LMA: label-based multi-head attentive model for long-tail web service classification

Guobing Zou, Hao Wu, Song Yang, Ming Jiang and Bofeng Zhang

School of Computer Engineering and Science,
Shanghai University,
Shanghai, China
Email: gbzou@shu.edu.cn
Email: woohaoshu@163.com
Email: yang65song71@126.com
Email: jimmyme@126.com
Email: bfzhang@shu.edu.cn

Yanglan Gan*

School of Computer Science and Technology,
Donghua University,
Shanghai, China
Email: ylgan@dhu.edu.cn
*Corresponding author

Abstract: With the rapid growth of web services, service classification is widely used to facilitate service discovery, selection, composition and recommendation. Although there is much research in service classification, work rarely focuses on the long-tail problem to improve the accuracy of those categories which have fewer services. In this paper, we propose a novel label-based attentive model LMA with the multi-head structure for long-tail service classification. It can learn the various word-label subspace attention with a multi-head mechanism, and concatenate them to get the high-level feature of services. To demonstrate the effectiveness of LMA, extensive experiments are conducted on 14,616 real-world services with 80 categories crawled from the service repository *ProgrammableWeb*. The results prove that the LMA outperforms state-of-the-art approaches for long-tail service classification in terms of multiple evaluation metrics.

Keywords: service classification; service feature extraction; long tail; label embedding; attention.

Reference to this paper should be made as follows: Zou, G., Wu, H., Yang, S., Jiang, M., Zhang, B. and Gan, Y. (2020) 'LMA: label-based multi-head attentive model for long-tail web service classification', *Int. J. Computational Science and Engineering*, Vol. 23, No. 2, pp.158–168.

Biographical notes: Guobing Zou is an Associate Professor and Dean of the Department of Computer Science and Technology, Shanghai University, China. He received his PhD in Computer Science from the Tongji University, Shanghai, China in 2012. He has worked as a Visiting Scholar in the Department of Computer Science and Engineering at Washington University in St. Louis from 2009 to 2011, USA. His research interests focus on service computing, edge computing, data mining, intelligent algorithms and recommender systems. He has published more than 70 papers on premier international journals and conferences, such as IEEE TSC, IEEE TCBB, *Information Sciences*, *Knowledge-Based Systems*, *Expert Systems with Applications*, *Journal of Web Services Research*, AAAI, IEEE ICWS, ICSOC and IEEE SCC. He served as an organisation chair, publicity chair and program committee member on several international conferences. He is a member of Technical Committee of Services Computing, China Computer Federation (CCF TCSC).

Hao Wu is currently a master's student in the School of Computer Engineering and Science, Shanghai University, China. He received his Bachelor's in Computer Science and Technology at Shanghai University, 2017. His research interests include service classification, and deep learning. He has published a paper on the International Conference on Service-Oriented Computing (ICSOC).

Song Yang is currently a master student in the School of Computer Engineering and Science, Shanghai University, China. He received his Bachelor's degree in Computer Science and Technology at Shanghai University, 2019. His research interests include service classification, mashup service creation, and deep learning.

Ming Jiang is currently a master student in the School of Computer Engineering and Science, Shanghai University, China. He received a Bachelor's degree in Computer Science and Technology at Shanghai University, 2017. His research interests include service recommendation, quality of service, and deep learning. He has published a paper on the *International Conference on Service-Oriented Computing (ICSOC)*.

Bofeng Zhang is a Full Professor in the School of Computer Engineering and Science at Shanghai University. He received his PhD degree from Northwestern Polytechnic University (NPU) in 1997, China. He worked as a Visiting Professor at University of Aizu from 2006 to 2007, Japan. He worked as Visiting Professor at Purdue University from 2013 to 2014, USA. His research interests include personalised service recommendation, intelligent human-computer interaction, and data mining. He has published more than 150 papers on international journals and conferences.

Yanglan Gan is an Associate Professor in the school of Computer Science and Technology, Donghua University, Shanghai, China. She received her PhD in Computer Science from Tongji University, Shanghai, China, 2012. Her research interests include bioinformatics, web services, and data mining. She has published more than 50 papers on premier international journals and conferences, including Bioinformatics, BMC Bioinformatics, IEEE TCBB, IEEE ICWS, IEEE SCC, neurocomputing, and knowledge-based systems. She is a member of Technical Committee of Bioinformatics, China Computer Federation (CCF TCBI).

1 Introduction

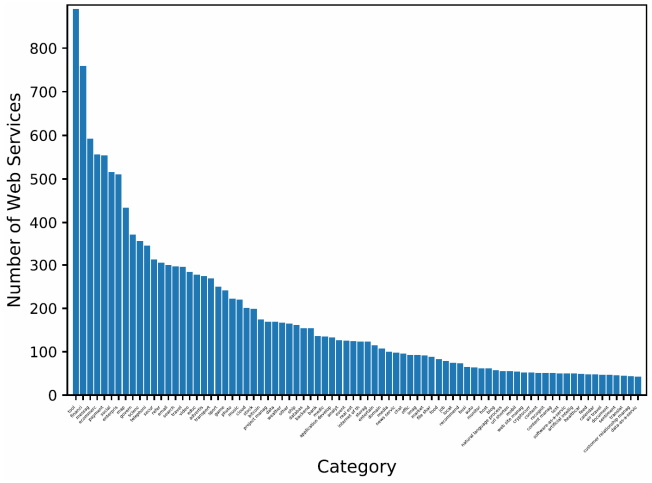
With the blossom of Web 2.0 technologies, the past decade has witnessed a rapid growth of web services and their compositions (e.g., mashups) on the internet (Cao et al., 2019). From July 2018 to December 2019, the number of registered services on the world's largest online web service repository, *ProgrammableWeb* (https://www.programmableweb.com/), increased from 17,923 to 23,038, increased by 23.5%. That means on average ten new services are registered in the repository every day. One of the reasons why web service has been able to develop so rapidly is web services provide a unified and loosely-coupled integration to reuse the heterogeneous software components (Yu et al., 2007). Software reuse is treated as a promising way to reduce the cost of software development (Yang et al., 2018), so more and more software developers are discovering reusable services from service repository.

In recent years, keyword-based methods are the first choice of service discovery in practice (Yang et al., 2018). Generally, service providers choose multiple keywords manually to describe the service so that software developers can quickly locate these services. For example, *PayPal API* is one of the services registered in *ProgrammableWeb*, the provider added four keywords to it, including one primary category *payments* and three secondary categories *ecommerce*, *financial* and *invoicing*. When a software developer implements a *transaction processing* requirement, he can use the keyword 'ecommerce, financial' to filter and finally find the service. However, the keywords assigned for the developers are not always reliable and appropriate, because manual labelling is limited by the service provider's perception. As to December 2019, the number of categories in *ProgrammableWeb* has increased to 488, such a large candidate pool makes it difficult to select several categories for a new service.

To address the challenge, machine learning methods are used to predict the keywords and recommend appropriate categories for service providers. Several works use traditional methods such as SVM (Wang et al., 2010) and LDA (Pang et al., 2019) for service classification. They have achieved good results on classification tasks of fewer than ten categories. As the number of categories increases, deep learning classification method is introduced, because traditional methods can only catch low-level features. In text classification tasks, convolutional neural network (CNN) (Kim, 2014) is often used to extract local features in a sentence and recurrent neural network (RNN) (Lai et al., 2015) is applied for classification by mining long-term dependent features. The work (Yang et al., 2018) adopts 2D CNN with bi-directional LSTM to get sentence representation and use it for classification.

Although the existing approaches can help service providers select keywords automatically, they are still limited in many service-oriented applications due to the long-tail distribution of web services among multiple categories. Anderson (2006) coined the term 'the long tail', as opposed to 20/80 rule, it is claimed that the internet makes it easier for consumers to purchase niche product (Yin et al., 2012). Taking an example in economics, success on Amazon often comes from selling in a niche with passionate buyers and fewer competitors. In service computing, *niche* mainly refers to categories that contain fewer web services. The long-tail marketing model is also appropriate for the field of service computing. It is observed that the distribution of the number of services on all categories is long-tailed. The histogram of the top 80 categories of web services is shown in Figure 1.

Figure 1 Histogram of the top 80 categories (see online version for colours)



In such case, a good service classifier needs not only to pay attention to the accuracy of classification but also to improve the diversity of classification. Therefore, the significance of service classification for service providers should include two facets:

1 help service providers find appropriate categories

2 make every category available, especially those ones with fewer services.

Since existing works primarily focus on the accuracy of the top 20% categories of services and neglect those niche ones, leading to ineffective service classification for those 80% niche categories. Upon the motivation, we investigate the problem of long-tail web service classification, and propose a novel approach named *label-based multi-head attentive* (LMA) Model for long-tail service classification.

In recent years, attention mechanism has been widely used for text feature extraction (Lin et al., 2017b; Vaswani et al., 2017). Most of these works are a self-attentive model, which means the attention information is extracted only from the text sequence. However, we observe that label embedding of web services can bring extra heuristic information, and the word-label attention can more directly promote the training of service classifier. Consequently, LMA introduces service label embedding and implements a label-attentive model to get the word-label attention. By integrating the attention vector, low-level representations abstracted to high-level features for the model training of service classifier. Furthermore, we propose a multi-head structure applied to the label-attentive model, which is informative for long-tail service classification. Additionally, to improve the overall performance of the classifier on uneven datasets, we use *focal loss* as our loss function. Extensive experiments are conducted on 14,616 services with 80 categories crawled from *ProgrammableWeb*, and the results demonstrate that LMA achieves higher classification performance than other machine learning-based methods.

To the best of our knowledge, this is the first work proposed to address the long-tail service classification problem. The main contributions of this paper are summarised as follows:

- We analyse the long-tail phenomenon in service classification. To further recommend niche categories for a new service, we propose a novel label-attentive model LMA that integrates service label embedding to attentive model for more precisely extracting abstract features and improving the accuracy in classification task of niche services.
- A multi-head structure is proposed to allow the attention model to learn information in different representation subspaces. That makes the model training in parallel and more efficient than the conventional deep learning methods.
- We perform extensive experiments on an *overall* testing set and three *niche-N* testing sets, demonstrating that our model outperforms the state-of-the-art in long-tail service classification.

The remainder of this paper is organised as follows. Section 2 illustrates the overall framework and elaborates the proposed LMA model for long-tail service classification. Section 3 shows the experimental evaluation. Section 4 reviews the related work. Finally, Section 5 concludes the paper and discusses the future work.

2 Approach

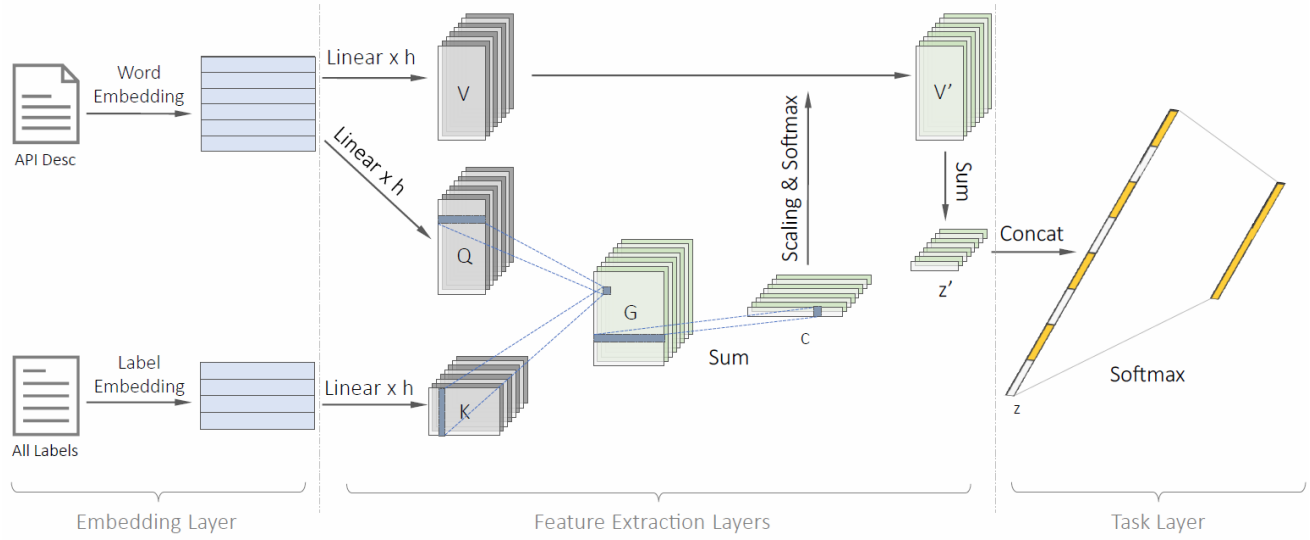
In this section, we first present the overall framework of LMA model, and then elaborate the label embedding and multi-head attention, which is followed by focal loss and model training.

2.1 The framework of approach

The framework of LMA is illustrated in Figure 2. The LMA employs a process similar to text classification, which consists of three crucial steps:

- *Embedding layer:* embedding the service descriptions and labels to their distributed representations. In this way, label embedding can learn dense label features during training and help improve the long-tail classification effect.
- *Feature extraction layers:* aggregating word embeddings and label embeddings into a fixed-length vector representation. Via these layers, low-level word representations are abstracted to high-level features, which is treated as the service representation.
- *Task layer:* using a classifier to annotate the service representation with a label. The applied classifier can be flexibly replaced according to the different applications.

Figure 2 The framework of LMA model for long-tail service classification (see online version for colours)



For the three steps, we focus on the first two steps in the following sections. In these two steps, we developed a series of techniques to make the obtained service representation more conducive to the long-tail classification problem. As for the third step, we adopt a softmax layer as the classifier, since it is concise and effective for classification tasks.

2.2 Label embedding

Since the inputted service descriptions are in the form of characters, they cannot be processed directly by the computer, so each word in the description needs to be encoded. We build a vocabulary to generate a unique numerical index for each word in the service description and represent the indexed word with one-hot encoding. Although one-hot encoding can be used as a representation of words, when the number of words in the vocabulary is large, the encoding vector is high-dimensional and sparse, which cannot be used for model training. So we use word embedding to transform one-hot vectors into distributed representations.

Word embedding is the basic building block in NLP, which can capture the regularities between words (Mikolov et al., 2013). Word embedding takes one-hot encoding as input, transforms high-dimensional vectors into low-dimensional distributed representations. The embedded vector using floating-point numbers to contain more dense information. We use random initialisation to embed the service description.

Given a service dataset $\mathbb{S} = \{(\boldsymbol{X}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$, where $\boldsymbol{X} \in \mathbb{X}$ is the service description, and $\boldsymbol{y} \in \boldsymbol{Y}$ is its corresponding label. $\boldsymbol{X} = \{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_L\}$ is a service description of length L, the word $\boldsymbol{w}_l$ is a one-hot vector in the space Δ^D, where D is the vocabulary size. Note that L is the padding length of the description sequence, we truncate long descriptions and complete short descriptions. The process of word embedding can be described as $\Delta^D \to \mathbb{R}_d^D$, where d is the dimensionality of the embedded vectors. Therefore, after embedding the one-hot vector, the embedded service description is represented as $\boldsymbol{X}' = \{\boldsymbol{x}_1', \cdots, \boldsymbol{x}_L'\}$, where $\boldsymbol{x}_l' \in \mathbb{R}_d^D$.

Several works (Tang et al., 2015; Wang et al., 2018) demonstrate that the integration of label embedding is informative for the downstream classification task. So we embed all labels into the distributed representation.

$\boldsymbol{Y}$ is the label set, and label $\boldsymbol{y} \in \boldsymbol{Y}$ is a one-hot vector in the space Δ^M, where M is the number of the service categories. Similar with word embedding, the process of label embedding can be described as $\Delta^M \to \mathbb{R}_d^M$, and the embedded labels are represented as $\boldsymbol{Y}' = \{\boldsymbol{y}_1', \cdots, \boldsymbol{y}_M'\}$, where $\boldsymbol{y}_m' \in \mathbb{R}_d^M$.

The label embedding $\boldsymbol{Y}'$ is updated iteratively during the training of the model to learn the corresponding label features. Label embedding can help enhance the compatibility of the service description and its associated category, thereby improving the accuracy of long-tail web service classification.

2.3 Multi-head attention

We propose a multi-head attention mechanism in the feature extraction layer. It consists of two procedures: first, we construct label-based attention to extract features, then we duplicate it as several copies to get the various word-label subspace attention.

Inspired by the work (Vaswani et al., 2017), we regard the service description as the query/value, the label set as the key, the attention vector $\boldsymbol{c}$ is calculated by the query and key, then $\boldsymbol{c}$ is used as the weight of value, and finally output the high-level service feature.

Specifically, we put the embedded sequence $\boldsymbol{X}'$ and embedded labels $\boldsymbol{Y}'$ into three linear layer to get the query $\boldsymbol{Q}$, value $\boldsymbol{V}$ and key $\boldsymbol{K}$:

$$\boldsymbol{Q} = relu\left(\boldsymbol{W}_Q \boldsymbol{X}' + \boldsymbol{b}_Q\right) \tag{1}$$

$$\boldsymbol{V} = relu\left(\boldsymbol{W}_V \boldsymbol{X}' + \boldsymbol{b}_V\right) \tag{2}$$

$$\boldsymbol{K} = relu\left(\boldsymbol{W}_K \boldsymbol{Y}' + \boldsymbol{b}_K\right) \tag{3}$$

where the output dimensionality of the linear layer is d_k, which is equal to the embedding dimensionality d. As a result, the shape of $\boldsymbol{Q}$ and $\boldsymbol{V}$ are both $L \times d_k$, and the shape of $\boldsymbol{K}$ is $M \times d_k$. Note that the parameters of the linear layer are independent with each other.

Then, we calculate the compatibility of label-word pairs via matrix multiplication:

$$\boldsymbol{G} = \boldsymbol{Q}\boldsymbol{K} \tag{4}$$

The shape of $\boldsymbol{G}$ is $L \times M$, where $\boldsymbol{G}_{lm}$ is the compatibility of word $\boldsymbol{w}_l$ and label $\boldsymbol{y}_m$. Based on the importance distribution of service labels on description words, we collect the largest compatibility value of each word:

$$m_l = Maxpooling\left(\boldsymbol{G}_l\right) \tag{5}$$

For the entire sentence, we obtain a vector $\boldsymbol{m}$ of length L. To generate the attention vector $\boldsymbol{c}$, $\boldsymbol{m}$ is put into a softmax layer:

$$\boldsymbol{c} = Soft\max(\boldsymbol{m}) \tag{6}$$

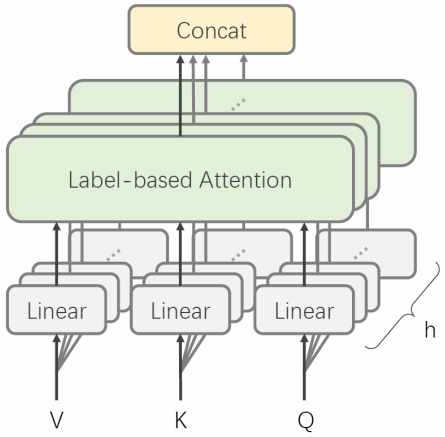
Finally, the high-level sequence representation can be summarised by weighting the value $\boldsymbol{V}$ with $\boldsymbol{c}$:

$$\boldsymbol{z} = \sum_l \boldsymbol{V}_l \boldsymbol{c}_l \tag{7}$$

where $\boldsymbol{z}$ is a vector for a service feature representation with the dimensionality of d_k.

To improve the performance and capture the word-label attention in multiple subspace, we perform a multi-head structure that is illustrated in Figure 3.

Figure 3 Label-based multi-head attention (see online version for colours)



Before generating the $\boldsymbol{Q}$, $\boldsymbol{V}$ and $\boldsymbol{K}$, we set the output dimensionality d_k as:

$$d_k = d/h \tag{8}$$

where h is the number of the heads. All the heads can be calculated parallelly, and we gained the high-level sequence representation $\boldsymbol{z}_h'$ with the number of h. The subspace features from all heads are concatenated as the service representation $\boldsymbol{z}$:

$$\boldsymbol{z} = concat\left(\boldsymbol{z}_1', \cdots, \boldsymbol{z}_h'\right) \tag{9}$$

Finally, $\boldsymbol{z}$ is used as an input of the softmax layer, which outputs the probability distribution of all categories. The category corresponding to the maximum value is selected as the classification result.

The multi-head structure is available to compute all heads in parallel, leading to more efficient extraction of high-level abstraction features of web services. Moreover, there is no additional increase in the total computational cost, since the dimension in multi-head structure is partitioned from single-head attention with full dimensionality.

2.4 Focal loss and model training

When training the LMA, we adopt the variant cross-entropy loss function taking into account long-tail distributions of web services. Cross-entropy is commonly used in classification tasks, which measures the difference between the true label distribution and the predicted one. Since ordinary cross-entropy loss function does not perform well on imbalanced service distributions among multiple service categories, we use the *focal loss* (Lin et al., 2017a) as loss function during the model training. Focal loss adds a weight based on the cross-entropy loss, thus it can penalise negative labelled service samples. The focal loss is formally defined as follows:

$$L = -\alpha_t \left(1 - p_t\right)^{\gamma} \log\left(p_t\right) \tag{10}$$

where p is the output of the task layer, and given the true label y,

$$p_t = \begin{cases} p & y = 1 \\ 1 - p & \text{otherwise} \end{cases} \tag{11}$$

Besides, there are two hyper-parameters in equation (10), where γ is a focusing parameter for modulating factor $(1 - p)$, and

$$\alpha_t = \begin{cases} \alpha & y = 1 \\ 1 - \alpha & \text{otherwise} \end{cases} \tag{12}$$

where α is the value between 0 and 1 to balance the positive and negative service samples. Since the formal focal loss function is mainly appropriate for binary classification, we fix the function in the multi-class long-tail service classification task by scaling the outputs from the task layer, where the service category probabilities of each sample sum to 1.

$$L_{fixed} = \sum_{i=1}^{M} -\alpha \left(1 - p_i\right)^{\gamma} y_i \log\left(p_i\right) \tag{13}$$

where M is the number of service categories. For those misclassified samples, the fixed focal loss function can increase the penalty value of categories containing fewer services while reducing the penalty value of categories containing more services.

We conduct *Adam* optimiser (Kingma and Ba, 2014) instead of stochastic gradient descent (SGD) to optimise the parameters during the model training for parameter optimisation in LMA model. Adam can adjust different learning rates for the different parameters, and update those with more frequent changes in smaller steps. Furthermore, we use the *early stopping* method to determine whether model training converges. It can monitor a certain metric and the model training is finished, when its values no longer rises any more.

3 Experimental evaluation

In this section, we first introduce the service dataset, and then illustrate the experimental results of LMA model, which includes the comparison with other machine learning methods as well as our own variant methods. Finally, we show the classification performance in terms of parameter impact.

3.1 Experimental setup and dataset

The experiment is implemented using Tensorflow 1.14 and Python 3.7.4. All the models are trained and tested on our workstation, which is equipped with NVIDIA GTX 1080 Ti, Intel Xeon Gold 6130 and 192 GB RAM.

The experimental data comes from web API platform *ProgrammableWeb*, which is the largest online RESTful service repository. We implemented a python crawler script with *scrapy* to collect services and use MySQL to manage the web services. As of 1 July 2018, this dataset contains 17,923 web services with 481 categories.

To validate the performance of LMA on the long-tail service classification problem, we use the top 80 categories with the most common web services as our dataset. The filtered dataset contains 14,616 services with 80 categories. After counting the services in each category, we find that *tools* contains 890 services, which has the maximum number of services, while the minimum category is *data-as-a-service*, which contains only 42 services. From Figure 1, it can intuitively see that the number of services involved in each category is rather uneven, indicating long-tailed distribution of web services.

Table 1 The comparison results of data selection methods

Data partitioning method	*Accuracy*			
	Overall	*Niche-20*	*Niche-30*	*Niche-40*
Random selection	0.5510	0.4544	0.3274	0.3010
Random selection by category (SMOTE)	0.5163	0.4431	0.3512	0.3329
Random selection by category	*0.5521*	*0.4943*	*0.4226*	*0.3995*

In the preprocessing of web services in the experiments, we filter the service description with stopwords and delete useless information in square brackets of several services, which indicates that these services are outdated. In addition, we do stem extraction on the service description that is effective as it may lessen redundancy of the stemmed words and their inflected words.

Because the service dataset is small and imbalance, we take dataset partitioning methods in the experiments. We apply three data selection methods, including random selection, random selection by category and oversampling with the smote algorithm. The three datasets are divided in a seven-to-three ratio, that is, the training set accounts for 70% of the total samples and the testing set accounts for 30%. The result of the dataset partitioning is shown in Table 1. The overall, niche-20, niche-30 and niche-40 are four testing sets for the long-tail service classification problem. We observe that random selection by category performs better than random selection, because it has the ability to divide training and testing sets evenly across all service categories. As for the SMOTE method, when the number of service categories is increased to 80, the number of web services in the training set is oversampled to nearly five times the original. It may lead to the model underfit on most common categories. Consequently, the dataset is divided using random selection by category. The training set and testing set contains 10,231 and 4,385 web services, respectively.

In order to verify the performance on the long-tail service classification problem, we further redistribute the testing set into four subsets. The basal testing set contains all the 80 categories services, which we call it overall. Based on the overall testing set, we delete the top 20, 30, and 40 categories of services. Therefore, we generate the three datasets named niche-20, niche-30 and niche-40, which contains 60, 50 and 40 categories, respectively. It helps us understand the long-tail service classification performance among competing methods. In the experiments, we use them to train and test the proposed model as well as other machine learning models. The number of web services per testing set is shown in Table 2.

3.2 Evaluation metrics

We evaluate the long-tail service classification performance from the perspective of accuracy and area under curve (AUC). Accuracy is a basic indicator for measuring the performance of a classifier, AUC is the probability that a random positive sample has a higher score than a random negative one, which measures the ranking ability of the model (Fawcett, 2006). It is defined as follows:

$$AUC = \frac{1}{u^+ u^-} \sum_{x^+ \in D^+} \sum_{x^- \in D^-} R\left(f(x^+), f(x^-)\right) \tag{14}$$

where D^+ is the set of all positive instances with size u^+, D^- is the set of all negative instances with size u-, $f(\cdot)$ is the prediction model and $R: \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a function to

calculate the ranking score of the input pair, which is defined as follows:

$$R(x_1, x_2) = \mathbb{I}(x_1 > x_2) + \frac{1}{2}\mathbb{I}(x_1 = x_2) \quad (15)$$

where $\mathbb{I}(\cdot)$ is the indicator function. In the case of unbalanced samples, AUC can better reflect the overall classification performance. It is a real number ranging from 0 to 1, where a higher value indicates better performance of a classifier.

Table 2 The distribution of training and testing dataset of web services

Training set	*Testing set*			
	Overall	*Niche-20*	*Niche-30*	*Niche-40*
10,231	4,385	1,835	1,233	811

3.3 Competing methods

To demonstrate the performance of the proposed method, we compare LMA with nine machine learning-based methods, including seven traditional models: CNN, LSTM, GRU, BiLSTM, CLSTM (Zhou et al., 2015), FastText (Joulin et al., 2016) and TextCNN (Kim, 2014), and two state-of-the-art models: LEAM (Wang et al., 2018) and ServeNet (Yang et al., 2018). To more deeply analyse the impact of the two components of LMA on the performance of the long-tail service classification, we implement our model into four variants: LMA, LMA (focal loss), LMA (multi-head), LMA (multi-head; focal loss). The details of the competing methods as well as our variant ones are described as below.

- *CNN:* it obtains phrase features through convolution, then the max-pooling layer extracts sentence features that are used for service classification.
- *LSTM:* it regards text data as sequence type, and can capture long-term dependencies of a sentence for service classification.
- *GRU:* it is a variant of LSTM, which can acquire long-term and short-term memory features for service classification through a simpler structure.
- *BiLSTM:* considering LSTM can only pass sequence information backwards, it concatenates a forward LSTM and a backward LSTM to extract more comprehensive features for service classification.
- *CLSTM (Zhou et al., 2015):* it utilises CNN to extract a sequence of higher-level phrase representations, which are fed into a long short-term memory recurrent neural network (RNN) (LSTM) to obtain the sentence representation. It can capture both local features of phrases as well as global and temporal sentence semantics for service classification.
- *FastText (Joulin et al., 2016):* it is a simple and fast classifier proposed by Facebook. It can achieve good service classification performance through simple linear combination of features.
- *TextCNN (Kim, 2014):* it uses convolution kernels of different sizes to extract key information in the sentence, and uses max-pooling to select the most influential high-dimensional features for service classification.
- *LEAM (Wang et al., 2018):* it introduces an attention framework that measures the compatibility of embeddings between text sequences and labels for service classification.
- *ServeNet (Yang et al., 2018):* it is the state-of-the-art method in service classification. It adopts 2D CNN with bi-directional LSTM rather than 1D CNN with one-directional LSTM for service feature extraction.
- *LMA:* it is our fundamentally self-developed method. Sentence features are extracted by single-head attention, and categorical cross-entropy is calculated as the loss function during model training.
- *LMA (focal loss):* it is our self-developed variant for service classification. Sentence features are extracted by single-head attention, and focal loss is used for parameter optimisation in model training.
- *LMA (multi-head):* it is our self-developed variant for service classification. Sentence features are extracted by multi-head attention, where it consists of five heads and categorical cross-entropy is used as loss function for model training.
- *LMA (multi-head; focal loss):* it is our self-developed variant with the best performance of service classification. Sentence features are extracted by multi-head attention, where it consists of eight heads and focal loss is applied for parameter optimisation in model training.

3.4 Experimental results and analysis

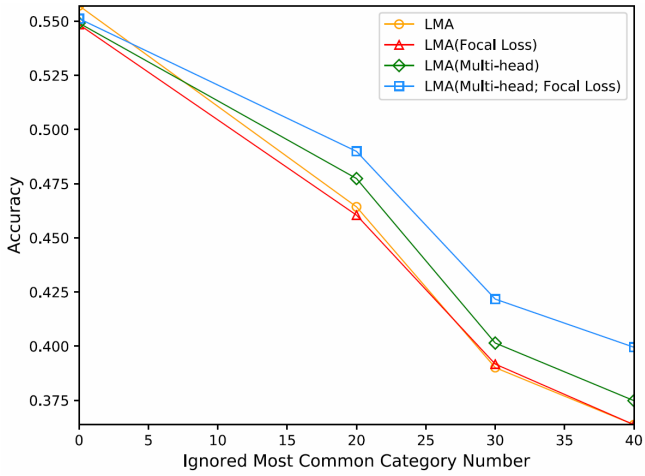
In this section, a set of experiments on a large-scale service dataset are conducted to validate the effectiveness of our proposed approach. Once a model is trained, we test it on all of the four testing sets. Results on the overall dataset reflect the basic performance of service classification, whereas results on niche-N datasets show its advantages on long-tail service classification.

Table 3 The experimental results of long-tail web service classification compared with conventional methods

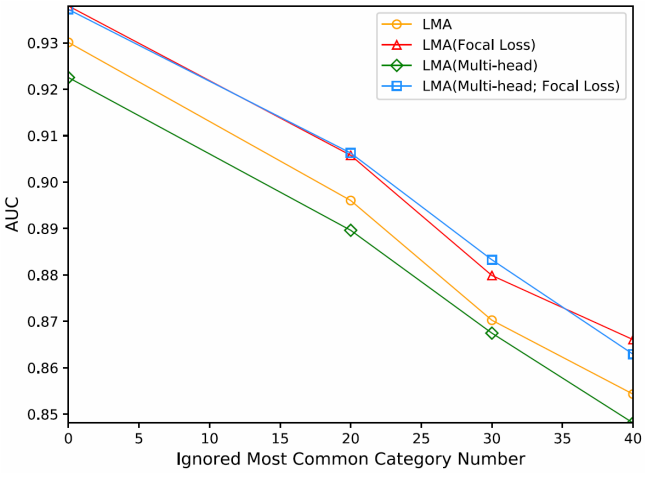
Method	*Overall*		*Niche-20*		*Niche-30*		*Niche-40*	
	ACC	*AUC*	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*	*AUC*
LSTM	0.4520	0.8404	0.3461	0.7932	0.2506	0.7604	0.2109	0.7433
GRU	0.5161	0.8777	0.4218	0.8450	0.3228	0.8091	0.2799	0.7879
BiLSTM	0.5298	0.9065	0.4447	0.8875	0.3512	0.8658	0.3033	0.8529
CLSTM	0.4990	0.8811	0.3973	0.8418	0.3025	0.8109	0.2663	0.8022
FastText	0.5473	0.9341	0.4431	*0.9089*	0.3552	*0.8825*	0.3083	*0.8715*
TextCNN	*0.5777*	0.9117	0.4796	0.8821	0.3942	0.8532	0.3453	0.8277
CNN	0.5238	0.8873	0.4327	0.8622	0.3682	0.8413	0.3329	0.8258
LEAM	0.5389	0.9081	0.4425	0.8700	0.3674	0.8409	0.3687	0.8256
ServeNet	0.5637	0.8804	0.4867	0.8535	0.4161	0.8272	0.3872	0.8217
LMA*	0.5521	*0.9364*	*0.4943*	0.9051	*0.4226*	0.8815	*0.3995*	0.8609

Note: LMA* is our best self-developed variant, LMA (multi-head; focal loss).

Figure 4 The experimental results on performance of long-tail web service classification among four self-developed methods, (a) accuracy (b) AUC (see online version for colours)



(a)



(b)

3.4.1 Comparison with conventional methods

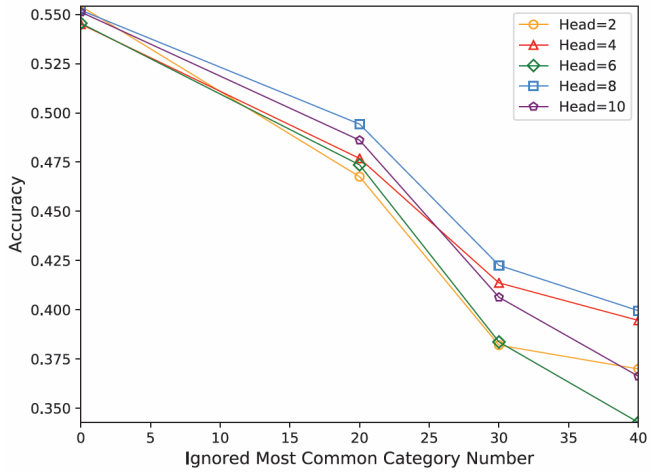
We compare our proposed LMA model with nine conventional methods for web service classification. The experimental results are shown in Table 3. For the classification accuracy, although TextCNN and ServeNet can get higher accuracy with 57.77% and 56.37% on the overall testing set, they perform poorly on the niche-N dataset. It is observed that our proposed model LMA with multi-head and focal loss can achieve the highest test accuracy on all the three niche-N testing sets. LMA has the classification accuracy with 49.43%, 42.26% and 39.95% on niche-20, niche-30 and niche-40, while the state-of-the-art method ServeNet receives 48.67%, 41.61% and 38.72%, respectively. The results prove that LMA is better than other machine learning based methods in long-tail service classification. As for AUC, LMA also reaches superior results among all benchmarks. Note that FastText has a good performance on AUC, and it is even slightly better than our method. However, its accuracy is much lower than LMA, which is unsuitable for long-tail service classification.

3.4.2 Comparison with self-developed methods

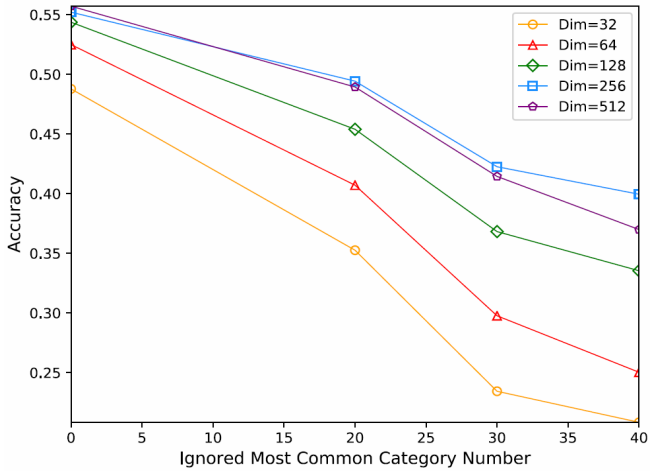
To further validate the capability of various components of our proposed model, we compare LMA (multi-head; focal loss) with three self-developed variants: LMA, LMA (focal loss) and LMA (multi-head). Note that LMA (multi-head) consists of five heads to achieve the optimal results.

The results show that variants with focal loss function obtain a higher AUC, which indicates that focal loss can improve the service classifier for uneven datasets. For those variants with multi-head attention, we find that the classification accuracy on niche-N testing set can be significantly improved, which demonstrates that multi-head attention is able to capture the features of niche services. When both multi-head and focal loss are used for LMA model, it reaches the best performance for long-tail service classification task.

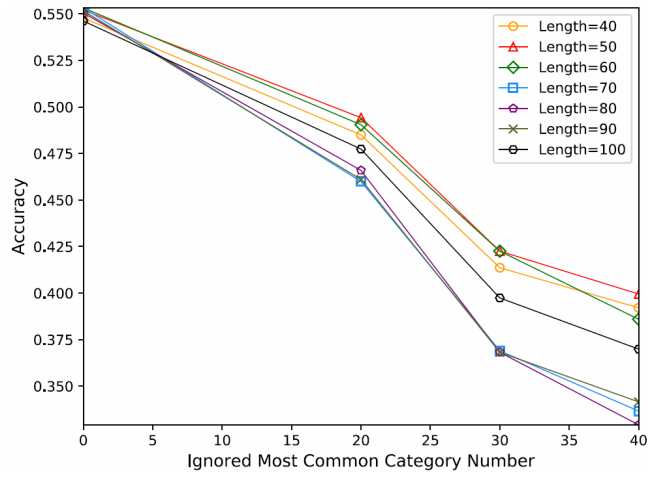
Figure 5 The performance impact on service classification accuracy along with the changes of different hyper-parameters (h, d, L), (a) multi-head number (b) embedding dimensionality (c) padding length (see online version for colours)



(a)



(b)



(c)

3.5 *Performance impact of hyper-parameters*

In the experiments, two groups of hyper-parameters influence the performance of long-tail service classification in out LMA model.

1 The quality of the service feature extraction of LMA is directly related to three hyper-parameters, including the number of multi-head attention h, the dimensionality of word and label embedding d, and the padding length of service description L.

2 α and γ are both the hyper-parameters in focal loss function, which impact the performance and need to be adjusted accordingly in different variants of our LMA model.

To find the best setting, we test the two groups of hyper-parameters and analyse how they impact the performance of long-tail service classification in our LMA model.

Figure 6 The performance impact on service classification accuracy along with the changes of different hyper-parameters (α, γ) (see online version for colours)

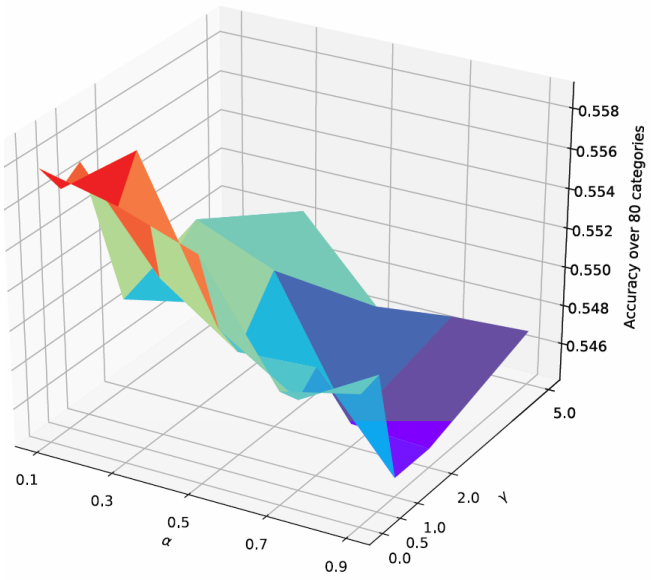


Figure 5 illustrates the performance impact on service classification accuracy along with the changes of h, d and L. The results as shown in Figure 5(a) demonstrate that the LMA can learn the long-tail attention subspace better when the number of the head is set as 8. In Figure 5(b), we can see the embedding dimensionality has a significant impact on service classification accuracy. More specifically, as the dimensionality increases, the accuracy of LMA on the niche-N dataset is accordingly improved. When d is set as 256, it receives the best classification accuracy. Since the padding length of the service description sequence plays an important role in service classification, we perform a set of experiments on different padding lengths. In Figure 5(c), it is observed that LMA achieves the best accuracy on long-tail service classification when L is set as 50. Padding

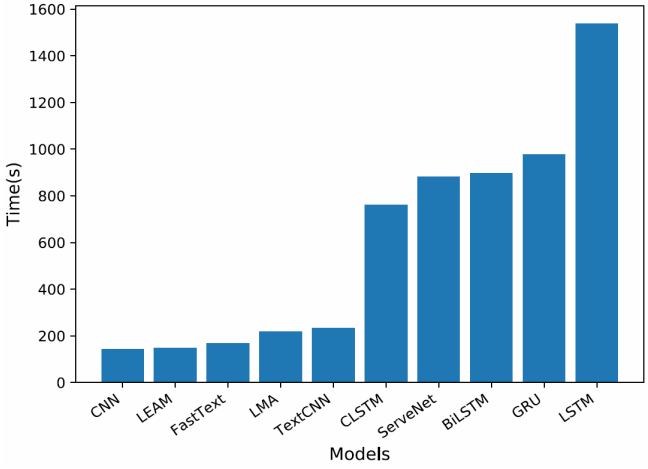
too long may cause the inputs to contain a lot of meaningless placeholder, which decreases the accuracy of long-tail service classification.

The effectiveness of focal loss is affected by two parameters, and each model has its unique best combination. To find the best combination of LMA, we test the model on the overall testing set. Figure 6 shows the performance impact on service classification accuracy with different combinations of α and γ. Along with the increase of α and γ, the classification accuracy gradually declines. When $\alpha = 0.3$ and $\gamma = 0.5$, LMA performs best for long-tail service classification.

3.6 Time overhead

To validate the efficiency of different service classification methods, we test the convergence speed on the training set. The histogram of comparison results is illustrated in Figure 7. It can be clearly observed that the training time of these models is divided into two levels: CNN, LEAM, FastText, LMA, and TextCNN can converge with approximately 200 seconds, while CLSTM, ServeNet, BiLSTM, GRU, and LSTM consume almost 4 times longer. Since CLSTM, ServeNet, BiLSTM, GRU and LSTM are based on the implementation of RNN, the calculation of the current moment depends on the result of the previous moment. As a result, they cannot be trained in parallel like CNN, which take them longer time to make the model converged.

Figure 7 The convergence speed of model training among different service classification methods (see online version for colours)



Typically, the state-of-the-art method ServeNet converges at 6 epochs in 881 seconds. As for our method, LMA has a network structure similar to transformer (Vaswani et al., 2017), which can process all words in parallel by stacking different activation functions to achieve fast convergence speed. In the experiment, our method LMA converges at 7 epochs in 218 seconds.

4 Related work

In recent years, service classification has been widely investigated and it fundamentally promotes advanced service-oriented applications, including service discovery, selection, composition and recommendation. Several works (Aznag et al., 2014; Liu et al., 2016; Pang et al., 2019) classify services with the LDA model, which is a classical classification model. The work (Shi et al., 2017) introduced a multi-label active learning approach for web service tag recommendation by learning the correlations among labels. Actually, since the service descriptions are text sequences, many semantic methods of service classification are based on natural language processing (NLP) techniques. The emergence of cloud computing has driven the transformation of service delivery and consumption. Furthermore, everything-as-a-service (XaaS) connects all things efficiently and flexibly by service. Benefiting from XaaS and service classification, much work has dedicated to the internet-of-things (IoT) research (Liang et al., 2019c, 2019b, 2019a) and changed people's lives.

Text classification has been widely explored based on NLP techniques. To reduce the cost of manual labelling, the work (Tong and Koller, 2001) proposed a pool-based active learning method with support vector machine (SVM) for text classification. Some researches work on attentive text classification. The work (Du et al., 2017) proposed a framework that combines RNN with a CNN-based attention model to capture the salient parts of sentences to improve the performance of text classification. The Google Brain group stacks several scaled dot-product attention in both the encoder and the decoder to build a self-attention model called the transformer (Vaswani et al., 2017) to improve the performance of text classification.

Moreover, label-based feature extraction gradually emerges as mainstream techniques for text classification. The predictive text embeddings (PTE) (Tang et al., 2015) use label embeddings to extract the relationship between words and labels. Label embedding attentive model (LEAM) (Wang et al., 2018) combined the label embedding into an attention model, which accelerates the learning speed and achieves high accuracy in text classification tasks.

Considering the classification tasks with the long-tail characteristics, the OLTR algorithm developed in the work (Liu et al., 2019) handled the imbalanced classification in computer vision, which integrates a dynamic meta-embedding to improve the classification accuracy on several long-tail distributed datasets.

Motivated by the above investigations, we aim at solving the problem of long-tail service classification and propose a novel LMA model.

5 Conclusions

To solve long-tail service classification task, we propose a novel approach called LMA model by integrating label embedding into multi-head attentive model, which can more

accurately extract service features for service classification with unbalanced service dataset. To demonstrate the performance of LMA, we conduct extensive experiments on a real-world service dataset. The results demonstrate that LMA outperforms the state-of-the-art methods for long-tail service classification in both accuracy and AUC. In the future, we continue developing more advanced approach based on the deep learning models to further improve the performance of service classification.

Acknowledgements

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

References

Anderson, C. (2006) *The Long Tail: why the Future of Business is Selling Less of More*, Hachette Books, Hyperion.

Aznag, M., Quafafou, M. and Jarir, Z. (2014) 'Multilabel learning for automatic web services tagging', *International Journal of Advanced Computer Science and Applications*, Vol. 5, No. 8, pp.182–191.

Cao, Y., Liu, J., Shi, M., Cao, B., Chen, T. and Wen, Y. (2019) 'Service recommendation based on attentional factorization machine', in *IEEE International Conference on Services Computing (SCC)*, IEEE, pp.189–196.

Du, J., Gui, L., Xu, R. and He, Y. (2017) 'A convolutional attention model for text classification', in *National CCF Conference on Natural Language Processing and Chinese Computing (NLPCC)*, Springer, pp.183–195.

Fawcett, T. (2006) 'An introduction to roc analysis', *Pattern Recognition Letters*, Vol. 27, No. 8, pp.861–874.

Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T. (2016) *Bag of Tricks for Efficient Text Classification*, arXiv preprint arXiv: 1607.01759.

Kim, Y. (2014) *Convolutional Neural Networks for Sentence Classification*, arXiv preprint arXiv: 1408.5882.

Kingma, D.P. and Ba, J. (2014) *Adam: A Method for Stochastic Optimization*, arXiv preprint arXiv: 1412.6980.

Lai, S., Xu, L., Liu, K. and Zhao, J. (2015) 'Recurrent convolutional neural networks for text classification', in *Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pp.2267–2273.

Liang, W., Long, J., Weng, T-H., Chen, X., Li, K-C. and Zomaya, A.Y. (2019a) 'TBRS: a trust based recommendation scheme for vehicular CPS network', *Future Generation Computer Systems*, Vol. 92, pp.383–398.

Liang, W., Tang, M., Long, J., Peng, X., Xu, J. and Li, K-C. (2019b) 'A secure fabric blockchain-based data transmission technique for industrial internet-of-things', *IEEE Transactions on Industrial Informatics*, Vol. 15, No. 6, pp.3582–3592.

Liang, W., Xie, S., Long, J., Li, K-C., Zhang, D. and Li, K. (2019c) 'A double PUF-based RFID identity authentication protocol in service-centric internet of things environments', *Information Sciences*, Vol. 503, pp.129–147.

Lin, T-Y., Goyal, P., Girshick, R., He, K. and Dollár, P. (2017a) 'Focal loss for dense object detection', in *IEEE International Conference on Computer Vision (ICCV)*, pp.2980–2988.

Lin, Z., Feng, M., dos Santos, C.N., Yu, M., Xiang, B., Zhou, B. and Bengio, Y. (2017b) *A Structured Self-Attentive Sentence Embedding*, arXiv preprint arXiv: 1703.03130.

Liu, X., Agarwal, S., Ding, C. and Yu, Q. (2016) 'An LDA-SVM active learning framework for web service classification', in *IEEE International Conference on Web Services (ICWS)*, IEEE, pp.49–56.

Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B. and Yu, S.X. (2019) 'Large-scale long-tailed recognition in an open world', in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.2537–2546.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. (2013) 'Distributed representations of words and phrases and their compositionality', in *Advances in Neural Information Processing Systems (NIPS)*, pp.3111–3119.

Pang, S., Zou, G., Gan, Y., Niu, S. and Zhang, B. (2019) 'Augmenting labeled probabilistic topic model for web service classification', *International Journal of Web Services Research*, Vol. 16, No. 1, pp.93–113.

Shi, W., Liu, X. and Yu, Q. (2017) 'Correlation-aware multi-label active learning for web service tag recommendation', in *IEEE International Conference on Web Services (ICWS)*, IEEE, pp.229–236.

Tang, J., Qu, M. and Mei, Q. (2015) 'PTE: predictive text embedding through large-scale heterogeneous text networks', in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, pp.1165–1174.

Tong, S. and Koller, D. (2001) 'Support vector machine active learning with applications to text classification', *Journal of Machine Learning Research*, November, Vol. 2, pp.45–66.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017) 'Attention is all you need', in *Advances in Neural Information Processing Systems (NIPS)*, pp.5998–6008.

Wang, G., Li, C., Wang, W., Zhang, Y., Shen, D., Zhang, X., Henao, R. and Carin, L. (2018) *Joint Embedding of Words and Labels for Text Classification*, arXiv preprint arXiv: 1805.04174.

Wang, H., Shi, Y., Zhou, X., Zhou, Q., Shao, S. and Bouguettaya, A. (2010) 'Web service classification using support vector machine', in *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, pp.3–6.

Yang, Y., Ke, W., Liu, P., Wang, W., Shen, B., Liu, B. and Zhao, Y. (2018) *ServeNet: A Deep Neural Network for Web Service Classification*, arXiv preprint arXiv: 1806.05437.

Yin, H., Cui, B., Li, J., Yao, J. and Chen, C. (2012) 'Challenging the long tail recommendation', *Proceedings of the VLDB Endowment*, Vol. 5, No. 9, pp.896–907.

Yu, Y., Lu, J., Fernandez-Ramil, J. and Yuan, P. (2007) 'Comparing web services with other software components', in *IEEE International Conference on Web Services (ICWS)*, IEEE, pp.388–397.

Zhou, C., Sun, C., Liu, Z. and Lau, F. (2015) *A CLSTM Neural Network for Text Classification*, arXiv preprint arXiv: 1511.08630.