CS224n Project Proposal

Luis A. Perez *
Department of Computer Science
Stanford Univeristy
Stanford, CA 94305
`luis0@stanford.edu`

Abstract

We propose working on the default final project, in the PCE-division, in an attempt to improve currint SOTA results in the SQuAD NLP task. In particular, we propose experimenting with several extensions of BERT [1] (such as a few of those proposed in the handout), performing thorough error analysis, and finally creating an ensemble from these extensions.

1 Research Paper Summary

We summarize the paper by Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova from the Google AI Language Lab titled "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" [1]. The paper is chosen since it is currently the foundation for the vast majority of high-performing models in the SQuAD 2.0 leaderboard. See Table 1 for a list of the top-ten models, and note that all of these models make use of the BERT architecture + weights as a foundation for their work.

1.1 Background and Context

The authors of this paper set-out on the task of creating a pre-trained, non-task-specific language model with the end-goal of using this model to further the SOTA results in task-specific metrics, such as GLUE and SQuAD v1.1. Their work can be considered an extension of ELMO [2], which introduced *deep contextualized* word representations, rather than static, fixed-size word representations used previously, such as those in word2vec. These representations can better model complex characteristics of word use and how these uses vary across lingustic contexts. BERT follows the same core idea as ELMO – extending the embeddings for a word be using the context in which it appears to modify them. The major contribution of the BERT [1] paper and its authors is to take this same idea, but rather than using traditional encoder-decoder architectures based on RNNs, they instead make use of the Transformer architecture.

With these pre-trained language representations, task-specific models can be utilized downstream through two main strategies (1) feature-based and (2) fine-tuning. The former essentially includes the pre-trained language representations along with additional features, to train a new model, while the latter simply fine-tunes the pre-trained parameters (without introducing too many new parameters).

Transformer had already demonstrated their utility and power in task-specific contexts, such as SQuAD and others. As such, using them for the task of creating context-aware word-embeddings was not far-fatched.

*

1.2 Problem and Proposed Solution

The paper [1] is specifically trying to solve the problem of learning and improved language model for use in word-representations. The proposed solution is to use a large corpus of text and a bi-directional transformer-centric architecture which leads to much improved word-embeddings.

The authors address issues with previous models (such as unidirectionality) by proposing two new pre-training objectives:

- The "masked language model".
- The "next sentence prediction".

The major contribution, then, from the BERT paper is (1) the introduction of pre-training objectives independent of final objectives, (2) utilization of bi-directional models to take full advantage of the language model, and (3) a pre-trained model which achieves SOTA results across multiple, distinct NLP tasks after fine-tuning, with a single additional layer. The key achievement is a 4.4% to 6.7% average accuracy improvement over previous SOTA models in the GLUE benchmarks. The full results (for all benchmarks) are reproduced in Table 3 from the paper. The best performing BERT system also outperforms the top leaderboard system by +1.5 F1 in ensembling and +1.3 F1 on a single system. For more details, see Table 3.

1.3 The BERT Model

BERT's model architecture is simply a multi-layer bidirectional Transformer encoder based on the original implementation by Vaswani et al [3] and as detailed in Figure 3. Transformers help reduce the number of operations required to learn long-distance dependencies to a constant number (at the cost of resolution). They make use of stacked self-attention and point-wise, FC layers for the encoder and decoder, as can be seen in Figure 3. A detailed discussion of this architecture (by annotating the paper that introduced it), can be found here, and is also covered in class. As such, we defer any further details.

1.4 The BERT Pre-training Tasks

For all of the tasks described below, a concatenation of the BooksCorpus (800M words) and English Wikipedia (2,500M words) is used (only text, no lists/tables/headers).

1.4.1 Masked Language Model MLM

In this context, the goal for the model is not task-specific (ie, such as SQuAD), but rather, input data has tokens randomly masked (removed), and the goal is to generate a model that, given this incomplete data, can correctly identify the original vocabulary id. One key aspect of this task is that the model has access to both previous words **and** future words. Further, it is different from auto-regressive models in that only the hidden state for the masked-word is used to compute the softmax probabilities over the vocabulary.

To soften the mismatch between pre-training on this task and fine-tuning on future tasks, the [MASK] token is not actually always used. As per the paper, it is used only 80% of the time, while 10% of the time the token is not masked at all, and the remaining times it is replaced by a random word.

1.4.2 Next Sentence Prediction

Interestingly enough, this task is utilized as an attempt to improve the performance of BERT on future QA and NLI tasks. In fact, this contradicts the fact that BERT is simply a language model without specific characteristics to the tasks it eventually achieves SOTA results in.

The task consists of pre-training a binarized *next sentence prediction* task on a large corpus. This task also helps explain the input representation chosen for the BERT model, which consists of 3 distinct pieces: (1) token embeddings, from WordPiece embeddings, (2) Segment Embeddings (only two segments are used, A and B, which assists on using BERT for the next sentence prediction task), and (3) position embeddings (BERT only supports sequences up to 512 in length).

1.5 BERT Fine-Tuning

A key insight of the paper is introducting the ability to use a single, pre-trainined system which can then be fine-tuned for specific tasks. Here, we summarlize the modifcations which were made in order to handle each of the NLP tasks which the system tackled. In broad-strokes, there are essentially two types of tasks which the system handles: (1) sequence classification and (2) question-answer pairs.

1.5.1 Sequence Classification

These are mostly from the GLUE benchmark, and consists of tasks where a sequence is fed into the model and a classification needs to occur (across C classes). The amazing thing about BERT is that it achieves state of the art results in these tasks only through fine-tuning. Specifically, the model takes as input a special [CLS] token as the first token (this was also done during pre-training), and the hidden state output from the transformer model (BERT) corresponding to this input token, is take as the hidden state input for a linear classifier.

That is to say, the fine-tuning consists simply of learning a $W \in \mathbb{R}^{K \times H}$, where K is the number of labels and H is the dimension of the hidden state, C, output from the transformer model (BERT). These new parameters (as well as the BERT parameters) can then be trained on this task.

1.5.2 Question and Answers

The BERT paper focuses on the SQuAD task, a collection of 100k corwdsourced question/answer pairs. For this system, we take advantage of the input representation used by BERT. The question tokens correspond to the sentence 1 encoding, and the paragraph to the sentence 2 encoding. Then each output transform state, T_i is dot-produced with two new parameter vectors, S and E for start and end. The a straightforward softmax is taken from this dot-product for all i, to determine the start and end of the corresponding answer in the text. Similar modifications are used for other tasks, which are detailed further in the paper.

1.6 Concluding Remarks

Overall, the paper presents an attention-based mechanism for pre-training on a large-corpus of data. Furthermore, due to the architectural choices and the input representations, this pre-trained model called BERT can be fine-tuned, with the addition of a few final layers, to achieve SOTA performance across many tasks.

2 Project Description

2.1 Goals

The primary goal of the project is to explore alternative, more complex architectures for fine-tuning the BERT model, and explore whether or note these modifications provide meaninful improvements.

In particular, we find the approach in [1] for classification tasks to be severely limiting, as it attempts to condence the entirety of the contextual information into a single hidden-state. We propose exploring a few natural extension to this approach to determine if it's possible to achieve better results with a more complex fine-tuning architecture. We believe doing this will provide valuable theoretical insight as to the power of BERT, as well as provice practical direction and guidance for future systems seeking to perform task-specific objectives through fine-tuning.

- Do we need fine-tuning layer? Here, we propose an extreme, where we actually seek to analyze and understand the results of using BERT with no additional fine-tuning layers.
- What if it were deeper? Here, we propose an extension, where additional intermediate convolutional layers are used rather than a single FC layer on the classification hidden state. We suspect that this will lead to some improved performance on the fine-tuned tasks, since the model will have increased flexibility and capacity.
- Can we make use of the other transformer hidden states? Our hypothesis is that currently fine-tuned BERT models are bottlenecked by the single hidden-state. Instead, we propose making use of all of the transformer hidden states for classification

- Both of the above approaches are also equally applicable to the other metrics used in the BERT paper.

Our primary goal is to explore these alternatives, analyze the results, and make an educated attempt at improving current SOTA systems.

As an extension goal, we'd like to approach the task of answer generation for the SQuAD datasets. This task is still not very well-defined, but the overall goal would be to, rather than pin-point to the part of the text where the answer to the question exists, instead create a generative model that can propose multiple answer and generate the output sequence itself. This can be an extension of the BERT model where the transformer states are attended to all out once and fed as input.

The metric would be to use BLEU against the correctly extracted sentence text. In a failure mode, the model would actually learn simply to find the answer text and generate it. However, we'd like to avoid this happening.

2.2 NLP Tasks

The NLP task is the same as those addressed in [1], with an emphasis on SQuAD V2.0 as this is the default project. We'd like to achieve good performance on this new task, by making use of the above proposed modifications.

2.3 Data

The dataset is already pre-specified by the default final project handout. However, for our own edification and for our stretch goals, we plan to make use of the GLUE datasets and mini-tasks to more accurately evaluate the validity of our results across multiple NLP tasks.

2.4 Varients

As discussed in the goal section, we propose making a few modifications to the fine-tuning aspect of BERT in order to make better use of the information available.

In particular, for classification tasks, we suspect that the single hidden state used is actually a bottleneck to be overcome. We propose overcoming this bottleneck through multiple possible mechanisms such as (1) an attention later to combine all of the final BERT outputs into a classification hidden state or (2) a RNN which takes the hidden outputs as well as the classificatio hidden state to generate the final classification scores (3) more deep/complex architectures, such as a residual block using 1D convolutions.

Furthermore, some of the above described varients can also be utilized to hopefully improve on the SQuAD v2.0 task.

2.5 Baselines

The baseline model for SQuAD V2.0 will be existing SOTA systems. The baseline model for the GLUE metrics will be the original results reported by the BERT [1] paper, as we will be doing an extension attempting to understanding the utility and extensiveness of fine-tuning.

2.6 Result Evaluation

As discussed above, SQuAD v2.0 will be used as the default, automatic evaluation metric for all of the modifications proposed above (for the SQuAD task). For the classification tasks, we will use the GLUE to evaluate our model's performance.

Model/Method	Category	EM	F1
BERT + MMFT+ ADA	BERT Ensemble	85.082	87.615
BERT + Synthetic Self-Training	BERT Ensemble	84.292	86.967
BERT finetune baseline	BERT Ensemble	83.536	86.096
Lunet + Verifier + BERT	BERT Ensemble	83.469	86.043
PAML + BERT	BERT Ensemble	83.467	86.035
Lunet + Verifier + BERT	BERT Extension	82.995	86.035
BERT + MMFT + ADA	BERT Extension	83.040	85.892
BERT + Synthetic Self-Training	BERT Extension	82.975	85.810
PAML + BERT	BERT Extension	82.577	85.810

Table 1: Table with Top 10 models/methods on SQuAD 2.0 as of 2/12/2019. See SQuAD Leaderboard for more details.

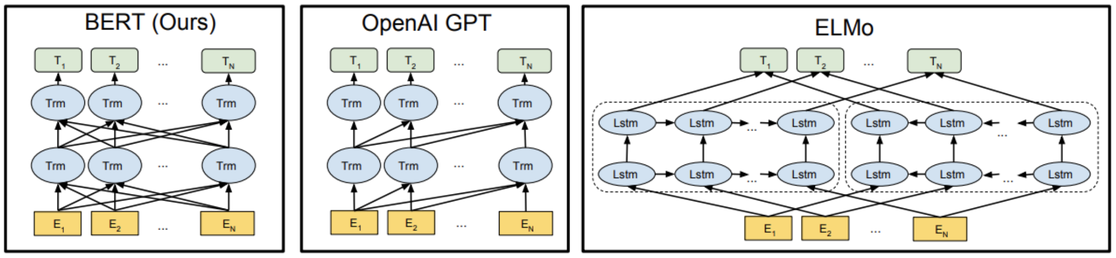


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

3 Appendix

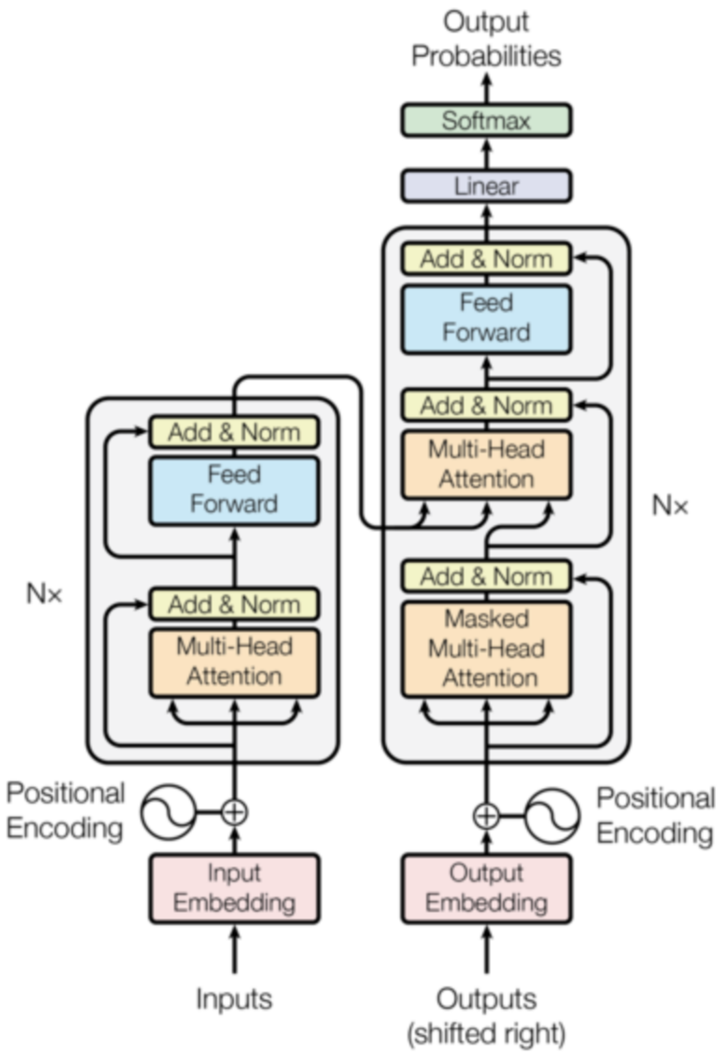
References

References

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[2] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.



System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
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$_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT$_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); BERT$_{\text{BASE}}$ = (L=12, H=768, A=12); BERT$_{\text{LARGE}}$ = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from `https://gluebenchmark.com/leaderboard` and `https://blog.openai.com/language-unsupervised/`.

System	Dev		Test	
	EM	F1	EM	F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
BERT$_{\text{BASE}}$ (Single)	80.8	88.5	-	-
BERT$_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
BERT$_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
BERT$_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
BERT$_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.