SC-GAT: Web Services Classification Based on Graph Attention Network

Mi Peng[1], Buqing Cao[1(✉)], Junjie Chen[1], Jianxun Liu[1], and Bing Li[2]

[1] School of Computer Science and Engineering & Hunan Provincial Key Laboratory for Services Computing and Novel Software Technology, Hunan University of Science and Technology, Xiangtan, China
pengm12138@qq.com, buqingcao@gmail.com, hnust_cjj@163.com, ljx529@gmail.com

[2] School of Computer, Wuhan University, Wuhan, China
bingli@whu.edu.cn

Abstract. The classification of Web services with high similarity is conducive to the promotion for service management and service discovery. With the increasing number of Web services, how to accurately and efficiently classify the Web services becomes an urgent and challenging task. Although the existing methods achieve significant results in the task for service classification via integrating the structure information of service network with the content features of service node, it fails to discriminate the importance of neighbor services in the service network on the service node needed to be classified. To solve this problem, we propose a Web services classification method based on graph attention network. Firstly, according to the composition and shared annotation relationship of Web services, it applies the description documents, tags of Web services and the call relationship between mashups and services to build a service relationship network. Then, the attention coefficient of service nodes in the network is calculated by the self-attention mechanism, and different service nodes in the neighborhood are assigned different weights to classify Web services. Through the graph attention network, the content features of Web service can be well integrated with its structure information. Also, the learned attention weight is more interpretable. The experimental results on the real dataset of ProgrammableWeb platform show that the precision, recall and macro-F1 of the proposed method are greatly improved compared to those of GCN, Node2vec, DeepWalk and Line.

Keywords: Web service · Attention mechanism · Graph attention network · Service classification

1 Introduction

As the main technology of implementing SOA architecture, Web services can perform distributed computing and integrate data effectively. Web API is a typical Web service, which refers to the API function interface on the network to meet the various needs of

H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 349, pp. 513–529, 2021.
https://doi.org/10.1007/978-3-030-67537-0_31

developers and users. Nowadays, Web APIs have become a core resource of Web and mobile applications. According to recent statistics of the ProgrammableWeb platform, its number has exceeded 20000, and more mashup developers use Web APIs for service composition. Therefore, how to quickly and accurately select the desired Web API from a large number of services has become a challenging issue for developers. Service classification can greatly reduce the time and space cost of Web service search and discovery [1, 2].

At present, many scholars have studied the classification of Web APIs. Most of them [3–5] are based on functional attributes to classify services, such as mining WSDL documents to build feature vectors, using TF-IDF, cosine similarity and other similarity measurement methods to calculate the similarity between vectors [6]. In addition, some researchers use LDA topic model or its extended model [7–9] to mine the hidden topic information of services, and use the topic vector to represent the description documents of services, and then carry out similarity calculation and service clustering [8, 9]. However, due to the short length of the WSDL document and the sparse features, these methods that only consider service content information are difficult to achieve good results [10]. Besides its content information, Web services have direct or indirect relationships with other information (such as tags, mashups, etc.), which can describe the functional attribute characteristics of Web services from multiple perspectives [11]. Therefore, some methods are put forward to exploit auxiliary relations such as tags into the process of service classification, and obtain the positive correlation between services by analyzing the tag relations between services with similar functions [6, 8]. At the same time, many mashup developers combine services with different functions to form some new applications (called Mashups), from which we can mine the negative correlation between different services to enhance the service classification. Therefore, this paper analyzes the relationship between service nodes in the relationship network, and uses the composition relationship and shared annotation relationship between Web services to form a complex service relationship network. In recent years, some related work has also carried out the network construction but basically did not consider the important weight of nodes in the service relationship network. Graph convolution network (GCN) [12] integrates the structure information of the graph with the features of the node to achieve good results in the task for node classification. However, GCN explicitly assigns a non-parametric weight to the adjacent nodes when combining with the adjacent nodes for feature aggregation, which is tough to assign different learning weights to different neighbor nodes.

In recent years, the attention mechanism [13] has been widely adopted in deep learning methods, which can use the self-attention mechanism to calculate different attention coefficients of neighbor nodes. And then, the linear combination of features and attention coefficients can better aggregate features according to the contribution of neighbor nodes. Therefore, some researchers propose graph attention network (GAT) [14], which applies self-attention mechanism to graph neural network. Through the parameterization of the weights between the nodes, the neural network is used for training and learning to assign more weights to important nodes. Inspired by this work, we introduce graph attention network into service classification, and propose a novel Web service classification method based on graph attention network, called SC-GAT. This

method takes the description document and service relationship network as input, then outputs the embedding features of Web service. We apply the learned low dimension vector to represent the service node, which can characterize not only the structure of service network, but also the attribute information of the service node, which to some extent solves the problem of feature sparsity in the previous methods. Finally, we adopt activation function to predict the classification of Web services.

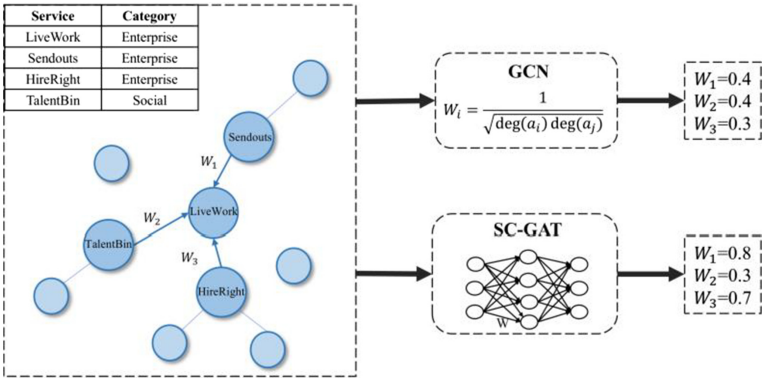


Fig. 1. The scene of service classification (take service "Livework" and its neighbor services "Sendouts", "Hireright" and "Talentbin" for example. Among them, a_i is the service, W_i is the weight of neighbor service).

Figure 1 shows an example of service classification method based on GCN and SC-GAT. Some methods, such as GCN, which consider the use of structural information to represent multi-node aggregation, usually calculate the preference weight of different neighbor nodes through the calculation of node degree. For example, the importance weights of neighbor services "Sendouts", "Hireright" and "Talentbin" calculated by the formula to the central service "Livework" are 0.4, 0.3 and 0.4 respectively, while it is obvious that the service "Sendouts" and "Hireright" are consistent with the category of "Livework", but they have not received good attention. The service "Talentbin" has a different category from "livework", but it gets similar weight to "Sendouts". This method not only needs to obtain the structural information of the service in advance, but this simple weight calculation method can not get the importance weight of the neighbor node well. In this paper, the proposed method SC-GAT parameterizes the weights of neighboring services, uses neural networks to train them according to the features of the nodes, and continuously adjusts the parameters during the training process. Finally, the services "Sendouts" and "Hireright", which are more similar to the central service "Livework", learned the importance weights of up to 0.8 and 0.7 respectively. The final learned parameters can well represent the importance of different neighbor nodes and consequently facilitate Web services classification.

In summary, the contributions of this paper are as follows:

- To our best knowledge, this is the first work to introduce graph attention network into Web service classification, which is significant to weigh the importance of neighbor services on the service node needed to be classified.
- We propose a novel Web service classification based on graph attention network, by exploiting the graph attention mechanism, which uses node features of the service to learn their parametric weights and mines the influence of Web service information on its structure calculation, avoiding costly matrix calculation or relying on the structure of the pre-known graph.
- Based on the real dataset on an online Web APIs repository ProgrammableWeb, we perform experimental comparison and analysis. The experimental results show that the proposed method is effective and outperforms baseline methods.

The rest of this paper is arranged as follows: the Sect. 2 is related work. The Sect. 3 introduces the application of graph attention network to service classification. The Sect. 4 is the experimental evaluation and analysis. The last section is the summary of this paper and the follow-up research work.

2 Related Work

In recent years, with the development of service computing and cloud computing, the discovery and mining of Web services has become a hot research direction. Research shows that efficient Web service classification can effectively improve the performance of Web service discovery [15]. At present, there are two main Web services classification methods, i.e., service classification based on functional semantics [16] and service classification based on QoS (quality of service) [17–19]. However, due to the short timeliness and difficult to obtain QoS information of services, it is rarely used in the Web services classification.

Web Services classification based on functional semantics divides services with high similarity into the same category by calculating the functional similarity of Web services. Chen et al. [6] proposed a Web services clustering method, called as wtcluster, which uses WSDL documents and tags to cluster Web services. Elgazzar et al. [20] designed a new WSDL documents mining technology, and successfully cluster it into similar Web service groups. This method extracts the key information of WSDL documents and clusters them according to service similarity. However, the limited information and sparse features of the service documents limit the clustering performance to some extent. In addition, some researchers exploit the auxiliary information of services into their service clustering process. For example, Wu et al. [21] presented a hybrid Web service tag recommendation strategy WSTRec considering tag co-occurrence, tag mining technology and semantic relevance measurement. Shi et al. [16] devised a probability topic model MR-LDA considering multiple Web service relationships, which can model the relationship between the combination of Web services and the relationship between Web services sharing tags. Cao et al. [22] put forward a Web service classification method based on the topic attention mechanism by combining the local hidden state vector with the global LDA topic vector. The above methods take into account service relationships into the training process of the model and improve service classification performance. However, the hidden relationships between Web services are not fully mined.

Actually, the composition relationship and shared annotation relationship of Web services form a complex service relationship network, and the research on network or graph data has attracted extensive attention. In the service relationship network, we regard the service itself as a node, and the edge is built through the mutual relationship between services. Nowadays, the network representation learning technology [23] can better learn the embedding vector of structure graph for the task of node classification. Therefore, inspired by the research of Yao [23], we propose a Web service classification method based on graph attention network. By weighing the importance of neighborhood service nodes, we use self-attention strategy to calculate the hidden representation of each service node in the service network.

3 Methods

The framework of the proposed method is shown in Fig. 2. It includes four parts: (1) data preprocessing; (2) service similarity computation; (3) graph attention network model construction; (4) service classification. In the data preprocessing part, we crawl Web services from the network and obtains meta information, such as service description text, tags and mashup after preprocessing. Then, API feature matrix and similarity matrix are obtained by calculating service meta information. Next, we construct the graph attention network model, input the API feature matrix and similarity matrix into the graph attention network model, and calculate the network embedding through self-attention mechanism. Finally, the softmax function is used to predict the categories of Web services.

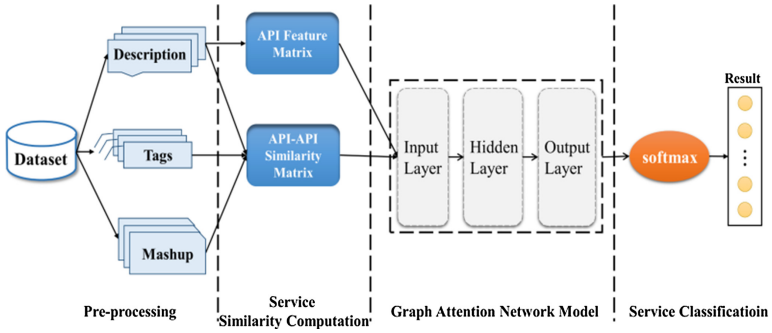


Fig. 2. The framework of service classification

3.1 Pre-processing

After crawling Web services from the network, we perform preprocessing to extract useful information of each Web service description document. The specific operations are as follows:

Remove Punctuation. Lowercase all the words in the description document to facilitate subsequent word stemming. At the same time, the punctuation in the text is removed, leaving only meaningful words.

Word Segmentation. Filter out the functional words in the language, remove the stop words in a sentence, and then list the remaining words. Each element is the word after removing the stop words.

Word Stemming. We implement word stemming for description document. For example, the root of create and created is create, and these words have the same meaning.

3.2 Web Service Similarity Computation

We define service relation network as $G = (V, E)$, which consists of a set of service nodes $V = \{a_i : 1 \leq i \leq N\}$ and a set of edge $E = \{(a_i, a_j) : 1 \leq i, j \leq N\}$. Among them, N is the number of Web services, and the edge (a_i, a_j) between nodes is determined by comparing the similarity and threshold of service nodes. When the calculated similarity is higher than a certain threshold, there is one edge between the two services.

In this paper, we use the word2vec [25] word embedding algorithm to learn the potential semantic vector of Web service description document. We define the text similarity between service a_i and service a_j as $Des(a_i, a_j)$, and combine the relationship between service and tag, mashup to determine the node similarity. According to our survey [24], Web services have the relationship of function composition and label common annotation, which is defined as follows:

Property 1 (Web Service Composition Relationship): a Web service may be called once or more by a mashup developer. If two services a_i and a_j at least one time are called by a user at the same time, it is considered that there is a composition relationship between a_i and a_j.

Property 2 (Web Service Sharing Annotation Relationship): each Web service contains several tags. For example, the label set of service a_i can be expressed as T_i. Different Web services may contain one or more identical tags. When two services share at least one tag, they are considered to have a shared annotation relationship.

According to property 2, Web services with similar tags should have higher similarity in function level, so they are more likely to belong to the same category. Therefore, after extracting all tags of each Web service, we employ the Jaccard coefficient to calculate the tag similarity:

$$Jac(a_i, a_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|} \tag{1}$$

Where T_i, T_j are the tags owned by service a_i and a_j. $|T_i \cap T_j|$ represents the number of common tags owned by two services, $|T_i \cup T_j|$ represents the union of the number of tags owned by two services respectively.

The total similarity matrix is obtained by the following formula (2):

$$Sim(a_i, a_j) = Neg(a_i, a_j)(\alpha Des(a_i, a_j) + (1 - \alpha) Jac(a_i, a_j)) \tag{2}$$

Where α is the user's preference weight for the service description document similarity, and $(1 - \alpha)$ is the weight for tag similarity. $Neg(a_i, a_j)$ describes the correlation between

Web services. This correlation is obtained from the property 1. If two Web services have been called by the same Mashup, it is determined that the two Web services have different functions and should be divided into different categories, and the value is 0. On the contrary, if the two Web services have not been called by the same Mashup, the value is 1.

Consequently, we can take the total similarity $Sim(a_i, a_j)$ as the element value of the matrix to build the API-API similarity matrix.

3.3 GAT Model

After performing pre-processing and service similarity computation, we adopt GAT to learn the network embedding of Web services. The specific framework is shown in Fig. 3, including four parts, i.e., input layer, linear layer, attention layer and output layer.

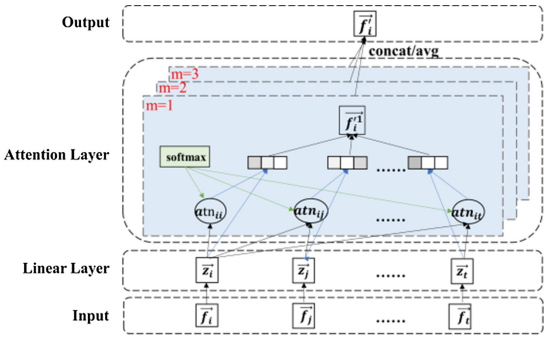


Fig. 3. The framework of GAT (take 3-heads attention as an example)

Input Layer. We define the initial node features of services as $F = \left\{\overrightarrow{f_1}, \overrightarrow{f_2}, \ldots, \overrightarrow{f_N}\right\}$, and the feature vector $\overrightarrow{f_i}$ of each service a_i is a one-hot coding vector $\{0, 1\}^L$, where L is the number of features in each node. It is worth noting that this initial feature vector is discrete and high-dimensional, and its dimension depends on the number of feature words extracted by word2vec. Our goal is to optimize it and learn the continuous and low-dimensional embedding vector.

Linear Layer. In order to make the features enough representable, we need at least one learnable linear process to transform the input features into higher-level features. Therefore, we apply a shared linear transformation parameterized by the weight matrix to each node through the linear layer $\overrightarrow{z_i} = W\overrightarrow{f_i}$, $W \in \mathbb{R}^{L' \times L}$. Where L' is the dimension of target output feature, and the weight matrix represents the relationship between input feature and output feature.

Attention Layer. Generally speaking, the relationship between the service in the network space with a certain range will be closer. We use masked attention to apply graph

structure to the model, that is, only the first-order neighbor nodes of each service node are calculated. The attention mechanism is realized by a single-layer feedforward neural network and parameterized by a weight vector $\vec{\theta} \in \mathbb{R}^{2L'}$. We concatenate the feature of the central service and the feature of neighbor service as the same query and key, so that we can learn the word dependency within the concatenating vector and automatically capture the correlation between the central node and the neighbor node. At the same time, the activation function *Leakyrelu* is applied to retain the coding of positive signal and small negative signal.

$$c_{ij} = \text{LeakyReLU}(\vec{\theta}^T [W\overrightarrow{f_i} \,||\, W\overrightarrow{f_j}]) \tag{3}$$

In order to make the coefficients between different services easy to compare, we utilize softmax to normalize the contribution of all neighbor services, and get the contribution of neighbor service a_j to the new feature of center service a_i. This attention mechanism is shared among all node pairs $atn : \mathbb{R}^{N'} \times \mathbb{R}^{N'} \rightarrow \mathbb{R}$, that is to say, the attention coefficient we finally get is a scalar. In the final calculation of service node features, neighbor services with higher contribution have more influence on the classification of current service. As shown in formula (4):

$$atn_{ij} = \frac{\exp(c_{ij})}{\sum_{h \in Ne_i} \exp(c_{ij})} \tag{4}$$

Where Ne_i is the neighbor nodes set of service node a_i, and || is the connection operation.

The weighted sum of attention coefficients and features of several neighbor services is the result of one calculation. In order to stabilize the learning process of self-attention mechanism, we adopt multi-head attention to calculate multiple representations of services several times. For the aggregation of multiple representations, there are different ways, such as cascading, weighted average, and so on. In the first attention level, we can simply get the embedding vector of the final service node by concatenating without additional parameter learning. The specific calculation way is shown in formula (5):

$$\overrightarrow{f_i'} = \Big\|_{m=1}^{M} \left(\sigma\left(\sum_{j \in Ne_i} atn_{ij}^m W^m \overrightarrow{f_j}\right)\right) \tag{5}$$

Where atn_{ij}^m is the attention coefficient standardized of the m-th calculation for service a_j to service a_i. $\sigma(\cdot)$ is the non-linear activation function. And $\|$ is the connection operation, connecting the multiple results calculated.

Output Layer. It is worth noting that the average value of several attention coefficients is used to derive the final feature representation of network embedding, which is shown in formula (6):

$$\overrightarrow{F_i} = \sigma\left(\frac{1}{M} \sum_{m=1}^{M} \sum_{j \in N_i} attn_{ij} \overrightarrow{f_j'}\right) \tag{6}$$

Where M is the number of attention heads, that is, the number of functions used to calculate the attention coefficient in the multi-head attention mechanism. m represents the m-th function to calculate the attention coefficient.

3.4 Service Classification

We input the embedding features of the service from the GAT model into a full connection layer, and output the probability distribution of all the candidate Web service categories by using the softmax function. Softmax transforms the output value of multi classification into a relative probability, which indicates the probability that the node belongs to a certain category. The calculation way of it is shown in formula (7), where K is the number of candidate Web service categories:

$$\text{softmax}\left(\overrightarrow{F_i'}\right) = \frac{\exp\left(\overrightarrow{F_i'}\right)}{\sum_{k=1}^{K} \exp\left(\overrightarrow{F_k'}\right)} \tag{7}$$

In the process of model training, we employ the cross-entropy loss function to learn the parameters of the model, which is usually chosen as the objective function of the multi classification problem. Cross-entropy describes the distance between the actual output probability and the expected output probability. The smaller the value is, the closer the two probability distributions are. The specific calculation function of it is shown in formula (8):

$$Loss = -\sum_{k=1}^{K} y_k \log \frac{1}{p_k} \tag{8}$$

Where K is the number of service categories. y_k is the indicator variable of service nodes, which is obtained by the one-hot coding. If the service node is consistent with the corresponding category k, $y_k = 1$, otherwise $y_k = 0$; p_k is the prediction probability for the service belonging to the category k.

4 Experiment

4.1 Dataset Description and Experimental Setup

We crawled 17,783 Web APIs from ProgrammableWeb platform as the dataset of service classification. For each Web API, its information includes name, description text, category, tags and other information. Because the experimental dataset is too large, the top 10, 15, 20, 25 and 30 categories with the largest number of Web APIs are selected as the experimental dataset. The distribution of the top 30 categories with the largest number is shown in Table 1. During training, we randomly reorganized the experimental data, and then 60% of the dataset is selected as the training set, 20% as the verification set and 20% as the test set. Adam [26] method is used as the optimizer of the model. The learning rate is equal to 0.005, the batch size is 1, the number of attention heads is 8, and service similarity threshold is set to 0.8.

4.2 Baselines

DeepWalk [27]: Deepwalk uses random walk to get the local information of the network, and learns the vector representation of the service nodes according to the local information. Then it also applies random walk to attain the vertex sequence as the corpus, adopt the word embedding model for training, and learn the one-dimensional feature representation of each service node.

Table 1. Top 30 categories order by number

Category	Number	Category	Number	Category	Number
Tools	850	Telephony	338	Games	240
Financial	758	Reference	308	Photos	228
Messaging	601	Security	305	Music	221
eCommerce	546	Search	301	Stocks	200
Payments	526	Email	291	Cloud	195
Social	501	Video	289	Data	187
Enterprise	472	Travel	284	Bitcoin	173
Mapping	437	Education	275	Other	165
Government	369	Transportation	259	Project Management	165
Science	368	Advertising	254	Weather	164

Node2Vec [28]: Node2vec is an improved method based on DeepWalk. It obtains the corresponding sequence of each service point by using a specific walk way, defines two parameters to balance the influence of BFS and DFS, and considers the local and global information of the service graph structure.

Line [29]: Line uses the first-order similarity model and the second-order similarity model between the training vertices, and employs the edge sampling method to measure the degree of close connection between the service nodes, so as to obtain the similar service nodes for classification..

GCN [12]: GCN takes the standardized graph structure and service node features as the input, extracts the spatial features of the topological graph by using the eigenvalues and eigenvectors of the Laplacian matrix of the graph, and finally uses softmax to score and predict the category of service.

4.3 Evaluation Metrics

*MacroF*1 is chose as the evaluation metric to measure the performance of the proposed method and baselines. By calculating the recall (Rec_i) and precision (Pre_i) of each service category, we get the average recall (Rec_{ma}) and average precision (Pre_{ma}) of all N service categories. Among them, the value of recall is the proportion of correctly classified Web APIs in all Web APIs of this service category, and the value of precision is the proportion of Web APIs of this service category in the final service classification result. *MacroF*1 is the harmonic average value of recall and precision. They can be defined as below:

$$Rec_{ma} = \frac{\sum_i^N Rec_i}{N} \tag{9}$$

$$Pre_{ma} = \frac{\sum_i^N Prec_i}{N} \tag{10}$$

$$F1_{ma} = \frac{2 \times Rec_{ma} Pre_{ma}}{Rec_{ma} + Pre_{ma}} \tag{11}$$

4.4 Experimental Result

The experiment respectively tests the top 10, 15, 20, 25 and 30 service categories with the greatest number of Web APIs, and the experimental results are shown in Table 2 and Fig. 4. It can be seen from the experimental results that the proposed method is superior to all four baseline methods in terms of precision, recall and Macro-F1.

More concretely, with the increase of the number of categories, the experimental performances of all methods gradually decline. The reason is that the increasing of the content information contained in service category and the corresponding selected features makes service classification more difficult. At the same time, the decreasing, uneven number of Web APIs in some categories also leads to the declining of service classification performance.

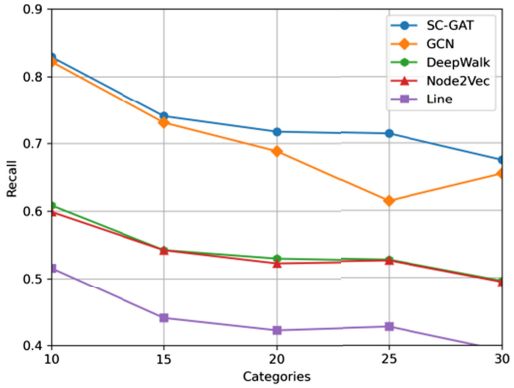
As for DeepWalk, Node2Vec and Line, they mainly obtain the structure information and classify the services into different categories according to the functional network graph of Web APIs. While SC-GAT not only contains the structural information, but also employs the content information of services to determine the importance of different neighbor nodes. And the classification result of SC-GAT is obviously better than the three methods using only a single structural information.

It can be found that when the number of service categories is small, such as 10, the classification result of GCN is similar to that of SC-GAT. However, when the number of service categories increases to 20 or 30, the classification result of SC-GAT superiors to that of GCN. The reason is that the weight calculation between neighbor nodes is single in GCN. With the increase of the number of service categories, the structure of graph becomes rich and complex, which raises the difficulty of service classification. In the SC-GAT, it utilizes the self-attention mechanism to mine the characteristics of the nodes themselves and calculate the contribution of the neighbor nodes, which is completely dependent on the features of nodes and can be independent of the graph structure. Therefore, the representation ability of SC-GAT is improved and service classification is more accurate.

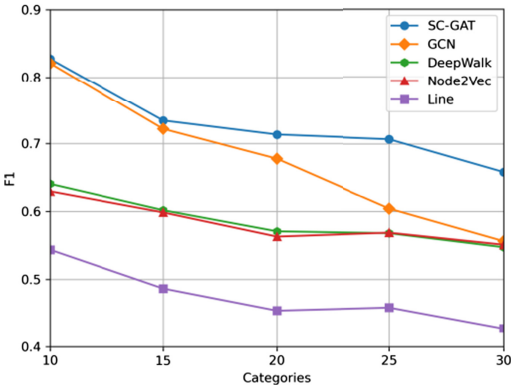
In addition, we make use of t-sne to visualize the classification results in our proposed method and the other four baseline methods. The top-10 categories with the largest number of services are adopted in the data. When the number of categories increases, the number of services contained in some categories decreases, and the division between categories has certain fuzziness, which is not conducive to the display of visual effect. We use the node's color to represent the category of the service node, and calculate the embedding of nodes by t-sne to get the location of nodes. As can be seen from Fig. 5, the service classification results of SC-GAT are basically better than the other four baseline methods. For the Line, DeepWalk and Node2Vec models, most of the service nodes are mixed together, which can not get a satisfactory result of classification. In the SC-GAT, the nodes of most service categories are very close to each other.

Table 2. F1-measure comparison of different serivce classification methods

Methods	Categories				
	10	15	20	25	30
Line	0.54371	0.48640	0.45376	0.45828	0.42714
Node2Vec	0.62957	0.59856	0.56300	0.56880	0.55107
DeepWalk	0.64052	0.60186	0.57076	0.56802	0.54734
GCN	0.82075	0.72173	0.67771	0.60425	0.55630
SC-GAT	**0.82716**	**0.73412**	**0.71343**	**0.70690**	**0.65837**

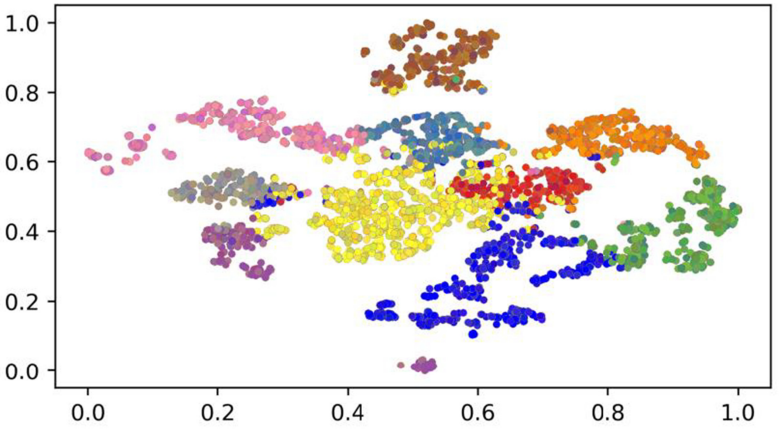


(b) Recall

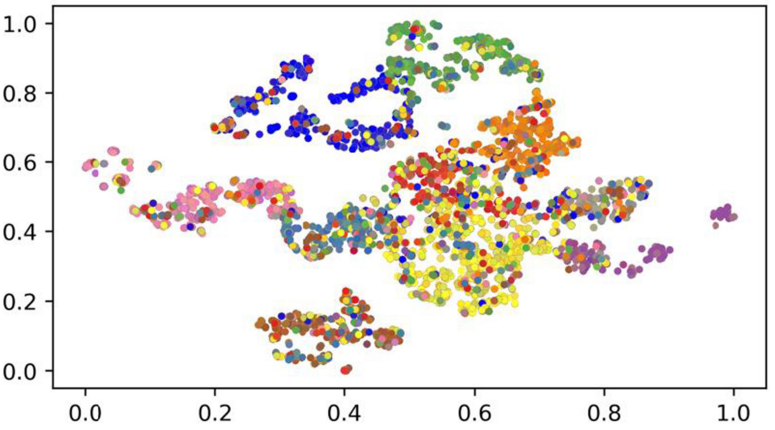


(c) F1

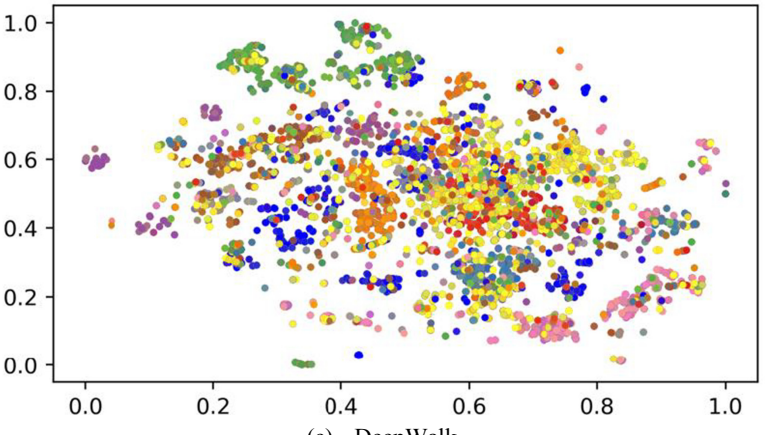
Fig. 4. Performance comparison of different service classification methods



(a) SC-GAT

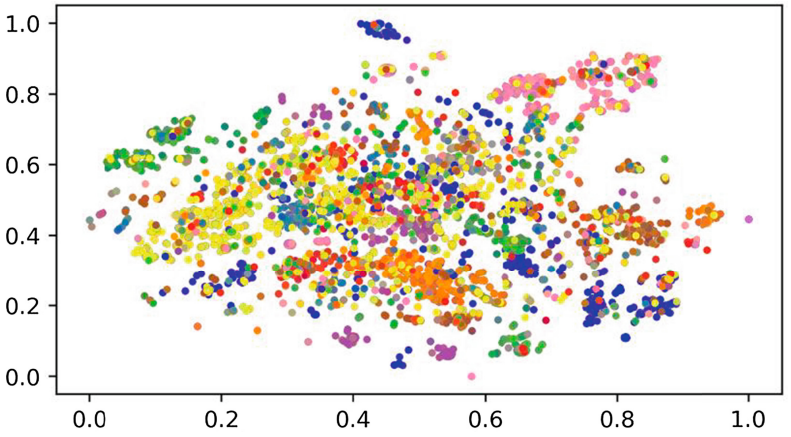


(b) GCN

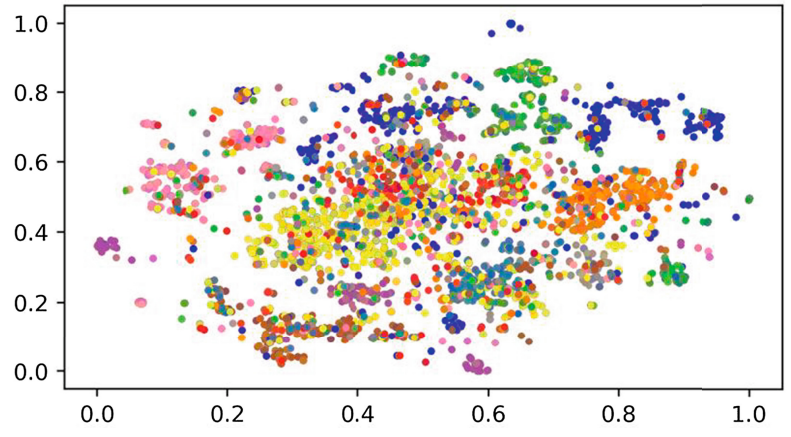


(c) DeepWalk

Fig. 5. Service classification visualization for different methods when the number of service categories is 10



(d) Node2Vec



(e) Line

Fig. 5. (*continued*)

4.5 Conclusion and Future Work

In this paper, we propose a Web service classification method based on graph attention network. It adopts self-attention mechanism to calculate the characteristics of service nodes to identify different contribution degrees for different service nodes in the neighborhoods, and determine the service category according to the influence of the first-order neighbor nodes. The experimental results on real datasets verify the effectiveness of the proposed method. In the future work, we will consider to extend the SC-GAT model to include edge features, in order to further improve the performance of service classification. At the same time, due to the heterogeneity of Web services information, we consider introducing heterogeneous information network in future work, and recommend services according to its rich node types and semantic information.

Acknowledgement. Our work is supported by the National Natural Science Foundation of China (No. 61873316, 61872139, 61832014 and 61702181), the National Key R&D Program of China (2018YFB1402800, 2017YFB1400602), and the Natural Science Foundation of Hunan Province (No. 2018JJ3190, 2018JJ2136). Buqing Cao is the corresponding author of this paper.

References

1. Zhou, Z., Sellami, M., Gaaloul, W., Barhamgi, M., Defude, B.: Data providing services clustering and management for facilitating service discovery and replacement. IEEE Trans. Autom. Sci. Eng. **10**(4), 1131–1146 (2013)
2. Skoutas, D., Sacharidis, D., Simitsis, A., Sellis, T.: Ranking and clustering web services using multicriteria dominance relationships. IEEE Trans. Serv. Comput. **3**(3), 163–177 (2010)
3. Zhang, M., Liu, X., Zhang, R., Sun, H.: A web service recommendation approach based on QoS prediction using fuzzy clustering. In: 2012 IEEE Ninth International Conference on Services Computing, pp. 138–145 (2012)
4. Wang, H., Yang, Z., Yu, Q.: Online reliability prediction via long short term memory for service-oriented systems. In: 2017 IEEE International Conference on Web Services, pp. 81–88 (2017)
5. Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J., Wu, C.: Category-aware API clustering and distributed recommendation for automatic mashup creation. IEEE Trans. Serv. Comput. **8**(5), 674–687 (2015)
6. Chen, L., et al.: WTCluster: utilizing tags for web services clustering. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 204–218. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25535-9_14
7. Min, S.H.I., Jian-Xun, L.I.U., Dong, Z., Bu-Qing, C.A.O., Yi-Ping, W.E.N.: Multi-relational topic model-based approach for web services clustering. Chin. J. Comput. **42**(4), 820–836 (2019)
8. Chen, L., Wang, Y., Yu, Q., Zheng, Z., Wu, J.: WT-LDA: user tagging augmented LDA for web service clustering. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 162–176. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_12
9. Shi, M., Liu, J., Zhou, D., Tang, M., Cao, B.: WE-LDA: a word embeddings augmented LDA model for web services clustering. In: 2017 IEEE International Conference on Web Services, pp. 9–16 (2017)

10. Cao, B., et al.: Mashup service clustering based on an integration of service content and network via exploiting a two-level topic model. In: 2016 IEEE International Conference on Web Services, pp. 212–219 (2016)
11. Cao, B., Liu, X.F., Rahman, M.M., Li, B., Liu, J., Tang, M.: Integrated content and network-based service clustering and web APIs recommendation for mashup development. IEEE Trans. Serv. Comput. **13**(1), 99–113 (2020)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 2017 International Conference on Learning Representations (2017)
13. Peng, C., Sun, Z., Bing, L., Wei, Y.: Recurrent attention network on memory for aspect sentiment analysis. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 452–461 (2017)
14. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
15. Li, H., Liu, J., Cao, B., Tang, M., Liu, X., Li, B.: Integrating tag, topic, co-occurrence, and popularity to recommend web APIs for mashup creation. In: 2017 IEEE International Conference on Services Computing, pp. 84–91 (2017)
16. Shi, M., Liu, J., Zhou, D., Tang, M., Xie, F., Zhang, T.: A probabilistic topic model for mashup tag recommendation. In: 2016 IEEE International Conference on Web Services, pp. 444–451 (2016)
17. Xiong, W., Wu, Z., Li, B., Gu, Q.: A Learning approach to QoS prediction via multi-dimensional context. In: 2017 IEEE International Conference on Web Services, pp. 164–171 (2017)
18. Zhang, Y., Zheng, Z., Lyu, M.R.: WSPred: a time-aware personalized QoS prediction framework for web services. In: 2011 IEEE 22nd International Symposium on Software Reliability Engineering, pp. 210–219 (2011)
19. Kang, G., Liu, J., Tang, M., Liu, X., Cao, B., Xu, Y.: AWSR: active web service recommendation based on usage history. In: 2012 IEEE 19th International Conference on Web Services, pp. 186–193 (2012)
20. Elgazzar, K., Hassan, A.E., Martin, P.: Clustering WSDL documents to bootstrap the discovery of web services. In: 2010 IEEE International Conference on Web Services, pp. 147–154 (2010)
21. Wu, J., Chen, L., Zheng, Z., Lyu, M.R., Wu, Z.: Clustering web services to facilitate service discovery. Knowl. Inf. Syst. **38**(1), 207–229 (2013). https://doi.org/10.1007/s10115-013-0623-0
22. Cao, Y., Liu, J., Cao, B., Shi, M., Wen, Y., Peng, Z.: Web services classification with topical attention based Bi-LSTM. In: Wang, X., Gao, H., Iqbal, M., Min, G. (eds.) CollaborateCom 2019. LNICST, vol. 292, pp. 394–407. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30146-0_27
23. Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 7370–7377 (2019)
24. Shi, M., Liu, J., Cao, B., Wen, Y., Zhang, X.: A prior knowledge based approach to improving accuracy of web services clustering. In: 2018 IEEE International Conference on Services Computing, pp. 1–8 (2018)
25. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
26. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: 2015 International Conference on Learning Representations (2015)
27. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710. Association for Computing Machinery, New York (2014)

28. Grover, A., Leskovec, J.: Node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864. Association for Computing Machinery, New York (2016)
29. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2015)