ISSN 2007-9737

WSC2RCNN: A Deep Learning Actions-based Classifier for Improved Web Service Discovery

Meghazi Hadj Madani, Aklouf Youcef

University of Science and Technology Houari Boumediene,
Research Laboratory in Informatics, Intelligence,
Mathematics and Applications, Algiers,
Algeria

{hmeghazi, yaklouf}@usthb.dz

Abstract. Due to the increasing popularity of Web services and their tremendous number, discovery tasks are shown to be the most important and difficult steps. The difficulty lies in the limited data provided on them, which is, in most cases, a short textual description. Based on these descriptions, many interesting works have been proposed trying to classify them in an efficient way. In this work, we propose a new approach based on deep learning WSC2RCNN that uses Web services descriptions and actions within these descriptions to improve the classification task, which has given promising results and outperforms state-of-the-art approaches using the same data.

Keywords. Web service, service classification, service actions, deep neural network.

1 Introduction

As a result of the strong demand for Web applications and the evolution of Web technologies, especially Service Oriented Architecture, providing more suitable solutions for interoperability, this led many software vendors to publish their applications as Web Services (WS). Like any good craftsman, if we have the right tools, we will necessarily get a quality end product; hence, the importance of the Web services discovery process.

Given their large number, the process of Web services discovery/selection has become problematic and inaccurate. Several research works have been done to solve this and have shown that Web services classification / clustering is the solution to reduce complexity, not only for the discovery process but also for recommendation, selection and composition. [21, 9, 6, 23, 4]. Early methods achieved clustering using textual WSDL descriptions as they are or by applying data mining techniques on them[18]; in the same context [9] approached the problem by selecting a set of Web service features such as WSDL contents, messages, types, and ports.

The problem here is that these kinds of methods are purely syntactic and devoid of semantics [2]. To remedy this, some formalisms have been proposed to add a semantic dimension, such as WSDL-S [1], OWL-S [15], and WSMO [12].

These formalisms have everything they need to shine but on a small scale, since they require a lot of effort for the development of high-level ontologies, the annotation of web services and associated inference operations.

Authors in [14] wanted to approach the problem from a social point of view and tried to exploit Web services interactions by building social networks defined on top of three relations categories which are: competition and substitution for web services having the similar functionalities, in addition to a collaboration category for those having different functionalities.

The social dimension was also exploited in [8] where a *global-social-service* network was created by associating to services related ones based on their functionalities, then capitalizing on service-to-service exploration.

We find also all the works that use the description documents of Web services to bring out the semantic features and classify them. Some used probabilistic topic model like LDA (Latent-Dirichlet-Allocation) to extract latent-topic features [7, 3, 20, 13, 17].

In the last few years, research in the field has changed course, especially after moving to the popular API style as a common mode of Web service consumption, and an additional category has emerged as a result of the great success of Deep Learning.

By exploiting deep learning techniques' potential and efficiency in natural language processing, new possibilities have emerged for Web services classification since their descriptions are available as short texts. Therefore, several approaches from the field [11, 25, 24, 27, 30, 16, 26, 22, 31] have dethroned traditional ones.

In this paper, we propose a new approach that belongs to Deep learning methods using service description. Named WSC2RCNN, our Web Service Classifier combines two Recurrent Convolutional Neural Networks for text classification.

The first one is applied to extract semantic features by capturing contextual information from Web services' embedded descriptions; the second network does the same thing but on words related to service actions generated from their original descriptions, this to highlight actions that reflect what a Web service really does.

After having done considerable experiments on a Dataset crawled from a well-known repository "ProgrammableWeb" with a good number of real-world web services, we have clearly shown that our proposed method can achieve a more precise classification and surpass the state-of-the-art methods. The rest of this paper is organized as follows.

Section 2 gives an overview of related work including Deep Learning approaches for Web services classification. Section 3 is dedicated to present our approach. Section 4 describes evaluation metrics used to measure different methods' classification performances. In section 5, we exhibit our experimental setup and results. Section 6 concludes the paper.

2 Related Work

The discovery of web services has aroused the interest of the research community and has seen significant activity. Considered as a flagship solution, research on Web services classification using Deep Learning techniques can be done using two main data sources [26]: Web services description documents and details collected from their ecosystem.

For instance, in [29] Web service clustering is accomplished by obtaining the effective words using information gain theory then combining them to an attention-based Bi-LSTM neural network. To seize the most relevant semantic information automatically in a Web service document, Cao et al. [5] adopted the Bi-LSTM architecture.

Then their model incorporated the Bi-LSTM architecture through a topical attention-mechanism to perform Web service classification prediction. In [30], a deep neural network model (DeepWSC) is trained based on probabilistic topic model and and taking advantage of the RCNN [11] capabilities (which has been customized), to get Web service description implicit contextual features.

The above methods share with ours the source of training data which is the description documents of Web services. However, in order to be able to provide a quality semantic information, they use, for the most, a probabilistic-topic-model to supervise this process. Although [30] benefits from RCNN, our work, however, differs from the latter in the trained-word-embedding model and the way of processing training data.

The rest of methods often combine the information collected from the Web services ecosystem with their descriptions. As a supervised method, Servnet [26] proposes a deep neural network to predict the service categories from service specifications, then automatically extract features from the service name and description to combine them under a unified feature in order to predict service classification.

In [22] authors realize automatic extraction of function-description-documents by providing a new deep neural network which integrates Graph-Convolutional-Network (GCN), to extract the global spatial features, with Bidirectional

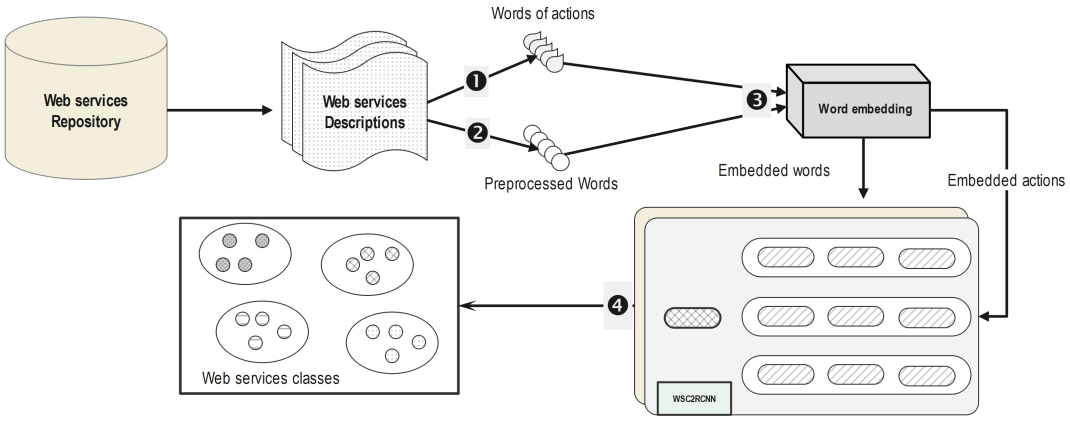

Fig. 1. WSC2RCNN framework for web services classification

Long Short Term Memory (Bi-LSTM) network, to learn sequential features. In an unsupervised manner, to achieve clustering, the second version of DeepWSC [31] gets Web services implicit features by combining deep neural network, to generate semantic features, with service composability relationship, extracted from an invocation convolutional network.

Authors in [28] also tried to design a supervised similarity knowledge graphs KSN to improve service description semantic. Associated to a CNN network, it is employed to extract context information. It is clear that the current trend is to develop graphs that attempt to capture different relationships and exploit them in order to improve the discovery process by finding specific types of patterns.

Even if DeepWSC [31] did good results by grafting a ”Service Composability Network” on the old version, our method was able to show better results. This is due, in the first place, to the way of treating the words of the descriptions, then, it has a great relation with the importance that we have given to the words which describe the actions of a web service.

As shown in section 5.3, our method gives very good results which exceed those mentioned, by taking into consideration the very nature of the words which appear in the descriptions of the web services.

3 Our Approach

Web Services are meant to execute operations and accomplish tasks, that’s what they are and that’s what defines them. However, in the most cases, and as representation data on them, we can only have short textual description. So to efficiently discover them through it, the question is:

“*which words (or parts) within these descriptions represent really what a service does?*”.

Our first intuition was that if we are in front of a text, actions contained in this text are represented naturally by ’verbs’. So, we started by extracting all possible verbs all over web services descriptions, by using pre-trained natural language models. Here, we faced two problems: The first one, is that these models can not detect all verbs correctly, for example the verb ’searches’ in the sentence:

“*Service X searches for all worldwide airlines that operate in a given country*”.

Is tagged as a ‘Noun’; What if we are going to take each word apart (and out of its context)?. The second problem is that since the majority of descriptions are short, it is hard to have significant number of verbs to work with, using these models.

What makes this process even harder is that after doing text cleaning and pre-processing step (especially for the Stemming part), many words loose their initial tag, so a lot of verbs are detected as nouns, adjectives ... etc.

Algorithm 1: Extract Actions

```
Input  : Web service textual description
Output: List of actions
Start parsing the sentence tree from the beginning;
if item is a verb and has xcomp then
    got to the xcomp node;
else
    if item is a verb and has dobj then
        add action to the list;
    else
        if item is a verb and has conjunction then
            go to the conjunction item; once you reach the noun, add action to the list;
        end
    end
end
```

For example, the following sentence: *"Service x is used to validate monetary transactions."* the words '*Service*', '*used*' and '*validate*' are tagged as follow:

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

This leads, in most cases, to changes in the semantic meaning of each word and consequently in the overall meaning of the description. Therefore, in our case, we tried to overcome this since the first step of our approach. Our method is based on four main steps (Figure 1):

We start in **Step 1** by a soft pre-processing of the services descriptions to preserve as possible words' original tags. Even this was not enough, because we get a very small number of verbs devoid of their context.

To overcome this, we used in **Step 2** the 'Extract-actions' algorithm, to extract what we qualify as actions and not only verbs.

After applying this process on every Web service description in the Dataset, in **Step 3**, we inject the pre-trained GloVe [19] model representation of both descriptions and extracted actions as inputs of the respectively two Recurrent Convolutional Neural Networks (RCNN) to be trained:

— **xcomp**: *An open clausal complement*,

— **dobj**: *The direct object*.

For both networks we used the original RCNN [11] which aims to capture text semantics by taking into account sequence words order. It makes the most of both RNN to learn local context of tokens and CNN for long-term dependencies. For the first RCNN network, we use $c_l(w_i)$ and $c_r(w_i)$ as, respectively, the left and right context of a word w_i, calculated using the following equations:

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

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

where f is a non-linear activation function. $e(w)$ is the word embedding vector of a word. $W^{(l)}$ is a matrix used to transform the context into the next hidden layer.

$W^{(sl)}$ is a matrix used to associate the semantic of the current word with the next word's left context. To get the latent semantic representation of the y_i vector, the equation (3) is used:

$$y_i = \tanh(Wx_i + b), \quad (3)$$

where x_i is the representation of a word w_i and

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)]. \quad (4)$$

For the second network, y_a (of actions) is calculated in a similar manner but applied to the words of the extracted actions. As targets of WSC2RCNN, both RCNN networks outputs are associated to the 20 high-quality Web services primary categories from the Dataset.

The results obtained of the two RCNN networks are then added to each other, in **Step 4**, to prepare the final output. Since we have a multi-class problem, a softmax activation layer is applied to normalize the final output named *M_output*. Figure 2 illustrates our model.

Table 1. The classification performances

Models	Purity	NMI	Recall	F1-Score
LDA+K	0.5200	0.4262	0.3199	0.3383
LDA	0.5285	0.4341	0.3321	0.3503
WE-LDA+K	0.5372	0.4363	0.3282	0.3466
WE-LDA	0.5420	0.4403	0.3370	0.3543
Text-CNN+WE-LDA+K	0.5553	0.4668	0.3572	0.3733
RCNN+WE-LDA+K	0.5708	0.4856	0.3821	0.3969
RCNN+WE-LDA+Heuristics	0.6379	0.5273	0.4186	0.4356
WSC2RCNN	*0.6288*	**0.5648**	**0.4754**	**0.4852**
WSC2RCNN+SC	**0.6470**	**0.5755**	**0.5045**	**0.5129**

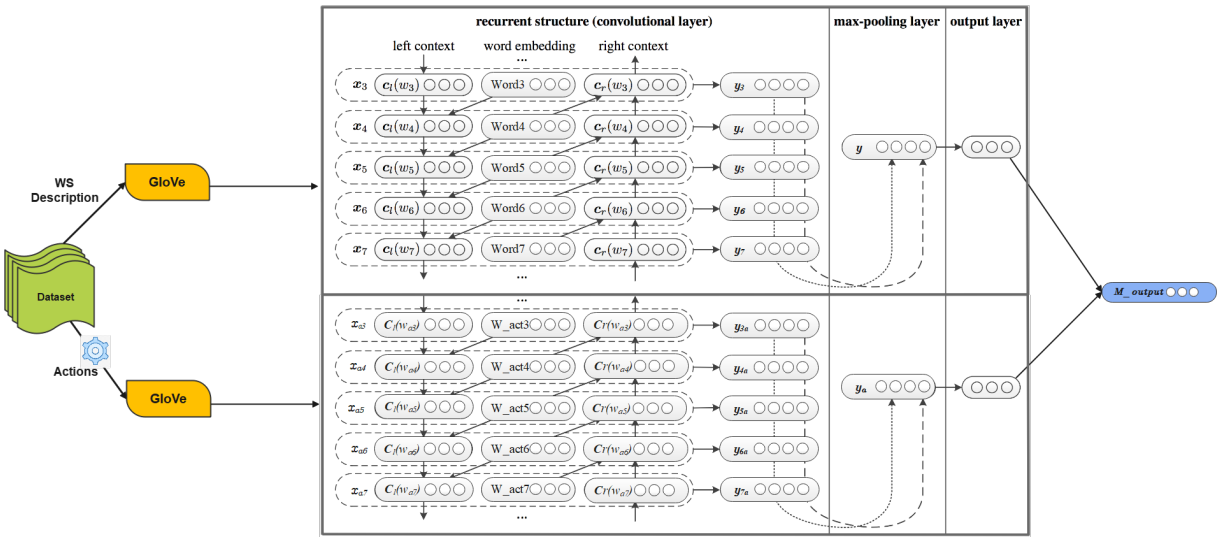

Fig. 2. Our model: WSC2RCNN

4 Used Evaluation Metrics

WSC2RCNN has been evaluated on four most used evaluation measures which are: Purity, Normalized-Mutual-Information (NMI), Recall and F_1-measure: **Purity** is one of the supervised cluster validation measures, calculated using the following formula:

$$\text{Purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{i=1}^{k} \max_{j} |\omega_i \cap c_j| , \qquad (5)$$

where $\Omega = \{w_1, w_2, ..., w_K\}$ represents the set of web services clusters and $\mathbb{C} = \{c_1, c_2, ..., c_J\}$ is the set of Web services classes. **NMI** is a measure based on the mutual information and defined as:

$$\text{NMI}(\Omega, \mathbb{C}) = \frac{2 \times I(\Omega; \mathbb{C})}{H(\Omega) + H(\mathbb{C})}, \qquad (6)$$

where we can get the mutual information *I* using:

$$I(\Omega; \mathbb{C}) = \sum_{i=1}^{k} \sum_{j=1}^{k} P(\omega_i \cap c_j) \log \frac{P(\omega_i \cap c_j)}{P(\omega_i) \cap P(c_j)}. \qquad (7)$$

Table 2. Distribution of Web services over the different categories

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

The entropy H is defined as:

$$H(\Omega) = -\sum_{i=1}^{k} P(\omega_i) \log P(\omega_i). \tag{8}$$

Recall is used to find out how much proportion of the true class is correctly predicted. It is calculated as follow:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{9}$$

where *TP* is the number of services assigned to their correct class and *FN* is the number of those where the model incorrectly predicts their positive class as negative. The last used measure is $\mathbf{F}_1$ which is calculated using the following formulas:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{10}$$

where *FP* refers to the number of services where the model incorrectly predicts their negative classes as positive:

$$F_1\text{-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{11}$$

5 Experiments and Results

5.1 Experimental Environment

We conducted our experiments to evaluate and show the effectiveness of WSC2RCNN, and all of them have been done using python and carried out on a platform with an Intel(R) Xeon(R) platinum 8259CL CPU@2.50GHz (32 cores) and 256 GB RAM.

Since ProgrammableWeb is the largest online Web Services registry with over 23K API, from it we used the Dataset[1] of [30, 31] which contains 17,923 crawled real-world web services.

The experimental data contains 8,459 best quality web services of the top 20 categories, see Table 2. We also used Glove6B with 100-dimensions word-vectors for our model's networks as pre-trained word-vectors.

5.2 Evaluation Methods

By applying the following evaluation metrics: Purity, Normalized-Mutual-Information (NMI), Recall and F1-measure; We have tested our model on the same Dataset and compared our results with the best existing state-of-art methods listed by the authors of [31] as follow:

LDA [7], each web service has its own vector of probabilistic-topic-distribution generated, Web services clustered together are those with the same latent topic. In **LDA+K** [3], the K-means++ algorithm uses the similarity between Web services' vectors from the previous method to cluster them.

For **WE-LDA** [20], Web services are clustered together when they have the same latent topic, after that, they will be attributed to their highest value latent topics, and for **WE-LDA+K**, the K-means++ algorithm uses the similarity between Web services' vectors from WE-LDA to cluster them.

The **Text-CNN+WE-LDA+K** [31] method is based on a Text-CNN [10] service feature-extractor trained with WE-LDA. The last two methods use both a trained RCNN network [11], the first one **(RCNN+WE-LDA+K)** [30] is

[1] https://github.com/aourhtnowvherlcaer/programmableWeb

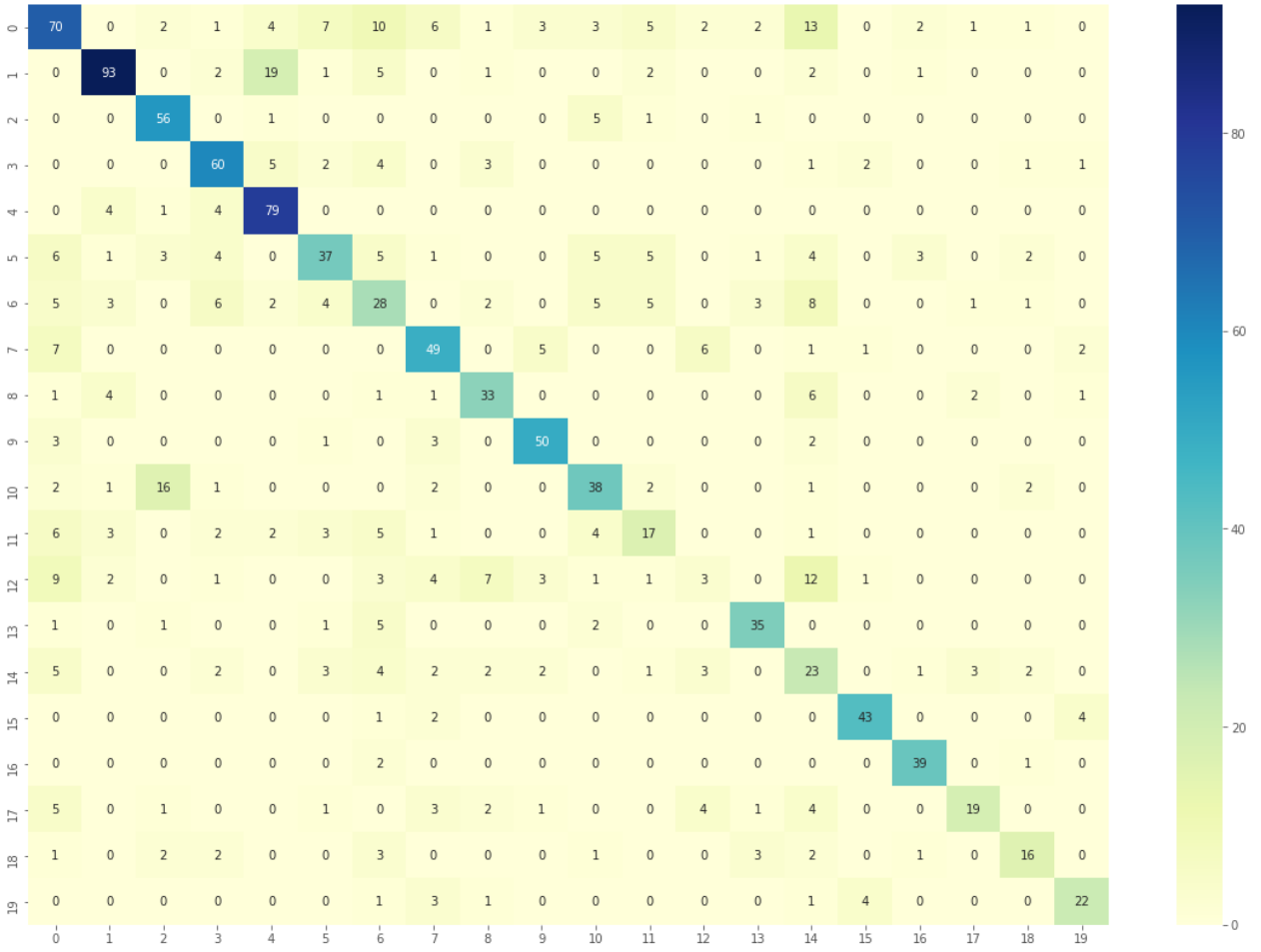

Fig. 3. M2RCNN Confusion Matrix

based on an LDA model. The second one **(RCNN+WE-LDA+Heuristics)** [31] uses service-deep-semantic-feature (RCNN) and service-composability-features (of a service composability built network) guided by a WE-LDA model.

5.3 Experimental Results and Discussions

By seeing Table 1, it is very clear that the first version of our approach exceeds all the candidate methods on the NMI, Recall and F_1 measures, and only the DeepWSC latest method with heuristics [31] exceeds it slightly on the Purity but it still gives very good results.

However, our second version, where we injected the embedded representation of secondary categories names (using GloVe), succeeded to outperform all candidate methods on all used measures.

Compared to LDA best performance, we notice an average improvement of 32.68% on all measures. If we take WE-LDA, the average of improvement is about 30.58%. On the best performance, which is DeepWSC with heuristics, we observed an average advantage of 7.66%.

The incorporation of the secondary categories names vectors to the combined [*actions*, *verbs*] feature on the second RCNN network did the work and improved obtained results.

Our new WSC2RCNN model outperforms the previous version with an average of 4.15% on all the evaluation measures. In addition to the good results we were able to have with our method, and if we look closely the confusion matrix, Figure 3, which is related to one of our models, we find that **WSC2RCNN** performs better than it looks.

For example, if we take line number 1, the model reacted well by making the prediction of the majority of web services in their correct class which is '***Financial***', whereas 19 Web services were predicted to belong to the class '***Payments***'. However, if we analyze the two classes from a semantic point of view, the two classes are quite close, which leads us to say that our model is not completely wrong on these predictions, since there is a relation between the two classes.

The same thing for the 16 services (line No.10) which have to be classified under the '***Telephony***' class but the model has assigned them to the '***Messaging***' class. From these results, we can see that our model gives positive results and takes into consideration the semantic aspect of Web services descriptions.

This leads us to believe that if we take into account the similarities between the classes, our method will give much better results than those already obtained. Without forgetting, the majority of works, dealing with the problem, perceive it as a classification problem where the classes are completely disjoint. This does not reflect the reality that a service can belong to several classes at the same time.

6 Conclusion and Future Works

In this paper, we propose a Web Service Classifier, named WSC2RCNN, based on Recurrent Convolutional Neural Networks for Text Classification (RCNN) and tested on a real-world Dataset with 8,459 web services.

The particularity of our approach is that it highlights the actions of web services extracted from their textual descriptions and accentuates the learning on these actions by training a dedicated and a customized RCNN Neural Network. Comparative experiments have shown that our model has been able to exceed all state-of-the-art approaches for web service classification on all the performance metrics. Since we have also shown that our model can give better results, in future work, we plan to extend our approach with more advanced techniques to take into account the semantic relations that exist between the classes of web services, in addition to an enhanced BERT embedding, adapted to our model to further results improvements.

In this work, the sequence of actions is analyzed and compared only within service descriptions. It will be more interesting to discover and capture sequences of inter-services actions in order to enrich the process of discovering web services with composite ones made up of other services and which offer the same expected functionalities.

References

1. **Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P., Verma, K. (2005).** Web service semantics - WSDL-S.
2. **Al-Masri, E., Mahmoud, Q. H. (2007).** Wsce: A crawler engine for large-scale discovery of web services. IEEE International Conference on Web Services (ICWS 2007), IEEE, pp. 1104–1111. DOI: 10.1109/ICWS.2007.197.
3. **Cao, B., Liu, X., Li, B., Liu, J., Tang, M., Zhang, T., Shi, M. (2016).** Mashup service clustering based on an integration of service content and network via exploiting a two-level topic model. IEEE International Conference on Web Services (ICWS), IEEE, pp. 212–219. DOI: 10.1109/ICWS.2016.35.
4. **Cao, B., Liu, X. F., Rahman, M. D. M., Li, B., Liu, J., Tang, M. (2020).** Integrated content and network-based service clustering and web apis recommendation for mashup development. IEEE Transactions on Services Computing, Vol. 13, No. 1, pp. 99–113. DOI: 10.1109/TSC.2017.2686390.
5. **Cao, Y., Liu, J., Cao, B., Shi, M., Wen, Y., Peng, Z. (2019).** Web services classification with topical attention based Bi-LSTM. International Conference on

Collaborative Computing: Networking, Applications and Worksharing, Springer International Publishing, pp. 394–407. DOI: 10.1007/978-3-030-30146-0_27.

6. **Cassar, G., Barnaghi, P., Moessner, K. (2014).** Probabilistic matchmaking methods for automated service discovery. IEEE Transactions on Services Computing, Vol. 7, No. 4, pp. 654–666. DOI: 10.1109/TSC.2013.28.
7. **Chen, L., Wang, Y., Yu, Q., Zheng, Z., Wu, J. (2013).** WT-LDA: user tagging augmented LDA for web service clustering. International conference on service oriented computing, Springer, Springer Berlin Heidelberg, pp. 162–176.
8. **Chen, W., Paik, I., Hung, P. C. K. (2015).** Constructing a global social service network for better quality of web service discovery. IEEE Transactions on Services Computing, Vol. 8, No. 2, pp. 284–298. DOI: 10.1109/TSC.2013.20.
9. **Elgazzar, K., Hassan, A. E., Martin, P. (2010).** Clustering WSDL documents to bootstrap the discovery of web services. 2010 IEEE international conference on web services, IEEE, pp. 147–154. DOI: 10.1109/ICWS.2010.31.
10. **Kim, Y. (2014).** Convolutional neural networks for sentence classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Association for Computational Linguistics, pp. 1746–1751. DOI: 10.3115/v1/d14-1181.
11. **Lai, S., Xu, L., Liu, K., Zhao, J. (2015).** Recurrent convolutional neural networks for text classification. Proceedings of the AAAI Conference on Artificial Intelligence, AAAI Press, Vol. 29, No. 1, pp. 2267–2273.
12. **Lara, R., Roman, D., Polleres, A., Fensel, D. (2004).** A conceptual comparison of WSMO and OWL-S. European Conference on Web Services, Springer, pp. 254–269. DOI: 10.1007/978-3-540-30209-4_19.
13. **Liu, X., Agarwal, S., Ding, C., Yu, Q. (2016).** An lda-svm active learning framework for web service classification. 2016 IEEE International Conference on Web Services (ICWS), IEEE, pp. 49–56. DOI: 10.1109/ICWS.2016.16.
14. **Maamar, Z., Faci, N., Wives, L., Badr, Y., Santos, P., De Oliveira, J. P. M. (2011).** Using social networks for web services discovery. IEEE Internet Computing, Vol. 15, No. 4, pp. 48–54. DOI: 10.1109/MIC.2011.27.
15. **Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M. (2005).** Bringing semantics to web services: The OWL-S approach. International Workshop on Semantic Web Services and Web Process Composition, Springer, pp. 26–42.
16. **Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., Gao, J. (2020).** Deep learning based text classification: A comprehensive review. arXiv preprint arXiv:2004.03705. DOI: 10.48550/ARXIV.2004.03705.
17. **Nabli, H., Djemaa, R. B., Amor, I. A. B. (2018).** Efficient cloud service discovery approach based on lda topic modeling. Journal of Systems and Software, Vol. 146, pp. 233–248. DOI: 10.1016/j.jss.2018.09.069.
18. **Nayak, R. (2008).** Data mining in web services discovery and monitoring. International Journal of Web Services Research (IJWSR), Vol. 5, No. 1, pp. 63–81. DOI: 10.4018/978-1-61520-684-1.ch012.
19. **Pennington, J., Socher, R., Manning, C. D. (2014).** Glove: Global vectors for word representation. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), Association for Computational Linguistics, pp. 1532–1543.
20. **Shi, M., Liu, J., Zhou, D., Tang, M., Cao, B. (2017).** WE-LDA: a word embeddings augmented lda model for web services clustering. 2017 IEEE International Conference on Web Services (ICWS), IEEE, pp. 9–16. DOI: 10.1109/ICWS.2017.9.

21. **Skoutas, D., Sacharidis, D., Simitsis, A., Sellis, T. (2010).** Ranking and clustering web services using multicriteria dominance relationships. IEEE Transactions on Services Computing, Vol. 3, No. 3, pp. 163–177. DOI: 10.1109/TSC.2010.14.

22. **Wang, X., Liu, J., Liu, X., Cui, X., Wu, H. (2020).** A spatial and sequential combined method for web service classification. Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data, Springer, pp. 764–778. DOI: 10.1007/978-3-030-60259-8_56.

23. **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.

24. **Xiong, R., Wang, J., Zhang, N., Ma, Y. (2018).** Deep hybrid collaborative filtering for web service recommendation. Expert systems with Applications, Vol. 110, pp. 191–205. DOI: 10.1016/j.eswa.2018.05.039.

25. **Xu, J., Wang, P., Tian, G., Xu, B., Zhao, J., Wang, F., Hao, H. (2015).** Short text clustering via convolutional neural networks. Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, pp. 62–69. DOI: 10.3115/v1/w15-1509.

26. **Yang, Y., Qamar, N., Liu, P., Grolinger, K., Wang, W., Li, Z., Liao, Z. (2018).** Servenet: A deep neural network for web services classification. 2020 IEEE International Conference on Web Services (ICWS), pp. 168–175. DOI: 10.48550/ARXIV.1806.05437.

27. **Yao, L., Mao, C., Luo, Y. (2018).** Graph convolutional networks for text classification. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, No. 1, pp. 7370–7377. DOI: 10.48550/ARXIV.1809.05679.

28. **Yu, Y., Zeng, J., Yao, J., Wen, J., Xing, B. (2020).** Web service discovery based on knowledge graph and similarity network. 2020 IEEE World Congress on Services, IEEE, pp. 231–236. DOI: 10.1109/SERVICES48979.2020.00054.

29. **Zhang, X., Liu, J., Cao, B., Xiao, Q., Wen, Y. (2019).** Web service discovery based on information gain theory and bilstm with attention mechanism. International Conference on Collaborative Computing: Networking, Applications and Worksharing, Springer, pp. 643–658. DOI: 10.1007/978-3-030-12981-1_45.

30. **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. 2019 IEEE International Conference on Web Services (ICWS), IEEE, pp. 434–436. DOI: 10.1109/ICWS.2019.00077.

31. **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 27/10/2021; accepted on 11/08/2022.
Corresponding author is Meghazi Hadj Madani.