CS224n Project Proposal

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

Abstract

In this paper, we explore several extensions of BERT for the Stanford Question and Answer v2.0 task in NLP, and summarize our analyzes of these results. In particular, we explore the benefits of using a fined-tuned BERT model for word-embdeddings underlying different, more complex question-and-answer architectures. We begin by setting a baseline system where BERT is used with a single additional layer, fine-tuned on the SQuAD dataset as presented in the original paper [1]. We demonstrate that such a system achieves great performance, as measures by F1, EM, and Answer-vs-No-Answer (AvNA) metrics. We continue by exploring different more traditional QA architectures, such as a BiDAF and a few other customized models. Additionally, we experiment with replacing the BERT embeddings with GPT-2 [3]. Finally, we conclude by performing an analysis of our results and experiments, and provide guidence of the future direction of this work.

1 Approach

In 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], BERT, an LM, is shown to have great promise in a wide variety of tasks. In particular, it perform extremenly well on the SQuAD v2.0 challenge. 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.

BERT is a pre-trained, non-task-specific language model which has been used to further the SOTA results in task-specific metrics, such as GLUE and SQuAD v1.1. Other similar language models exists, such as ELMO [2] and GPT-2 [3], each making use of a large corpus of language traning data in order to generate *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, specifically, follows the same core idea as ELMO – extending the embeddings for a word be using the context in which it appears to modify them. All of these large language models vary mostly in the architecture of the neural networks.

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).

*

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

1.1 Detailed Architecture

The BERT paper [1] specifically tries to solve the problem of learning an improved language model for use in word-representations. In this paper, we take a similar approach and extend it. We use the pre-trained version of the BERT weights, which have been ported to PyTorch [2]. We provide a brief overview of the architecture and the pre-training tasks.

The BERT embeddings are context-aware and are contructed from two tasks.

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

With these 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 4 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 4. In this paper, we use these original values as our baselines.

BERT's model architecture is simply a multi-layer bidirectional Transformer encoder based on the original implementation by Vaswani et al [4] and as detailed in Figure 4. 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 4. A detailed discussion of this architecture (by annotating the paper that introduced it), can be found here.

This model is then trained on the BooksCorpus (800M words) and English Wikipedia (2,500M words) (only text, no lists/tables/headers). The model we use is trained on the MLM task. In this context, the 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. We defer further details in this task to [1], but suffice it to say that the learned, context-aware embeddings are quite effective at capturing the most important properties of language.

1.1.1 Our Modifications

All of our code for the experiments is publically available here. The paper explores two main objectives, all of which rely on pre-trained contextualized embeddings:

1. Network architecture comparison for fine-tuning [single-layer model, BiDAF, QANet] of BERT word-embeddings.
2. Contextualized word-embedding effect [ELMo vs BERT vs GPT-1 vs GPT-2] on most promising architecture.
3. Hyperparameter tuning during training using Bayesian Learning (see Spearmint Package) on BERT + BiDAF

1.1.2 Network Architecture Comparisons

With the network architecture comparison, we see to understand the effect of further network architectures on-top of pre-trained contextual word-embeddings. In particular, this paper answer the following questions.

- 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.

[2] PyTorch Open-Source Port of Multiple Models

- 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

The key contribution in our paper is mostly experimental. We begin by using the BERT word-embeddings with full-network fine-tunining to perform question-and-answering. Our baseline model consists of exactly this model. We have already trained this model [3].

1.1.3 Linear Model

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.

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.

1.1.4 BiDAF Model

For this system, we compare the performance of BERT embeddings using a more complex model on top. In particular, we make use of the BiDAF model which makes use of highway layers as well as other layers in order to improve in the context of question and answering.

1.1.5 Custom Model

For the final comparison, we make use of a model which consists of a simple additional attention mechanism on the embeddings produced by BERT.

2 Experiments

Current experimental results have been submitted to the PCE-division of the DEV board and consists only of having full-trained a baseline model using the BERT context embeddings. Training was done for 2 epochs, and achieved decent results (though not as claimed in the original paper). This is likely due to fine-tuning for only 2 epochs.

2.1 Concrete Results and Other Parameters

For the training of the base-line model we used the same parameters as specified in the original paper [1]. In particular, we trainined our models for a total of 2 epochs (for fine-tuning).

1. Only fine-tuning the final layers. This took on the order of 2 hours on 2 GPUs. This achieved EM: 50.921 (+50.921) and F1: 50.996 (+50.996), which is actually worse than the provided baselines. We used learning reate of 1e-5 with a fixed schedule.
2. Fine-tuning the entire BERT model in addition to the final layer. This took on the order of 10 hours on 2 GPUs and achieved our best results so far, with a score of EM: 72.688 (+21.767) and F1: 76.071 (+25.075).

We performed additional experiments, but none which have demonstrated any promising approaches. We're currently still running experiments on the BiDAF model with BERT embeddings.

[3] code

2.2 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).

3 Future Work

This section is still to be completed. In particular, we're interested in training the same models above but using different embeddings. Specifically, it will be useful to make use of the GPT-2 pre-trained models recently released by OpenAI. Furthermore, we'd like to make use of Bayesian Hyper Parameter optimization *during* training – in particular, in regards to the learning rate schedule. Specifically, we'd like to have the learning rate schedule dynamic, but not-prespecified, but rather learned based on each batch of data.

3.1 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.

4 Appendix

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.

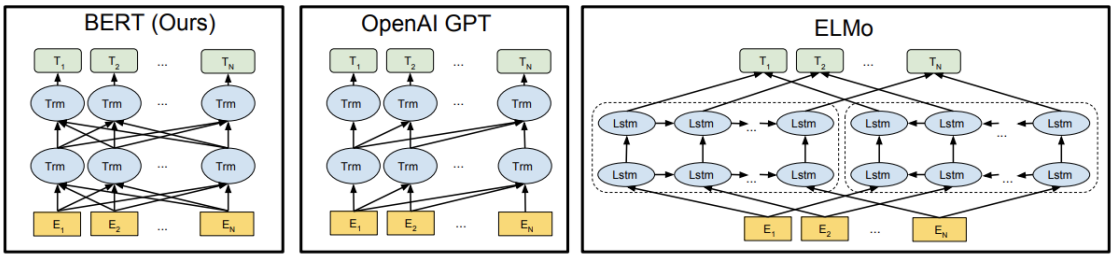
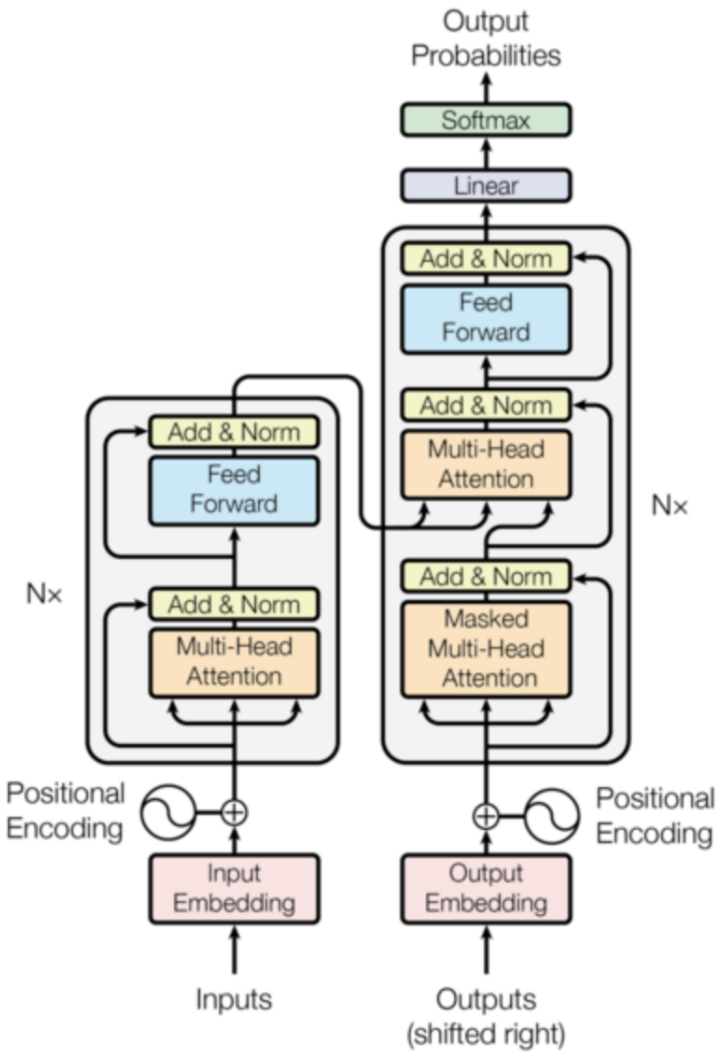


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations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



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. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[4] 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) 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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**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); $\text{BERT}_{\text{BASE}}$ = (L=12, H=768, A=12); $\text{BERT}_{\text{LARGE}}$ = (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				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (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.