The CSI Journal on
Computer Science and Engineering
Vol. 18, No. 2, 2021
Pages 53-60
Regular Research Paper

Recommendation in Online Q&A Communities Based on BERT Pre-training Technique

Navid Khezriyan Jafar Habibi Issa Anamoradnejad

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Abstract

Open-source and online Q&A communities make use of tags and keywords for indexing, classification, and thematic search. In this study, we propose TagBERT, a novel model for recommending tags on new questions, which makes use of a combination of deep learning and BERT models. In this model, first, the processed sentences are converted into numerical vectors by means of the BERT tokenizer. Next, the attributes are extracted using the CNN network, and, afterward, the DNN network is trained on the extracted attributes in order to recommend tags. To evaluate our model, we used four datasets, i.e. Free-code, UNIX, WordPress, and Software Engineering. Our proposed model obtained the highest precision score over baseline deep-learning and conventional methods. In contrast to previous studies in which precision was significantly reduced as a result of increased recommended tags, the precision of our model did not remarkably vary with an increase in the number of tags.

Keywords: Tag recommendation, Online Q&A communities, Open-source communities, Classification, BERT.

1. Introduction

Tags have recently become popular as a useful way of organizing and summarizing user-generated content due to their ease of indexing and user involvement. A tag is a keyword that describes the content and facilitates the classification of keywords as well as the search for information. The unique ability of tags to group and share information has changed the way people use information [1,2]. Tag recommendation is aimed at describing and organizing the content of objects. It not only improves the user experience by contributing to the disambiguation of queries but also may enhance the quality of the produced tags and indirectly improve the quality of information retrieval services so that the users could achieve more appropriate results in their queries. In addition, effectiveness is of great importance, given that weak recommendations will both lead to user's dissatisfaction and ultimately cause damage to information retrieval services [3].

In open-source or Q&A communities, objects include project name or question title, project description or reply to questions, and tags. When a developer sends a question to a Q&A community or shares a project in an open-source community, the website requires them to select several tags for the post. Free-code, for example, allows the users to create more than 10 tags for each post [4].

In this study, we make use of a BERT-based deep-learning model. BERT [5], which stands for Bidirectional Encoder Representations from Transformer, has proved to be useful for various NLP tasks such as questions and answering natural language inference. In contrast to directional models, which read the input text either from the left or from right, the transformer encoder reads the words without direction, which helps it to learn the context of a word both on the left and on the right.

In this paper, we will report the results of the implementation of our model on four datasets, i.e. Free-code, Wordpress, Unix, and Software Engineering, which are online Q&A and open-source communities, and compare the results with those of TagCNN (based on the convolutional neural network) [6], TagRNN (based on recurrent neural network) [7], TagHAN (based on hierarchical attention networks) [8], and TagRCNN (recurrent convolutional neural network) [9] which are widely used for classification of texts. We also compare our results

with traditional approaches to tag recommendation including TagMulRec [10], EnTagRec [11], and FastTagRec[12].
Given the conclusion drawn in [4] that all deep-learning methods cannot provide better results than traditional methods, our main question is whether a BERT-based deep-learning method can lead to better results than traditional tag recommendation methods.
The paper is organized as follows. Section 2 reviews the previous studies of tag recommendation. Section 3 describes the proposed method. In Section 4, the evaluation results are presented and Section 5 draws some conclusions.

2. Literature Review

In recent years, researchers have proposed various methods of tag recommendation. In general, tag recommendation methods can be divided into six categories based on their underlying techniques. This categorization was provided by Belém et al. [3] in 2016. By surveying the majority of studies conducted after [3], we observed that these categories are implied in almost all of the studies. Following is a brief description of the six categories.

2.1. Co-occurrence-Based Methods

These methods recommend tags for a new object by using the community rules and the previously allocated tags to existing objects. This makes them dependent on a large set of tagged data. The main challenge here is the high complexity of computation due to the number of tags which grows exponentially. Co-occurrence-based methods may be seriously affected by a lack of previous information about the target (i.e. cold start). Examples of these methods are [4, 13, 14].

2.2. Graph-Based Methods

Graph-based methods extract the recommended tags from the neighborhood of the object in question. The nodes are objects and the vertices represent the similarity between objects. Graph-based methods are less affected by the cold start problem than are the co-occurrence-based methods. However, they suffer from the noise produced by content and textual features. Studies that fall within this category are [15,16,17].

2.3. Matrix Factorization-Based Methods

These methods take the tag as a model matrix and try to reduce the dimensions of the matrix. Their aim is to recommend tags by predicting the relationships among users, tags, and objects. These methods work well in cases where there is sufficient data about users, tags, and objects. Also, the noise will decrease as the matrices become smaller in size. Examples are [19,20].

2.4. L2R-Based Methods

As tag recommendation is modeled as a ranking problem in which more appropriate results are recommended first, these methods are considered as a natural approach to this task. L2R-based methods recommend tags as a feature vector by learning automatically from a training set. These methods seek to automatically combine the qualitative variables and features of the tag and produce a model which ranks these features in terms of scores or positions based on the aim of the recommendation. L2R-based methods can make use of many features of ranking functions and, in addition, can be easily extended to cover more objective functions and features. Their main disadvantage is the time required for learning tag recommendation models. However, the training phase can be performed off-line [21,22].
These methods have been pursued in [21,22,23,24].

2.5. Content-Based Methods

Content-based methods recommend tags based on the content of the target and its related features or the target user's features. These methods are usually used to mitigate the cold start problem. In problems which are approached through content-based methods, there is much noise in content and textual features to be dealt with [25]. Examples of these methods are [26,27,28,29].

2.6. Clustering-Based Methods

These methods group objects and tags and recommend representative tags from the cluster of the target object. Clustering is an interesting strategy for reducing the dimensions of the problem. These methods produce representative tags by using the relationships among clusters. Representative tags may not be general enough; therefore, clustering-based methods are the least useful for describing specific content or separating such content from others. In recommending a tag for a new object, the object is first put in a cluster (or class) and the tags attributed to that cluster will be then recommended for that object. An instance of these methods can be observed in [30].
As stated in [3], what is important about the studies conducted so far is that most of them seek to combine multiple methods and data sources to achieve an optimum solution. In [33,34,35,36], for instance, co-occurrence-based, content-based, and L2R-based methods are used simultaneously for tag recommendation.
After putting the problem in one of the above six categories, it is important to adopt a suitable approach to solve the problem. In recent years, deep-learning algorithms have been successfully utilized in some cases to find more optimum answers than traditional methods.
In 2019, Zhou et al. [4] compared traditional tag recommendation methods with deep-learning algorithms on different datasets. They compared three traditional methods, i.e. En-TagRec, TagMulRec, and FastTagRec, with TagCNN, TagRNN, TagHAN, and TagRCNN.They concluded that TagCNN and TagRCNN were superior to the traditional methods on all their data sets while the other deep-learning methods did not achieve desirable results.

3. The Proposed Solution

In this section, we first state the problem of tag recommendation for textual data and then apply our TagBERT method to solve this problem.

The main question in the tag recommendation problem is how we can automatically recommend a set of tags for a new object in a set of objects which are already tagged. We developed the TagBERT model to answer this question. This model falls within the category of co-occurrence-based and content-based methods.

Figure 1 illustrates how TagBERT solves this problem. The figure is divided into eight parts (A-G).

Part A is the input unit of TagBERT. The input is composed of three parts including ID, text, and tag. We have selected the data used in [4] so that we could compare our results with that study. The data are XML files that hold different pieces of information about the above-mentioned open-source and Q&A communities. As described earlier in the introduction, we need to extract project name or question title, project description or answer to the question, and tags. The Data Cleaning component processes the extracted attributes through actions such as transliterating, removing HTML tags, removing characters other than numbers and digits (?, [,], =, <, etc.), removing less frequent tags [4], removing stop words, removing the space before and after opening or closing HTML tags, and removing multiple spaces. Next, the title of the project or the question title, the project description or, and every tag are given a binary value. The value is 1 if the tag has been assigned to an object; otherwise, it is 0. Each row of the table is considered as an object. The set of these objects makes up the input.

Part B of Figure 1 illustrates the preparation of objects for being delivered to the BRTT module. For this purpose, the ID and text sections of the object set are divided into several portions. Each portion has a size of 32.

In part C, each portion created in B undergoes tokenization. In this operation, an ID is allocated to each token and the length of all objects is set to the length of the largest object in that portion. In doing so, tokens with an ID of 0 are added to the end of shorter objects in order to make all objects of equal length. Each portion is now sent to theBERT module. The BERT component consists of 12 encoder layers and a large feed-forward network comprised of 768 hidden units and 12 attention heads. BERT takes a sequence of words as input which are constantly streaming on the stack. Each layer applies self-attention and transmits the results to the next encoder through a feed-forward network. Finally, BERT module performs the sentence embedding of objects.

In part D, a CNN and a DNN receive the sentence embedding created by theBERT module along with the tags in order to perform classification and specify the probability of each tag being appropriate for each object.

Part E shows the CNN model. In this part, each embedded sentence is given to CNN as input. In CNN, each input is inserted into a matrix in which each row represents an embedded word and then pushed through the convolution layer. The convolution layer scans the text and decides whether or not each attribute is compatible with the corresponding tag. Finally, the max pooling technique is utilized to reduce computation complexity in CNN. This technique decreases the output size from one layer to another by selecting the largest element in the pooling window. TagBERT makes use of a CNN in which the size of regions is set as (2,3,4) and there are 50 filters for each region.

Part F shows the structure of DNN. DNN has been designed to learn by multi-layer connection in a way that each layer only receives the connection from the previous layer and provides the connections of the next layer in the hidden unit. In TagBERT, the output of CNN is given as input to DNN with 256 units in order to conduct the learning process. The limitation in the output layer is a multi-label problem in which each object can have several tags. By examining all tags for all objects, TagBERT turns the multi-label problem into a binary problem. Thus, for every tag, it asks whether the tag belongs to the object in question or not. As a result, the sigmoid function is used for two classifications in the output layer.

In part G, a limit is determined for the values produced by sigmoid function which range between 0 and 1. This limit is then used for tag recommendation. In our study, this limit was specified as 0.92. Tags with a value greater than this limit will be recommended for the object; otherwise, they will not be recommended.

4. Experiments

As in [4], we repeated the model 10 times and reported the average values as the results.

We randomly selected 10000 objects from the datasets as our test set (V) and used the rest of the objects for the training process.

For every object in our test set ($O_i \in V$), a maximum of K tags were presented for creating the set of recommended tags (TR_i^K) if the object was qualified as being able to have K tags.

4.1. Evaluation

All experiments were performed on Ubuntu v18.04 on a 64-bit system with Intel Core i7 and 64 GB of RAM. The model was implemented using the open-source library TensorFlow.

To be compatible with the results presented in [12,31,32], we evaluated the recommended and actual tags using Recall@K, Precision@K, and F1-score@K.

We calculate Recall@K_i using equation (1). Also, equation (2) is used to calculate Recall@K for all objects in the test set (V).

$$Recall@K_i = \begin{cases} Recall@K_i = \frac{|TR_i^K \cap O_iT|}{K}, & |O_iT| > K \\ Recall@K_i = \frac{|TR_i^K \cap O_iT|}{O_iT}, & |O_iT| \leq K \end{cases} \quad (1)$$

$$Recall@K = \frac{\sum_{i=1}^{|V|} Recall@K_i}{|V|} \quad (2)$$

Precision@K_i is calculated using (3) and equation (4) is used to calculate Precision@K for all objects in the test set (V).

$$Precision@K_i = \frac{|TR_i^K \cap O_iT|}{K} \quad (3)$$

$$Precision@K = \frac{\sum_{i=1}^{|V|} Precision@K_i}{|V|} \quad (4)$$

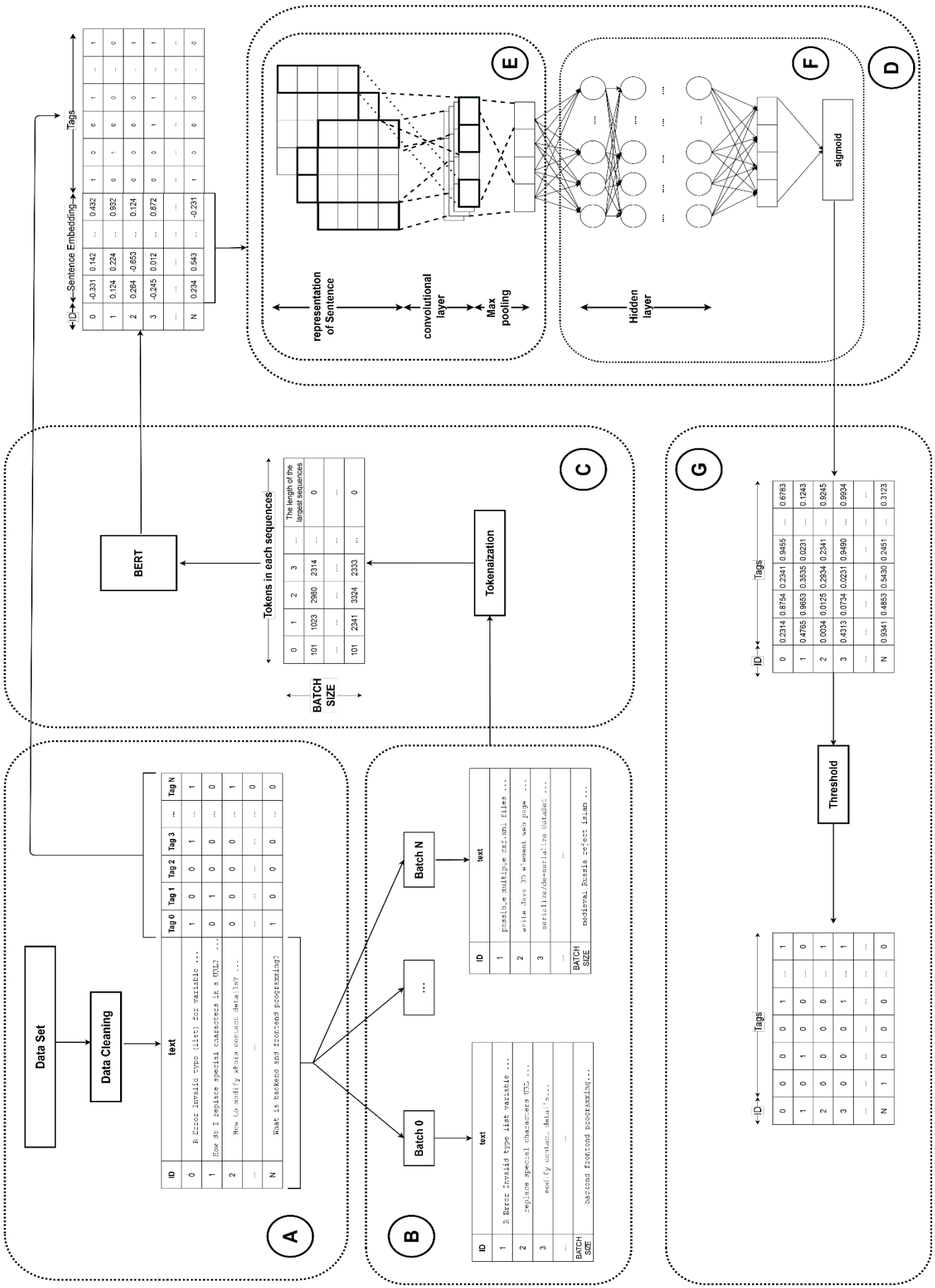

Figure. 1. The workflow of TagBERT

Finally, F1-score@ K_i is calculated by combining Recall@K_i and Precision@K_i according to equation (5). F1-score@K for all objects in the test set (V) can be calculated using (6).

$$F1\text{-}score\ @K_i = 2.\frac{Precision@K_i\ .\ Recall@K_i}{Precision@K_i + Recall@K_i} \quad (5)$$

$$F1\text{-}score\ @K = \frac{\sum_{i=1}^{|V|} F1-score\ @K_i}{|V|} \quad (6)$$

We use TagBERT for training on pre-processed data and make use of the test set for evaluating the model based on the mentioned criteria with K=10. The results of the experiments are presented in Tables 1 to 6.

5. Results

Tables 1 to 3 contain five columns. The first column represents the methods of tag recommendation. Rows 1-3 in this column are traditional methods while rows 4-7 are deep-learning methods and row 8 is our proposed method, i.e. TagBERT. Columns 2-5 in these tables represent the four datasets investigated in our study.

Table 1 lists the results of F1-score. As this criterion is a combination of recall and precision values, it is regarded as an important evaluation criterion. As can be seen in Table 1, TagBERT performs better than the best performance of traditional methods by 10.1 percent on Free-code dataset. It also shows a slight improvement in comparison with the best deep-learning method. Overall, Software Engineering dataset has resulted in a 9.8-percent improvement in the best method of previous studies. TagBERT has also created 5.8 and 11.3 percent of improvement on Wordpress and Unix datasets, respectively. It should be noted that, due to the large number of classes and fine-grained classification of objects in tag recommendation problems, it is extremely difficult to achieve high levels of precision and F1-score in this type of problem.

Table 1: Comparison of TagBERT with traditional and deep-learning methods on F1-score@10

Method	Site Name			
	Free code	Software Engineering	Wordpress	Unix
TagMulRec	36.4	25.2	26.6	27.1
EnTagRec	36	24.8	30.5	-
FastTagRec	33.2	25.7	28.3	28.2
TagCNN	45.3	36.5	35.6	36.7
TagRNN	20.8	21.6	27.6	29.1
TagHAN	23.3	22.1	28.0	29.2
TagRCNN	39.2	28.7	32.8	35.2
TagBERT	46.5	46.3	41.4	**48.0**

Table 2 presents the results of precision. As can be seen, TagBERT has created an improvement of 10.2, 14.1, 9.2, and 16.7 percent on Free-code, Software Engineering, Wordpress, and Unix.

Table 2: Comparison of TagBERT with traditional and deep-learning methods on Precision@10

Method	Site Name			
	Free code	Software Engineering	Wordpress	Unix
TagMulRec	24.5	15.3	16.3	16.9
EnTagRec	23.9	15.1	18.6	-
FastTagRec	21.9	15.7	17.3	18.2
TagCNN	29.7	22.5	21.9	23.1
TagRNN	13.8	13.5	17.0	18.4
TagHAN	15.1	13.6	17.2	18.3
TagRCNN	25.8	17.7	20.3	22.2
TagBERT	**40.2**	**36.6**	**31.1**	**38.9**

Table 3 presents the results of recall. It can be seen that our model is weaker than the other models in terms of recall. The reason is that we have attempted to establish a balance among the criteria and achieve acceptable precision. As a result, the parameters of our model have been set in a way that an acceptable level of accuracy could be achieved for both recall and precision, in contrast to previous models in which precision decreases significantly as the number of recommendations increase.

Table 3: Comparison of TagBERT with traditional and deep-learning methods on Recall@10

Method	Site Name			
	Free code	Software Engineering	Wordpress	Unix
TagMulRec	75.8	70.4	72.5	68.2
EnTagRec	77.3	69.7	86.4	-
FastTagRec	82	70.8	76.5	72.2
TagCNN	**94.9**	**96.1**	**94.5**	**89.9**
TagRNN	41.6	53.5	73.1	69.0
TagHAN	48.4	59.9	76.0	72.7
TagRCNN	81.6	76.4	86.3	84.8
TagBERT	64.4	76.0	75.6	74.3

Tables 4 to 7 contain three columns. The first column represents the methods of tag recommendation. Rows 1-3 in this column are traditional methods while rows 4-7 are deep-learning methods and row 8 is our proposed method, i.e. TagBERT. Columns 2 and 3 show the results of evaluation of precision for K=5 and K=10.

Table 4 lists the results for Free-code dataset. TagBERT has led to a reduction of only 1.63 percent in precision with increase in the number of recommendations whereas TagCNN, which has the highest precision among previous methods, has led to a reduction of 21.1 percent. Table 5 shows the results of Software Engineering dataset. In this dataset, TagBERT shows a reduction of 0.9 in precision while TagCNN, which has performed better than the the other previous methods on this dataset, has led to a reduction of 17.8 percent. The results of Wordpress dataset are displayed in Table 6. Reduction in precision is only 1.7

percent for TagBERT while TagCNN has led to a reduction of 17.8 percent. Finally, the results of Unix dataset are represented in Table 7. On this dataset, our proposed model has only led to a reduction of 0.5 percent in precision with increase in the number of recommendations while TagRCNN, which has had the highest performance with K=5, has faced a reduction of 19.4 percent as the number of recommendations increased to K=10. Overall, it can be concluded that, with increase in the number of recommendations, TagBERT has led to the least amount of reduction in precision while the other models have suffered a significant reduction.

Table 4: Comparison of the performance of TagBERT with traditional and deep-learning methods (with increase in the number of recommended tags from K=5 to K=10) in terms of precision for Free-code dataset.

Method	Precision@5	Precision@10
TagMulRec	26.5	16.3
EnTagRec	34.8	18.6
FastTagRec	27.8	17.3
TagCNN	39.7	21.9
TagRNN	28.3	17.0
TagHAN	28.5	17.2
TagRCNN	37.4	20.3
TagBERT	**32.8**	**31.1**

Table 5: Comparison of the performance of TagBERT with traditional and deep-learning methods (with increase in the number of recommended tags from K=5 to K=10) in terms of precision for Software Engineering dataset.

Method	Precision@5	Precision@10
TagMulRec	25.3	15.3
EnTagRec	24.5	15.1
FastTagRec	25.2	15.7
TagCNN	43.6	22.5
TagRNN	21.7	13.5
TagHAN	22.1	13.6
TagRCNN	32.5	17.7
TagBERT	**37.5**	**36.6**

Table 6: Comparison of the performance of TagBERT with traditional and deep-learning methods (with increase in the number of recommended tags from K=5 to K=10) in terms of precision for Wordpress dataset.

Method	Precision@5	Precision@10
TagMulRec	26.5	16.3
EnTagRec	34.8	18.6
FastTagRec	27.8	17.3
TagCNN	39.7	21.9
TagRNN	28.3	17.0
TagHAN	28.5	17.2
TagRCNN	37.4	20.3
TagBERT	**32.8**	**31.1**

Table 7: Comparison of the performance of TagBERT with traditional and deep-learning methods (with increase in the number of recommended tags from K=5 to K=10) in terms of precision for Unix dataset

Method	Precision@5	Precision@10
TagMulRec	29.4	16.9
EnTagRec	-	-
FastTagRec	30.9	18.2
TagCNN	30.9	23.1
TagRNN	30.3	18.4
TagHAN	30.2	18.3
TagRCNN	41.6	22.2
TagBERT	**39.4**	**38.9**

6. Conclusion

In this study, we sought to propose a novel BERT-based deep learning approach to the task of tag recommendation in online Q&A and open-source communities. In addition to better performance than other deep-learning and traditional methods of tag recommendation, TagBERT could also remedy a common deficiency of previous methods, i.e. the fact that their precision dropped significantly with increase in the number of recommended tags.

The results show that our proposed model outperforms the traditional and the best deep-learning methods.

We believe that deep learning can make a great contribution to the quality of online Q&A and open-source communities. Therefore, it is suggested that different deep-learning methods be closely examined and analyzed to achieve even better results for the problem of tag recommendation.

References

[1] F. Figueiredo, H. Pinto, F. Belém, J. Almeida, M. Gonçalves, D. Fernandes, E. Moura, Assessing the quality of textual features in social media, Information Processing & Management 49 (1) (2013) 222–247. doi:10.1016/j.ipm.2012.03.003.

[2] X. Li, L. Guo, Y. E. Zhao, Tag-based social interest discovery, in: and others (Ed.), Proceedings of the 17th international conference on World Wide Web, (2008), pp. 675–684.

[3] F. M. Belém, J. M. Almeida, M. A. Gonçalves, A survey on tag recommendation methods, Journal of the Association for Information Science and Technology 68 (4) (2017) 830–844. doi:10.1002/asi.23736.

[4] P. Zhou, J. Liu, X. Liu, Z. Yang, J. Grundy, Is deep learning better than traditional approaches in tag recommendation for software information sites?, Information and Software Technology 109 (2019) 1–13. doi:10.1016/j.infsof.2019.01.002.

[5] J Devlin, MW Chang, K Lee, K Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805* (2018).

[6] N Kalchbrenner, E Grefenstette, P Blunsom, A convolutional neural network for modelling sentences, *arXiv preprint arXiv:1404.2188* (2014).
[7] P Liu, X Qiu, X Huang, Recurrent neural network for text classification with multi-task learning, *arXiv preprint arXiv:1605.05101* (2016)
[8] Z Yang, D Yang, C Dyer, X He, A Smola, Hierarchical attention networks for document classification, Proceedings of the 2016 confer- ence of the North American chapter of the association for computational linguistics: human language technologies (2016).
[9] S Lai, L Xu, K Liu, J Zhao, Recurrent convolutional neural networks for text classification, Twenty-ninth AAAI conference on artificial intelligence (2015).
[10] S Wang, D Lo, B Vasilescu, A Serebrenik, EnTagRec++: An enhanced tag recommendation system for software information sites, Empirical Software Engineering 23 (2018) 800–832.
[11] P Zhou, J Liu, Z Yang, G Zhou, Scalable tag recommendation for software information sites, IEEE 24th International Confer- ence on Software Analysis, Evolution and Reengineering (SANER) (2017).
[12] J Liu, P Zhou, Z Yang, X Liu, J Grundy, FastTagRec: fast tag recommendation for software information sites, Automated Software Engineering 25 (2018) 675–701.
[13] GV Menezes, JM Almeida, F Belém, Vale, Demand-driven tag recommendation, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2010.
[14] H Yu, B Zhou, M Deng, F Hu, Tag recommendation method in folksonomy based on user tagging status, Journal of Intelligent Information Systems 50 (2018) 479–500.
[15] X Xia, D Lo, X Wang, B Zhou, Tag recommendation in software information sites, 10th Working Conference on Mining Software Repositories (MSR) (2013).
[16] X Cai, J Zhu, B Shen, Y Chen, Greta: Graph-based tag assignment for github repositories, IEEE 40th Annual Computer Software and Applications Conference (COMPSAC) 1 (2016).
[17] M. Hmimida, R. Kanawati, A graph-based meta-approach for tag recommendation, Springer, Cham, (2016).
[18] R Jäschke, L Marinho, A Hotho, Tag recommendations in folksonomies, *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, Berlin, Heidelberg, (2007).
[19] M Shi, J Liu, D Zhou, Y Tang, A topic-sensitive method for mashup tag recommendation utilizing multi-relational service data, IEEE Transactions on Services Computing (2018).
[20] S. Rendle, L. Schmidt-Thieme, Pairwise interaction tensor factorization for personalized tag recommendation, Proceedings of the third ACM international conference on Web search and data mining (2010).
[21] SD Canuto, FM Belém, JM Almeida, A comparative study of learning-to-rank techniques for tag recommendation, Journal of Information and Data Management 4 (2013) 453–453.
[22] T. Qin, T.-Y. Liu, H. Li, A general approximation framework for direct optimization of information retrieval measures, Information Retrieval 13 (4) (2010) 375–397. doi:10.1007/s10791-009-9124-x.
[23] H Cao, M Xie, L Xue, C Liu, F Teng, Social tag prediction base on supervised ranking model, Proceeding of ECML/PKDD 2009 Discovery Challenge Workshop (2009).
[24] F Belém, E Martins, T Pontes, J Almeida, Associative tag recommendation exploiting multiple textual features, Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval (2011).
[25] F. Figueiredo, H. Pinto, F. Belém, J. Almeida, M. Gonçalves, D. Fernandes, E. Moura, Assessing the quality of textual features in social media, Information Processing & Management 49 (1) (2013) 222– 247. doi:10.1016/j.ipm.2012.03.003.
[26] SK Maity, A Panigrahi, S Ghosh, A Banerjee, DeepTagRec: A Content-cum-User Based Tag Recommendation Framework for Stack Overflow, Springer, Cham, 2019.
[27] Y Wu, S Xi, Y Yao, F Xu, H Tong, J Lu,Guiding supervised topic modeling for content based tag recommendation, Neurocomputing 314 (2018) 479–489.
[28] D Kowald, S Kopeinik, P Seitlinger, T Ley, Refining frequency-based tag reuse predictions by means of time and semantic context, in: Mining, Modeling, and Recommending'Things' in Social Media, Springer, 2013, pp. 55–74.
[29] F. M. Belém, A. G. Heringer, J. M. Almeida, M. A. Gonçalves, Exploiting syntactic and neighbourhood attributes to address cold start in tag recommendation, Information Processing & Management 56 (3) (2019) 771–790. doi:10.1016/j.ipm.2018.12.009.
[30] Y Song, Z Zhuang, H Li, Q Zhao, J Li, WC Lee, Real-time automatic tag recommendation, Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, (2008), pp. 515-522
[31] S. Wang, D. Lo, B. Vasilescu, A. Serebrenik, EnTagRec: An Enhanced Tag Recommendation System for Software Information Sites, in: 2014 IEEE International Conference on Software Maintenance and Evolution, 2014, pp. 291–300.
[32] P Zhou, J Liu, Z Yang, G Zhou, Scalable tag recommendation for software information sites, IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER) (2017).
[33] FM Belém, CS Batista, RLT Santos, Beyond relevance: explicitly promoting novelty and diversity in tag recommendation, ACM Transactions on Intelligent Systems and Technology (TIST) 7 (3) (2016) 1–34.
[34] SD Canuto, FM Belém, JM Almeida, A comparative study of learning-to-rank techniques for tag recommendation, Journal of Information and Data Management 4 (2013) 453–453.
[35] L Wu, L Yang, N Yu, XS Hua, Learning to tag, Proceedings of the 18th international conference on World wide web (2009).
[36] E. F. Martins, F. M. Belém, J. M. Almeida, M. A. Gonçalves, On cold start for associative tag recommendation, Journal of the Association for Information Science and Technology 67 (1) (2016) 83–105. doi:10.1002/asi.23353.

Navid Khezrian is currently an M.Sc. student in Computer Networks at Sharif University of Technology. He received his B.Sc. degree in Software Engineering from Bu-Ali Sina University.
Email: navidkhezrian@gmail.com

Jafar Habibi is an Associate Professor in the Department of Computer Engineering, Sharif University of Technology, and the managing director of Intelligent Information Solutions Center. He is a supervisor of Sharif's Robo-Cup Simulation Group and Software Engineering Lab. His research interests are mainly in the areas of software engineering, simulation systems, MIS, DSS, and evaluation of computer systems performance.
Email: jhabibi@sharif.edu

Issa Annamoradnejad is currently a Ph.D. candidate in Software Engineering at Sharif University of Technology. He received his M.Sc. degree from the Sharif University of Technology, Tehran, Iran. He has led multiple research projects aimed at utilizing machine-learning approaches in multi-disciplinary areas.
Email: moradnejad@ce.sharif.edu

Paper Handling Data:
Submitted: 07-25-2020
Received in revised form: 12-28-2020
Accepted: 02-03-2021
Corresponding author: Dr. Jafar Habibi
Department of Computer Engineering, Sharif University of Technology