BYteam at SemEval-2020 Task 5: Detecting Counterfactual Statements with BERT and Ensembles

Yang Bai, Xiaobing Zhou*
School of Information Science and Engineering
Yunnan University, Yunnan, P.R. China
*Corresponding author:zhouxb@ynu.edu.com

Abstract

We participate in the classification tasks of SemEval-2020 Task: Subtask1: Detecting counterfactual statements of semeval-2020 task5(Detecting Counterfactuals). This paper examines different approaches and models towards detecting counterfactual statements classification. We choose the Bert model. However, the output of Bert is not a good summary of semantic information, so in order to obtain more abundant semantic information features, we modify the upper layer structure of Bert. Finally, our system achieves an accuracy of 88.90 % and F1 score of 86.30 % by hard voting, which ranks 6th on the final leader board of the in subtask 1 competition.

1 Introduction

Counterfactual statements describe events that did not happen or cannot happen, as well as the possible consequence if the events have had happened. More specifically, counterfactuals describe events counter to facts and hence naturally involve common sense, knowledge, and reasoning. Tackling this problem is the basis for all down-stream counterfactual related causal inference analysis in natural language. In this work, we present our results for the SemEval 2020 Shared Task 5 (Yang et al., 2020): Modelling Causal Reasoning in Language: Detecting Counterfactuals. Subtask 1 focuses on the binary distinction if a sentence is counterfactual or not. Our submission for Subtask A ranks 6th. For Subtask 1, we experiment with different neural network-based models such as Recurrent Neural Network(RNN) (Zaremba et al., 2014), Attention (Vaswani et al., 2017) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018). For the imbalanced data set, we find that the BERT performs best on our development set. So, we combine the results by different BERT models.

The rest of the paper is organized as follows. Section 2 briefly shows the related work. Section 3 elaborates on our approach. It shows the pre-processing step and architecture of our model. Section 4 describes the data set, hyper-parameters adopted research methodology and our results. Finally, Section 5 concludes our work.

2 Related Work

Detecting the counterfactual statement is becoming more and more important. To build an effective classifier, one of the major problems is to find the appropriate features. Normally, two types of features are applied: surface features like n-grams and word representations trained by the neural network. Traditionally, the classifiers are trained on different types of surface features with approaches like SVM, Random Forest, and Logistic Regression. Using pre-trained word embeddings for feature extraction is effective in multiple NLP tasks.(Wang et al., 2018; Wang et al., 2016) Traditionally word embeddings are extracted from shallow neural networks trained on a large swathe of texts required to learn the contextual representations of words. For example, there are skip-grams (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). However, these embeddings are learned from an aggregation of all possible word contexts, which may gloss over semantic nuances in representations. Recently, models like ELMo (Peters et al., 2018) and BERT significantly advance the state-of-the-art in language modeling by

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 640–644
Barcelona, Spain (Online), December 12, 2020.

learning context-sensitive representations of words. ELMo goes beyond word embeddings by learning representations that are functions of the entire input sentence. However, ELMo is still considered shallow with two bidirectional LSTM layers, and more recent transformer-based language models such as the OpenAI Generative Pre-trained Transformer (GPT) (Radford et al., 2019) and Bidirectional Encoder Representations from Transformers (BERT) have been extended to a depth of up to twelve layers. The BERT is trained to be bidirectional with two novel prediction tasks, Masked LM and Next Sentence Prediction. The pre-trained BERT model has been shown to achieve significant improvements in a series of downstream tasks over ELMo and OpenAI GPT.

3 Our Approach

3.1 Pre-processing

The official organizer provides the training set and test set for subtask-1. We split the training set into a new training set and a validation set by using the stratified 5-fold cross-validation[1]. BERT uses the wordpiece tool for word segmentation and inserts special separators ($[CLS]$, which are used to separate each sample) and separator ($[SEP]$, which are used to separate different sentences in the sample). For each fold of the data set, the input data format is as follows: $[CLS]$+$sentence$+$[SEP]$.

3.2 Model Description

BERT (Devlin et al., 2018) is a deep bidirectional network built using Transformers (Vaswani et al., 2017). It is pre-trained to detect (a) a masked word from its the left and right context, and (b) the next sentence. We select the BERT-Base as the underlying BERT model. The BERT-Base comprises 12 Transformer blocks, 12 self-attention heads, and 768 hidden dimensions with a total parameter of 110M. It is trained on the BookCorpus (800M words) and the English Wikipedia (2,500M words). For classification tasks, the output of BERT-Base($pooler\ output$) is obtained by its last layer hidden state of the first token of the sequence (CLS token) further processed by a linear layer and a Tanh activation function. However, the pooler output is usually not a good summary of the semantic content of the input.[2] In order to make the model obtain abundant semantic content, we try the following model architecture to relieve this problem.

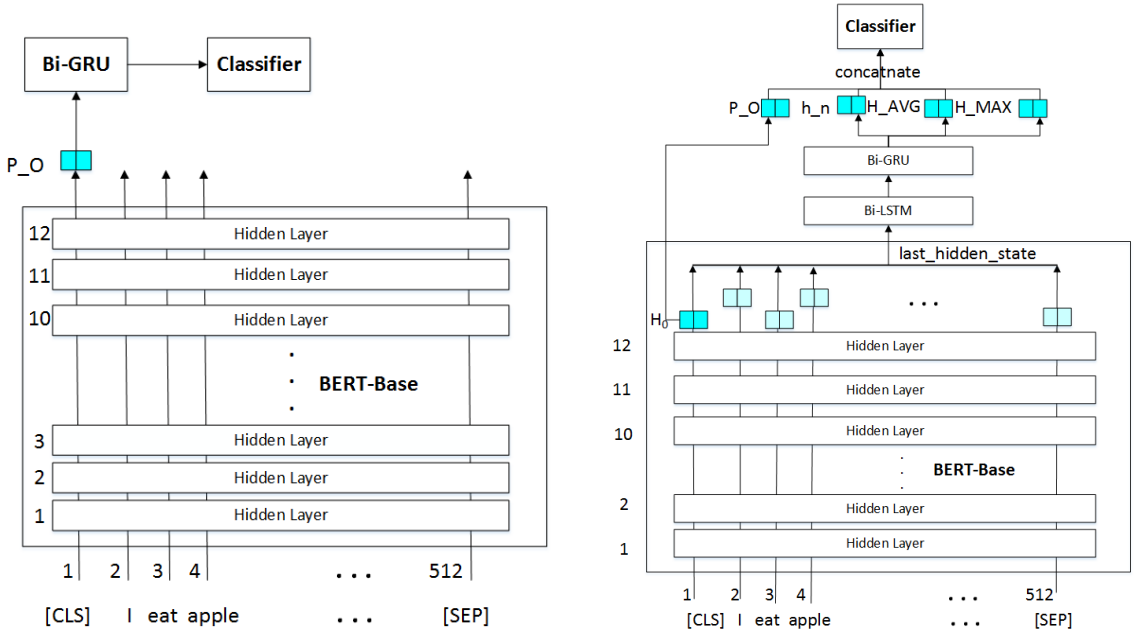
3.2.1 Method_A

In order to make the model obtain more abundant semantic information features, we implement the two models as shown in Figure 1 by obtaining higher-dimensional features. For Figure1(a), we divide the model into two parts. The first part is to get the $pooler\ output$(P_O). The second part is the Bi-GRU module with input P_O. Firstly, we input the data into the model to get the P_O, which is a two-dimensional vector. Secondly, we reshape P_O as a three-dimensional vector and input it into BiGRU to get three-dimensional vector output. Finally, we reshape the output as a two-dimensional vector and input it into the classifier. For figure1(b), firstly, we get the sequence of hidden states at the output of the last layer of the BERT-Base, also known as $last\_hidden\_state$[2]. Secondly, we input the $last\_hidden\_State$ into Bi-LSTM and Bi-GRU to get the output of Bi-GRU and the hidden state of Bi-GRU(h_n)[3]. Thirdly, after getting the output of Bi-GRU, we get the H_AVG by average-pooling and the H_max by max-pooling. Fourthly, we concatenate P_O, h_n, H_AVG, and H_max into the classifier.

3.2.2 Method_B

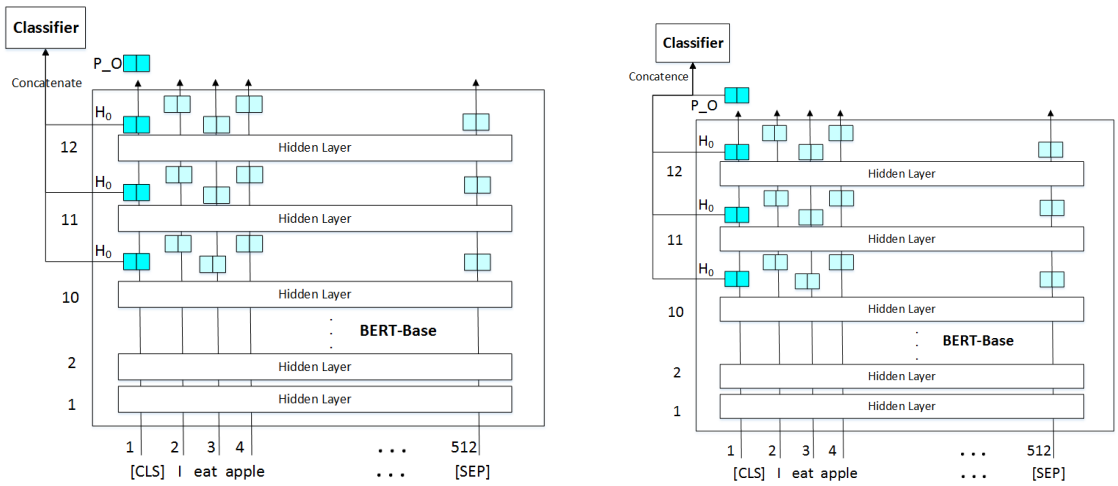
Recent studies have shown that the top layer of Bert can learn more abundant semantic information features.(Jawahar et al., 2019) Therefore, we try two models as shown in Figure 2 to get abundant semantic information features. For Figure 2(a), we concatenate H_0 of the last three hidden layers into the classifier. For Figure 2(b), we concatenate P_O and H_0 of the last three hidden layers into the classifier after obtaining P_O.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html
[2] https://huggingface.co/transformers/model_doc/bert.html#bertmodel
[3] https://pytorch.org/docs/stable/nn.html?highlight=nn%20gru#torch.nn.GRU



(a) (b)

Figure 1: Method_A(P_O is the pooler output. $last\_hidden\_State$ is the sequence of hidden-states at the output of the last layer of the BERT-Base model. H_0 is hidden-state of the first token of the sequence(CLS token) at the output of the hidden layer of the model. h_n is the second output returned by Bi-GRU: the hidden state of the last time step. H_AVG is the average-pooling of Bi-GRU output. H_MAX is the max-pooling of Bi-GRU output.)



(a) (b)

Figure 2: Method_B(P_O is the pooler output, H_0 is hidden-state of the first token of the sequence(CLS token) at the output of the hidden layer of the model.)

4 Experiments

4.1 Task details

The subtask1 is a classification of the existence of counterfactual statements, '1' refers to counterfactual, and '0' refers to non-counterfactual. The criteria evaluation using F1-score as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{Precision * Recall * 2}{Precesion + Recall}$$

4.2 Hyper-parameters and Results

In this work, our models are implemented based on Pytorch. We use the Bert-Base-Cased[4] as our pre-trained model. For all models, in order to save GPU memory, the batch size parameter of GPU in fine-tuning is set to 4 and the gradient_accumulation_steps is set to 4, so that each time a sample is an input, the gradient is accumulated 4 times, and then the back-propagation update parameters are performed. The memory is saved by sacrificing a certain training speed; the parameter learning_rate is set to 1e-5, and we use the triangular learning rate. First, the learning rate is gradually increased through warm_up, and then the linear learning rate is gradually reduced through linear learn rete decay, which effectively improves the training effect. The hyper-parameters of each model and the results on the validation set are shown in Table 1.

		Hyperparameters	F1-macro on our dev set
1	5-fold data with 42 random seeds Method_A(a)	output_hidden_states=False dropout=0.1 learning rate=1e-5 epoch=3 per_gpu_train_batch_size=4 gradient_accumulation_steps=4	0.9210
2	5-fold data with 42 random seeds Method_A(b)		0.9286
3	5-fold data with 42 random seeds Method_B(a)	output_hidden_states=True dropout=0.1 learning rate=1e-5 epoch=3 per_gpu_train_batch_size=4 gradient_accumulation_steps=4	0.9265
4	5-fold data with 42 random seeds Method_B(b)		0.9244

Table 1: The hyper-parameters of each model and the results on the validation set.

As can be seen from table 1, firstly we get the new training set and the validation set by using the stratified 5-fold cross-validation with 42 random seeds. Stratified sampling ensures that the proportion of samples in each data set remains unchanged. Secondly, for each fold of data set, we choose the model with the highest F1 value in the validation set to predict the test set and then average the probability of the five prediction results to get the prediction results of the model. Then we input data into one to four models for training with the training set and predict one to four results($result1 - 4$) with the test set. Finally, we combine $result1 - 4$ by hard voting to get the final result. In subtask 1, our system F1 score is 0.8630 and we rank sixth on the competition leaderboard.

[4]https://huggingface.co/bert-base-cased

5 Conclusion

This paper introduces the overall idea and specific scheme of BYteam in SemEval-2020-task5. In the competition, we adopted multiple methods to modify the upper structure of Bert model, perform stratified 5-fold cross-validation based on each model, and finally achieve good performance by hard voting. In the future, our work will focus on solving the problem of imbalanced data and how to make the model obtain abundant semantic information features. The code is available online.[5]

Acknowledgments

This work was supported by the Natural Science Foundations of China under Grants 61463050, the NSF of Yunnan Province under Grant 2015FB113.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language?

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2016. Community-based weighted graph model for valence-arousal prediction of affective words. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):1957–1968.

Jin Wang, Bo Peng, and Xuejie Zhang. 2018. Using a stacked residual lstm model for sentiment intensity prediction. *Neurocomputing*, 322:93–101.

Xiaoyu Yang, Stephen Obadinma, Huasha Zhao, Qiong Zhang, Stan Matwin, and Xiaodan Zhu. 2020. SemEval-2020 task 5: Counterfactual recognition. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

[5] https://github.com/byew/commme