NLP Project Summary

Pingwei Sun, Hanwen Sun, Han Liu

NEU-NLP Lab

Abstract

In this stage of project, our team build up a sentiment classification system based on the pretrained BERT model.

Experiments are done to find more efficient strategies for BERT fine-tuning. With these methods, the performance of our model on test dataset is improved from 87% to 94%.

Besides, we try to analyse why the strategy work, all the experience is collated and presented in this paper.

1 Introduction

In the task of natural language processing, the expression of token and the semantic relationship of context are always two key sessions.

In terms of word expression, the method has developed from dictionary to vector representation, and then to today's word embedding method.

In context semantics fields, people began to manage to make the model learn features from the aspect of sentences and paragraphs through the structures designed for memory, from the early RNN, LSTM, and other models.

Since then, some bi-direction methods have also been developed.

However, with the launch of **Transformer** (Vaswani et al., 2017), the model's ability has risen to a new level in terms of learning word similarity and contextual semantic information.

At the same time, with the improvement of computing power, more and more organizations are also trying to use larger models and huge datasets to obtain better results.

After completing the pretrained word vector expression (Mikolov et al., 2013), Google launched a more powerful pretraining model **BERT** (Devlin et al., 2018) in 2018, which has a great ability to represent words and understand semantics in the context.

BERT is undoubtedly a milestone in the field of natural language processing, however, there are too many parameters that will affect the model's performance. During this period of experiment, we tried various strategies and tricks to make BERT work better in the task of sentiment classification.

Initiate work of BERT:

- Using transformer as the framework of the model to better capture the bidirectional information.
- Multitask training strategies and inspiration on pretraining.
- Being competent for many downstream tasks after fine-tuning.

Our work:

- Try different efficient classifiers.
- Explore hyperparameters settings.
- Find efficient fine-tuning strategies during training models.

2 Methods

2.1 Efficient Classifiers

The structure of classifier is important for our model. We may experiment on the type of classifier(FFN, SVM, LSTM), the layer number, width, and the activation function of classifier.

What's more, we will send different features form bert into the classifier. Different layers in bert may contain different level of information, we can combine them in different ways and observe the results.

2.2 Hyperparameters Tuning

We mainly focus on the base learning rate, decay factor(learning rate decay of each transformer layer) , and the scheduler.

Bert consists of 12 transformer layers, we need to find an appropriate way to set and schedule our learning rate to fine-tune our model. Empirically, we set the base learning rate as 2e-5 or 2.5e-5.

Different layers of BERT contain different information, the lower layer of the BERT may contain more general information, so we can fine-tune them with different learning rates.

We use the decay factor to split the learning rate into 12 parts, corresponding to 12 transformer layers. The 11th layer has the highest learning rate, and the learning rate decay by decay factor with the layer decrease.

Additionally, we try different learning rate schedulers and combine them with the warm-up trick to make the model upgrade steadily.

2.3 Data Augmentation

In order to improve the robustness of our model, we will try some way of data augmentation.

EDA(Easy Data Augmentation) is often used in nlp. We apply three main ways to our train data. We randomly replace words with their synonyms, shuffle words in each sentence and delete each word with a probability.

In addition, we apply Mask Predictions to the training dataset. We mask some words in each sentence and then use the BERT to predict the specific word based on the text.

2.4 Fine-Tuning Strategies

After doing experiments on classifiers and hyperparameters, we also try some strategies, such as further pretraining and adversarial learning, to improve our model.

2.4.1 Further Pretraining

In order to enable the model to be further pretrained on our dataset and better understand the dataset, it is designed to run the *MLM* task, which is also called *masked language model* in BERT. While, for the other task, we design one called *SIM* instead of reusing the *next sentence prediction* task to conduct the unsupervised further pretraining.

In the aspect of *SIM* task, we are inspired by the Simcse (Gao et al., 2021) and try to apply the method as an unsupervised pretraining task. Specifically, we use the mask and dropout structure in BERT to obtain different embeddings of the same sentence. Sentence embedding from the same origin sentence is treated as a positive sample to each other, while the other embeddings in the batch are naturally set as the negative ones, the process of which is presented in figure 1.

By building a similarity matrix, optimization will be done to make the model better understand the sentiment information in the dataset.

During the part of further pretraining, the *bert-base* model is trained on the *MLM* task and the *SIM* task, results of which will be shown in section 3.6, to help with its performance.

2.4.2 Adversarial learning

As for the adversarial learning part, we introduce various popular methods, including *FGM,*

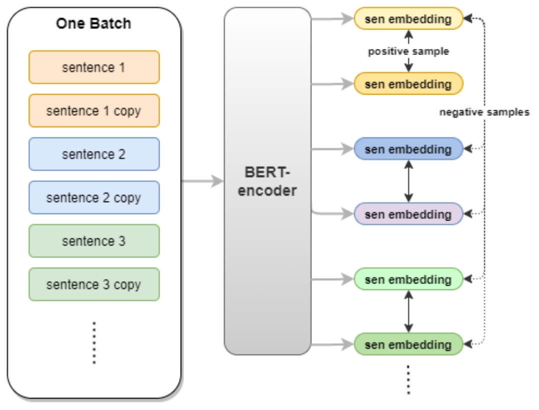

Figure 1: SIM pretrain

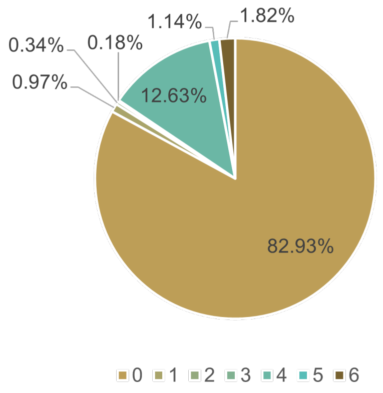

Figure 2: Proportion of samples of different categories in the training dataset

PGD and FreeLB, to make it more robust. Related experiments are also presented in section 3.6.

3 Experiments

3.1 Datasets

The dataset is provided by the neu-nlp group. We have train data, valid data and test data which have 80000, 10000, and 10000 sentences separately.

There are seven different classes of sentences, labeled zero to six and represent different sentiments. We find that the train data and valid data are seriously unbalanced, as shown in figure 2.

3.2 Hyperparametes

We use the BERT-base model, with the hidden size of 768, 12 Transformers blocks, and 12 self-attention heads. We experiment on the RTX 6000 and set the batch size as 16, the scheduler warm-up portion is set to 0.1. Empirically, we set the max number of the epoch to 4 and save the best model on the validation set for test.

3.3 Exp-I: Investigating Different Classifiers and Features

3.3.1 Different Classifiers

In order to explore the impact of different classifiers on the accuracy, we propose several experiments on the effectiveness of different types of models (structures are presented in figure 3 and 4), and the results are shown in table 1.

model	val acc	time
Linear	0.8743	3.5 min per epoch
Linears	0.8764	4 min per epoch
LSTM	0.8717	5 min per epoch
BiLSTM	0.8730	5 min per epoch

Table 1: Performance of different classifiers

From the table, it can be seen that the basic linear model fits better than the timing model. The model with two linear-layers performs the best, which costs suitably and gets the strongest result, reaching 87.65% on our Validation dataset.

So as to pursue the consistency with Bert model as much as possible, we use the same activation function *tanh* & *glue* and shortcut connection to maintain the stability of the gradient.

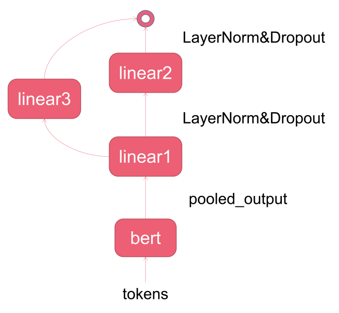

Figure 3: Structure of linear classifier

In fact,the BERT-model itself is made of 12 encoders and has the ability to process long term information.Further semantic information modeling may lead to the additional parameters and deterioration of the result.

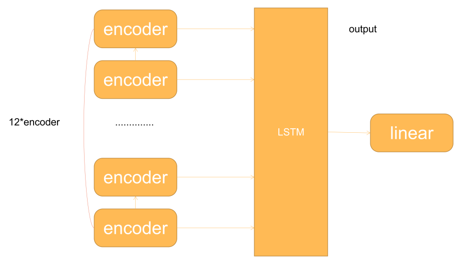

Figure 4: Structure of lstm classifier

3.3.2 Different Features

Bert is composed of 12 layers of transformer modules. It is argued that the low level is biased toward grammatical feature learning, and the high level is biased towards semantic feature learning. Bert's own pooling output is obtained through a simple net with [CLS] of the last layer. In this section, we try to find the best combination of 12 [CLS].

Table 2 shows the performance of fine-tuning BERT with different layers. The feature from the last layer of BERT and last four layers with max pool achieve good results.

Table 2: Fine-tuning BERT with different layers

Layer 0	0.8341
Layer 1	0.8472
Layer 2	0.8657
Layer 3	0.8710
Layer 4	0.8714
Layer 5	0.8697
Layer 6	0.8690
Layer 7	0.8721
Layer 8	0.8736
Layer 9	0.8729
Layer 10	0.8738
Layer 11	**0.8755**
First four Layers with concat	0.8693
First four Layers with mean pool	0.8712
First four Layers with max pool	0.8697
Last four Layers with concat	0.8717
Last four Layers with mean pool	0.8740
Last four Layers with max pool	**0.8743**
Even Layers with max pool	0.8739
Odd Layers with max pool	0.8741
All 12 Layer2 with concat	0.8709

3.3.3 Analysis

Through the above experiments, we believe that the concise classifier is more suitable for BERT. And the last layer of BERT gives the best performance. Thus, we use this setting in subsequent experiments

3.4 Exp-II: Layer-wise Learning Rate

Table 3 shows the performance of different base learning rates and decay factors on our validation dataset. We find that assigning a lower learning rate to the lower layer is effective to fine-tune BERT, and an appropriate setting is lr = 2.0e-5 and df = 0.95.

Learning rate	Decay factor	Acc
2.5e-5	1.00	0.8704
2.5e-5	0.95	0.8762
2.5e-5	0.90	0.8719
2.5e-5	0.85	0.8714
2.0e-5	1.00	0.8758
2.0e-5	0.95	0.8726
2.0e-5	0.90	0.8684
2.0e-5	0.85	0.8670

Table 3: Layer-wise learning rate

3.5 Exp-III: Investigating Different Data Augmentation

We do all the experiments using learning rate as 2.0e-5 and decay rate as 0.95.

Augmentation	Validation acc
None	0.8762
Delete	0.8727
Syn	0.8729
Shuffle	0.8679
Mask	0.8642

Table 4: Data Augmentation

To our surprise, We find data augmentation have no help to our model. Actually, we think this is because of the imbalance of our train data, the augmentation strength the imbalance and influence the result.

3.6 Exp-IV: Investigating Different Fine-Tuning Strategies

3.6.1 Further Pretraining

In order to verify the improvement brought by this method, an ablation experiment is conducted.

As is shown in figure 5, further pretraining definitely helps the model to converge faster and better.

Besides, comparing the one pretrained on *MLM* task with that on *MLM* and *SIM* task, it is obvious that the *SIM* task improves the model's ability of understanding sentences in our datasets.

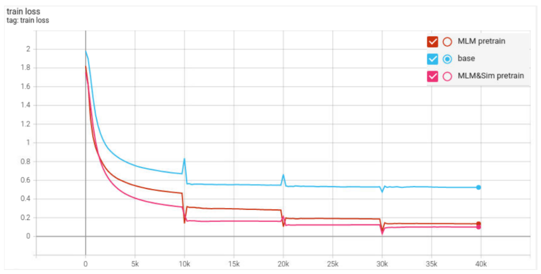

Figure 5: Training loss of different further pretrain methods, representing the ones with no further pretraining, further pretrained on the *MLM* task and on both *MLM and SIM* tasks respectively.

3.6.2 Adversarial learning and other attempts

Adversarial training,such as *FGM* (Miyato et al., 2017) and *FreeLB* (Zhu et al., 2019) is an effective trick by adding disturbances to construct some confrontation samples, then put them to the model for training.In some way it can enhance the robustness of the model when encountering confrontation samples, and improve the performance and generalization ability of the model to a certain extent.However,after many attempts and fine-tuning, the adversarial training did not significantly increase the points of the model in the validation dataset6.

It is speculated that it may be due to the imbalance of the dataset,causing it difficult to find a suitable gradient rising point.

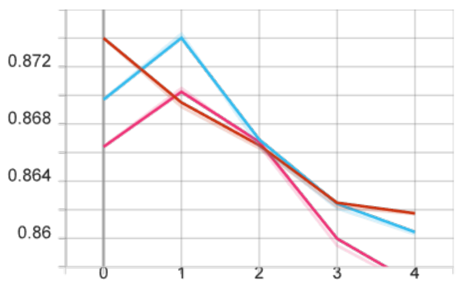

Figure 6: Adversarial-training accuracy

4 Conclusion

In this paper, we conduct extensive experiments to investigate the different approaches to using BERT for the sentiment classification task. We find that the two linear-layers classifier fit BERT most and the top layer of BERT is more useful for sentiment classification. What's more, an appropriate layer-wise decreasing learning rate can improve our model to some extent. Last but not least, further pretraining on the dataset can help our model converge faster and better.

In the feature, we will probe more insight of BERT into how it works.

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*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. *arXiv: Machine Learning*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2019. Freelb: Enhanced adversarial training for natural language understanding. *arXiv preprint arXiv:1909.11764*.