NLU 2020 Spring – CoLA Task

xx 019xxxxxxxxx
Shanghai Jiaotong University
huntersx@sjtu.edu.cn

Abstract

Natural Language Understanding (NLU) is a promising but challenging research in Artificial Intelligence, and GLUE is a collection of tools designed for evaluating the performance of models across a diverse set of existing NLU tasks. In this experiment, I mainly use four pre-training models including BERT, RoBERTa, ALBERT and ELECTRA to evaluate the performance of CoLA task in GLUE Benchmark. From the results we can see that Large models usually perform better than Base models, and different setups of parameters such as learning rate, random seeds and batch size will lead to different performance. Among all these tested models, ELECTRA-Large can achieve best performance. By adjusting the parameters, my single ELECTRA-Large model achieves **71.9** points on the CoLA task, surpassing ELECTRA's official performance on GLUE Benchmark Leaderboard (71.7). The whole code and setup of this experiment is available at `https://github.com/Huntersxsx/SJTU-NLU2020-CoLA`, and I also zipped code and fine-tuned model at `https://pan.baidu.com/s/1siiKqY_WxIalJkMBRSzpYQ`, the password is rvyr.

1 Task Description

This experiment is to do the CoLA task (Warstadt et al., 2019) in GLUE Benchmark (Wang et al., 2018). The purpose of CoLA (Corpus of Linguistic Acceptability) is to judge whether an English sentence is grammatically acceptable or not. The CoLA dataset consists of 10,657 sentences from 23 linguistic publications, expertly annotated for grammatically acceptability by their original authors. In the published data set, the training set contains 8,851 sentences, the validation set contains 1,043 sentences, and the test set contains 1,063 sentences. The COLA task is a binary task at sentence level and the evaluation metric is Matthew's Corr. What we need do is to upload the formatted predictions for the test set to the GLUE website [1], and only the score of CoLA task does matter.

2 Model Introduction

The CoLA task in this experiment is a Single Sentence Classification Task. I mainly used the pre-training models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019) and ELECTRA (Clark et al., 2020) for this experiment. The following is a brief introduction to these pre-training models.

2.1 BERT

BERT (Devlin et al., 2018) is short for Bidirectional Encoder Representation from Transformers, which is a multi-layer bidirectional Transformer Encoder (Vaswani et al., 2017). BERT is a deep bidirectional pre-training language model and can be directly used for the fine-tuning of many downstream NLP tasks. The appearance of BERT makes the application of pre-training technology in NLP develop rapidly and widely used in the industry.
The model is composed of Embedding layer, hidden layers (Tansformer Encoder) and output layer, as shown in the left of Figure 1. The input embeddings is the sum of Token Embeddings, Segmentation Embeddings and Position Embeddings. The hidden layer is formed by multi-layer Transformer Encoder, which has both extraction long-distance dependencies ability of RNN and parallel computing ability of CNN, which is mainly due to the self-attention mechanism utilized in Transformer. When calculating the current word and word from its context, Transformer can extract long-distance dependencies between words, and can compute all words at the same time due to the

[1]GLUE Benchmark: https://gluebenchmark.com.

calculation of each word is independent. To get a better bidirectional representation, the author used the joint training of two novel tasks during pre-training stage, including masked language model (MLM) task and sentence coherence determination (NSP) task.

(1) Masked Language Model (MLM)

The MLM task randomly covers 15% of the tokens in the input, and replaces them with token [MASK]. The purpose of this task is to predict the covered tokens through other words, thus learning the context features, grammatical structure features, and syntactic features of the words through iterative training. However, if always using token [MASK] to substitute random tokens, it can lead to a pretrain-finetune discrepancy because the artificial symbols like [MASK] used during pretraining are absent from real data at finetuning time. To alleviate this kind discrepancy, the author replaces the word with the [MASK] token in 80% of the time, replaces the word with a random word tokens in 10% of the time and keep the word unchanged in 10% of the time. This makes the model rely more on context information to predict the current word, thus giving the model a certain ability to correct errors.

(2) Next Sentence Prediction (NSP)

The NSP task is aimed to train a model that understands sentence relationships, when choosing the sentences A and B for pre-training, 50% of the time B is the actual next sentence that follows A, and 50% of the time it is a random sentence from the corpus. Through iterative training, the relationship between sentences can be learned, which is particularly important for text matching tasks such as Question Answering (QA) and Natural Language Inference (NLI).

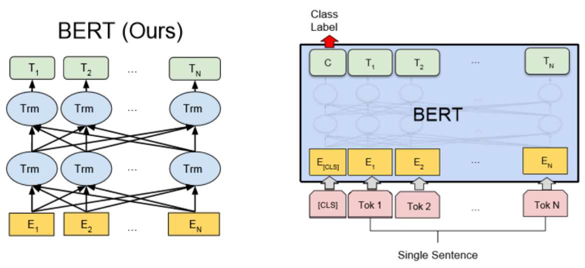


Figure 1: (Left) The architecture of BERT. (Right) Single Sentence Classification Tasks: SST-2, CoLA.

BERT can be transferred to different downstream tasks in NLP field through fine-tuning. Among these downstream tasks, CoLA is a binary classification task of a single sentence and can be predicted by using the structure shown in the right of Figure 1. The category can be classified by the output of the special [CLS] word embedding.

Overall, BERT is the SOTA model up to October 2018, which has swept 11 NLP tasks through pre-training and fine tuning. It uses bidirectional Transformer, which can simultaneously extract features using the context information of the current word and dynamically adjust the word embedding according to different context information, thus solving the polysemological problem. As BERT uses Transformer Encoder as a backbone, it has the ability of parallel computing, and is more efficient than RNN. When do the downstream tasks, it only needs to load the pre-trained model as the word embedding layer, and then simply modify the model for the specific task to get good results.

2.2 RoBERTa

RoBERTa (Liu et al., 2019) improves BERT in the training method, which was mainly reflected in four aspects: changing the way of Mask, discarding NSP task, optimizing training hyperparameters and using larger training data.

(1) Using dynamic Mask way

BERT randomly selected 15% of tokens at the beginning and then never changed throughout the pre-training process, this is static Masking. RoBERTa proposes dynamic Masking by preparing ten copies of the data in the beginning, thus the same sentence has 10 different ways of masking. This dynamic way adds training data indirectly and helps to improve model performance

(2) Discarding NSP loss

In order to capture the relationship between sentences, BERT used the NSP task for pre-training, which was proved not effective in many tasks (Lample and Conneau, 2019). RoBERTa, on the other hand, removes the NSP loss and adpots Full-Sentences training, which means each input is packed with full sentences sampled contiguously from one or more documents, such that the total length is at most 512 tokens. It was found that removing the NSP loss matches or slightly improves downstream task performance. Using Full-Sentence training is able to capture longer dependencies, which is more friendly to downstream tasks with long sequences.

(3) Larger batche size

The batch size of $BERT_{BASE}$ is 256 and the training steps is 1M. RoBERTa increases the batch size to 8K and finds that training with large batches improves perplexity for the masked language modeling objective, as well as end-task accuracy. What's more, large batches are also proved to be easier to parallelize via distributed data parallel training

(4) More training data

RoBERTa uses 160G data, which is 10 times of BERT (16G). Obviously, more training data increases data diversity and improves model performance.

2.3 ALBERT

ALBERT (Lan et al., 2019) adopts a new parameter sharing mechanism, which not only improves the overall effect of the model, but also has much fewer parameters than BERT. For the pre-training model, improving the size of the model can improve the effect of downstream tasks to some extent. However, if the model is further improved, OOM problems will inevitably occur, and long-term training may also lead to model degradation. ALBERT proposes two methods to reduce memory and increase training speed, and also proposes a novel task replacing NSP task in BERT.

(1) Factorized embedding parameterization

In BERT,the embedding size E is tied with the hidden layer size H, i.e., $E \equiv H$. but this decision appears suboptimal for both modeling and practical reasons. From a modeling perspective, untying the E from H allows us to make a more efficient usage of the total model parameters, which means $E << H$. From a practical perspective, the model will have billions of parameters if $E \equiv H$. Therefore, ALBERT uses a factorization of the embedding parameters, first projecting the one-hot vectors into a lower dimensional embedding space of size E, and then project it to the hidden space H. This trick reduces the embedding parameters from $O(V \times H)$ to $O(V \times E + E \times H)$, and this parameter reduction is significant when $H << E$.

(2) Cross-layer parameter sharing

Cross-layer parameter sharing is another way to improve parameter efficiency. The structure unit of BERT is Transformer, and Transformer has many ways to share parameters, only sharing feed-forward network parameters, or only sharing attention parameters. The default decision for ALBERT is to share all parameters across layers. Although the performance may drop a little, but the amounts of parameters has gone down a lot and training speed has improved a lot. Moreover, experiments prove that weight-sharing has an effect on stabilizing network parameters.

(3) Sentence Ordering Prediction (SOP)

Previous research (Lample and Conneau, 2019) has found the NSP task is not effective in many tasks, the reason is that NSP actually contains two subtasks, topic prediction and coherence prediction. However, topic prediction is easier to learn and also overlaps more with what is learned using MLM loss. ALBERT uses a sentence-order prediction (SOP) loss, which avoids topic prediction and instead focuses on modeling inter-sentence coherence.

(4) Discarding Dropout Layer

In addition, the author also found that the model still had no overfitting after 100W training steps, so the he removed Dropout Layer to improve the effect of downstream tasks. This is also the first time that dropout has a negative impact on large-scale pre-training models.

2.4 ELECTRA

Although the MLM pre-training method in BERT achieves better results than LM with the help of bidirectional representation, it also brings a lot of calculation costs. This pre-training method only learns from 15% tokens, and its calculation efficiency is too low. Therefore ELECTRA (Clark et al., 2020) uses the replaced token detection task, which is to predict whether each token in the corrupted input is replaced by a generator sample or not, making model learn from all the token. Figure 2 shows the framework of Replaced Token Detection task proposed in ELECTRA.

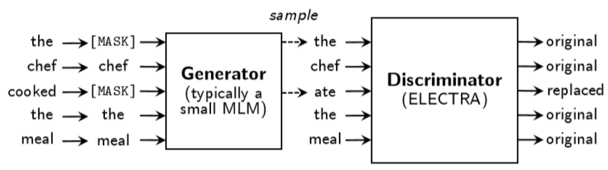


Figure 2: The framework of Replaced Token Detection task in ELECTRA.

As shown in Figure 2, ELECTRA contains two neural networks, a generator G and a discriminator D. Each of them consists of an encoder based on Transformer. For a given input $\mathbf{x} = [x_1, x_2, \cdots, x_n]$, the generator does MLM tasks, choose some random location for mask, get $\mathbf{x}^{masked}$, and then pre-

dicts the original representation of these masked tokens, get $\mathbf{x}^{corrupt}$. The discriminator D will predict which tokens in $\mathbf{x}^{corrupt}$ match the original input $\mathbf{x}$.
Although similar to the training objective of a GAN, there are several key differences. First, if the generator happens to generate the correct token, that token is considered "real" instead of "fake". More importantly, the generator is trained with maximum likelihood rather than being trained adversarially to fool the discriminator. Lastly, ELECTRA does not supply the generator with a noise vector as input, as is typical with a GAN.
In summary, this Replaced Token Detection task is more effective than MLM because the model learns from all the input tokens instead of just the ones that are concealed. As a result, with the same model size, data and calculation conditions, context representation learned by ELECTRA is significantly better than context representation learned by methods such as BERT and XLNet, thus can achieve better performance on downstream tasks. Moreover, ELECTRA works well even with a relatively small amount of computation.

3 Experiments

In this experiment, I evaluated the performance of BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019) and ELECTRA (Clark et al., 2020) on CoLA task with the help of two open source code, Huggingface Transformers [2] and ELECTRA-pytorch [3].
First, I downloaded the Github Repositories and prepared required environment, and then I rewrite some parts of the code to make it be able to evaluate the performance on CoLA test set. I did not pre-train the model from scratch and directly used pre-trained models offered by official source to fine-tune on CoLA task. In order to achieve better performance, I fine-tuned these models with different hyperparameters. The fine-tune hyperparameters I used are listed in Table 1, and I mainly adjusted Max Sequence Length, Batch Size, Learning Rate and Random Seed to obtain different performance. With ablation studies, I found that models perform better when Max Sequence Length is set to 80, and Batch Size is set to 32. The effect of Learning Rate is not obvious so I only tried 2e-5 or 1e-5. However, the performance differs a lot when Random Seed is set to different values, therefore, I tried many Random Seed values for better performance.

[2]Huggingface: https://github.com/huggingface/transformers.
[3]ELECTRA: https://github.com/lonePatient/electra_pytorch

Hyperparameter	**CoLA Value**
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.99
Warmup proportion	0.1
Weight Decay	0.01
Max Sequence Length	128/ 80
Learning Rate	2e-5/ 1e-5
Batch Size	32/16
Train Epochs	3.0
Random Seed	42/ 50/ $\cdots$

Table 1: Fine-tune hyperparameters

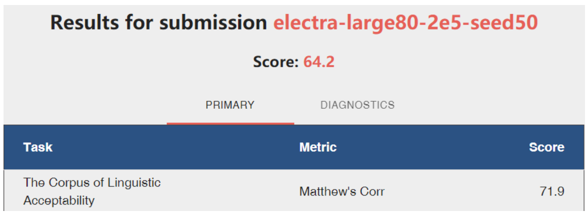
Model	**Matthew's Corr**
BERT-Base	43.4
BERT-Base cased	51.8
RoBERTa-Base	59.3
ELECTRA-Base	**66.4**
BERT-Large cased	62.3
ALBERT-Large	61.0
RoBERTa-Large	61.4
ELECTRA-Large	**71.9**

Table 2: Results for models on CoLA test set. Only models with single task finetuning (no ensembling, task specific tricks, etc) and best results are shown. The screenshots of results can be found in Appendix

Hyperparameter	**Base/ Large**
Max Sequence Length	80/ 80
Learning Rate	2e-5/ 2e-5
Batch Size	32/ 32
Random Seed	50/ 50

Table 3: Some hyperparameters of best performance ELECTRA-Base and ELECTRA-Large

I choose the best performance evaluated by different parameters, and show the results of BERT-Base, BERT-Base cased, RoBERTa-Base, ELECTRA-Base, BERT-Large cased, ALBERT-Large, RoBERTa-Large and ELECTRA-Large in Table 2. In fact, I also used XLNet to do the experiment, but I got very poor performance perhaps for my incorrect hyperparameters setup. And for RoBERTa and ALBERT, I only tried one set of parameters to get the result, so the performance of RoBERTa and ALBERT is much poorer than


Results for submission electra-large80-2e5-seed50

Score: 64.2

PRIMARY DIAGNOSTICS

Task	Metric	Score
The Corpus of Linguistic Acceptability	Matthew's Corr	71.9

Figure 3: The screenshot of my best submission.

the official results. However, considering the limitation of time and GPU, as well as the satisfying performance of ELECTRA, I gave up continuing trying with ALBERT, RoBERTa and XLNet, and mainly focused on the ELECTRA model. With some effort, I achieved 66.4 with ELECTRA-Base and 71.9 with ELECTRA-Large, both of which are higher than offical performance. You can see Figure 3 for confirmation.

The hyperparameters used in the best performance ELECTRA-Base and the best performance ELECTRA-Large are shown in Table 3, and other models' are available in the Appendix.

4 Conclusion

In this experiment, I did a research on Pre-training Models in NLP, and used four of them including BERT, RoBERTa, ALBERT and ELECTRA to complete CoLA task. I directly used the pre-trained models offered by official source and fine-tuned them with CoLA task. With different setups of parameters, I achieved different performance. According to the results, I find that Large models tend to have better performance because of their much more parameters, and ELECTRA-Large comes first among all the models. With the fine-tuning of parameters, I finally achieved **71.9** points in CoLA task using ELECTRA-Large single model, which is higher than ELECTRA Team's. In the future, I think I can try more pre-training models or utilize ensemble tricks to get better performance.

References

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

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*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

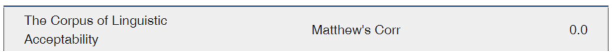
Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

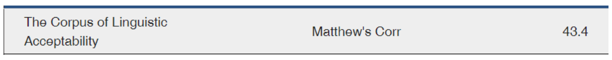
Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

A Appendix



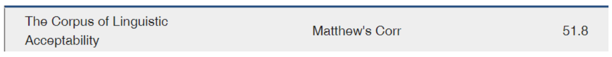
The Corpus of Linguistic Acceptability	Matthew's Corr	0.0

(a) All one



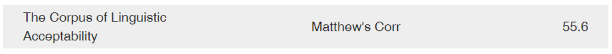
The Corpus of Linguistic Acceptability	Matthew's Corr	43.4

(b) Bert base uncased: max seq length=128, batch size=32, learning rate=2e-5, epochs=3.0



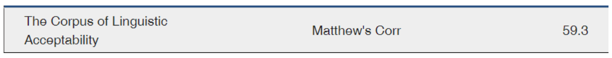
The Corpus of Linguistic Acceptability	Matthew's Corr	51.8

(c) Bert base cased: max seq length=128, batch size=32, learning rate=2e-5, epochs=3.0



The Corpus of Linguistic Acceptability	Matthew's Corr	55.6

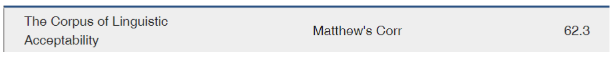
(d) Roberta base: max seq length=128, batch size=32, learning rate=2e-5, epochs=3.0



The Corpus of Linguistic Acceptability	Matthew's Corr	59.3

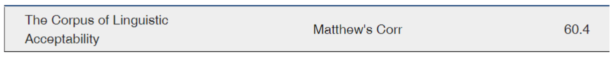
(e) Roberta base: max seq length=80, batch size=32, learning rate=2e-5, epochs=3.0

Figure 4: Some screenshots of my submitted results



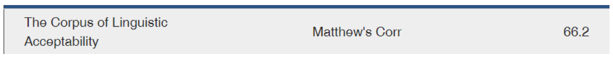
The Corpus of Linguistic Acceptability	Matthew's Corr	62.3

(a) Bert large cased: max seq length=80, batch size=16, learning rate=2e-5, epochs=3.0



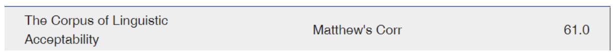
The Corpus of Linguistic Acceptability	Matthew's Corr	60.4

(b) ELECTRA base: max seq length=80, batch size=16, learning rate=1e-4, epochs=3.0



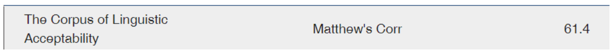
The Corpus of Linguistic Acceptability	Matthew's Corr	66.2

(c) ELECTRA base: max seq length=80, batch size=32, learning rate=2e-5, epochs=3.0



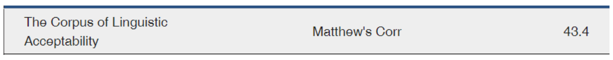
The Corpus of Linguistic Acceptability	Matthew's Corr	61.0

(d) ALBERT large vv2 max seq length=80, batch size=16, learning rate=2e-5, epochs=3.0



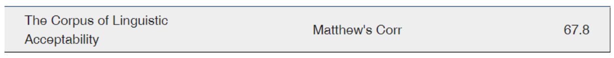
The Corpus of Linguistic Acceptability	Matthew's Corr	61.4

(e) RoBERTa large: max seq length=80, batch size=16, learning rate=1e-5, epochs=3.0



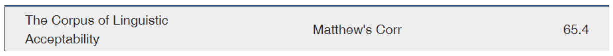
The Corpus of Linguistic Acceptability	Matthew's Corr	43.4

(f) Bert base uncased: max seq length=128, batch size=32, learning rate=2e-5, epochs=3.0



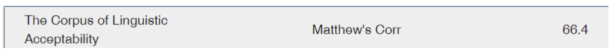
The Corpus of Linguistic Acceptability	Matthew's Corr	67.8

(g) ELECTRA large: max seq length=80, batch size=16, learning rate=2e-5, epochs=3.0, seed=42



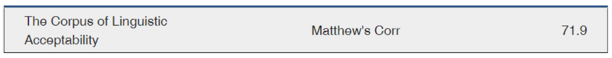
The Corpus of Linguistic Acceptability	Matthew's Corr	65.4

(h) ELECTRA base: max seq length=80, batch size=32, learning rate=2e-5, epochs=3.0, seed=36



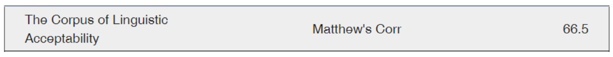
The Corpus of Linguistic Acceptability	Matthew's Corr	66.4

(i) ELECTRA base: max seq length=80, batch size=32, learning rate=2e-5, epochs=3.0, seed=50



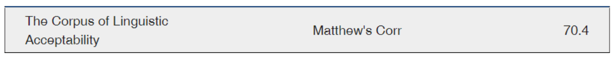
The Corpus of Linguistic Acceptability	Matthew's Corr	71.9

(j) ELECTRA large: max seq length=80, batch size=32, learning rate=2e-5, epochs=3.0, seed=50



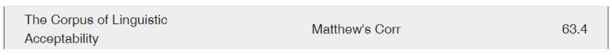
The Corpus of Linguistic Acceptability	Matthew's Corr	66.5

(k) ELECTRA base: max seq length=80, batch size=32, learning rate=2e-5, epochs=3.0, seed=64



The Corpus of Linguistic Acceptability	Matthew's Corr	70.4

(l) ELECTRA large: max seq length=80, batch size=32, learning rate=2e-5, epochs=3.0, seed=64



The Corpus of Linguistic Acceptability	Matthew's Corr	63.4

(m) ELECTRA base: max seq length=80, batch size=32, learning rate=2e-5, epochs=3.0, seed=72

Figure 5: More screenshots of my submitted results