ISSN 2007-9737

Deep Learning-Based Text Classification to Improve Web Service Discovery

Hadj Madani Meghazi[1,2,3,*], Sid Ahmed Mostefaoui[1,2], Moustafa Maaskri[1,2], Youcef Aklouf[3]

[1] University of Tiaret, Tiaret,
Algeria

[2] University of Tiaret, Laboratory of Research in Artificial Intelligence and Systems,
Algeria

[3] University of Science and Technology Houari Boumediene,
Algiers

{h.meghazi, s_mostefaoui, moustafa.maaskri}@univ-tiaret.dz, yaklouf@usthb.dz

Abstract. Due to the rising number of firms and organizations offering access to their business data or resources on the internet through APIs, there has been a significant increase in the number of web APIs. This poses a difficulty in swiftly and effectively finding online APIs. In order to tackle this problem, the introduction of service classification has been implemented to streamline the process of finding services within a vast array of options. Prior approaches have endeavored to classify web services based on semantic characteristics, although their precision has been constrained. This work introduces a novel strategy named "DeepLAB-WSC" to improve the identification of web services. The approach specifically emphasizes actions derived from textual descriptions of web services and utilizes advanced techniques from deep learning-based text classification. The suggested methodology was evaluated using a real-world web API dataset and achieved superior results compared to existing state-of-the-art research.

Keywords. Service classification, action extraction, text classification, deep learning, web services discovery.

1 Introduction

Service-Oriented-Architecture (SOA) and its primary implementation technology, Web Services (WS), have revolutionized the process of designing and developing corporate applications for software suppliers. Functioning as fundamental units that can transmit and modify data, they interconnect to generate novel composite value-enhanced services that may be accessed as needed.

The most crucial step of the Web services' consumption cycle was and remains service discovery. This is because of their sheer number, which grows exponentially, as well as the fact that better services or mashups will be produced if we can make a good discovery. Research on this subject can be grouped into three (03) categories: syntactic, semantic, and social. The first one covers syntactic techniques for measuring the degree of services similarity that is mostly based on WSDL descriptions, which are published in and processed, word-for-word, by a UDDI directory [21]. Semantic techniques fall within the second category. Several formalisms have been suggested in this context to incorporate a semantic aspect, starting with the straightforward annotation of WSDL descriptions (WSDL-S) [1], moving on to the proposal of a high-level WS ontology (OWL-S) [14], and finally giving birth to a new conceptual model (WSMO) [18]. The third category includes methods that propose a model based on a social network of web services to intercept and exploit the

history of their interactions in order to improve their discovery [13].

To reduce complexity, several studies have been conducted to address this issue, and research has demonstrated that Web service classification/clustering is the optimal approach for not just the discovery process but also for recommendation, selection, and composition. [22, 4, 24, 20]. In this work, we will propose an approach for the classification of web services that falls under the semantic category but without resorting to a domain ontology.

This is not specific to our work, but it is associated with the remarkable success of Deep Learning methods and applications. More precisely, word embedding techniques like Word2Vec [19], Glove [17], or even BERT [3], which are great ways to capture semantics without using ontologies. These latter have considerably slowed down the development of proposals in the field given the extensive work involved in developing high-level ontologies, annotating Web services, and performing related inference processes.

In Our approach, we started by analyzing the textual descriptions of web services in order to extract, with an algorithm, what we will qualify as service actions. Then, we extended and exploited a selection of stat-of-art text classifiers and accentuated their learning with these actions. Finally, we conducted extensive experiments on more than 8,400 real-world web services from ProgrammableWeb[1] to evaluate the effectiveness of our proposal, and we have unequivocally demonstrated that our suggested technique can get a higher level of precision in classification and outperform the most advanced methods currently available. The subsequent sections of this work are structured in the following manner.

Section 2 provides a comprehensive review of relevant research, with a specific emphasis on the application of Deep Learning methods for the classification of Web services. Section 3 is specifically devoted to showcasing our methodology. Section 4 provides a comprehensive examination and interpretation of the experimental findings. Section 5 serves as the final section of the report, providing a conclusion and addressing potential future research.

[1] www.programmableweb.com

2 Related Work

The emergence of web APIs has captured the attention of the academic community and has witnessed substantial engagement. The main sources of data for researching Web services categorization using Deep Learning techniques are Web services description documents and data collected from their environment [26].

For instance, in [28] Cao et al. proposed a Wide and Bi-LSTM model combining all the discrete features in the description documents of Web services and performing the breadth prediction of Web service category to automatically extract the most pertinent semantic information from a Web service document.

Then, a Bi-LSTM architecture uses a topical attention-mechanism to mine the word order and context information of the words in the Web service description documents in order to do Web service classification prediction. In [32], the authors provide a DeepWSC framework for web service clustering that is heuristics-based.

In order to cluster web services effectively, within this method, a signed graph convolutional network is used to extract service composability characteristics from service invocation associations, and an upgraded recurrent convolutional neural network (RCNN) [10] is used to extract deep semantic features from service descriptions.

This is an extension of paper [31], the key difference is that the new DeepWSC utilizes the composability characteristics of services, which are then inputted into the deep neural network with the deep semantic features of web services. These features are combined using an approach to generate integrated implicit properties of web services.

The paper [26] introduces ServeNet, a deep neural network that can automatically extract high-level features from service names and descriptions without the need for feature engineering or length restrictions. ServeNet is

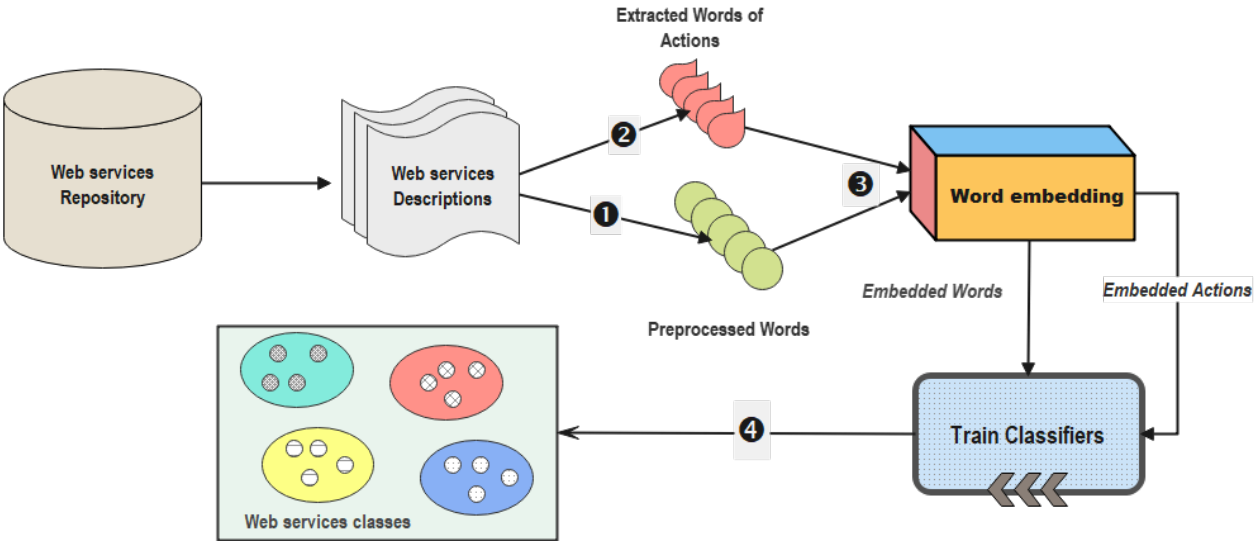

Fig. 1. Overall framework of our DeepLAB-WSC

capable of predicting the classification of services into 50 different categories.

In their study, Kang et al. [9] utilize the attention mechanism to combine the local implicit state vector of a Bidirectional Long Short-Term Memory Network (BiLSTM) with the global Hierarchical Dirichlet Process (HDP) topic vector. They propose a topical attention-based BiLSTM technique for classifying Web services.

The BiLSTM model is designed for the purpose of automatically acquiring the keyword feature representations of Web services. The topic vectors of Web service documents are acquired using HDP during offline training.

A topic attention technique is employed to improve the feature representation by discerning the significance or weight of different keywords in Web service documents.

An attention-based BiLSTM model is also coupled with Information Gain theory to present a Web service classification in [29]. The suggested technique focuses on intricate elements inherent in Web services, such as the significance of various words and the sequential semantic connections between words, through the utilization of IG theory and the attention-based BiLSTM model.

The authors of [23] have developed a new deep neural network that combines a Graph-Convolutional-Network (GCN) and a Bidirectional-Long-Short-Term-Memory (Bi-LSTM) network. This network aims to automatically extract function-description-documents by capturing different relationships in graphs and exploiting them. The GCN is used to extract global spatial features, while the Bi-LSTM network learns sequential features.

Also, Peng et al. [16] present a graph attention network-based Web services classification approach. To begin, it capitalizes on description documents, Web service tags, and the call relationship between mashups and services to create a service relationship network based on Web service composition and shared annotations. The self-attention mechanism then determines the attention coefficient of each service node in the network, and different service nodes in nearby areas are allocated different weights to classify Web services. The graph attention network combines a Web service's content characteristics with structure information.

X Yong et al. introduce LDNM [25], a comprehensive framework for classifying web services. This framework utilizes a deep

Algorithm 1: Extract Actions

Input: Web service textual description
Output: List of Actions[ ]

```
function SGET_ACTION(Sentence)
    if (Item is Verb) and (has dobj) then
        Concatenate the verb with dobj;
        return as Action;
    if (Item is Verb) and (has conjuction) then
        Go to the conjunction item;
        Once you reach the Noun: concatenate the Verb with the Noun;
        if this Noun has conjuction then
            Concatenate also the Verb with the next Noun;
    return Actions;
procedure GET_ACTION(Sentence)
    if (Item is Verb) and (has xcomp) then
        go to the xcomp node;
        concatenate the Verb with each action of SGet_Action(xcomp);
        Add all as Actions to the list of Actions[ ];
    else
        Add all Actions of SGet_Action(Sentence) to the list of Actions[ ];
for each sentence in Web service textual description do
    Get_Action(Sentence);
```

fusion technique to combine structured and unstructured characteristics. The initial step included transforming each service document into a feature vector using two different document representation techniques: topic distribution based on LDA (Latent Dirichlet Allocation) [2] and Doc2vec [11], which is a document embedding model based on neural networks.

Then, using Node2vec [6], they obtained structured representation vectors extracted from service invoking and tagging graphs. Finally, they employ an MLP neural network to fuse these features and train a service classifier.

The methods we just discussed attempt to classify web services based on their textual descriptions or by combining them with other features. When we looked more closely at these methods, we discovered that they treat these descriptions as a collection of words rather than taking into account and differentiating between significant words and their placements in these descriptions.

Original Tags	Stem Tags
(‘Service’, ‘proper noun’)	(‘servic’, ‘adjective’)
(‘used’, ‘**verb**’)	(‘use’, ‘adjective’)
(‘validate’, ‘**verb**’)	(‘valid’, ‘adjective’)

Furthermore, some techniques simply use the stems of words, which can distort the meaning. In this light, we extracted the actions of web services from their textual descriptions using an algorithm, which we will describe in more detail in the following section.

Then, a number of models from the ”Text Classification” [8, 10, 3, 30, 12] domain are extended and adjusted to better fit the classification of these services using these actions as additional features.

3 The Classification Process of the Proposed Approach

The process of our approach, which we named DeepLAB-WSC (Deep Learning Actions Based Web Service Classification), is illustrated in Figure 1. It consists of four (04) steps.

3.1 Step 1: Pre-processing of Web Services Descriptions

Web Services are characterized by their capacity to execute actions and perform tasks. Nevertheless, in the majority of situations, when it comes to presenting information about them, we can only offer a concise written explanation.

In order to effectively identify them using this method, the key question to ask is: "Which specific words or phrases in these descriptions accurately convey the essence of a service?" Initially, we believed that when we encounter a text, the acts described within it are inherently represented by 'verbs'. We began the process by extracting all possible verbs from web service descriptions using pre-existing natural language models that had been trained beforehand.

At this juncture, we encountered two predicaments: The primary concern is that these models lack the ability to precisely identify all verbs. In the statement "Service X searches for all worldwide airlines that operate in a given country", the term 'searches' is identified as a noun. If we isolate each word and disregard its context, what would happen?

The technique becomes more challenging due to the loss of start tags for many words throughout the text cleaning and pre-processing steps, especially after the Stemming step. Consequently, verbs are often misidentified as nouns, adjectives, and so on. Take into account the statement that follows: The words 'Service', 'used', and 'validate' in the statement "Service X is used to validate monetary transactions." are marked as follows:

In most cases, this has the effect of changing the semantic meaning of each word and, as a result, the overall meaning of the description. Because of this, we attempted to overcome this in our situation from the very beginning of our approach. That's why we begin by softly pre-processing the service descriptions in order to retain as many words as possible with their original tags.

3.2 Step 2: Obtain Actions Via Web Service Descriptions

Another problem we faced was the scarcity of verbs in the majority of brief descriptions, making it challenging to utilize these models effectively. In order to tackle this issue, we employed the 'Extract-actions' algorithm at this phase to extract activities as per our definition where xcomp denotes an open-clausal-complement and **dobj** symbolizes the direct object. The algorithm is demonstrated in the following examples:

- **Case 1:**
  - Description: Service X **annotates** text.
  - Actions: [**annotates** text].
- **Case 2:**
  - Description: Service X **annotates** text and images.
  - Actions: [**annotates** text, **annotates** images].
- **Case 3:**
  - Description: Service X uses Twitter API to track followers.
  - Actions: [**utilizes** Twitter, **utilizes track** followers].

3.3 Step 3: Actions and Descriptions Embedding

Of the five (05) classification models we trained (see Section 3.3), we used two methods for the Word embedding process: Glove and BERT.

- GloVe (Global-Vectors for word representation) [17] is an unsupervised learning method developed by Stanford University researchers with the goal of generating Word Embeddings by aggregating global word co-occurrence matrices from a given corpus.

  We used the variant with a length equal to 100 and trained on 6B tokens including 400K

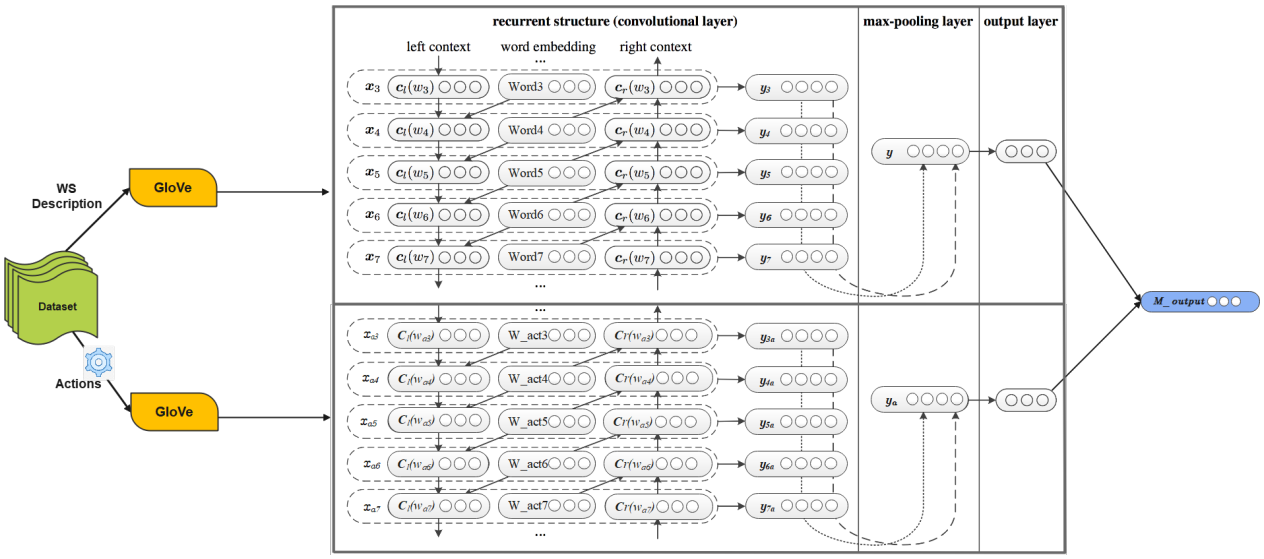

Fig. 2. WSC2RCNN architecture [7]

vocab. This variant is used on the WSC2RCNN and TextING-Based [30]classification models.

– BERT(Bidirectional-Encoder-Representations from Transformers) [3] is a state-of-the-art language representation model.

It is a huge deep bidirectional encoder-based transformer model that has been pre-trained on more than 110 million parameters. BERT is used as the word embedding model for our BERT and BERT-GCN based [12] classification models.

3.4 Step 4: Classification based on Services' Actions and Descriptions Embedding

Our approach's learning process consists of training our classifiers using embedding representations of web service descriptions, then attempting to accentuate their learning with the actions extracted in Step **2** and embedded in Step **3**. After making a connection between the work seen in Section 2 and the results obtained by [15] and [5], the tested models are:

3.4.1 WSC2RCNN

In our previous work [7], we used a Web Service Classifier with two (02) RCNN deep neural networks. We gave, pre-trained GloVe model, representations for both descriptions and collected actions into a pair of Recurrent-Convolutional-Neural Networks (RCNN) that would be trained.

For both networks, we employed the original version of RCNN [10], which seeks to capture textual semantics by considering sequence word order. This approach utilizes both Recurrent Neural Networks (RNN) to understand the local context of tokens and Convolutional Neural Networks (CNN) to capture long-term dependencies. The left and right contexts of a word w_i, which we determined using the following equations, were represented by $c_l(w_i)$ and $c_r(w_i)$, respectively, for the first RCNN network:

$$c_l(w_i) = f(W^{(l)}c_l(w_{i-1}) + W^{(sl)}e(w_{i-1})), \quad (1)$$

$$c_r(w_i) = f(W^{(r)}c_r(w_{i+1}) + W^{(sr)}e(w_{i+1})), \quad (2)$$

where the activation function f is non-linear. A word's word embedding vector is represented by $e(w)$. The context is transformed into the following hidden layer using a matrix called $W^{(l)}$. $W^{(sl)}$ is a

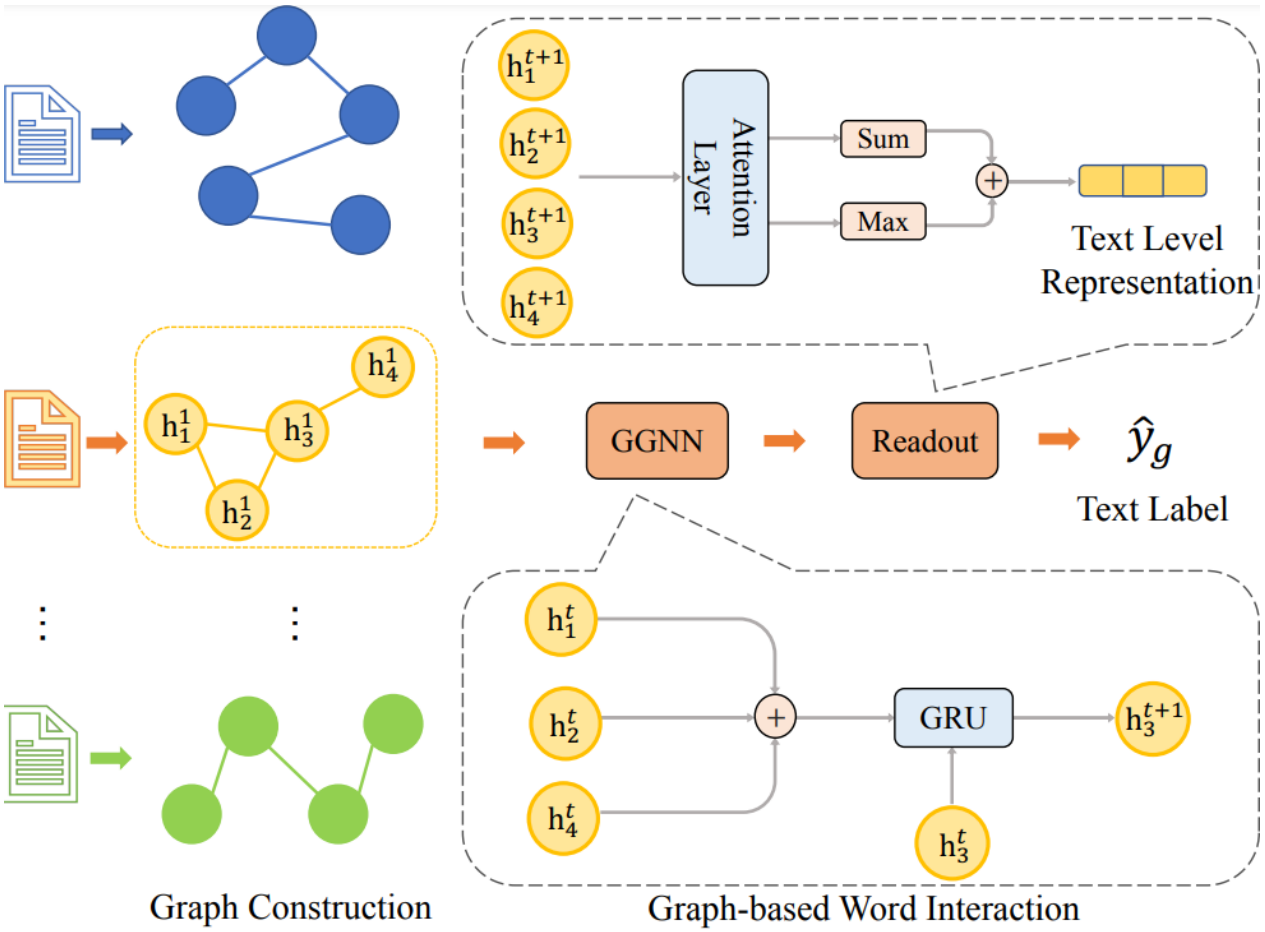

Fig. 3. TextING architecture [30]

matrix that links the semantic of the present word to the left context of the word that comes after it.

Equation (3) is used to obtain the latent semantic representation of the y_i vector:

$$y_i = \tanh(Wx_i + b), \tag{3}$$

where a word w_i is represented by x_i:

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)]. \tag{4}$$

A similar calculation is made for y_a (of actions) for the second network, but it is applied to the words of the extracted actions. The 20 key categories of high-quality Web services from the dataset are linked to the outputs of both RCNN networks, which are goals of WSC2RCNN. The final output is then prepared by summing the outputs from the two RCNN networks.

A softmax activation layer is used to normalize the final output, M_output, because we have a multi-class issue. We present our model in Figure 2.

3.4.2 TextING Classifier

We conducted experiments on WS descriptions using TextING. The authors of [30] provide a novel graph neural network for text categorization. In this network, each document is treated as a separate graph, allowing for the learning of word interactions at the text level. In this process, distinct graphs are generated for each text, and

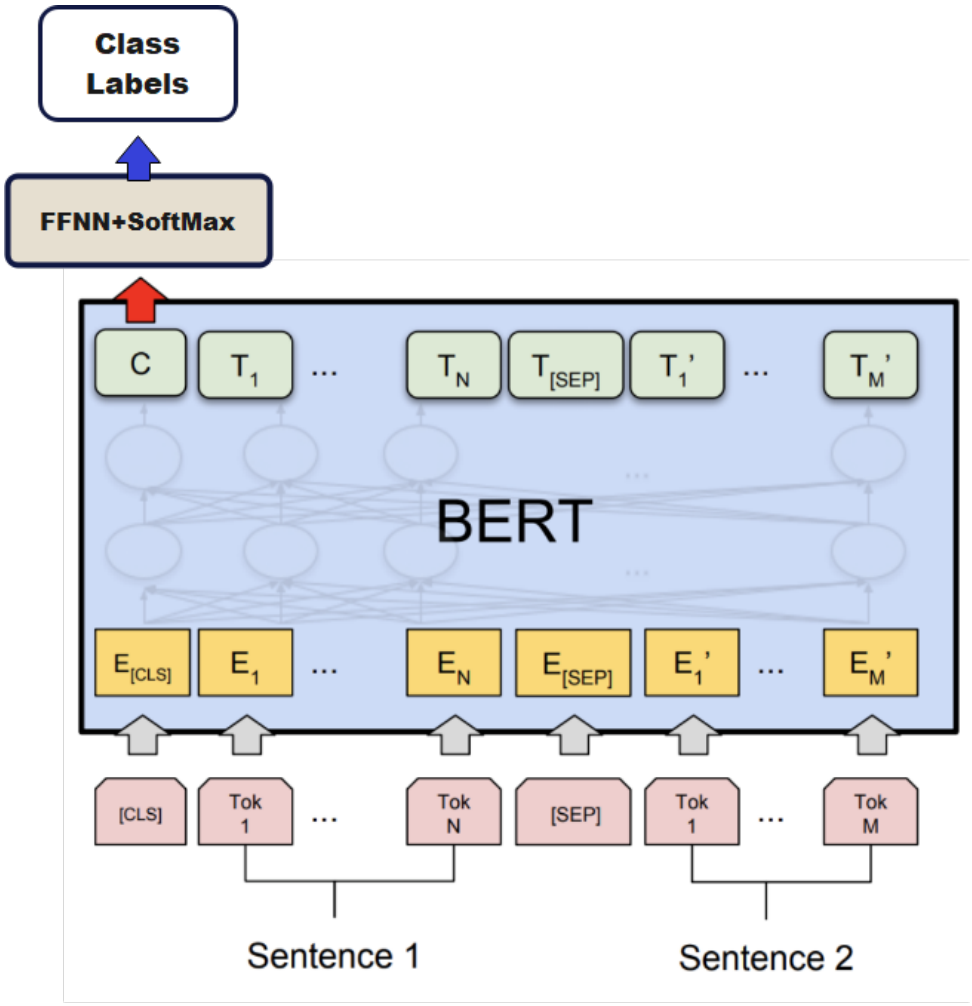

Fig. 4. BERT classifier

subsequently, Graph Neural Networks (GNN) are employed to acquire detailed word representations by considering their local structures. Additionally, GNN has the capability to produce embeddings for words that have not been seen before in the new document. Ultimately, the word nodes are included into the document embedding.

3.4.3 BERT Classifier

A fine tuned BERT is used and a number of layers are added to elaborate a BERT Web Service classifier, shown in figure 4. The model used is bert-base-uncased, which consists of 12 layers, 768 hidden units, 12 attention heads, and a total of 110 million parameters, and a softmax activation layer to normalize the final output.

3.4.4 BERT_With_Actions Classifier

We made multiple attempts to incorporate the extracted actions efficiently while trying to increase the performance of the “BERT Classifier” and validate our approach. The solution found consists of concatenating the sentences of the web service descriptions with the actions repeated “Rep” times

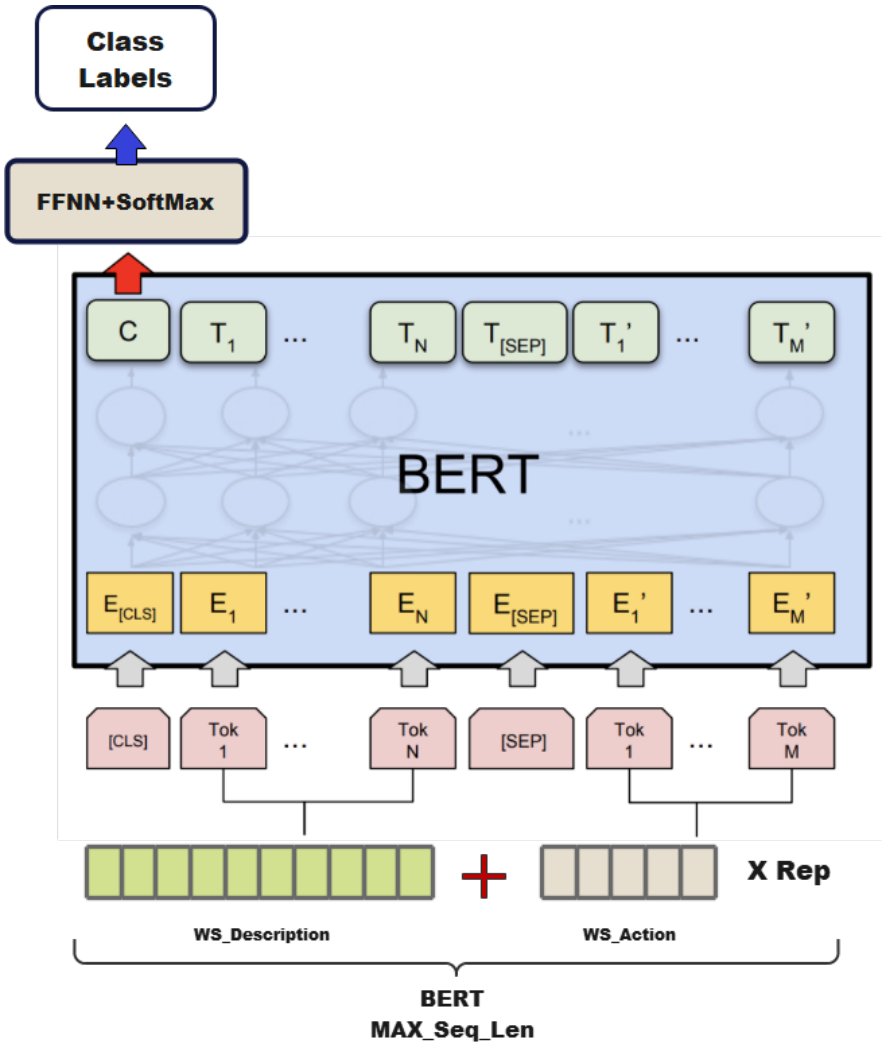

Fig. 5. BERT + Actions classifier

while not exceeding the BERT variant's maximum sequence length (See figure 5):

$$\text{Input}_{\text{BERT\_with\_Actions}} = \text{WS\_Description} + \text{WS\_Actions} \times \text{Rep}. \quad (5)$$

3.4.5 Classifiers based on BERT-GCN

We also put Yuxiao Lin et al.'s proposition (BERT-GCN) to the test. In [12], BERT with his large-scale pretraining, and GCN [27] with his transductive learning, are trained together.

First, a graph with both word nodes and document nodes is built. The weight of an edge that connects two nodes i and j is defined in the following manner:

$$A_{i,j} = \begin{cases} \text{PPMI}(i,j), & i, j \text{ are words and } i \neq j, \\ \text{TF} - \text{IDF}(i,j), & i \text{ is a document, } j \text{ is a word,} \\ 1, & i = j, \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

The word-document edges and word-word edges are determined using the term frequency-inverse document frequency (TF-IDF)

Table 1. The allocation of Web services across selected categories

ID	Primary Cat	# of services	ID	Primary Cat	# of services
0	Tools	887	10	Telephony	342
1	Financial	757	11	Security	312
2	Messaging	591	12	Reference	304
3	eCommerce	553	13	Email	299
4	Payments	553	14	Search	290
5	Social	510	15	Travel	294
6	Enterprise	509	16	Video	281
7	Mapping	429	17	Education	277
8	Government	371	18	Advertising	274
9	Science	357	19	Transportation	269

and positive point-wise mutual information (PPMI) measures, respectively.

Then, the GCN layers iteratively update document nodes after initializing them using BERT-style embeddings, and the main training aim is to do linear interpolation between the BertGCN prediction and the BERT prediction. This may be expressed as:

$$Z = \lambda\, Z_{\mathrm{GCN}} + (1-\lambda)\, Z_{\mathrm{BERT}}, \tag{7}$$

where λ regulates the tradeoff between the two models. The same process of 3.4.4 is applied to BERT_GCN_With_Actions Classifier in terms of the integration of actions while respecting the maximum length imposed by BERT used variant.

4 Experiments

4.1 Used Evaluation Metrics

The evaluation of selected models has been conducted using four widely used evaluation measures: Purity, NMI, Recall, and F_1-measure. **Purity** is a metric used for guided cluster validation. It is calculated using the following formula:

$$\mathrm{Purity}(\Omega,\, \mathbb{C}) = \frac{1}{N}\sum_{i=1}^{k} \max_{j} \left|\omega_i \cap c_j\right|, \tag{8}$$

where $\Omega = \{w_1, w_2, ..., w_K\}$ represents the collection of clusters of web services, whereas $\mathbb{C} = \{c_1, c_2, ..., c_J\}$ represents the collection of classes of web services. **NMI** is a metric that relies on mutual information and is precisely defined as:

$$\mathrm{NMI}(\Omega,\, \mathbb{C}) = \frac{2 \times I(\Omega;\, \mathbb{C})}{H(\Omega) + H(\mathbb{C})}, \tag{9}$$

where can we obtain the mutual information I using:

$$I(\Omega;\, \mathbb{C}) = \sum_{i=1}^{k}\sum_{j=1}^{k} P(\omega_i \cap c_j) \log \frac{P(\omega_i \cap c_j)}{P(\omega_i) \cap P(c_j)}. \tag{10}$$

And the entropy H using:

$$H(\Omega) = -\sum_{i=1}^{k} P(\omega_i) \log P(\omega_i). \tag{11}$$

Recall is a metric used to determine the accuracy of predicting the real class by measuring the proportion of properly predicted instances. It is computed as follows:

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{12}$$

where TP represents the count of services properly allocated to their respective class, and FN represents the count of services where the model wrongly forecasts their positive class as negative.

Table 2. The classification performance of tested methods

Models	Purity	NMI	Recall	F1-Score
DeepWSC	0.5708	0.4856	0.3821	0.3969
DeepWSC + Heuristics	0.6379	0.5273	0.4186	0.4356
RCNN*	0.6438	0.5704	0.6438	0.6247
WSC2RCNN	**0.6595**	**0.5795**	**0.6588**	**0.6512**
TextING	0.6887	0.5986	0.6864	0.6714
BERT	0.7746	0.6924	0.7746	0.7712
BERT + Actions	**0.7785**	**0.6931**	**0.7786**	**0.7759**
BERT-GCN	0.7788	0.6935	0.7788	0.7717
BERT-GCN + Actions	**0.7810**	**0.6972**	**0.7811**	**0.7745**

The last used measure is $\mathbf{F}_1$ which is calculated using the next formulas:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{13}$$

The term "FP" represents the count of APIs where the model wrongly forecasts their negative classes as positive:

$$\text{F}_1\text{-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{14}$$

4.2 Experimental Environment

We performed tests to assess and showcase the efficiency of our methodology. All experiments were executed on a platform equipped with an Intel (R) Xeon (R) platinum 8259CL CPU@2.50GHz (32 cores) and 256GB RAM.

Utilizing the Dataset[2] of [31, 32] from ProgrammableWeb, the leading online registry for Web services with a vast collection of over 23,000 APIs, we accessed a compilation of 17,923 authentic web services that were obtained by web crawling. The experimental data comprises 8,459 high-quality WS from the top 20 classes (refer to Figure 1).

[2]github.com/aourhtnowvherlcaer/programmableWeb

4.3 Experimental Results and Discussions

We compared our results to those of [31, 32] in order to correctly interpret them. Table 2 summarizes all of the results obtained by the various studied methods. We started by putting in tests for RCNN, TextING 3.4.2, BERT 3.4.3, and BERT-GCN 3.4.5 classifiers. The results were satisfactory, starting with the RCNN classifier, which alone exceeded those of [32] on all metrics.

This is because the authors of [31] wanted to get closer to the semantics of WS descriptions by trying to capture the context of each word with an RCNN deep neural network. However, by applying strict stemming, the meaning of many words is lost (See Section 3.1).

This is what we could observe in their pre-processed dataset. This issue has been fixed and the results of our RCNN classifier exceeded the best of [32]'s by an average of 26.57% on all the evaluation metrics.

The first tested method of the GNN class (TextING) achieves an average improvement of 34.89%. On [32], BERT and BERT-GCN had average advantages of 53.7% and 54.2%, respectively. To complete the evaluation of our approach's effectiveness, each of the proposed classifiers has gone through an extra training process in which we have attempted to include the actions retrieved by our "**Extract-Actions**" algorithm (See Section 3.2).

The results have demonstrated that the classifiers' performance improves each time these actions are combined. Our WSC2RCNN Classifier outperforms our RCNN and presents an average advantage of 5.51% across all the evaluation metrics. Furthermore, our BERT+Actions classifier achieves an average advantage of 0.43% over the BERT classifier. Ending with BERT-GCN+Actions, which outperforms all classifiers seen in this work and shows a 0.37% improvement over BERT-GCN.

5 Conclusion and Future Research

This work introduces a novel method for classifying web services, named DeepLAB-WSC, that relies on the concise textual descriptions of the services. Our methodology stands out by prioritizing the activities executed by web services, which are derived from their descriptions, and use deep learning text classification techniques to categorize the services.

Our comparison trials have demonstrated that DeepLAB-WSC surpasses current cutting-edge techniques for classifying web services in terms of all performance parameters. Our technique has a key benefit in that it specifically targets the most crucial aspect of the description, namely the actions, in order to enhance the accuracy of categorization.

Looking ahead, we plan to extend this work by leveraging the power of BERT and GNN to build social networks of web services based on the similarities (BERT based classifiers) and complementarities (predicting the next sentence/document) of their descriptions. Making this social dimension profitable will allow usto enhance both the process of finding and combining online services.

References

1. **Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P., Verma, K. (2005).** Web service semantics - WSDL-S. www.w3.org/submissions/WSDL-S/.
2. **Blei, D. M., Ng, A. Y., Jordan, M. I. (2003).** Latent dirichlet allocation. The Journal of Machine Learning Research, Vol. 3, pp. 993–1022.
3. **Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2019).** BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologie, pp. 4171–4186. DOI: 10.18653/v1/n19-1423.
4. **Elgazzar, K., Hassan, A. E., Martin, P. (2010).** Clustering WSDL documents to bootstrap the discovery of web services. IEEE International Conference on Web Services, pp. 147–154. DOI: 10.1109/icws.2010.31.
5. **Gasparetto, A., Marcuzzo, M., Zangari, A., Albarelli, A. (2022).** A survey on text classification algorithms: From text to predictions. Information, Vol. 13, No. 2, pp. 83. DOI: 10.3390/info13020083.
6. **Grover, A., Leskovec, J. (2016).** node2vec: Scalable feature learning for networks. Proceedings of the 22nd ACM Special Interest Group on Knowledge Discovery and Data Mining and International Conference on Knowledge Discovery and Data Mining, pp. 855–864. DOI: 10.1145/2939672.2939754.
7. **Hadj-Madani, M., Youcef, A. (2022).** WSC2RCNN: A deep learning actions-based classifier for improved web service discovery. Computación y Sistemas, Vol. 26, No. 4. DOI: 10.13053/cys-26-4-4069.
8. **Joachims, T. (1998).** Text categorization with support vector machines: Learning with many relevant features. Proceedings of the European Conference on Machine Learning, pp. 137–142. DOI: 10.1007/bfb0026683.
9. **Kang, G., Xiao, Y., Liu, J., Cao, Y., Cao, B., Zhang, X., Ding, L. (2021).** Tatt-BiLSTM: Web service classification with topical attention-based BiLSTM. Concurrency

and Computation: Practice and Experience, Vol. 33, No. 16. DOI: 10.1002/cpe.6287.

10. **Lai, S., Xu, L., Liu, K., Zhao, J. (2015).** Recurrent convolutional neural networks for text classification. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 29, No. 1, pp. 2267–2273.
11. **Lau, J. H., Baldwin, T. (2016).** An empirical evaluation of doc2vec with practical insights into document embedding generation. Proceedings of the 1st Workshop on Representation Learning for NLP, pp. 78–86. DOI: 10.18653/v1/w16-1609.
12. **Lin, Y., Meng, Y., Sun, X., Han, Q., Kuang, K., Li, J., Wu, F. (2021).** BertGCN: Transductive text classification by combining GNN and BERT. Proceedings of the Association for Computational Linguistics International Joint Conference on Natural Language Processing. DOI: 10.18653/v1/2021.findings-acl.126.
13. **Maamar, Z., Wives, L. K., Badr, Y., Elnaffar, S., Boukadi, K., Faci, N. (2011).** LinkedWS: A novel web services discovery model based on the metaphor of "social networks". Simulation Modelling Practice and Theory, Vol. 19, No. 1, pp. 121–132. DOI: 10.1016/j.simpat.2010.06.018.
14. **Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K. (2004).** Owl-s: Semantic markup for web services. www.w3.org/submissions/OWL-S/.
15. **Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., Gao, J. (2021).** Deep learning–based text classification: A comprehensive review. ACM Computing Surveys, Vol. 54, No. 3, pp. 1–40. DOI: 10.1145/3439726.
16. **Peng, M., Cao, B., Chen, J., Liu, J., Li, B. (2021).** SC-GAT: Web services classification based on graph attention network. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 349, pp. 513–529. DOI: 10.1007/978-3-030-67537-0_31.
17. **Pennington, J., Socher, R., Manning, C. (2014).** Glove: Global vectors for word representation. Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543. DOI: 10.3115/v1/d14-1162.
18. **Roman, D., de-Bruijn, J., Mocan, A., Lausen, H., Domingue, J., Bussler, C., Fensel, D. (2006).** WWW: WSMO, WSML, and WSMX in a nutshell. Proceedings of the Asian Semantic Web Conference, pp. 516–522. DOI: 10.1007/11836025_49.
19. **Rong, X. (2014).** word2vec parameter learning explained. arXiV. DOI: 10.48550/ARXIV.1411.2738.
20. **Shi, M., Tang, Y., Liu, J. (2019).** Functional and contextual attention-based LSTM for service recommendation in mashup creation. IEEE Transactions on Parallel and Distributed Systems, Vol. 30, No. 5, pp. 1077–1090. DOI: 10.1109/tpds.2018.2877363.
21. **Walsh, A. E. (2002).** UDDI, SOAP, and WSDL: The web services specification reference book. Prentice Hall Professional Technical Reference.
22. **Wang, H., Shi, Y., Zhou, X., Zhou, Q., Shao, S., Bouguettaya, A. (2010).** Web service classification using support vector machine. Proceedings of the 22nd IEEE International Conference on Tools with Artificial Intelligence, pp. 3–6. DOI: 10.1109/ictai.2010.9.
23. **Wang, X., Liu, J., Liu, X., Cui, X., Wu, H. (2020).** A spatial and sequential combined method for web service classification. Proceedings of the Asia Pacific Web and Web-Age Information Management Joint International Conference on Web and Big Data, pp. 764–778. DOI: 10.1007/978-3-030-60259-8_56.
24. **Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J., Wu, C. (2015).** Category-aware API clustering and distributed

recommendation for automatic mashup creation. IEEE Transactions on Services Computing, Vol. 8, No. 5, pp. 674–687. DOI: 10.1109/tsc.2014.2379251.

25. **Xiao, Y., Liu, J., Kang, G., Cao, B. (2021).** LDNM: A general web service classification framework via deep fusion of structured and unstructured features. IEEE Transactions on Network and Service Management, Vol. 18, No. 3, pp. 3858–3872. DOI: 10.1109/tnsm.2021.3084739.
26. **Yang, Y., Qamar, N., Liu, P., Grolinger, K., Wang, W., Li, Z., Liao, Z. (2020).** ServeNet: A deep neural network for web services classification. IEEE International Conference on Web Services, pp. 168–175. DOI: 10.1109/icws49710.2020.00029.
27. **Yao, L., Mao, C., Luo, Y. (2018).** Graph convolutional networks for text classification. Proceedings of the 33rd AAAI Conference on Artificial Intelligence, pp. 7370–7377. DOI: 10.48550/ARXIV.1809.05679.
28. **Ye, H., Cao, B., Peng, Z., Chen, T., Wen, Y., Liu, J. (2019).** Web services classification based on wide and Bi-LSTM model. IEEE Access, Vol. 7, pp. 43697–43706. DOI: 10.1109/access.2019.2907546.
29. **Zhang, X., Liu, J., Cao, B., Shi, M. (2021).** Web service classification based on information gain theory and bidirectional long short-term memory with attention mechanism. Concurrency and Computation: Practice and Experience, Vol. 33, No. 13. DOI: 10.1002/cpe.6202.
30. **Zhang, Y., Yu, X., Cui, Z., Wu, S., Wen, Z., Wang, L. (2020).** Every document owns its structure: Inductive text classification via graph neural networks. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 334–339. DOI: 10.18653/v1/2020.acl-main.31.
31. **Zou, G., Qin, Z., He, Q., Wang, P., Zhang, B., Gan, Y. (2019).** Deepwsc: A novel framework with deep neural network for web service clustering. IEEE International Conference on Web Services, pp. 434–436. DOI: 10.1109/icws.2019.00077.
32. **Zou, G., Qin, Z., He, Q., Wang, P., Zhang, B., Gan, Y. (2022).** DeepWSC: Clustering web services via integrating service composability into deep semantic features. IEEE Transactions on Services Computing, Vol. 15, No. 4, pp. 1940–1953. DOI: 10.1109/tsc.2020.3026188.

Article received on 20/03/2023; accepted on 21/04/2024.
* *Corresponding author is Hadj Madani Meghazi.*