Introduction

BERT (Bidirectional Encoder Representations from Transformers) [1] is a 2018 paper published by researchers at Google AI Language. It has inspired a whole new point of view in the Machine Learning community by presenting state-of-the-art performance in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), etc.

In this review, some critical concepts introduced by Transformer, which is a popular attention model used by BERT will be briefly presented. After that, part of the techniques such as bidirectional training of Transformer and Masked LM (MLM), etc., applied by BERT and result comparison will be gone through

Transformer

Before going deep dive the detail of BERT, we need briefly talk about the concept of Transformer since this algorithm used by BERT comes from this article in December 2017, Attention Is All You Need [2]. This article is to solve the problem of translation, such as translating from Chinese to English. This article completely abandons the RNN and CNN [3] that are often used in the past and proposes a new network structure, namely Transformer, which includes encoder and decoder.

One of the key concepts of transformer is Multi-headed Attention (Figure 1). The structure "matches" the query and key-value pairs, and outputs the sum of the weights of the value. The weight of the value comes from the attention value of the query and key. The Transformer structure uses a multi-head mechanism to calculate specific attention values in parallel.

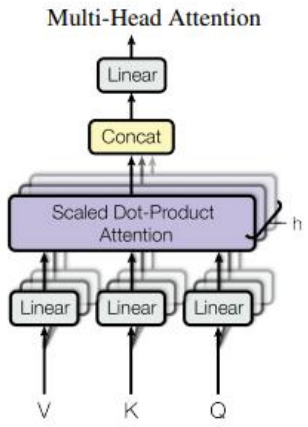

Figure 1: Multi-Head Attention

Another one of the key concepts of transformer is Positional Encoding. As the model has no recurrence and no convolution, A layer was added to make use of the sequence order. At the end of the encoder and decoder stacks, the information injected contains information about the relative or absolute position of the token in this sequence.

BERT

Before having BERT, the structure of the previous pre-training models was limited by the unidirectional language model (left-to-right or right-to-left), which also limits the representation ability of the model, so that it can only obtain unidirectional context information. BERT firstly constructed by a stack of Transformers (Figure 2), uses MLM for pre-training and a deep two-way Transformer component to build the entire model, thus finally generating a deep bidirectional language representation that can fuse the left and right context information.

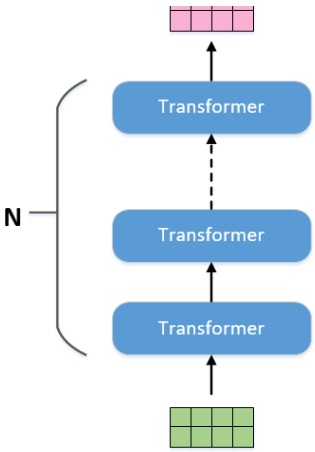

Figure 2: High-level Structure of BERT

Input of BERT

The input of BERT is the representation corresponding to each token, and the word dictionary is constructed using the WordPiece [3] algorithm. In order to complete the specific classification task, in addition to the token of the word, the author also inserts a specific classification token ([CLS]) at the

sequence, and the last Transformer layer output corresponding to the classification token is used to play the role of aggregating the entire sequence characterization information.

Since BERT is a pre-training model, it is required to be adapted to a variety of natural language tasks, the sequence input to the model must be capable of containing one sentence (text sentiment classification, sequence labeling tasks) or more than two sentences (text summarization, natural language inference, question answering tasks). So how to make the model have the ability to distinguish which range belongs to sentence A and which range belongs to sentence B? BERT uses two methods to solve it:

1. A [SEP] token is inserted after each sentence in the sequence of tokens to separate different sentence tokens.
2. Add a learnable Segment embedding to each token representation to indicate whether it belongs to sentence A or sentence B.

Other than the methods mentioned above, the practical implementation in BERT is combining Token, Segment and Position Embedding all together to enrich the information of embedding results

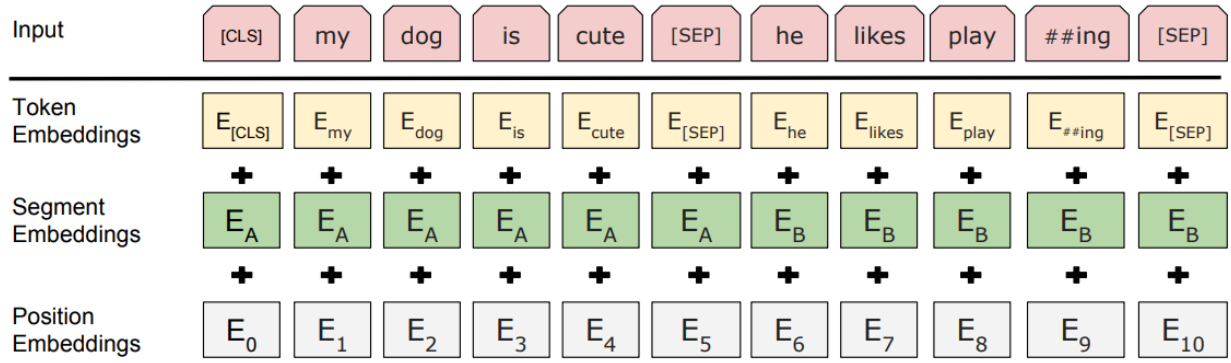

Figure 3:BERT input representation

Output of BERT

C is the output of the classification token ([CLS]) corresponding to the last Transformer and represents the output of other tokens corresponding to the last Transformer. For some token-level tasks (e.g., sequence labeling and Q&A tasks), the input is fed into an additional output layer for prediction. For some sentence-level tasks (such as natural language inference and sentiment classification tasks), C is input into an additional output layer, which explains why a specific classification token is inserted before each token sequence.

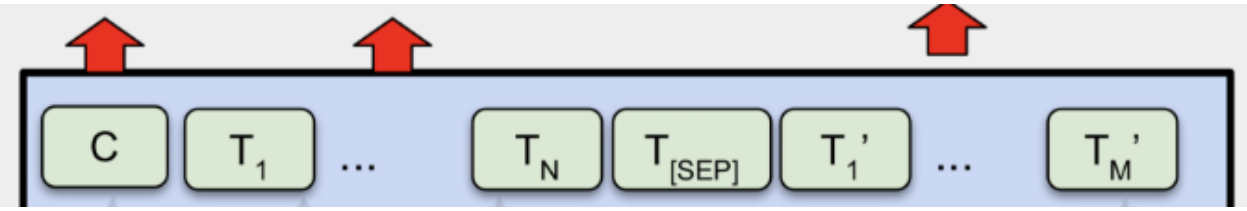

Figure 4: Output of BERT

Pre-Training Task of BERT

In fact, the concept of pre-training is very mature in CV (Computer Vision, computer vision) and is widely used. The pre-training task used in CV is generally the ImageNet image classification task. The premise of completing the image classification task is to be able to extract good image features. At the same time, the ImageNet dataset has the advantages of large scale and high quality, so it is often possible to obtain good results.

Although the field of NLP does not have high-quality human-annotated data like ImageNet, the self-supervised nature of large-scale text data can be used to construct pre-training tasks. Therefore, BERT builds two pre-training tasks, Masked Language Model and Next Sentence Prediction.

Masked Language Model (MLM)

MLM is the reason why BERT is not limited by one-way language models. Simply put, it randomly replaces the token in each training sequence with a mask token ([MASK]) with a probability of 15%, and then predicts the original word at the [MASK] position. However, since [MASK] does not appear in the fine-tuning stage of the downstream task, there is a mismatch between the pre-training stage and the fine-tuning stage (it is well explained here that the pre-training target will make the Linguistic representations are sensitive to [MASK], but not to other tokens). Therefore, BERT adopts the following strategies to solve this problem:

First, a token position is randomly selected for prediction with a probability of 15% in each training sequence. If the i-th token is selected, it will be replaced with one of the following three tokens:

1. 80% of the time it is [MASK]. For example, my cat is cute -> my cat is [MASK]
2. 10% of the time it is a random other token. For example, my cat is cute -> my cat is tree
3. 10% of the time is the original token. For example, my cat is cute -> my cat is cute

This strategy makes BERT no longer only sensitive to [MASK], but to all tokens, so that the representation information of any token can be extracted.

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

Figure 5: Ablation over different masking strategies

Next Sentence Prediction (NSP)

Some tasks such as question answering and natural language inference need to understand the relationship between two sentences, while MLM tasks tend to extract token-level representations, so sentence-level representations cannot be directly obtained. In order for the model to be able to understand the relationship between sentences, BERT uses the NSP task for pre-training, which is simply to predict whether two sentences are connected together. The specific method is that for each training example, we select sentence A and sentence B in the corpus to form, 50% of the time sentence B is the next sentence of sentence A (marked as IsNext), and the remaining 50% of the time Sentence B is a random sentence in the corpus (labeled NotNext). Next, input the training example into the BERT model, and use the C information corresponding to [CLS] to make two-class predictions.

Fine-Tuning

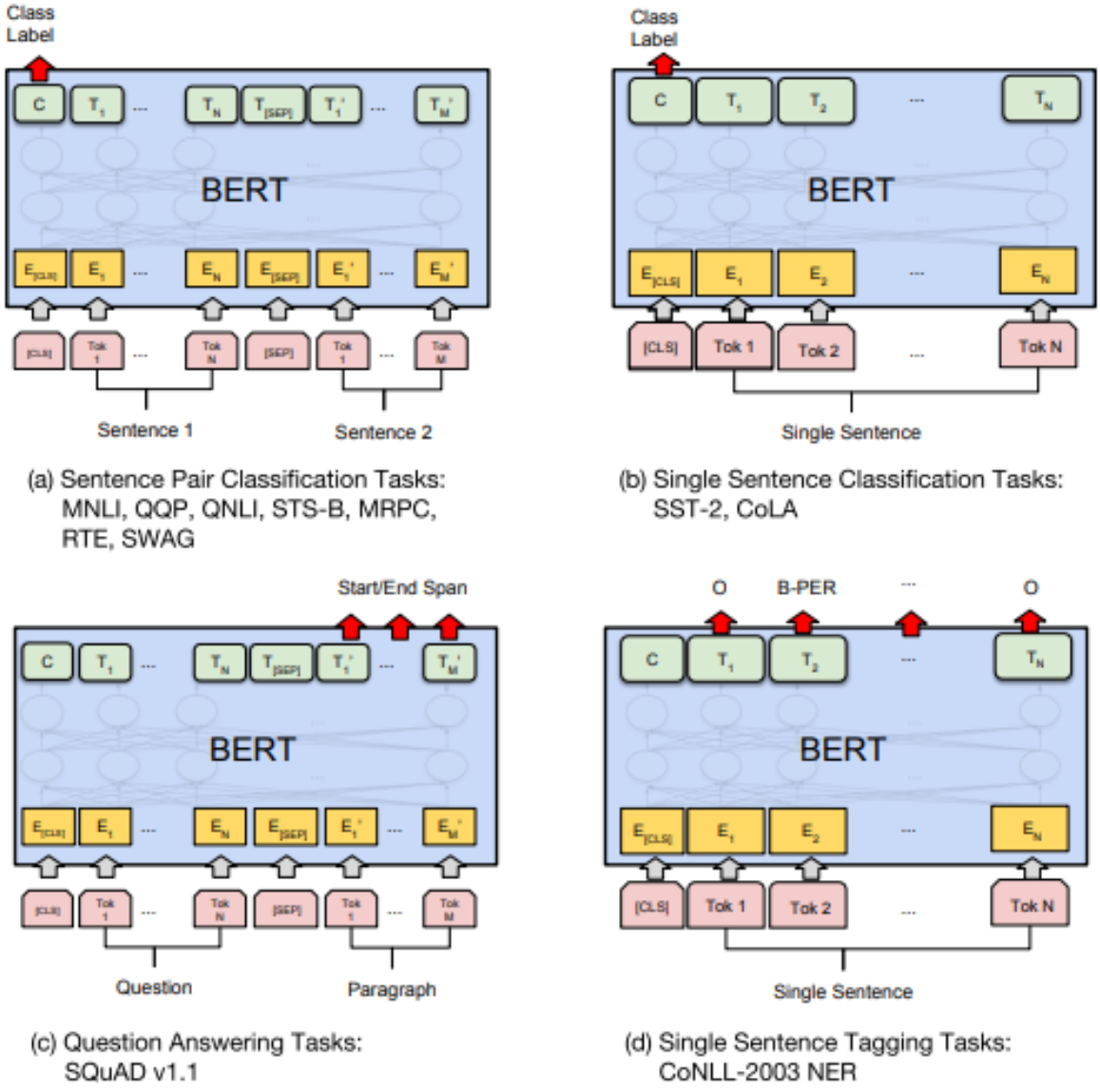

Figure 6: Illustrations of Fine-tuning BERT on Different Tasks

After completing the pre-training, it is necessary to fine-tune for specific tasks. There are 4 examples in the paper. First, let's talk about the classification task. The classification task includes the task of classifying a single sentence, such as judging whether a movie review whether positive or negative. Figure 6(a)(b) are the example of this kind of tasks, the left shows the classification of two sentences, and the right shows the classification of single sentences. In the output hidden vector, take out the vector C corresponding to CLS, add a layer of network W, and feed it to the softmax for classification to obtain the prediction result. All parameters of the Transformer model and the network W are jointly trained on the task-specific dataset until convergence.

For Q&A tasks, given a question and a paragraph, and then mark the specific location of the answer from the paragraph, as shown in Figure 6(c). It is necessary to learn a start vector S, the dimension of which is the same as that of the output hidden vector, and then do a dot product with all the hidden vectors to take the word with the largest value as the start position. In the meanwhile, learning another end vector E to do the same operation to get the end position with an additional condition that the end position must be greater than the start position.

For entity naming recognition (NER) tasks, as shown in Figure 6(d), a layer of classification network is added to make a judgment on each output hidden vector. It can be seen that these tasks only need to add a small number of parameters to train on a specific data set. From the experimental results, even a small dataset can achieve good results.

Result

SQuAD1.1 Leaderboard

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Oct 05, 2018	BERT (ensemble) *Google AI Language* https://arxiv.org/abs/1810.04805	87.433	93.160
2 Sep 09, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.356	91.202
3 Jul 11, 2018	QANet (ensemble) *Google Brain & CMU*	84.454	90.490

Glue Benchmark Leaderboard

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$BERT_{BASE}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Figure 7: BERT Performance in SQuAD1.1 and Glue

On SQuAD v1.1, BERT achieves 93.2% F1 score (a measure of accuracy), surpassing the previous state-of-the-art score of 91.6% and human-level score of 91.2%: BERT also improves the state-of-the-art by 7.6% absolute on the very challenging GLUE benchmark, a set of 9 diverse Natural Language Understanding (NLU) tasks.

Conclusion

BERT is a language representation model. It is trained with huge data, big model, and heavy computational overhead to achieve the state-of-the-art (SOTA) results in 11 natural language processing tasks in 2018 and it does inspire and provide us with a lot of valuable experience:

1. Deep learning is representation learning
   "*We show that pre-trained representations eliminate the needs of many heavily engineered task-specific architectures*". Among the 11 BERT tasks that have reached a new level, most of them only add a linear layer as the output based on the fine-tuning of the pre-trained representation. In the task of sequence labeling (e.g., NER), even the dependencies of the sequence output are ignored, and still outperform by comparing with the previous works.
2. Scale matters
   One of the core claims is that the deep bidirectionality of BERT, which is enabled by masked LM pre-training, is the single most important improvement of BERT compared to previous works. MASK technique in language models may not be new, but the author of BERT who has verified

its powerful representation learning ability on the basis of such a large-scale data + model + computing power.

3. Pre-training is important
   This is a great work to demonstrate that scaling to extreme model sizes also leads to large improvements on very small-scale tasks, provided that the model has been sufficiently pre-trained.

References

1. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
2. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. arXiv preprint arXiv: 1706.03762, 2017.
3. Keiron O'Shea, Ryan Nash. An Introduction to Convolutional Neural Networks. arXiv:1511.08458, 2015
4. Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv preprint arXiv:1609.08144, 2016.