Scholarly Text Classification with Sentence BERT and Entity Embeddings

Guangyuan Piao[0000−0003−0516−2802]

Department of Computer Science, Hamilton Institute
Maynooth University,
Maynooth, Co Kildare, Ireland
guangyuan.piao@mu.ie

Abstract. This paper summarizes our participated solution for the shared task of the text classification (scope detection) of peer review articles at the SDPRA (Scope Detection of the Peer Review Articles) workshop at PAKDD 2021. By participating this challenge, we are particularly interested in how well those pre-trained word embeddings from different neural models, specifically transformer models, such as BERT, perform on this text classification task. Additionally, we are also interested in whether utilizing entity embeddings can further improve the classification performance. Our main finding is that using SciBERT for obtaining sentence embeddings for this task provides the best performance as an individual model compared to other approaches. In addition, using sentence embeddings with entity embeddings for those entities mentioned in each text can further improve a classifier's performance. Finally, a hard-voting ensemble approach with seven classifiers achieves over 92% accuracy on our local test set as well as the final one released by the organizers of the task. The source code is publicly available at https://github.com/parklize/pakdd2021-SDPRA-sharedtask.

Keywords: Text classification · Word embeddings · Sentence embeddings · Entity embeddings.

1 Introduction

Text classification is a crucial task in Natural Language Processing (NLP) with a wide range of applications such as scope detection of scholarly articles [5], sentiment classification [7], and news topic classification [11]. For example, identifying the topics or category of scientific articles, can help us efficiently manage a large number of articles, retrieve related papers, and build personal recommendation systems. In this report, we present the details of our solution for scholarly text (abstract) classification in the context of a shared task at the SDPRA (Scope Detection of the Peer Review Articles) workshop [8][1] at PAKDD 2021. By participating this task, we are particularly interested in understanding (1) how good the transfer learning performance is based on pre-trained transformer networks

[1] https://sdpra-2021.github.io/website/

designed for sentence/text embeddings, and (2) whether entity embeddings for those mentioned entities in a text help to improve the classification performance.

1.1 Task description

The shared task is a multi-class classification problem, which aims to classify each given abstract of a scholarly article into one of seven classes. Overall, there are 35,000 abstracts in total where 16,800 for training, 11,200 for validation, and 7,000 for final testing [9]. Table 1 shows the distribution of each class of the dataset.

For final submission, each participated team allows to submit results with three different runs in which the best-performing result will be chosen for the final ranking. The official evaluation metric for the shared task is *weighted-average F1 score*[2].

Table 1. Dataset statistics.

Class	Train	Validation	Test
Computation and Language (CL)	2,740	1,866	1,194
Cryptography and Security (CR)	2,660	1,835	1,105
Distributed and Cluster Computing (DC)	2,042	1,355	803
Data Structures and Algorithms (DS)	2,737	1,774	1,089
Logic in Computer Science (LO)	1,811	1,217	772
Networking and Internet Architecture (NI)	2,764	1,826	1,210
Software Engineering (SE)	2,046	1,327	827
Total	**16,800**	**11,200**	**7,000**

2 Proposed Approach

Figure 1 illustrates a simplified architecture of a base model where a set of base models can be used for an ensemble approach. The key intuition is that maximizing transfer learning via pre-trained sentence encoders available on the Web so that we can obtain text (abstract) embeddings via those models in a straightforward manner. Afterwards, we can train our separate FNNs (Feed-forward Neural Networks) on top of those text embeddings for our classification task.

2.1 Pre-trained sentence encoders

In the following, we briefly introduce different pre-trained sentence encoders used in our approach.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

Sentence transformers. Sentence-BERT [10] is a modification of the BERT [3] network using siamese and triplet networks that are able to derive semantically meaningful sentence embeddings. SentenceTransformers[3] is a Python framework for state-of-the-art sentence and text embeddings. The models in SentenceTransformers are based on transformer networks like BERT and RoBERTa, and to facilitate transfer learning, the framework also provides a wide range of pre-trained sentence transformers which can be used for encoding a sentence/text. Therefore, we tested and used some of the pre-trained sentence transformers to encode each abstract, and used those encoded sentences (embeddings) as an input to additional FNNs for our task. The list of used model names can be found in Table 3.

SciBERT [1]. Although the pre-trained models in SentenceTransformers can be directly used for obtaining abstract embeddings for our task, they are trained on general domain corpora such as news articles and Wikipedia, which might have some limitations for the obtained embeddings as the domain of our task is the Computer Science domain. In contrast to those aforementioned sentence transformers, SciBERT is trained on papers from the corpus of the Semantic Scholar[4] which contains 1.14 million papers with 3.1 billion tokens using the full text of the papers for training.

Universal Sentence Encoder [2]. Similar to sentence transformers, this approach also provides the functionality for encoding sentences into corresponding embeddings that specifically target transfer learning to other NLP tasks. The model (`universal-sentence-encoder/4` in Table 3) is available in Tensorflow Hub[5], which is a repository of trained machine learning models ready for fine-tuning and deployable anywhere.

Encoding text with entity embeddings. Motivated by the recent studies, which have shown that leveraging entities mentioned in a short text improves text classification performance [11, 12], we obtain text embeddings via the set of entities mentioned in them. There are four main steps in this process.

1. Wikipedia entities mentioned in a text are extracted using TagMe [4]. Figure 2 shows the set of extracted Wikipedia entities using TagMe from a given abstract. In addition to an entity, TagMe also provides its *confidence score*, e.g., `Graph (discrete mathematics)`:0.43.
2. For each entity, we obtain pre-trained corresponding embeddings provided from wikipedia2vec [14]. wikipedia2vec is a tool used for obtaining embeddings of words and entities from Wikipedia, and also provides pre-trained word and entity embeddings[6].

[3] https://www.sbert.net/
[4] semanticscholar.org
[5] https://www.tensorflow.org/hub
[6] https://wikipedia2vec.github.io/wikipedia2vec/pretrained/

3. Despite of the efficiency of TagMe for extracting mentioned entities from a text, some of those extracted entities can be noisy. To cope with this problem, we further apply k-means clustering approach for those extracted entity embeddings with two clusters - one relevant cluster with the majority of related entities for a given text and the other cluster which is noisy - with the assumption that the larger one should contain the most of high-quality entities. Therefore, we only consider the entities belong to the larger cluster and do not consider those in the other cluster.
4. Finally, a text embedding can be obtained with the set of entity embeddings for those entities mentioned in the text (abstract) via a weighted mean pooling. That is, the entity embeddings are weight averaged based on their confidence scores where those scores can be treated as attention weights [13].

We use `entity-emb` to indicate this encoding approach for deriving a text embedding from a raw text. As we can see from Figure 1, the text embedding obtained with `entity-emb` can be optionally concatenated together with another text embedding obtained via a pre-trained sentence encoder such as SciBERT as an input to FNNs for classification.

fastText. The base models with pre-trained sentence encoders do not allow fine tuning of word embeddings in the context of our task. To better understand whether using those pre-trained sentence encoders provide good transfer learning quality, we also train a fastText [6][7] text classifier, which is a fast and efficient text classifier from Facebook using "bag of" tricks. It is often on par with deep learning classifiers in terms of accuracy, but with many orders of magnitude faster for training and evaluation.

Hard-voting ensemble. For the final classification with the predictions of multiple base models, we adopt the hard-voting ensemble. It sums the votes from those models, and then, the class with the most votes is used as the predicted class.

2.2 Training

We train those base models introduced in Section 2.1 using the training set provided by the challenge organizers with the objective of maximizing the evaluation metric: weighted-average F1 score. We further divide the validation set into *internal* validation and test sets where each set contains 5,600 examples as

[7] https://fasttext.cc/

Table 2. Dataset statistics where the validation set (11,200) is further split into internal validation and test sets evenly.

Train	Internal validation	Internal test	Final test
16,800	5,600	5,600	7,000

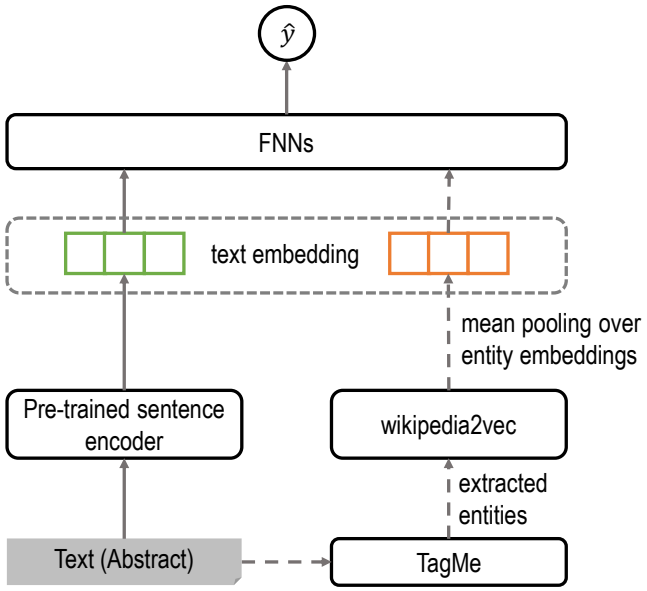

Fig. 1. Simplified illustration of a base model for encoding text into a text embedding for classification. The components connected via dashed lines are optional.

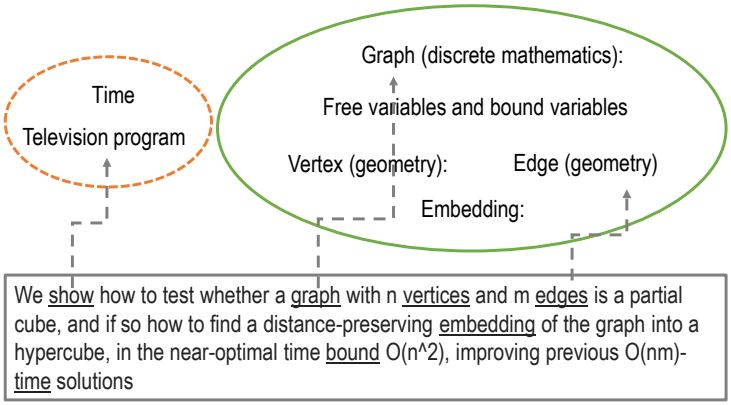

Fig. 2. Example of applying k-means clustering for the extracted entities of a text with $k = 2$. Those entities in the larger cluster–the green one–are chosen while those in the other cluster are not used.

shown in Table 2. The internal validation set is used for hyper-parameter tuning if needed for some base models, and the internal test set is used for testing only for different base models and ensemble approaches. All experiments are run on an Intel(R) Core(TM) i5-8365U processor laptop with 16GB RAM and the

Google Colab [8] environment. The implementation details can be found in our github repository[9].

3 Results

Table 3 shows the classification results on our *internal* test set using different transfer learning approaches. As we can see from the first part of the table, `sci-bert & entity-emb` provides the best performance in terms of the weighted-average F1 score of 0.9158, which outperforms the `sci-bert` (0.9140). We also notice similar trends when we incorporate the sentence embeddings with `entity-emb` for other models. For instance, `distilbert-base-nli-stsb-mean-tokens & entity-emb` improves the weighted-average F1 score over `distilbert-base-nli-stsb-mean-tokens` from 0.8496 to 0.8595. This indicates that utilizing entity embeddings can further improve the classification performance.

The bottom part of Table 3 shows the classification results with ensemble approaches using three different sets of models where the numbers in the bracket of each ensemble model indicate the base models in the first part of the table. For instance, `ensemble[4,5,7,8,9]` indicates the (hard-voting) ensemble of 4th, 5th, 7th, 8th, and 9th models in Table 3, i.e., `distilbert-base-nli-stsb-mean-tokens`, `distilroberta-base-msmarco-v2`, `universal-sentence-encoder/4`, `fasttext`, `sci-bert`, and `sci-bert & entity-emb`, respectively. As we can see from the table, the hard voting scheme with the classification results from different base models can further improve the performance. Note that incorporating the weakest base model `distilbert-base-nli-stsb-quora-ranking` also helped improve the performance as we can see from the three ensemble approaches. Overall, we observe that the `ensemble[4,5,6,7,8,9,10]` provides the best classification performance on our *internal* test set. These three ensemble models are used for predicting the labels on the *final* test set and submission.

Figure 3 further illustrates the confusion matrix using `ensemble[4,5,6,7,8,9,10]` on our *internal* test set. Each cell in the confusion matrix indicates the number of samples predicted as the corresponding column label which have the true row label. For example, 570 in the top-left cell indicates 570 samples out of 5,600 samples (10.8%) in the *internal* test set have been classified as LO where LO is the true label. The first cell in the second row shows that 4 (0.07% of the 5,600) samples have been classified as LO using `ensemble[4,5,6,7,8,9,10]` where the true label is DC. As we can see from the figure, DS (Data Structures and Algorithms) and DC (Distributed and Cluster Computing) are the most confusing labels followed by NI (Networking and Internet Architecture) and DC. The following abstract shows an example

[8] https://colab.research.google.com/
[9] https://github.com/parklize/pakdd2021-SDPRA-sharedtask

Table 3. Classification results on the *internal* test set.

No.	Model name	Weighted-average F1 score
0	`distilbert-base-nli-mean-tokens`	0.8389
1	`distilroberta-base-paraphrase-v1`	0.8641
2	`xlm-r-distilroberta-base-paraphrase-v1`	0.8625
3	`roberta-base-nli-stsb-mean-tokens`	0.8413
4	`distilbert-base-nli-stsb-mean-tokens`	0.8496
5	`distilroberta-base-msmarco-v2`	0.8591
6	`distilbert-base-nli-stsb-quora-ranking`	0.8228
7	`universal-sentence-encoder/4`	0.8931
8	`fasttext`	0.8816
9	`sci-bert`	0.9140
10	`sci-bert & entity-emb`	**0.9158**
11	`ensemble[4,5,7,8,9]`	0.9201
12	`ensemble[4,5,7,8,9,10]`	0.9252
13	`ensemble[4,5,6,7,8,9,10]`	**0.9258**

with the ground truth label as DC and with the predicted label as DS by our approach.

> "We present DegreeSketch, a semi-streaming distributed sketch data structure and demonstrate its utility for estimating local neighborhood sizes and local triangle count heavy hitters on massive graphs. DegreeSketch consists of ..."

As we can see from the example, this abstract is about an article proposing a data structure in the context of distributed computing, which is difficult to classify since the label DS also makes sense here.

For the *final* evaluation of different approaches from participated teams, the task organizers allow three prediction results on the *final* test set. We used the prediction results given by `ensemble[4,5,7,8,9]`, `ensemble[4,5,7,8,9,10]`, `ensemble[4,5,6,7,8,9,10]` for the *final* test set provided by the task organizers, and our best-performing approach achieves the weighted-average F1 score of 0.92 on the *final* test set.

4 Conclusions

This paper shows how the proposed solution with different sentence encoders powered by transformers together with entity embeddings achieves the best classification performance in the context of a scholarly text classification task on the internal test set, which we believe can be generally applied to other text classification tasks as well. To the best of our knowledge, this is the first work using both sentence and entity embeddings for a text classification in the context of shared task or challenge, which might provide interesting insights for designing solutions for similar text classification tasks or challenges.

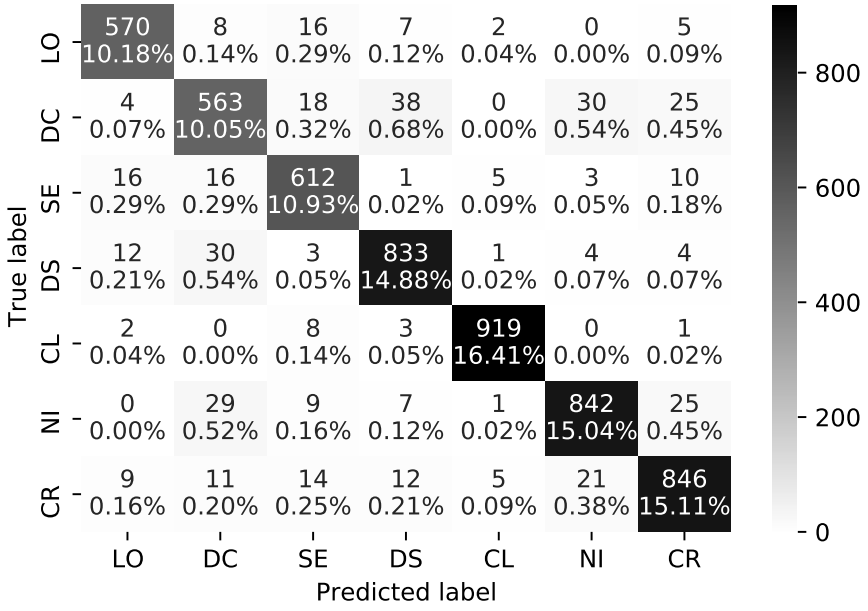

Fig. 3. The confusion matrix of classification using `ensemble[4,5,6,7,8,9,10]` on the *internal* test set (with 5,600 samples). Each cell contains the number of samples and its corresponding percentage of the 5,600 samples for the corresponding predicted and true labels.

References

1. Beltagy, I., Lo, K., Cohan, A.: Scibert: Pretrained language model for scientific text. In: EMNLP (2019)
2. Cer, D., Yang, Y., Kong, S.y., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al.: Universal sentence encoder. arXiv preprint arXiv:1803.11175 (2018)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
4. Ferragina, P., Scaiella, U.: Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In: Proceedings of the 19th ACM international conference on Information and knowledge management. pp. 1625–1628 (2010)
5. Ghosal, T., Sonam, R., Saha, S., Ekbal, A., Bhattacharyya, P.: Investigating domain features for scope detection and classification of scientific articles
6. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
7. Piao, G., Breslin, J.G.: Domain-aware sentiment classification with grus and cnns. In: Buscaldi, D., Gangemi, A., Reforgiato Recupero, D. (eds.) Semantic Web Challenges. pp. 129–139. Springer International Publishing, Cham (2018)

8. Reddy, Saichethan; Saini, N.: Overview and insights from scope detection of the peer review articles shared tasks 2021. In: The First Workshop & Shared Task on Scope Detection of the Peer Review Articles (SDPRA 2021) (2018)
9. Reddy, Saichethan; Saini, N.: Sdpra 2021 shared task data (2021), mendely Data, V1, doi: 10.17632/njb74czv49.1
10. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (11 2019), https://arxiv.org/abs/1908.10084
11. Türker, R., Zhang, L., Alam, M., Sack, H.: Weakly supervised short text categorization using world knowledge. In: International Semantic Web Conference. pp. 584–600. Springer (2020)
12. Türker, R., Zhang, L., Koutraki, M., Sack, H.: " the less is more" for text classification,. In: SEMANTICS Posters&Demos (2018)
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
14. Yamada, I., Asai, A., Sakuma, J., Shindo, H., Takeda, H., Takefuji, Y., Matsumoto, Y.: Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 23–30. Association for Computational Linguistics (2020)