necessary. Besides these, we also conducted experiments to test the effectiveness of different feature convergence functions.

In the future, we plan to improve our proposed approach by integrating more information about Web services, e.g., developers and development status, for classification. Moreover, we also plan to conduct experiments on more datasets of Web services.

CRediT authorship contribution statement

Yongqiang Liu: Methodology, Software, Writing – original draft. **Bing Li:** Conceptualization, Writing – review & editing, Supervision. **Jian Wang:** Conceptualization, Methodology, Writing – review & editing. **Duantengchuan Li:** Methodology, Writing – review & editing. **Yutao Ma:** Conceptualization, Writing – review & editing.

Declaration of competing interest

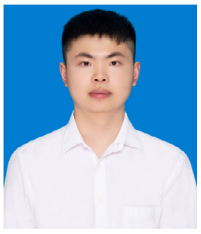
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Key R&D Program of China (No. 2018YFB1402800) and the National Natural Science Foundation of China (Nos. 62032016, 61972292, and 61832014).

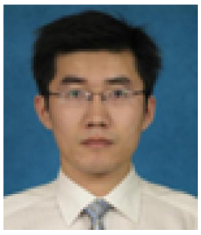
References

[1] S.J. Vaughan-Nichols, Web services: Beyond the hype, Computer (2002) 18–21.
[2] Q. He, R. Zhou, X. Zhang, Y. Wang, D. Ye, F. Chen, J.C. Grundy, Y. Yang, Keyword search for building service-based systems, IEEE Trans. Softw. Eng. (2017) 658–674.
[3] U. Qamar, R. Nisa, S. Bashir, F.H. Khan, A majority vote based classifier ensemble for web service classification, Bus. Inf. Syst. Eng. (2016) 249–259.
[4] Y. Yang, W. Ke, W. Wang, Y. Zhao, Deep learning for web services classification, in: IEEE International Conference on Web Services, Milan, Italy, 2019, pp. 440–442.
[5] J. Liu, Z. Tian, P. Liu, J. Jiang, Z. Li, An approach of semantic web service classification based on Naive Bayes, in: IEEE International Conference on Services Computing, San Francisco, CA, USA, 2016, pp. 356–362.
[6] H. Wang, Y. Shi, X. Zhou, Q. Zhou, S. Shao, A. Bouguettaya, Web service classification using support vector machine, in: 22nd IEEE International Conference on Tools with Artificial Intelligence, Arras, France, 2010, pp. 3–6.
[7] I. Katakis, G. Meditskos, G. Tsoumakas, N. Bassiliades, I.P. Vlahavas, On the combination of textual and semantic descriptions for automated semantic web service classification, in: Artificial Intelligence Applications and Innovations III, Proceedings of the 5TH IFIP Conference on Artificial Intelligence Applications and Innovations, Thessaloniki, Greece, 2009, pp. 95–104.
[8] Y. Yang, N. Qamar, P. Liu, K. Grolinger, W. Wang, Z. Li, Z. Liao, ServeNet: A deep neural network for web services classification, in: IEEE International Conference on Web Services, Beijing, China, 2020, pp. 168–175.
[9] Y. Wang, Q. Yao, J.T. Kwok, L.M. Ni, Generalizing from a few examples: A survey on few-shot learning, ACM Comput. Surv. (2020) 63:1–63:34.
[10] F. Li, R. Fergus, P. Perona, One-shot learning of object categories, IEEE Trans. Pattern Anal. Mach. Intell. (2006) 594–611.
[11] M. Fink, Object classification from a single example utilizing class relevance metrics, in: Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, Vancouver, British Columbia, Canada], 2004, pp. 449–456.
[12] B. Cao, X.F. Liu, B. Li, J. Liu, M. Tang, T. Zhang, M. Shi, Mashup service clustering based on an integration of service content and network via exploiting a two-level topic model, in: IEEE International Conference on Web Services, San Francisco, CA, USA, 2016, pp. 212–219.
[13] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 2017, pp. 1126–1135.
[14] F. Sung, Y. Yang, L. Zhang, T. Xiang, P.H.S. Torr, T.M. Hospedales, Learning to compare: Relation network for few-shot learning, in: IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 1199–1208.
[15] J. Snell, K. Swersky, R.S. Zemel, Prototypical networks for few-shot learning, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, Long Beach, CA, USA, 2017, pp. 4077–4087.
[16] R. Geng, B. Li, Y. Li, Y. Ye, P. Jian, J. Sun, Few-shot text classification with induction network, 2019, CoRR arXiv:1902.10482.
[17] S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, Long Beach, CA, USA, 2017, pp. 3856–3866.
[18] V.G. Satorras, J.B. Estrach, Few-shot learning with graph neural networks, in: 6th International Conference on Learning Representations, Vancouver, BC, Canada, Conference Track Proceedings, 2018.
[19] Y. Bao, M. Wu, S. Chang, R. Barzilay, Few-shot text classification with distributional signatures, in: 8th International Conference on Learning Representations, Addis Ababa, Ethiopia, 2020.
[20] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2019, pp. 4171–4186.
[21] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, D. Wierstra, Matching networks for one shot learning, in: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems, Barcelona, Spain, 2016, pp. 3630–3638.
[22] L. Bertinetto, J. ao F. Henriques, P.H.S. Torr, A. Vedaldi, Meta-learning with differentiable closed-form solvers, in: 7th International Conference on Learning Representations, New Orleans, la, USA, 2019.
[23] X. Wang, L. Yang, D. Wang, L. Zhen, Improved TF-IDF keyword extraction algorithm, Comput. Sci. Appl. (2013) 64–68.
[24] R. Socher, D. Chen, C.D. Manning, A.Y. Ng, Reasoning with neural tensor networks for knowledge base completion, in: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems, Lake Tahoe, Nevada, United States, 2013, pp. 926–934.
[25] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, T. Mikolov, Fasttext.zip: Compressing text classification models, 2016, CoRR arXiv:1612.03651.
[26] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations, San Diego, CA, USA, Conference Track Proceedings, 2015.



Yongqiang Liu received his Bachelor degree in Computer Science from Wuhan University, China, in 2019. He is currently a Master in the School of Computer Science, Wuhan University, China. His current research interests include services computing and software engineering.

Bing Li received the Ph.D. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2003. He is currently a Professor and the Deputy Dean of the School of Computer Science, Wuhan University. His main research areas are service computing, software engineering, artificial intelligence, complex systems, and cloud computing.



Jian Wang received his Ph.D. degree in Computer Science from Wuhan University, China, in 2008. He is currently an Associate Professor in the School of Computer Science, Wuhan University, China. His current research interests include services computing and software engineering.

Duantengchuan Li is currently pursuing the Ph.D. degree with the School of Computer Science, Wuhan University. His research interests include recommendation system, representation learning, natural language processing and their applications in services computing and software engineering.

Yutao Ma received his Ph.D. degree in Computer Science from Wuhan University, China, in 2007. He is now an Associate Professor in the School of Computer Science, Wuhan University. Dr. Ma was with the Institute of China Electronic System Engineering Corporation (Beijing) as a post-doctoral fellow and has been a visiting scholar in the Department of Electronic and Computer Engineering, Lehigh University, USA. His research focus is on the development and maintenance of large-scale software service systems.