Final Report

SONG Huancheng 20635473 hsongag@connect.ust.hk

Abstract

BERT is a widely used pretraining model released by google has achieved many significant progresses in NLP tasks such as language generation and language understanding.

Text similarity analysis is a major subject of language understanding and It's found that few people had paid attention to similarity of academic papers, which encourage me to explore this area. By understanding and identifying similarity information between academic papers, we can improve the understanding of scientific research context and achieve a progress on scientific research analysis and automatically abstract generating.

In this work, an introduction as well as research on BERT and its derived model – ALBERT has been performed. Besides, these two models are implemented to do the classification of research text similarity and compared with each other to further explore the effectiveness of BERT.

Background

Structure of transformer

BERT is designed based on the structure of transformer, which uses a set of encoders that contain multi-head attention sublayer and a feed forward sublayer. The connection of two sublayers is a residual connection and each sublayer is followed by a Layer Normalization which takes (x+sublayer(x)) as input. The input of encoder is a set of matrices named quires (Q), keys (K) and values (V) respectively. Since in the encoder, the attention layers are all self-attention, the Q, K, V are all generated from inputs. Besides, due to the reason that attention does not sequentially encodes the input and loses the information about positions of inputs, a positional encoding is used to record the order of sequence.

In decoder layers, a multi-head attention sublayer is added between feed forward sublayer and self-attention sublayer, which takes Q, K from encoder and V from self-attention layer.

BERT

Bidirectional Encoder Representations from Transformers (BERT) is a method of pre-training language representations, meaning that we train a general-purpose "language understanding" model on a large text corpus (like Wikipedia), and then use that model for downstream NLP tasks that we care about (like question answering) [7]

The steps of using BERT are mainly divided into two parts: pretraining and fine-tuning. During pretraining procedure, unlabeled data is used to train the model on several different tasks and both single sentences as well as sentence-pairs are used as input so that the pretrained model will be able to fit in a large range of downstream fine-tunings. The author proposed two tasks in pretraining procedure, the first is Masked Language Model (MLM), which is proved the most significant modification in this paper. It's based on the thought of cloze test that randomly mask a proportion of tokens and use [mask] token to replace them and the task of model is to predict these masked tokens correctly. This task helps the model free from unidirectional and turn to literally bidirectional.

Another task of BERT in during pretraining is Next Sentence Prediction (NSP), which is proved less effective in the future work. NSP pretrained the model with text pair representations, which is composed of sentences A and sentences B joined by a [SEP] token and 50% of B are chosen from the actual next sentences of A and the other 50% are randomly chosen from corpus.

During Fine-tuning procedure, the parameters in encoder is locked and we only train the parameters in decoder, which is fine tuned end to end by plug the output of ender to a specific task as its input.

ALBERT

After BERT was released, a lot of researches had been done to further explore the effects and improvements of BERT. A Lite BERT (ALBERT)[6] is one of the improved model that make a significant improvement to the training speed as well as parameters size.

From BERT, There are two major directions of improving the performance, the first is increasing training size and the second is implementing a better designed structure of BERT. In ALBERT, it is proposed that although enlarge training size helps, the longer running time and limitation of GPU/TPU as well as latent model degradation makes new difficulties. Therefore, ALBERT focused on designing a better model and proposed two parameter reduction techniques to reduce the consumption of memory and enhance the speed of BERT training. Besides, ALBERT used a self-supervised loss to modeling inter-sentence coherence which made a progress during the training of multi-sentence inputs.

1) The first technique is Factorized embedding parameterization which decompose the embedding matrix into two sub matrices. Through this way, the hidden layers are separated from vocabulary embedding, so that the vocabulary embedding size is no longer in need to increase with hidden size.
2) The second technique is Cross-layer parameter sharing which prevent parameters from increasing with depth of model. It's noted that in the experiments in ALBERT, the main reduction of parameters is due to Cross-layer parameter sharing, which may bring a decreasing of performance. However, the impact to the performance is relatively small, comparing with the large speed fostering.

Furthermore, ALBERT introduce a self-supervised loss to perform sentence-order prediction （SOP） as the replacement of NSP in original BERT in order to improve the low effectiveness of NSP.

Meeting Record

week1

An overall planning for the project time as well as the content of independent project had been made and it is confirmed that the meeting time was once a week.

Next week assignment: Making a present introducing BERT and FRAGE

week2

4 papers are read, one about attention [2], one transformer [3], one BERT [1] and one FRAGE [4], then on meeting time, I presented the overview of BERT and FRAGE with background introduction of transformer and attention. After that, a QA section was made for our three group mates to discuss our topic and ask questions to TA.

week3

The assignment was to read paper about RoBERTa [5] and ALBERT, compare the progress they had made and give a brief overview of these two models. And another task was to find a suitable Chinese corpus for BERT training, compute and estimate the possible run time using ALBERT based on the experiment settings/results in ALBERT, the size of Chinese corpus and GPU numbers. Also, a presentation was made to display the reading feedback and estimation result followed by a QA section.

week4/5

A group working was set up to apply BERT model to a neural machine translation task. After that, a task was released to connect BERT to SYNST model where BERT was used as encoder and SYNST was used as decoder. However, we didn't finish this task and stop at successfully encoding the input dataset with BERT but the connecting with SYNST was still under finished.

Dataset Introduction

The dataset used in this paper is from DigSci Scientific Data Mining Competition [8]. This dataset is composed of two sub datasets, the first is candidate.csv, which contains 200,000 papers for us to training. The second is train_release.csv. It stores the description text which contains cite to papers in candidate.csv. In feature engineering, the abstract of papers in candidate.csv and description text in train_release.csv are used as input to perform text similarity. We do classification of whether two text are relevant with 0/1 value.

Experiment

feature engineering

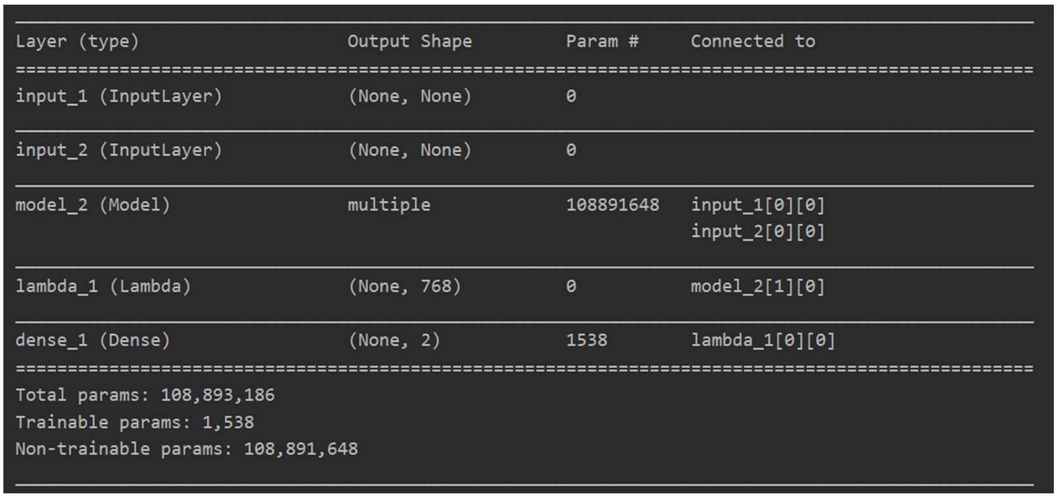
Due to the reason that the dataset is not designed for sentence pair classification between abstract and description text, a set of feature engineering operations need to be done. At first, abstract of each paper and its corresponding description text are extracted and the citing part in each description text is replaced with the title of paper. Then, all the records are labelled as 1, which denotes that the abstract of a paper is relevant with the description text. After that, a set of data with label 0 is creating by randomly shuffle the original data and shift its abstract to 1. Finally, merging the dataset with label 0 to the dataset with label 1 and a training dataset with size of (60623,3) is got. However, in order to reduce the training time, only 10% size of data is used and 20% of the input data is used as test dataset.

model setting

Keras is used as the deep learning tool of this work and keras_bert [3] is used to load the BERT model into keras.

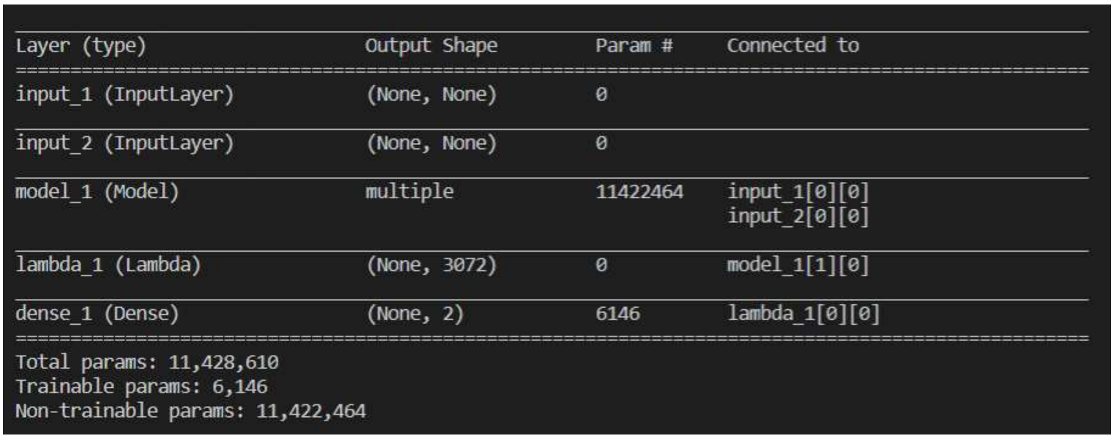
Figure 1 shows the structure of BERT model used in this work, from which we can see that before connecting to BERT, two Input layers are set which denotes each input sentences and the length of tokens of each sequence is 256 so that the the max_len of input is 512. In BERT model, a BERT base uncased model is chosen to be implemented. The specific parameters are <L=12, H=768, A=12> and the total parameter number is about 110M. However, pretraining procedure is not done in this work, so the parameters of MLM and NSP are not trainable. Then, a Dense layer is used to perform the classification. Besides, categorical cross entropy is used as loss function and Adam is used as optimizer with learning rate of 1e-5.

Figure 2 shows the structure of ALBERT model, if we compare the parameters of ALBERT with BERT, it can be found that ALBERT contains parameters almost ten times less than BERT, which may improve the speed of pretraining a lot. However, the number of parameters is still too large to be trained due to the limitation of hardware (causing a Resource exhausted error). Therefore, both the parameters of ALBERT and BERT are set to nontrainable.



```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, None)         0
__________________________________________________________________________________________________
input_2 (InputLayer)            (None, None)         0
__________________________________________________________________________________________________
model_2 (Model)                 multiple             108891648   input_1[0][0]
                                                                 input_2[0][0]
__________________________________________________________________________________________________
lambda_1 (Lambda)               (None, 768)          0           model_2[1][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 2)            1538        lambda_1[0][0]
==================================================================================================
Total params: 108,893,186
Trainable params: 1,538
Non-trainable params: 108,891,648
__________________________________________________________________________________________________
```

figure 1



```
________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================================
input_1 (InputLayer)            (None, None)         0
________________________________________________________________________________________________
input_2 (InputLayer)            (None, None)         0
________________________________________________________________________________________________
model_1 (Model)                 multiple             11422464    input_1[0][0]
                                                                 input_2[0][0]
________________________________________________________________________________________________
lambda_1 (Lambda)               (None, 3072)         0           model_1[1][0]
________________________________________________________________________________________________
dense_1 (Dense)                 (None, 2)            6146        lambda_1[0][0]
================================================================================================
Total params: 11,428,610
Trainable params: 6,146
Non-trainable params: 11,422,464
```

figure 2

result analysis

In this work, the mean_absolute_score (score), accuracy_score (acc) and f1 score (f1) are used to evaluate the performance of models.

table 1 shows the result of the experiment, we can see that the overall performance of these two models are both not very good, the possible reason may come from the dataset for the matching of abstract with description_text is not preciseness. Besides, BERT has an attribute that the larger training size, the better performance it achieves, so the small size of our dataset may not well stimulate the potential of BERT.

Another finding is that BERT outperforms ALBERT much. The possible reason may be the technique of Cross-layer parameter sharing ALBERT used brings a degradation of performance, whose influence has been enlarged due to the small data size and insufficient training iterations. Also, the frozen parameters of encoding layers of BERT and ALBERT brings a bad impact for the final evaluation and belies the advantages of ALBERT (the fast pretraining speed), which is reflected from the similar training time between BERT and ALBERT in this experiment.

	mean_absolute_score	accuracy_score	f1 score
BERT	0.8394	0.8087	0.7734
ALBERT	0.7639	0.6908	0.4086

table 1

Reference

[1] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[2] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

[3] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

[4] Gong, Chengyue, et al. "Frage: Frequency-agnostic word representation." Advances in neural information processing systems. 2018.

[5] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).

[6] Lan, Zhenzhong, et al. "Albert: A lite bert for self-supervised learning of language representations." arXiv preprint arXiv:1909.11942 (2019).

[7] DigSci Scientific Data Mining Competition https://www.biendata.com/competition/digsci2019/

[8] Keras-BERT https://github.com/CyberZHG/keras-bert/tree/master/keras_bert