A Novel Dual-Graph Convolutional Network based Web Service Classification Framework

Xin Wang[1], Jin Liu[1]✉, Xiao Liu[2], Xiaohui Cui[3], Hao Wu[4]✉
[1]*School of Computer Science, Wuhan University, China*
[2]*School of Information Technology, Deakin University, Australia*
[3]*School of Cyber Science and Engineering, Wuhan University, China*
[4]*School of Information Science and Engineering, Yunnan University, China*
{xinwang0920, jinliu}@whu.edu.cn, xiao.liu@deakin.edu.au, xcui@whu.edu.cn, haowu@ynu.edu.cn

***Abstract*—Automated service classification is the foundation for service discovery and service composition. Currently, many existing methods extracting features from functional description documents suffer the problem of data sparsity. However, beside functional description documents, the Web API ecosystem has accumulated a wealth of information that can be used to improve the accuracy of Web service (API) classification. At the moment, there is an absence of a unified way to combine functional description documents with other sources of information (e.g., attributes, interactions and external knowledge) accumulated in the Web API ecosystem for API classification. To address this issue, we present a dual-GCN framework that can effectively suppress the noise propagation of textual contents by distinguishing functional description documents and other sources of information (specifically Mashup-API co-invocation patterns by default in this paper) for API classification. This framework is extensible with the ability to include different sources of information accumulated in the Web API ecosystem. Comprehensive experiments on a real-world public dataset demonstrate that our proposed method can outperform various representative methods for API classification.**

***Keywords*-Web Service; API Classification; Web API Ecosystem; Graph Convolutional Network; Knowledge Graph**

I. Introduction

Web APIs are basic building blocks in software applications enabling automated interactions among heterogeneous components. As a new paradigm, Web APIs significantly reduce software development costs by providing storage services, message services, location services, and so on. In the era of big data, Web service is a resource with great value-added potential, which has become a common concern in academia and industry. With the rapid development of the digital economy, various kinds of Web services have been published to meet user requirements. Therefore, how to efficiently select Web APIs satisfying user requirements is a very important task in the field of service-oriented computing. Accurate service classification can greatly reduce the search space of Web services, thus effectively promoting the discovery and composition of Web services.

In recent years, researchers in the field of service computing have made significant efforts on Web service classification. Many methods have been proposed to extract useful information from functional description documents for service classification [1], [2]. For example, the LDA topic model [3] is utilized to obtain the topic distributions of Web services. Then Web services are classified by calculating the similarity among them based on topic vectors. With the rise of deep learning, some researchers try to classify Web services using deep learning techniques [4], [5]. They firstly represent words as numeric vectors through the word embedding technology, and then use CNN-based or RNN-based neural networks to extract implicit features for service classification. However, these methods mainly use functional description documents to classify services, which are often vulnerable to data sparsity. If description documents are insufficient, these methods will not be able to achieve the desired results. Based on the data analysis of the ProgrammableWeb[1], which is the largest online Web service registry, we observe that functional description documents of Web services are often short, sparse, with little information or even missing texts.

Recently, knowledge graph (KG) has been widely used in many scenarios such as text classification [6] and movie recommendation [7] due to its ability to contain fruitful information. Knowledge graph is a type of heterogeneous graph consisting of plenty of entity-relation-entity triplets. The Web API ecosystem has accumulated a wealth of knowledge that can be used to enhance the accuracy of service classification [8]. To the best of our knowledge, the use of other sources of information (interactions, attributes and external knowledge) accumulated in the Web API ecosystem for service classification is still limited. In particular, there is an absence of a unified way to combine functional description documents with other sources of information for API classification. Recently, some existing works have utilized dual neural networks (such as dual-CNN [9], dual-LSTM [10] etc.) to extract features from different perspectives. In

✉ Corresponding author.

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

978-1-7281-8786-0/20/$31.00 ©2020 IEEE
DOI 10.1109/ICWS49710.2020.00043

Authorized licensed use limited to: Wuhan University. Downloaded on October 08,2021 at 01:39:21 UTC from IEEE Xplore. Restrictions apply.

addition, if the description documents of APIs use the same words which have no practical meaning as other entities in the Web API ecosystem, the single-graph evolution will generate noises through the propagation of word co-occurrence information. As more useful information accumulated in the Web API ecosystem is introduced, it is difficult for single-graph evolution to decide what information to use or not. How to exploit key knowledge instead of all the knowledge to benefit service classification needs to be further investigated [11].

Inspired by these, we propose a unified and extensible Web service classification framework. Specifically, we firstly use a graph to capture functional description documents of APIs and then we construct another graph to capture the most useful knowledge (namely Mashup-API co-invocation patterns by default in this paper) accumulated in the Web API ecosystem, which can be flexibly extended to include other useful knowledge. It is worth mentioning that dual-graph evolution can isolate API description documents from other entities (Mashup documents by default in this paper) to prevent word-level noise propagation. Because the noise generated by capturing relationships between APIs and other entities through word co-occurrence is far greater than the benefit, especially when more other information is introduced. Finally, we use dual Graph Convolution Networks (GCN) [12] consists of two parallel branches which respectively capture word co-occurrence information, document-word relationships in API description documents and entity association information for joint training of parameters. In this way, we can effectively address the problem of data sparsity, providing a unified way to combine functional description documents with other sources of information in the API ecosystem, avoiding word-level noise propagation and achieving better service classification performance.

The main contributions of this paper can be summarized as follows:

- We propose a dual-GCN based Web service classification framework that can effectively suppress noise propagation of textual contents by distinguishing functional description documents and Mashup-API co-invocation patterns accumulated in the Web API ecosystem.
- Our proposed service classification framework is extensible, and it can introduce more useful knowledge (such as interactions, attributes and others) accumulated in the Web API ecosystem for service classification.
- We conduct comprehensive experiments on a real-world public dataset, which demonstrates that our proposed method can not only achieve better service classification accuracy but also have stronger robustness than various representative methods.

The rest of the paper is organized as follows. Section II provides some backgrounds about the Web API ecosystem and graph convolutional network. Section III reviews some recent related studies. Section IV presents the details of our proposed method. Section V demonstrates the experimental results. Finally, section VI concludes our paper and point out some future research directions.

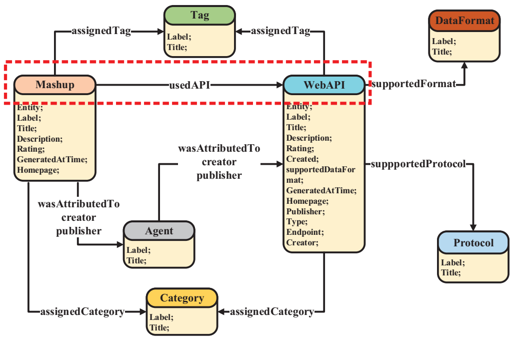


Figure 1. A full schema of knowledge graph in the Web API ecosystem. The rex dotted area denotes the co-invocation relationships between Mashups and APIs.

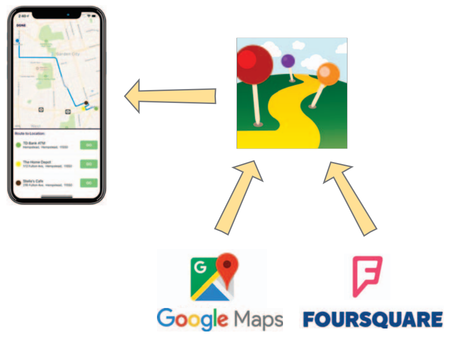


Figure 2. A sample schema of Mashup: the $OnTheWay$ application integrates the Google Maps API and the Foursquare API to provide route optimization services.

II. Background

A. Web API Ecosystem and Mashup-API Co-invocation

In the past few years, the number of Web APIs has increased dramatically. For example, as of February 2020, more than 23,000 Web APIs have been published on Programmableweb, an increase of about 80% from five years ago [13]. Nevertheless, the lack of semantic description of the Web APIs seriously affects their discovery, sharing and integration. To deal with this issue, Dojchinovski et al. have presented the *Linked Web APIs dataset*[2] including provenance, temporal, technical, functional, and non-functional aspects of Web APIs [8], which is shown in Figure 1. The red dotted area denotes the co-invocation relationships between Mashups and APIs.

A Mashup application is usually created by combining multiple Web APIs containing different functionalities. Ac-

[2]http://linked-web-apis.fit.cvut.cz/

Authorized licensed use limited to: Wuhan University. Downloaded on October 08,2021 at 01:39:21 UTC from IEEE Xplore. Restrictions apply.

cordingly, a co-invocation pattern can be found among these Web APIs. For a more intuitive understanding of Mashup-API co-invocation patterns, Figure 2 shows a Mashup scenario for a mobile application called *OnTheWay*[3]. Basically, *OnTheWay* can help travelers to create quick and easy personalized road trips. It integrates the *Google Maps API*[4] and *Foursquare API*[5] to facilitate an effective route optimization based on geographic location and public comments shared in social platform. Specifically, the *Google Maps API* provides basic functions related on maps for searching and marking specific locations, while the *Foursquare API* utilizes social capabilities focusing on public comments among friends on specific locations. By combining advantages of two types of APIs, *OnTheWay* can provide users with satisfying road trips and improve user experience. For example, a user can easily find a restaurant with a high public comment rate nearby.

B. Graph Convolution Networks

Graph Neural Networks have received increasing attentions in recent years. Kipf et al. [12] extended the mature neural network model (such as CNN) to graphs of arbitrary structure, and proposed a simplified graph neural network model, named Graph Convolutional Network (GCN). GCN can obtain the embedding vectors of entities through the adjacency features between entities, which can effectively capture the higher-order structure information in a heterogeneous graph. Specifically, we define an undirected graph $G(V, E)$, $V(|V| = N)$ as a set of N nodes, and E as a set of edges. We assume that all nodes are self-connected. Then, we use $A \in \mathbb{R}^{N\times N}$ and D to represent the adjacency matrix and degree matrix of G respectively, where $Dii = \sum_j Aij$. We assume that $X \in \mathbb{R}^{N\times M}$ is a feature matrix of all entities, where M is the dimension of the feature vectors. By stacking multiple layers of GCN, higher-order features can be obtained. The propagation rule of a multilayer GCN network is as follows:

$$L_{(l+1)} = \rho\left(D^{-\frac{1}{2}} A D^{-\frac{1}{2}} L_{(l)} W_{(l)}\right) \tag{1}$$

Where l indicates the number of layers. $W_{(l)}$ represents the trainable weight matrix of the l^{th} layer. $\rho(.)$ denotes an activation function and $L_{(0)} = X$.

III. Related Work

The effective and accurate classification of Web services is a basic research issue. To this end, researchers have invested a lot of efforts. General speaking, existing works on Web service classification can be divided into two categories including one based on conventional machine learning methods, and the other based on deep learning methods.

[3]http://www.onthewayapp.com

[4]https://developers.google.com/maps/

[5]https://api.foursquare.com/

A. Conventional Machine Learning based Methods

Liu et al. [3] leveraged SVM as the basic classifier, then they utilized latent dirichlet allocation (LDA) to reduce the feature dimension to solve the sparsity problem and improve the efficiency of service classification. Faced with the rich knowledge accumulated in the web ecosystem, Liang et al. [14] constructed a heterogeneous network with multiple types of relationships and used the RWR (random walk with restart) model to capture the degree of association between all types of entities. However, as the number of entities increases, the time cost will become very large. Liu et al. [2] proposed a semantic Web service classification method based on Naive Bayes. They used three stages of bayesian classification to classify semantic Web services in terms of service interfaces and execution capabilities. Finally, they used the information gain theory to determine the classification effects of different features.

B. Deep Learning based Methods

As deep learning technology shines in various fields, Yang et al. [4] proposed a deep neural network framework stacked 2-D CNN and Bi-LSTM to capture features of API description documents in small regions and sequences. Cao et al. [1] used Bi-LSTM to automatically learn about features of Web services. Then they used topic vectors of Web service documents obtained through offline training to enhance topic attention processings of Web services. Ye et al. [5] firstly used a wide learning model to make breadth predictions for Web services, then they used a Bi-LSTM model to mine semantic information of Web service documents to perform depth predictions. Finally, all the prediction results of breadth and depth are integrated into the final result of service classification by a linear regression algorithm.

IV. Dual-GCN API Classification Framework

In this section we will introduce details of our proposed Dual-GCN Web service classification framework.

A. Co-invocation Graph Component

We construct a co-invocation relationship graph according to the historical co-invocation records between Mashups and APIs. The number of nodes in the co-invocation graph is the sum of the number of Mashups and APIs. We assign weights of edges based on co-invocation relationships. Specifically, for source node i and target node j, the number of links from i to j can be defined as $\eta(i, j)$. Similarly, k in $\eta(i, k)$ denotes the node connected to i. In our refined knowledge graph (Mashup-API co-invocation graph), each entity has at most one link with another entity, while in the Web API ecosystem, there may be multiple links between two entities due to other sources of information. Therefore, to enable the model for the inclusion of more useful knowledge in the

Authorized licensed use limited to: Wuhan University. Downloaded on October 08,2021 at 01:39:21 UTC from IEEE Xplore. Restrictions apply.

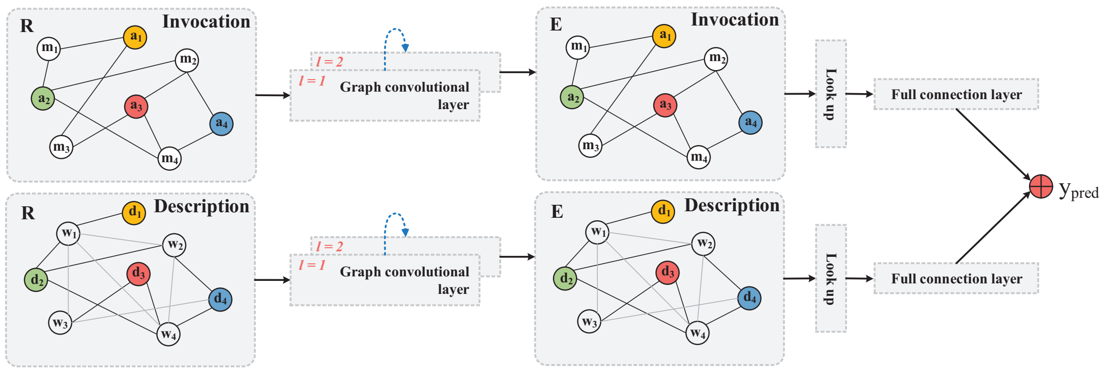


Figure 3. A sample illustration of the proposed Dual-GCN model. Nodes begin with "a" are API nodes, nodes begin with "m" are Mashup nodes, nodes begin with "d" are description document nodes, nodes begin with "w" are word nodes. The API and its description document have the same color. Different colors mean different API and document classes. For ease of distinction, in the description graph, gray lines are word-word edges, black bold lines are document-word edges. A multi-layer GCN learns features from R raw channels to E feature embedding channels.

future, the weight assignment between Mashups and APIs is given by matrix A^1:

$$A^1_{ij} = \begin{cases} \dfrac{\eta(i,j)}{\sum_k \eta(i,k)} & if \quad \eta(i,j) > 0, \\ 0 & otherwise \end{cases} \tag{2}$$

We initialize the feature matrix $X^1 = I^1$ ($I^1 \in \mathbb{R}^{N^1 \times N^1}$, is a identity matrix, where N^1 is the sum of the number of Mashups and APIs) as the input of the GCN.

B. Text Graph Component

The creation of a text graph is similar to co-invocation graph except that the number of nodes in the graph is the sum of the number of unique words and documents. In addition, the weight distribution between nodes needs to be reconfigured [6]. Term frequency-inverse document frequency (TF-IDF) is a widely used measure to reflect the contribution of a single word to a document. Point-wise mutual information (PMI) is often used to measure the correlation between two variables. In this component, we leverage TF-IDF and PMI respectively to define weights for document-word edges and word-word edges only considering positive PMI values. The weight assignment between node i and node j is initialized as follows:

$$A^2_{ij} = \begin{cases} \text{TF-IDF}_{ij} & i \text{ is document, } j \text{ is word} \\ \text{PMI}(i,j) & i,j \text{ are words, } \text{PMI}(i,j) > 0 \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

We remove words with a frequency lower than 5 (by default). We set up a fixed-size (10 by default) window to capture the word co-occurrence information. We initialize the feature matrix $X^2 = I^2$ ($I^2 \in \mathbb{R}^{N^2 \times N^2}$, is a identity matrix, where N^2 is the sum of the number of word nodes and document nodes) as the input of GCN.

C. Learning the Proposed Model

After initializing two adjacency matrixes (A^1 and A^2) and two feature matrixes (X^1 and X^2), training steps of the model are executed from left to right as shown in Figure 3.
Graph Convolutional Layer. Two or more layers of GCN allow non-adjacent information exchange, although there are no direct Mashup-Mashup, API-API and document-document edges in the graph. The propagation rules of two-layer GCN models are defined by the following equations:

$$Z^1 = f(X^1, \hat{A}^1) = \hat{A}^1 \ \rho(\hat{A}^1 X^1 W^1_{(0)}) W^1_{(1)} \tag{4}$$

$$Z^2 = f(X^2, \hat{A}^2) = \hat{A}^2 \ \rho(\hat{A}^2 X^2 W^2_{(0)}) W^2_{(1)} \tag{5}$$

Here, we can calculate $\hat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ ahead of time. $W_{(0)}$ and $W_{(1)}$ are learnable weight parameter matrixes. ρ is an activation function, e.g. a ReLU $\rho(x)$ =max$(0, x)$.
Look Up Layer. Look up layer is a linear layer that does not involve any parameter learning. The created graphs contain different types of entities such as Mashup entities and API entities in co-invocation graph, word entities and API document entities in text graph. In our task, we only care about feature vectors of APIs (API entities and API description documents). In the look up layer, we filter out the corresponding feature vectors based on the index list of APIs, e.g., $a = (a_1, a_2, a_3...a_n)$ and the index list of API documents, e.g., $d = (d_1, d_2, d_3...d_n)$, where n denotes the number of APIs. The index of the API entity corresponds to the index of the document, e.g., $a_1 \rightarrow d_1$. Feature matrixes of API entities and API documents are obtained by:

$$V^1 = Z^1[a] \tag{6}$$

$$V^2 = Z^2[d] \tag{7}$$

where V^1 denotes the feature matrix of API entities. V^2 denotes the feature matrix of API documents.

Authorized licensed use limited to: Wuhan University. Downloaded on October 08,2021 at 01:39:21 UTC from IEEE Xplore. Restrictions apply.

Full Connection Layer. We combine features extracted from the two components by matrix addition. Then the result is placed into a full connection layer (f_{fc}):

$$O = f_{fc}(V^1 \bigoplus V^2) \tag{8}$$

where $\bigoplus$ denotes the element-wise addition. $O \in \mathbb{R}^{n \times c}$ denotes the output of the full connection layer, c is the number of categories.

Output Layer. Finally, the result of Web API classification is obtained through a $softmax$ activation function:

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{9}$$

$$Y = softmax(O) \tag{10}$$

where Y denotes all predictions results. The loss function is defined as the cross-entropy error of all labeled APIs:

$$Loss(Y, T) = -\sum_{n_i=1}^{n} \sum_{c_j=1}^{c} T_{n_i c_j} \ln Y_{n_i c_j} \tag{11}$$

where T is the label indicator matrix. All the parameters in the model are trained via gradient descent.

V. Evaluation

We conduct comprehensive experiments to evaluate our proposed method for Web service classification. Specifically, three Questions need to be answered:

1) Q1: Whether our model of using dual-graph evolution performs better than models only using single-graph evolution for Web service classification?
2) Q2: What is the performance of our proposed method in comparison with some representative methods for Web service classification?
3) Q3: Can our proposed method achieve satisfactory results on limited labeled data?

Table I
Statistics of dataset.

Item Type	Statistics
Number of Mashups	7,415
Number of Web APIs	11,339
Number of Mashups participating in co-invocation	7,328
Number of Web APIs participating in co-invocation	1,366
Mashup-API co-invocation matrix density	1.3646×10^{-4}
Number of Mashup used in co-invocation graph	7,094
Number of Web API used in co-invocation graph	1,236
Mashup-API co-invocation matrix density (new)	1.4096×10^{-4}

A. Dataset Statistics and Preparation

We evaluate our method using the *Linked Web APIs dataset*. It consists of over half million of RDF triplets, which contains 11,339 Web APIs, 7,415 Mashups and nearly 7,717 Mashup profiles of developers crawled from ProgrammableWeb. Entities (Mashups, APIs, publishers, creators, homepage et al.) and various types of relationships

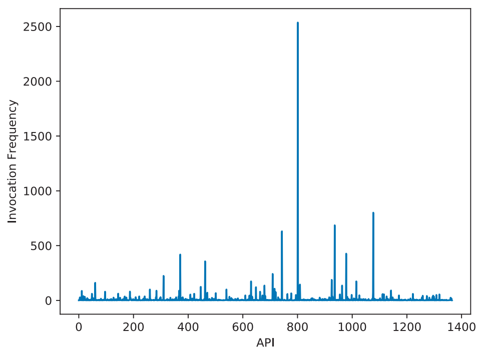


Figure 4. The distribution of co-invocation frequency for APIs.

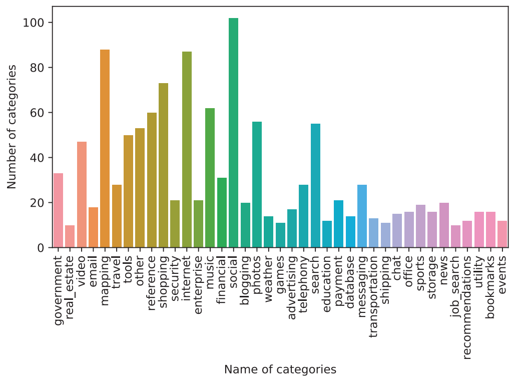


Figure 5. The number distribution of APIs in different categories.

(*usedAPI*, *assignedTag*, *assignedCategory* et al.) between them are described with a unique URL. Most of Mashups and APIs only have a short description document.

We further explore the dataset, which reveals two main characteristics in Mashup-API co-invocation patterns:

- **Sparsity.** We observe that only about 12% of APIs have been included once or more in the history of Mashup developments. This is the reason behind the low density of Mashup-API co-invocation matrix. A Mashup development contains only a limited number of APIs. The average number of APIs used per Mashup is 2.1. More specifically, 92.4% of Mashups use fewer than 5 APIs.
- **Imbalance.** We find that existing Mashups invoke Web APIs at an uneven rate. Some popular APIs are invoked very frequently. For example, the *Google Map API* has been invoked more than 2,000 times in the dataset. On the contrary, some less popular APIs are invoked less than 5 times. Figure 4 shows the frequency distribution of APIs which have been invoked in Mashup developments. We further find out that Top-100 popular APIs are involved in 80.8% of Mashup developments.

In our experiment, we remove Mashups and APIs without

Authorized licensed use limited to: Wuhan University. Downloaded on October 08,2021 at 01:39:21 UTC from IEEE Xplore. Restrictions apply.

description documents and APIs in small categories with less than 10 APIs same as [4]. After that, we remove Mashups without invoking APIs. Finally, the dataset contains 7,094 Mashups and 1,236 APIs with 39 categories. Due to the small size and imbalance of the dataset, randomly splitting the dataset cannot make the training set and testing set follow the same distribution in small categories. We randomly select 80% of APIs in each category for training, and the remaining 20% of APIs in each category for testing same as [4]. Finally, the dataset was divided into 1002 training APIs and 234 testing APIs. Some basic statistics of the dataset are given in Table I. The category distribution of APIs is shown in Figure 5. For API functional description documents, we remove punctuations, stop words and transform all words to their root forms using the lemmatizer in the NLTK toolkit[6].

B. Representative Methods for Comparison

We select various representative methods and some state-of-the-art methods mentioned in related work for comparisons with Top-1 accuracy, Top-5 accuracy and F1-macro metrics. Specifically, we adopt 6 conventional machine learning based methods including AdaBoost (AB), K-NearestNeighbor (KNN), Logistic Regression (LR), Random Forest (RF), LDA-L-SVM [3] and (TS-NB) [2]. We also adopt 5 deep learning-based methods for Web service classification including CNN [15], C-LSTM [16], R-CNN [17], TA-BiLSTM [1], ServerNet [4]. In addition, we also compare our dual-graph evolution model with three single-graph evolution models, e.g., T-GCN [6] (Text Graph), I-GCN (Invocation Graph) and C-GCN (Comprehensive Graph) containing all information in one graph, to show the advantages of our proposed method. In C-GCN, we use two different adjacency matrices to represent the text relationships and invocation relationships respectively. C-GCN-w means sharing parameters, C-GCN-non-w means not sharing parameters.

C. Experimental Parameters and Environment

In our proposed method, the two-layer GCN contains 256 hidden units and 128 hidden units respectively, and the full connection layer contains 39 hidden units. An Adam optimizer with a learning rate of 0.001 is used. In order to avoid overfitting, we add a dropout layer between every two layers, with a drop probability of 0.5. We set L_2 loss weight as 4×10^{-5}. For graph-based method, any pre-trained word embedding is not needed. For text-based methods, we exploit 200-dimensional GloVe[7]word vectors for text representation. We utilize the sklearn library[8] to implement conventional machine learning algorithms with optimal parameters. We implement all deep learning methods based on PyTorch[9].

[6]http://www.nltk.org/

[7]https://nlp.stanford.edu/projects/glove/

[8]https://scikit-learn.org

[9]https://pytorch.org/

D. Performance of Web API Classification

In the experiments, we use the Top-N accuracy metric to evaluate the performance of all methods. Top-1 accuracy can reflect the precision of the method. However, for multi-category tasks (39 categories in our task), Top-5 accuracy can reflect the performance of the Top-5 category list containing real results. In addition, we evaluate the performance of methods by F1-macro, which is a harmonic mean of precision and recall. Our task can be boiled down to the classification of small-size, sparse, and severely unbalanced data. To further explore the robustness of our method, we test the model performance with different proportions of training data. Table II shows the experimental results of Top-1 accuracy, Top-5 accuracy and F1-score with 25%, 50% and 100% of the original dataset for training. We calculate a measure reduction from using 100% to 25% of the dataset to reflect the sensitivity of the model to the size of labeled data. The optimal results are marked in bold. The underlined values are the suboptimal results.

1) Discussion of Experimental Results for Q1: T-GCN has a strong ability to capture global word co-occurrence and document-word information achieving satisfactory results. Currently, the Web ecosystem has accumulated a wealth of knowledge (interactions and attributes) that can be utilized to benefit service classification. I-GCN captures implicit features of Mashup-API co-invocation patterns using graph convolutional network, and obtains comparable results. C-GCN combines description documents with co-invocation patterns by the single-graph evolution achieving better performance. Our method, called Dual-GCN, combines co-invocation patterns with description documents by dual-graph evolution, and obtains best results as shown in Table II. Compared with C-GCN, our method has achieved at least 5.8%, 1.7% and 5.0% improvement in Top-1 accuracy, Top-5 accuracy and F1-score metrics when using all training data. Our proposed method can effectively isolate API description documents from other entities to prevent word-level noise propagation. It proves that our proposed dual-graph evolution method can perform better than single-graph evolution methods on Web service classification.

2) Discussion of Experimental Results for Q2: In the conventional machine learning methods, LDA-L-SVM and KNN achieve better performance. Most deep learning-based methods do not perform well on small dataset as shown in the experimental results. Among them, the performances of CNN, C-LSTM and R-CNN are very close and low. TA-BiLSTM uses topic vectors to enhance topic attention processings of Web services, thereby further improving the classification performance. ServerNet extracts the features of more regions by stacking 2-CNN and BiLSTM to obtain better classification results, better than CNN, C-LSTM, R-CNN and TA-BiLSTM. T-GCN and I-GCN leverage description documents and co-invocation patterns respectively achieving

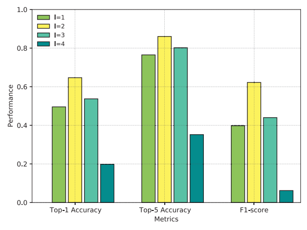
Authorized licensed use limited to: Wuhan University. Downloaded on October 08,2021 at 01:39:21 UTC from IEEE Xplore. Restrictions apply.

Table II
TOP-1 ACCURACY, TOP-5 ACCURACY AND F1-MACRO OF OUR METHOD COMPARED WITH VARIOUS REPRESENTATIVE METHODS

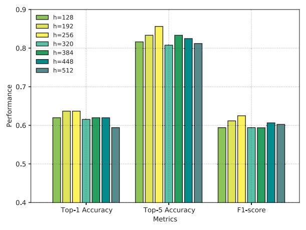
Methods	⋆100%			50%			⋆25%			**⋆Sensitivity**		
	Top-1	Top-5	F1	Top-1	Top-5	F1	Top-1	Top-5	F1	Top-1	Top-5	F1
AB	0.3077	0.5265	0.2368	0.2179	0.4726	0.1569	0.1880	0.4145	0.1125	-38.9%	-21.3%	-52.5%
KNN	0.5598	0.7863	0.5260	0.5043	0.7479	0.4368	0.3761	0.6239	0.2723	-32.8%	-20.7%	-48.2%
LR	0.4274	0.6923	0.2773	0.3590	0.5769	0.1746	0.2521	0.5214	0.1300	-41.0%	-24.7%	-53.1%
RF	0.4872	0.7863	0.4436	0.4701	0.7179	0.3748	0.3803	0.6410	0.2637	-21.9%	-18.5%	-40.6%
LDA-L-SVM	0.5347	0.8413	0.4355	0.4189	0.6888	0.2577	0.3015	0.5343	0.1987	-43.6%	-36.5%	-54.4%
TS-NB	0.4289	0.6456	0.3034	0.3217	0.5343	0.2161	0.2676	0.4811	0.1798	-37.6%	-25.5%	-40.7%
CNN	0.3547	0.6376	0.2646	0.2752	0.5068	0.1364	0.1598	0.4060	0.0625	-54.9%	-36.3%	-76.4%
C-LSTM	0.3393	0.6256	0.2577	0.2692	0.5402	0.1695	0.2385	0.5017	0.1431	-29.7%	-19.8%	-44.5%
R-CNN	0.4162	0.7000	0.3622	0.3256	0.5855	0.2129	0.2368	0.5265	0.1480	-43.1%	-24.8%	-59.1%
TA-BiLSTM	0.4397	0.7185	0.3847	0.3477	0.5976	0.2369	0.2573	0.5448	0.1771	-41.5%	-24.2%	-54.0%
ServerNet	0.4586	0.7369	0.4087	0.3668	0.6139	0.2610	0.2811	0.5531	0.1902	-38.7%	-24.9%	-53.5%
T-GCN	0.5897	0.8034	0.5755	0.5214	0.7463	0.4565	0.4373	0.6994	0.4003	-25.8%	-12.9%	-30.4%
I-GCN	0.4402	0.7350	0.3248	0.4274	0.6652	0.2940	0.3632	0.6456	0.2621	-17.5%	-12.2%	**-19.3%**
C-GCN-w	0.6111	0.8462	0.5927	0.5556	0.7906	0.5084	**0.5128**	0.7436	0.4241	**-16.1%**	-12.1%	-28.4%
C-GCN-non-w	0.5855	0.8248	0.5774	0.5598	0.8077	0.5007	0.4744	0.7265	0.4208	-19.0%	-11.9%	-27.1%
Dual-GCN	**0.6468**	**0.8604**	**0.6222**	**0.5812**	**0.8205**	**0.5605**	0.4786	**0.7608**	**0.4621**	-26.0%	**-11.6%**	-25.7%
Gains	+5.8%	+1.7%	+5.0%	+3.8%	+1.6%	+10.2%	-6.7%	+2.3%	+9.0%	-	-	-

competitive results. C-GCN uses both description documents and co-invocation patterns in single-graph achieving suboptimal results. Dual-GCN performs best and outperforms all representative methods.

3) Discussion of Experimental Results for Q3: From the Table II, we can clearly see that Dual-GCN obtains higher Top-1 accuracy, Top-5 accuracy and F1-macro with limited labeled data. For example, Dual-GCN respectively obtains test accuracies of 47.86%, 76.08% and 46.21% in Top-1 accuracy, Top-5 accuracy and F1-macro with only 25% of dataset for training, which is even higher than some representative methods using 100% of training data. Some representative methods vary greatly with the size of the training dataset. On a quarter of the training data, Top-5 accuracy of Dual-GCN only decreases by 11.6% compared to using all training dataset, which demonstrates the robustness of our proposed method.



(a)



(b)

Figure 6. Impact of number of convolutional layers and hidden units.

E. Impact of Parameters Settings

1) Impact of the Number of Convolutional Layers: The number of convolutional layers determines the ability to capture higher-order features from adjacency entities in the graph. We study the impact of the number of convolutional layers l for Web service classification. The number of convolutional layers l ranges from 1 to 4. Figure 6(a) represents the experimental results of all metrics for Web service classification. More convolutional layers does not bring about performance improvement. From the figure, we can see that the optimal setting of l is 2 in all metrics, indicating that a 2^{nd} order information propagation in our constructed graphs is sufficient.

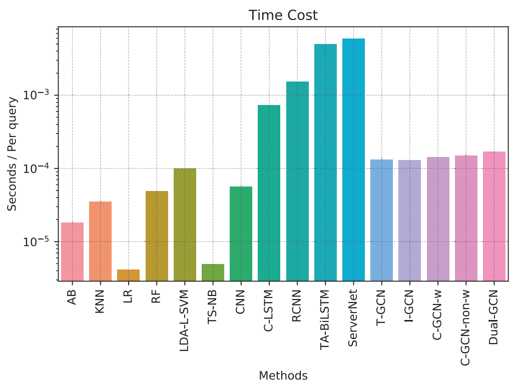


Figure 7. Average computational overheads of all methods, where y-axis is in logarithmic scale.

2) Impact of the Number of hidden units: Similar to most neural networks, the architecture of our model is designed to follow the tower pattern. In our method, a two-layer GCN, we set the number of hidden units in the first layer to h and the number of hidden units in the second layer to $\frac{h}{2}$. The number of hidden units h varies in the set $\{128, 192, 256, 320, 384, 448, 512\}$. From Figure 6(b), we can see that the optimal setting of the number of hidden units is close to 256. This means that a larger number of hidden units is not always benefical for the improvement of classification accuracy and a moderate number of hidden units is sufficient.

Authorized licensed use limited to: Wuhan University. Downloaded on October 08,2021 at 01:39:21 UTC from IEEE Xplore. Restrictions apply.

F. Computational Overhead

To investigate the computational overhead of service classification in all methods, we conduct experiments on a laptop equipped with i5-8300H CPU, GTX1050Ti GPU and 16-GB memory. All methods are implemented in Python. From the average computational overhead of each sample in Figure 7, we can see that our method obtains a competitive time cost. When compared with C-LSTM, RCNN, TA-BiLSTM and ServerNet, the time cost is greatly reduced, but the classification performance is improved.

VI. Conclusion and Future Work

This paper presented a novel unified and extensible Dual-Graph Convolutional Network based Web service classification framework which can combine functional description documents and other sources of information (Mashup-API co- invocation patterns by default in this paper) accumulated in the Web API ecosystem. Specifically, we defined a dual-GCN model to extract features of API documents and Mashup-API co-invocation patterns respectively to prevent word-level noise propagation. This service classification framework is flexible and extensible with the ability to include other useful information (interactions, attributes and external knowledge) accumulated in Web API ecosystem. Comprehensive experiments on a real-world public dataset successfully demonstrated that our method can not only achieve better service classification accuracy but also have stronger robustness than various representative methods.

As part of our future work, we will further explore other elements such as *DataFormat, Rating* and *Protocol* in the Web API ecosystem to enhance the proposed method. Meanwhile, since the GCN model does not consider the order of words, we will attempt to introduce sequential features into our method in the future.

Acknowledgment

This work was supported by the grands of the National Natural Science Foundation of China (Nos.61972290, U163620068, 61962061, 61562090).

References

[1] Y. Cao, J. Liu, B. Cao, M. Shi, Y. Wen, and Z. Peng, "Web services classification with topical attention based bi-lstm," in *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Springer, 2019, pp. 394–407.

[2] J. Liu, Z. Tian, P. Liu, J. Jiang, and Z. Li, "An approach of semantic web service classification based on naive bayes," in *2016 IEEE International Conference on Services Computing (SCC)*. IEEE, 2016, pp. 356–362.

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

[4] Y. Yang, W. Ke, W. Wang, and Y. Zhao, "Deep learning for web services classification," in *2019 IEEE International Conference on Web Services (ICWS)*. IEEE, 2019, pp. 440–442.

[5] H. Ye, B. Cao, Z. Peng, T. Chen, Y. Wen, and J. Liu, "Web services classification based on wide & bi-lstm model," *IEEE Access*, vol. 7, pp. 43 697–43 706, 2019.

[6] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 7370–7377.

[7] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 950–958.

[8] M. Dojchinovski and T. Vitvar, "Linked web apis dataset," *Semantic Web*, vol. 9, no. 4, pp. 381–391, 2018.

[9] J. Pan, S. Liu, D. Sun, J. Zhang, Y. Liu, J. Ren, Z. Li, J. Tang, H. Lu, Y.-W. Tai, and M.-H. Yang, "Learning dual convolutional neural networks for low-level vision," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3070–3079.

[10] Y. Tay, M. C. Phan, L. A. Tuan, and S. C. Hui, "Learning to rank question answer pairs with holographic dual lstm architecture," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 695–704.

[11] X. Wang, H. Wu, and C.-H. Hsu, "Mashup-oriented api recommendation via random walk on knowledge graph," *IEEE Access*, vol. 7, pp. 7651–7662, 2018.

[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[13] L. Yao, X. Wang, Q. Z. Sheng, B. Benatallah, and C. Huang, "Mashup recommendation by regularizing matrix factorization with api co-invocations," *IEEE Transactions on Services Computing*, 2018.

[14] T. Liang, L. Chen, J. Wu, and A. Bouguettaya, "Exploiting heterogeneous information for tag recommendation in api management," in *2016 IEEE International Conference on Web Services (ICWS)*. IEEE, 2016, pp. 436–443.

[15] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[16] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A c-lstm neural network for text classification," *arXiv preprint arXiv:1511.08630*, 2015.

[17] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015, pp. 2267–2273.

Authorized licensed use limited to: Wuhan University. Downloaded on October 08,2021 at 01:39:21 UTC from IEEE Xplore. Restrictions apply.