1、Language processing model design

In recent years, the trend of pre-training has changed from learning embedding to learning the entire network, that is, the transition from word2vec to BERT. Another trend is that large models are very important to the final effect . It is not difficult to see that the stronger the model, the larger the parameter. Since the parameter quantity of the model is so important, we initially made a very straightforward assumption: the larger the parameter quantity, the higher the performance of the model.

If the hardware resources and training time are not considered, and if the above assumptions are true, then we only need to find ways to expand the model scale. To verify the hypothesis, we doubled the size of the hidden layer of the BERT-large model and constructed the BERT-xlarge model. The parameter amount of this model is doubled compared with BERT-large, but unfortunately, as shown in Figure 1, BERT-xlarge has a model degradation phenomenon, and the performance does not increase but decreases. Therefore, simply pushing up the amount of parameters will not only face more severe problems of insufficient hardware resources and excessive training time, but also fail to obtain better results.

Therefore, we found that on BERT, adjusting the model size to a certain level can no longer make the effect better, but will worsen. As shown below:

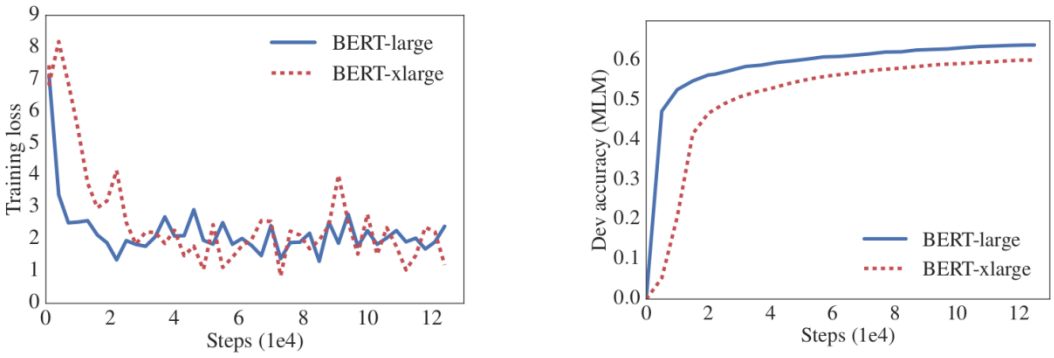

Figure 1 Comparison of the effects of BERT and xBERT models

Our approach is to "do subtraction" on the basis of BERT to reduce the amount of parameters, but improve the efficiency of parameter utilization and maximize the effect of parameters. The ALBERT model we use is based on "doing subtraction" and fully surpasses BERT in performance. It has the following characteristics:

Matrix Factorization

In the original BERT, because of the residual, the Hidden size of each layer is the same, and it also makes the embedding size the same as the

hidden size. However, because the vocabulary is relatively large, the word piece of 3w is used in BERT, which will cause embedding to account for a large part of the parameters of the model.

From the perspective of the model, embedding parameters learn context-free expressions, while BERT's model structure learns context-sensitive expressions. Previous research results show that the good effects of BERT and similar models are more derived from the capture of contextual information. Therefore, embedding occupies so many parameters, which will result in very sparse final parameters.

Therefore, instead of using such a large embedding size, we use a smaller size, but in this way, it is impossible to make residuals in the first layer. It doesn't matter, we can use another parameter matrix to project the embedding size onto the hidden size. That is, assuming that V is the vocabulary size, E is the embedding size, and H is the hidden size, then E=H in the original BERT, the number of parameters is VxH, and the parameter data after decomposition is VxE+ExH, because E is much smaller than H, So after the decomposition, the parameters of this part are much less.

Parameter Sharing Between Levels

The level of BERT is very deep, at least 10 levels or more. In this way, if the parameters between levels can be shared, the amount of parameters will be greatly reduced.

Not only that, after sharing the parameters, the model will be more stable. The performance in this respect is that the L2 distance between the input and output of each layer of the model is smaller. As shown below:

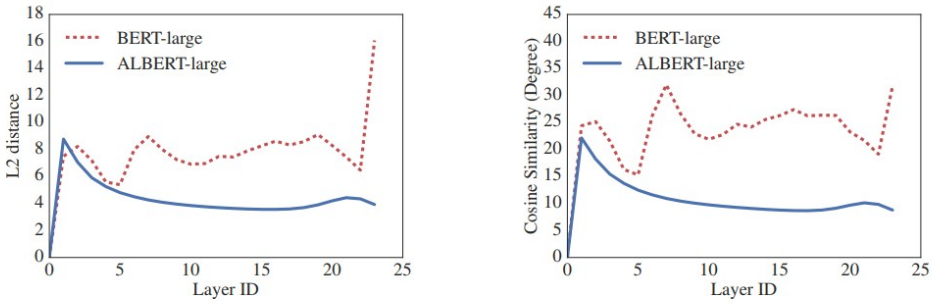

Figure 2 The L2 distances and cosine similarity (int terms of degree) of the input and output embedding of each layer fir BERT-large and ALBERT-large

Although ALBERT solves the problem of model size, the problem of calculation amount is not solved, because parameter sharing between levels

does not omit calculation. Therefore, we can also try to improve the sparsity of the network here.

2、Model Training

The ALBERT-xxlarge model we finally adopted finally achieved 89% accuracy on the valid data set given by the competition, which was higher than the 86% accuracy of BERT-xxlarge .

We took two 8G memory of the GPU distributed training, training batch_size is 2 , learning_rate as 1E-5 , num_train_epochs to 4 .

3、Code Architecture

The structure of our code is as follows:

```
├── script
│  ├──data                //data in .pt format
│  │  ├──raw_data          //raw data in .json format
│  ├──debug-exp  //code running log
│  ├──model               //pre-training nd final result model
│  ├──pretrained_roberta //model definition module
│  │  ├──__init__.py       //module identification file
│  │  ├──__main__.py
│  │  ├──__modeling__.py          //model architecture
│  ├──prediction                  //prediction of the test data
│  ├──data_util.py //preprocessing for original data
│  ├──main.py                                  //main file
│  ├──README                                   //README
```

4、Testing Method

The judges can run the main.py(as in README)to run the prediction function on the test dataset again and the generated answer file answer.json is in the test directory.