ITNLP at SemEval-2021 Task 11: Boosting BERT with Sampling and Adversarial Training for Knowledge Extraction

Genyu Zhang, Yu Su, Changhong He, Lei Lin, Chengjie Sun, Lili Shan
Intelligence Technology and Natural Language Processing Lab,
School of Computer Science and Technology,
Harbin Institute of Technology
{gyzhang, suyu, changhong.he}@insun.hit.edu.cn
{cjsun, linl, shanll}@insun.hit.edu.cn

Abstract

This paper describes the winning system in the End-to-end Pipeline phase for the NLP-ContributionGraph task. The system is composed of three BERT-based models and the three models are used to extract sentences, phrases and triples respectively. Experiments show that sampling and adversarial training can greatly boost the system. In End-to-end Pipeline phase, our system got an average F1 of 0.4703, significantly higher than the second-placed system which got an average F1 of 0.3828.

1 Introduction

The Knowledge Graph (KG) describes the concepts, entities and their relationships in the objective world in a structured form, expresses Internet information in a form closer to the human cognitive world. Information extraction is the first step of the KG construction. Information extraction is a technology that extracts structured information such as entities and relationships from semi-structured or unstructured data automatically. Similarly, as the rate of research publications increases, it is critical to construct Knowledge Graphs to represent scholarly knowledge efficiently. The target of the NLPContributionGraph task (D'Souza et al., 2021) is to find a systematic set of patterns of subject-predicate-object statements for the semantic structuring of scholarly contributions that are generically applicable for NLP research articles, then apply the discovered patterns in the creation of a larger annotated dataset for ingesting the dataset into the Open Research Knowledge Graph infrastructure to assist users manually manage their article contributions. Our task consists of three sub-tasks: Sentences Extraction (SE), Phrases Extraction (PE) and Triples Extraction (TE).

The dataset used in the NLPContributionGraph task contains hundreds of Natural Language Processing (NLP) scholarly articles annotated for their contributions. Each article is written in English and contains three types of annotation information: 1) contribution sentences; 2) scientific term and predicate phrases from the sentences; and 3) subject-predicate-object triple statements from the phrases toward KG building.

Our code is available at https://github.com/itnlp606/nlpcb-graph.

2 Related Work

In recent years, pretrained language models (Peters et al., 2018; Devlin et al., 2019; Sun et al., 2019; Lan et al., 2020) have achieved impressive performance in various NLP tasks including information extraction. BERT (Devlin et al., 2019) uses Bidirectional Transformers (Vaswani et al., 2017) to pretrain the model on the Masked Language Model (MLM) task and the Next Sentence Prediction (NSP) task and advances the state-of-the-art for eleven NLP tasks. The system presented in this paper is based on fine-tuning BERT. This section will introduce two strategies to boost the BERT model.

2.1 Sampling

In classification tasks, we often encounter uneven distribution of positive and negative samples. Under such distribution, the model may not be able to make accurate predictions. The trained model naturally tends to predict the majority set, and the minority set may be considered as noise. Compared with the majority set, the minority set is more likely to be misclassified. Modifying loss function (Lin et al., 2017; Li et al., 2019) and sampling methods (Chawla et al., 2002; Liu et al., 2009) are valid approaches to solve this problem, and the later was adopted in our system. Oversampling achieves sample balance by increasing the number of minority samples in classification.

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 485–489
Bangkok, Thailand (online), August 5–6, 2021.

The most direct method is to simply copy the minority samples to form multiple records. The disadvantage of this method is that if the sample features are few, the model is easy to overfit. SMOTE (Chawla et al., 2002) interpolates between samples of the minority class to generate additional samples. Under-sampling achieves sample balance by reducing the number of samples of the majority class in classification. The most direct method is to randomly remove some samples of the majority class. EasyEnsemble (Liu et al., 2009) divides the majority samples into several parts randomly, so the data of each part is equal to the number of minority samples. Then, multiple models are trained on different parts of data, and the output of each model will be integrated. BalanceCascade (Liu et al., 2009) combines a subset of the majority class with the minority class to train the model, then discards the samples that are correctly classified in the next round, so that the subsequent base learner can pay more attention to those samples that are incorrectly classified. Our model uses under-sampling for sentences extraction and triples extraction. For different tasks, diverse sampling strategies have been adopted.

2.2 Adversarial training

As machine learning model is vulnerable to some small worst-case perturbations, adversarial training (Goodfellow et al., 2014) aims to make the AI systems safer by improving the robustness of the model. In Computer Vision tasks, adversarial training usually hurts the generalization of the model. However, Miyato et al. (2017) adopted adversarial training in text classifying by applying perturbations to the word embeddings, which can improve both generalization and robustness of the NLP models.

Considerable efforts have been made to find better adversarial perturbations. The Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) generates adversarial examples by formulation:

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{r}_{adv}$$
$$\boldsymbol{r}_{adv} = \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right)$$

where $\boldsymbol{x}$ are the embeddings of the input text, $\boldsymbol{r}_{adv}$ are the adversarial perturbations, $\boldsymbol{\theta}$ are model parameters, $\hat{\boldsymbol{x}}$ are embeddings of adversarial examples that are used to update the model. The Fast Gradient Method (FGM) (Miyato et al., 2017) is another generation of FGSM in which the perturbations are normalized by gradients:

$$\boldsymbol{r}_{\text{adv}} = -\epsilon \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}$$

where $\boldsymbol{g} = \nabla_{\boldsymbol{x}} \log p(y \mid \boldsymbol{x}; \hat{\boldsymbol{\theta}})$.

Madry et al. (2018) used a min-max formulation as follows to cast both attacks and defenses into a common theoretical framework,

$$\min_{\theta} \left\{ \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{r\in\mathcal{S}} L(\theta, x+r, y) \right] \right\}$$

in this formulation, the inner maximization problem describes attack which aims to find the most adversarial data leading to a high loss, the outer minimization problem describes defense which aims to find the most robust model. They also proposed Projected Gradient Descent (PGD) that uses a iterative algorithm to generate the most adversarial data.

The Friendly Adversarial Training (FAT) (Jingfeng et al., 2020) adopted by our team is an early-stopped version of PGD, its adversarial data was generated by a min-min formulation as following:

$$\tilde{x}_i = \arg \min_{\tilde{x}\in B(x_i)} \ell\left(f(\tilde{x}), y_i\right)$$
$$\text{s.t. } \ell\left(f(\tilde{x}), y_i\right) - \min_{y\in\mathcal{Y}} \ell\left(f(\tilde{x}), y_i\right) \geq \rho$$

different from PGD, FAT generates friendly adversarial data rather than the most adversarial data, $\rho > 0$ is a margin that indicates the confidence of adversarial data being misclassified. FAT is more computationally effecient than PGD, and model trained with FAT can reach higher accuracy.

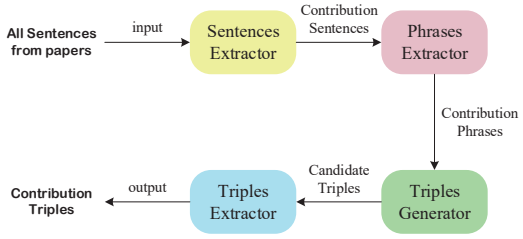
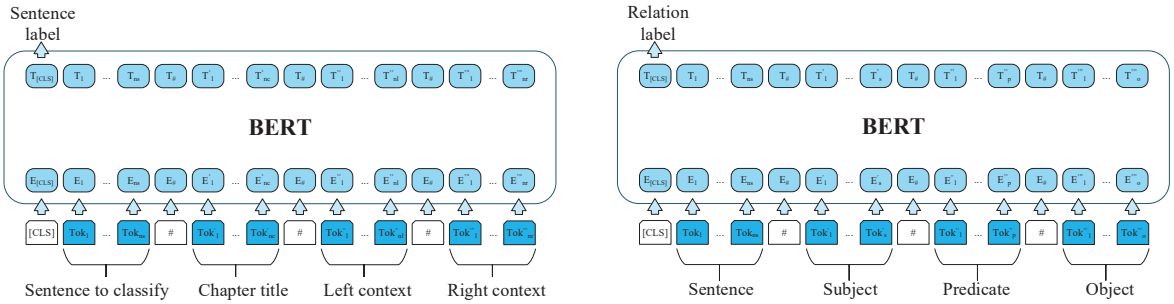


Figure 1: System Overview

3 System Description

For three sub-tasks in NLPContributionGraph, we designed four modules to implement these tasks. These modules use fine-tuning BERT with FAT as



(a) Sentences Extractor Framework

(b) Triples Extractor Framework

Figure 2: Both of the Sentences Extractor and the Triples Extractor are build by the BERT-based classification model.

the base model and adopt different boosting methods to improve overall results. The full framework of the system is shown in Figure 1. The three extractors can extract contribution information by classification. The Triples Generator can convert discrete phrases into triples. These modules will be explained further in following sections.

3.1 Sentences Extractor

The system uses the Sentence Extractor to solve the SE task. The extractor can extract the sentences that have the contribution information such as research problem, code, etc. As shown in Figure 2a, the Sentences Extractor uses the sentence context and paragraph heading as additional features and uses BERT as a binary classifier to determine whether the sentence contains the contribution information. In an annotated paper, most sentences do not contain the contribution information. Therefore, we adopted an under-sampling strategy. In the training process, the ratio of positive samples and negative samples is fixed to an integer for each batch to ensure that the model will not overfit on negative samples. The ratio is a hyperparameter that needs to be tuned in the training process.

3.2 Phrases Extractor

The Phrases Extractor can extract contribution phrases from the sentences to solve the PE task. For this task, the BERT-based sequence labeling model is effective. The phrases predicted by the trained model will sometimes be incomplete, resulting in high recall and low precision. Fortunately, ensembling learning can solve this problem well. In the competition, we trained ten different models by ten-fold cross-validation. After training, the trained models will make their own predictions, and the module will count the number of votes for each phrase. Only phrases with more than a certain number of votes will be seen as a valid output.

3.3 Triples Generator

After the Phrases Extractor completes the prediction, we can obtain discrete phrases. The role of the Triples Generator is to convert these phrases into triples through permutation and combination. This section will introduce two methods to finish this task.

Language Model Approach Language models are usually used to evaluate the probability of a sentence. The triples to be extracted are composed of subject, predicate and object, which are components of a sentence. This approach uses a language model to evaluate the probability of triples. The input of the module is all permutations of contribution phrases, and the permutation that has the highest probability will be the output of the module, which are candidate triples.

Combination Approach Due to the lack of data, the prediction made by language model is not accurate. In the annotated data, the order of about ninety percent of the triples is sequential. In order to deal with the insufficient representation ability of the language model, we directly use the combination of all serial phrases as the output of this module. In the competition, we adopted Combination Approach as the Triples Generator.

3.4 Triples Extractor

The Triples Extractor can classify all candidate triples based on the BERT model. As shown in Figure 2b, sentences and triples are separated by

hash marks, and inputted into BERT for classification. Unfortunately, the trained model will easily overfit on negative samples due to the large number of combinations. Therefore, we need to adopt the under-sampling strategy to boost the base BERT. For complex combinations, the module combines two strategies to select negative samples:

Random Replacement (RR) For each of the three phrases in the positive sample, we will randomly select one and replace it with another phrase.

Random Selection (RS) Randomly select three phrases that are not positive samples.

The module combines the two sampling methods above to generate negative samples. For each batch, it fixes the ratio of positive and negative samples to generalize better.

4 Experimental Setup

In this section, we did some ablation analysis on the validation set for the boosting methods proposed in this paper, and gave some analysis through the experimental results.

4.1 Adversarial Training

We tested the performance of different adversarial training approaches. Table 1 shows that FAT achieved the best results in all three tasks. Especially in the SE task, adversarial training can greatly improve the model's performance. During the experiment, we also found that if adversarial training is not applied, training will converge in an average of five epochs. If the system uses adversarial training, the training will last for about twenty epochs and will continuously improve the performance on the validation set. The perturbations added by the adversarial training make the model generalize better. In addition, ensembling is not applied in the PE task, so the F1 score of this task is low. This issue would not affect the experimental results.

Task	Natural	FGM	FAT
SE	0.4112	0.5527	**0.5615**
PE	0.2011	0.2128	**0.2231**
TE	0.4641	0.4740	**0.5176**

Table 1: F1 scores of Natural training (no adversarial training), FGM and FAT.

4.2 Sentences Features

We randomly selected five domains of papers to test the effect of different features on the SE task. These domains are Question Answering (QA), Relation Extraction (RE), Sentence Classification (SC_1), Sentence Compression (SC_2) and Text Generation (TG).

Table 2 shows that adding either context or title can significantly improve the accuracy of classification and the best results can be achieved by concatenating both of them.

Domain	Natural	Title	Context	T&C
QA	0.4068	0.5294	0.6471	**0.6977**
RE	0.4516	0.5128	0.5781	**0.5827**
SC_1	0.3636	0.5417	0.7391	**0.7826**
SC_2	0.5600	0.5714	0.5926	**0.6667**
TG	0.4681	0.5424	0.5385	**0.5763**

Table 2: F1 scores while adding different features. T&C means adding both Title and Context.

4.3 Triples Extractor Sampler

In the TE task, RR and RS are applied as the sampling methods. Without the under-sampling strategy, the model is difficult to converge. Table 3 shows the performance of different sampling methods on papers in various domains. Among them, the combination of RR and RS strategies achieved the best results. The diversity of the sampling strategies improves the generalization.

Domain	RR	RS	RR&RS
QA	0.3621	0.3592	**0.4267**
RE	0.3782	0.4033	**0.4163**
SC_1	0.3359	0.3505	**0.3692**
SC_2	0.4585	0.4623	**0.4777**
TG	0.4900	0.5351	**0.5748**

Table 3: Macro-F1 scores with different triple sampling methods

4.4 Evaluation Results

In End-to-end Pipeline phase, our system got F1 scores of 0.5619, 0.4522 and 0.1379 in tasks SE, PE, TE, respectively. Our system has achieved good results on tasks SE and PE, but task TE can still be improved. Since our system only considers sequential triples, some triples will be missed, which can be a defect of our system.

5 Conclusion

BERT is a powerful model that has considerable applications in numerous fields of NLP. Using BERT in the NLPContributionGraph task allows researchers to read papers more efficiently. The methods of adversarial training and sampling proposed in this paper can greatly boost the performance of BERT on this task and can also offer some thoughts for future work on knowledge extraction of papers.

Acknowledgements

This work was supported by the National Key R&D Program of China via grant 2018YFC0830700, National Natural Science Foundation of China (NSFC) via grant 61772156.

References

V. Nitesh Chawla, W. Kevin Bowyer, O. Lawrence Hall, and Philip W. Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, pages 321–357.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jennifer D'Souza, Sören Auer, and Ted Pedersen. 2021. SemEval-2021 task 11: Nlpcontributiongraph - structuring scholarly nlp contributions for a research knowledge graph. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). Association for Computational Linguistics.

J. Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *international conference on learning representations*.

Zhang Jingfeng, Xu Xilie, Han Bo, Niu Gang, Cui Lizhen, Sugiyama Masashi, and Kankanhalli Mohan. 2020. Attacks which do not kill training make adversarial learning stronger. *ICML*, pages 11278–11287.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *ICLR*.

Buyu Li, Yu Liu, and Xiaogang Wang. 2019. Gradient harmonized single-stage detector. *national conference on artificial intelligence*.

Tsung-Yi Lin, Priya Goyal, B. Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. *ICCV*, pages 318–327.

Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. 2009. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, pages 539–550.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. *international conference on learning representations*.

Takeru Miyato, M. Andrew Dai, and J. Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. *international conference on learning representations*.

E. Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and S. Luke Zettlemoyer. 2018. Deep contextualized word representations. *north american chapter of the association for computational linguistics*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv: Computation and Language*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, N. Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 30 (NIPS 2017)*, pages 5998–6008.

Appendix

This section will list the hyperparameters used in the competition, which can help researchers replicate the experiments conducted in this paper.

- **Global**
  - Batch size: 16
  - Learning rate (Adam): 5e-5
  - Pretrained model: BERT-base
  - Word embedding size: 512
  - Hidden layer size: 768
- **Task SE**
  - Number of senteces in the context: 2
  - Positive and negative sample ratio: 1:3
- **Task TE**
  - Positive and RS sample ratio: 1:3
  - Positive and RR sample ratio: 1:3
  - Positive and negative sample ratio: 1:6