Question Representation and Mapping In Transformers (QeRMIT)

Mark Chang **Di Ai**

Stanford University
CS224N - Natural Language Processing with Deep Learning
{mrkchang,diai}@stanford.edu

Abstract

We introduce the Question Representation and Mapping In Transformers (QeRMIT) model that builds on the previous works of Bidirectional Encoder Representations from Transformers (BERT). We illustrate that QeRMIT is tailored towards improving performance on specific tasks such as reading comprehension, where the meaning of the query can be captured without a sequence. In this paper, we focus specifically on the popular Stanford Question Answering Dataset (SQuAD) v2.0 to objectively demonstrate the significance of better capturing the meaning of a question and mapping it onto the final BERT Pre-trained Contexual Embeddings (PCE) layer of the paragraph. We discuss the results and challenges in implementing QeRMIT, and successful approaches that increased EM and F1 scores on SQuAD v2.0.

1 Introduction

Prior to the advent of PCE in 2017, various Natural Language Processing (NLP) tasks such as text classification and Named Entity Recognition (NER) required task-specific, inflexible models. On the other hand, PCEs, most famously Embeddings from Language Model (ELMo) and BERT, demonstrated the ability of their PCEs to generalize for a multitude of NLP tasks and outperform non-PCE models in every category.

Our paper focuses on improving BERT. Specifically, we hypothesize that utilization of these PCEs can be further enhanced by better capturing and projecting meaning from segment A (e.g. question) to segment B (e.g. paragraph). For the scope of this paper, we only focus on SQuAD v2.0. Compared to SQuAD v1.0, the new version now contains question-answer pairs that contain no answers, enhancing the required level of understanding of both the given context and question.

2 Related Work

We based our work on the BERT model. BERT demonstrated that it was possible to train a general-purpose transformer language model using unsupervised data. With minimal modifications to the output of the pretrained model, BERT achieved state of the art performance on 11 separate NLP tasks. Notably, as of March 2019, 10 out of the top 10 teams on the SQuAD 2.0 challenge leaderboard incorporated BERT in some fashion. Further information on BERT can be found in its paper [7].

Prior to PCEs, the Bi-Directional Attention Flow (BiDAF) model outperformed all existing models on SQuAD v1.0. It utilized a concatenation of GLOVE word vectors and character-level embeddings that are passed through bidirectional LSTMs and have attention layers from both the question-to-context and context-to-question directions. We found this work to be highly relevant in our hypothesis that

the question itself could be better represented and mapped. We also saw opportunity to replace the GLOVE word embeddings with BERT PCE in our study. Further information on BiDAF can be found in its paper [4].

3 Approach

We started our implementation of BERT large uncased using Huggingface's open-source git repository [8] and obtained a baseline F1 score of 80.407 and EM of 77.526. The class non-PCE baseline is the BiDAF model (non-PCE), which achieved a F1 score of 59.291 and EM of 55.991 [4].

3.1 BERT (baseline)

BERT can be characterized with two segments in its training: pre-training and fine-tuning. During pre-training, the major difference between BERT and other PCE models is that BERT utilizes a deep bidirectional model, whereas others were unidirectional (OpenAI's GPT) or at most a shallow concatenation of a left-to-right and right-to-left language model (ELMo). BERT achieves this by randomly masking prediction words and also having a sentence prediction task during its training. Note that all these PCE models utilize deep transformers, which allows the model to learn much deeper relationships between a word and its context regardless of word distance and at a highly optimized computational speed.

In its fine-tuning layer for SQuAD, BERT has a single fully-connected linear layer that outputs probability of each paragraph token being the <start> or <end> token of the answer to the question. This is especially important for a question answering task like SQuaD, which relies heavily on bidirectional context to select the correct span. Constraints are applied to ensure that the <end> token occurs after the <start> token and that the answer length does not exceed a pre-defined maximum.

Given its simplicity in its fine-tuning to a specific task, we saw opportunity to improve on the BERT model while taking advantage of its highly versatile PCEs.

3.2 Question Representation and Mapping In Transformers (QeRMIT)

We hypothesize that self-attention alone is not enough to fully encode the meaning of the question and have it be appropriately mapped in the final predictions. As described in the preceding section, during fine-tuning BERT utilizes only the paragraph tokens in predicting the answer spans. While the deep transformer layers do allow BERT to implicitly encode information about the question onto the context paragraph, we believe a more explicit mapping will improve performance, especially in SQuAD 2.0 where some of the no answer questions are quite subtle, as shown later in the Analysis section. Therefore, we propose three novel methods of representing the question and a skip-connection method for projecting the question into the context. We take advantage of how the baseline BERT uses only a single fully-connected layer during fine-tuning to predict <start> and <end> tokens of the answer without much emphasis to the question beyond the basic transformer encoder model.

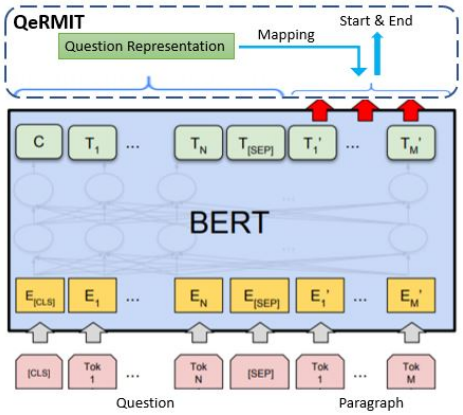
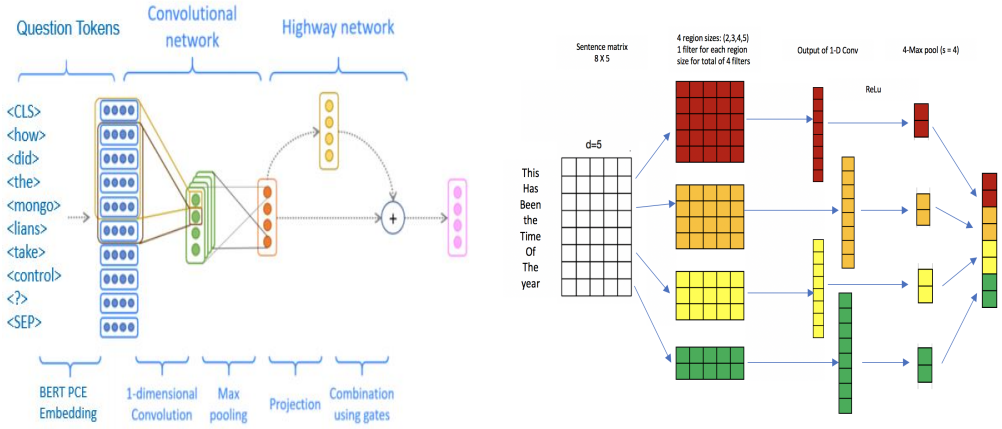


Figure 1: QeRMIT w. BERT

Dense Convolutional Question Representation We propose that the meaning of a question can optimally be densely packed using a convolutional encoder similar to Kim et al.'s work in Character-Aware Neural Language Models [9]. In order to prepare the tensor for convolution, we first create a masking layer to mask out the output of the pretrained BERT model for the paragraph tokens while keeping both the <CLS> and first <SEP> tokens. We then truncate the tensor to the max question length to optimize for speed. See Figure 2a.

Idea of the convolution is simple. For each batch of pretrained question vectors, we perform a 1-D convolution across the questions with a kernel size of 5. We use a padding size of 2 to account for questions that are shorter than 5 words, including the <CLS> and first <SEP> tokens. Then, we pass the output through a highway network. The resulting matrix is a densely packed question vector of the same hidden layer size as the BERT pretrained context layer. We later project this question vector to the paragraph token embeddings to emphasize the representation of the question in the model predictions.

We also implemented a more complex CNN. QeRMIT - CNN v2 has 4 kernel sizes of 2, 3, 4, 5 with 1 filter per region. The idea here is to use different kernel sizes to capture meaning of shorter vs. longer phrases in the question. We pass the output of each filter through ReLu and a 4-Max pool with a stride length of 4 before concatenation to get a densely packed question vector of the same hidden layer size as the BERT pretrained context layer. See Figure 2b.



(a) QeRMIT - CNN v1

(b) QeRMIT - CNN v2

Figure 2: QerMit - CNN Question Representations

<CLS> Token Question Representation As an alternative to the dense convolutional question representation, we take into consideration that BERT is trained to capture the holistic meaning of the question in the <CLS> token vector. The <CLS> token is found at the beginning of every sequence for BERT. This pretrained <CLS> contextual vector can be used to map the question onto the BERT fine-tuning layer. The benefit here is that there are no additional layers that need be trained in addition to the existing BERT baseline model for SQuAD, thereby being a computationally cheaper representation.

MaxPool Question Representation As a compromise of both the more complex dense convolutional question representation and the simpler <CLS> Token Question Representation, a MaxPool Question Representation can utilized. This representation takes advantage of how max pool layers have traditionally been able to capture the biggest signal across different dimensions and output a representation that captures overall meaning. The large embedding size of BERT also aids this representation.

Question Mapping through Skip Connection Once the question representation is prepared, the vector mapping to paragraph is done in two steps. First, we extract similarity between the question vector and each paragraph token vector from pretrained BERT by using element-wise multiplication. Here, the paragraph dimensions that are dissimilar to the question will be attenuated. Second, we

sum this output with the original pretrained context vector using a skip connection. The goal here is increase the representation of paragraph vector. We believe that the question is an essential input to determining the correct answer span and should be able to improve the performance of the baseline BERT model.

Question Mapping through Concatenation with Additional Layers To explore other avenues of question mapping into the context, we explored using concatenation of the question representation with the context vectors. Since this alone would simply have an effect of splitting up the weight matrix into two sums, we add 3 additional fully connected linear layers. We also created another mapping structure of having a 1-D convolution across the the example span.

3.3 Using BERT as Feature Vectors & Incorporating Char Embeddings

BiDAF emphasizes the attention flow between the question and the context and therefore was of interest to our hypothesis about having better question representation and mapping. This view aligns with what we believe will increase the performance of NLP models in Question Answering (QA) tasks. Authors of BERT note that BERT PCEs could be used like wordvec to characterize contextual meaning of a word. Therefore, as a side investigation, we incorporated BERT PCEs into the BiDAF model to assess its effect on performance on SQuAD v2.0.

Furthermore, we incorporated character-level embeddings in an attempt to further augment BiDAF with BERT PCE's performance. BERT uses a WordPiece tokenizer to pre-process the input text prior to training, as described in [5]. While using sub-word units provides an advantage for unknown and rare words by breaking up words into sub-components, it is not as effective as a Character-based model that has even greater granularity for words that are structurally similar. In particular, character-based models will be able to handle words with apostrophes, quotation marks, and other punctuation with greater proficiency. We expect that the addition of a character-based model will be able interpret phrases like "Edgar's sister Margaret" and "President George Beadle's office", both taken from the SQUAD dataset, better than a Word Piece model.

3.4 Data Augmentation

More quality training data is expected to regularize a neural net from overfitting and improve the overall performance. Indeed, top BERT models in the SQuAD v2.0 leaderboard mention use of synthetic self-training, likely to overcome the "un-scalability" of hand-annotated training data. Below we present three ideas. TriviaQA has been implemented into our model, and the other two are research ideas outside the scope of this paper.

Incorporating TriviaQA We can take advantage of other annotated training examples created by another QA competition called TriviaQA [3]. We feed this training data to our best models to further improve performance.

English Dictionary Embedding Augmentation Another area of potential research involves using the BERT embeddings and augmenting that with a neural network trained on the dictionary definitions of each word. We can train a character-based LSTM encoder and decoder to produce the target dictionary word based on the source dictionary definitions. After the model is trained, we can augment the BERT embeddings with the character embeddings for the final fine-tuning layers.

Synthetic Data We propose a novel method of generating synthetic data for a wider variety of question-answer pairs. We take the pre-trained BERT model and fine-tune on Paragraph only instead of Question-Paragraph pairs during fine-tuning. This allows us to create a model that can generate Answer spans solely from Paragraphs. Then, we train a seq2seq model to take in Paragraphs and Answers as input and output Questions. We can use the model to generate "new" Answers and Questions pairs.

Furthermore, we propose another simple method of augmentation by switching out questions between different contexts. By doing so, we can with high confidence assume that these switched question-context pairs now have <no answer>. To create even stronger negative examples, we can use part-of-speech (POS) tagging to replace words with another, different word of the same POS from the same paragraph, which has the advantage of creating realistic questions that sound like they have

answers. The analysis section has an example of a question that due to the replacement of a single noun rendered the question unanswerable. By accomplishing this, we can generate scalable synthetic training data.

4 Experiments

4.1 Data

The baseline pretrained BERT model used concatenation of BooksCorpus (800M words) and English Wikipedia (2,500M words). For our fine-tuned models built on top of BERT's precontextual word vectors, we used data provided by Stanford's CS224N default project [1]. This dataset includes train (130k examples), development (6k), and test data (6k) for SQuAD v2.0.

4.2 Evaluation method

We use the Exact Match (EM) and F1 score as a performance evaluation metric to SQuAD v2.0 as outlined in [1].

4.3 Experimental details

In our re-implementation of BERT, we modified hyperparameters to accommodate for the lack of computational resources as compared to the TPU hardware that the original authors of BERT used. The most powerful VM we had access to was the NV24 instance on Microsoft Azure, which contains 4x M60 GPUs.

After freezing the pretrained BERT layers, we were able to use a batch size of 1024, and trained for 2 epochs. To gain additional efficiency, we used FP16 instead of FP32, which uses half as many bits to represent floating point numbers, and a loss scaling factor of 128, which was shown to result in a similar loss training curve as FP32 in [7]. The loss scaling factor shifts the range of FP16 numbers downward to account for the loss of very small numbers, which otherwise would be represented as 0 and inhibit gradient calculation. Using FP16 instead of FP32 also allowed us to increase batch size due to the reduced memory footprint of FP16 compared to FP32.

Freezing the pretrained layers led to poor performance since PCEs have yet learned to perform well on question-answering systems. Fine-tuning the original model with all the layers took around 24 hours on a NV6 instance with a batch size of 4 and a gradient accumulation step of 4.

We chose to use BERT-large because of the proven performance gains from running a larger and more powerful model. Also, we chose to use the Uncased version because both Cased and Uncased had the same number of model parameters and we did not believe that the gain in word granularity was offset by the effective loss in parameters (i.e. you would need to increase the number of parameters to get an equally powerful model for cased vs. uncased due to the increased vocabulary size).

Hyperparameter Search We used the recommended hyperparameter settings as outlined in the BERT paper [7] for each model to ensure results are comparable across different models. With limited resources, we did not conduct an exhaustive hyperparameter search. We purposefully diverted our limited budget towards training a variety of models. If we had access to more powerful computing hardware, we recommend performing a thorough Bayesian Optimization parameter search.

In all cases, we noticed that tuning the no answer threshold using the provided *evaluate.py* script improved F1 scores by ~1.2. However, it was interesting to note that increasing the max sequence length, max query length, and max answer length actually resulted in a decrease in F1 score. We hypothesize that this is due to the low prevalence of questions, contexts, and answers that exceed the default settings (max seq length = 384, max query len = 64, max answer len = 30). In fact, we found that out of 130,571 training examples, there were 1,625 (1.2%) instances where the sequence exceeded the default max sequence length of 384, 0 instances where the query exceeded the default max query length of 64, and 152 (0.1%) instances where the answer exceeded the default max answer length of 30. These examples are too rare and unique for the model to be able to generalize or learn anything substantive from.

Table 1: Hyperparameter Search

Name	Parameters	NoAns Tuning	EM	F1
BERT Large (baseline)	Default	No	77.312	80.243
BERT Large (baseline)	Default	Yes	79.138	81.599
BERT Large Max Seq Length	Max Seq Length = 512	No	77.361	80.334
BERT Large Max Seq Length	Max Seq Length = 512	Yes	79.121	81.585
BERT Large Max	Max Seq = 512 Max Query, Ans = 256	No	77.279	80.275
BERT Large Max	Max Seq = 512 Max Query, Ans = 256	Yes	79.056	81.551

4.4 Results

Our final test leaderboard submission had a EM score of 72.730 and a F1 score of 76.294. Our final dev leaderboard submission had a EM score of 79.138 and a F1 score of 81.599. The large drop of approximately 5 points from the dev to test set can be attributed to differences in data distribution. Instructors of the course acknowledged that some of the examples in the test set were hand-selected and may be harder in difficulty, resulting in reduced performance for the models.

We had initially expected the EM and F1 scores to go up by training on top of the fine-tuned baseline model while freezing the layers that had the embedding. However, the results show worse performance for all three QeRMIT models. In hindsight, this worsening in performance is expected because the baseline fine-tuning is done to aggregate the meaning of the question and paragraph to the contextual vectors of the paragraph tokens. This means that the vectors representing the questions could lose meaning over time as they are being fine-tuned. Therefore, when our models are using these question vectors to represent the question, these vectors do not necessarily have a comprehensive and dense representation of the question.

As we retrained our models without freezing the embedding layers, we noticed that performance still did not increase. After careful consideration, we came to the conclusion that simply adding linear layers was too simple and did not actually model interactions between word tokens. The linear layer simply resulted in taking a weighted average of the question and context hidden layers. As shown in the training loss over time (Figure 3), the model was unable to make any progress in learning.

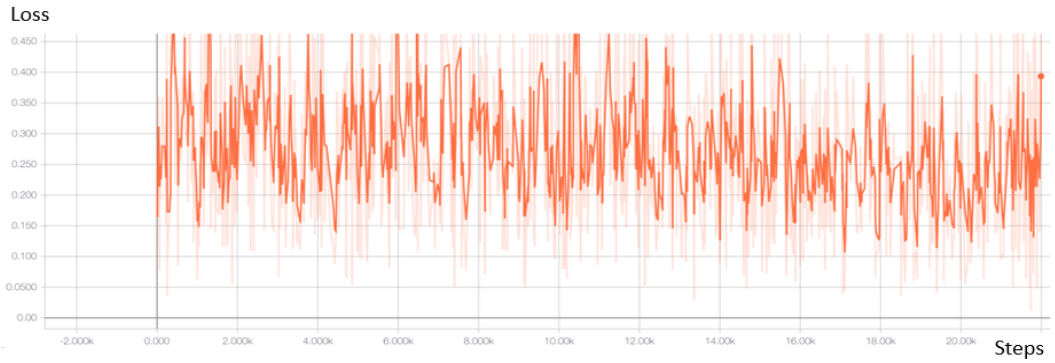


Figure 3: QeRMIT - CLS (+BERT base) Training Loss

Based on how adding QeRMIT dropped performance of the baseline BERT model, we found BERT's model inflexible to use. The CNN representation of the question tokens is a proven method for character embeddings, yet its mapping into the context vectors through methods of skip connection and concatenation have all lowered EM and F1 scores. We have also tried adding additional layers to the final fine-tuning layer, including layers of token span-wise CNN and another model with 3 additional fully-connected linear layers.

Table 2: Model results

Name	Layers trained	EM	F1
BiDAF (baseline)	All layers	55.991	59.291
BERT	All layers	77.526	80.407
BERT - frozen embeddings	qa_output (only)	18.921	21.113
QeRMIT - CNN v1 (+baseline)	qa_output & CNN	77.493	80.359
QeRMIT - MaxPool (+baseline)	qa_output (only)	76.505	79.601
QeRMIT - CLS (+baseline)	qa_output (only)	77.196	80.102
QeRMIT - CLS (+BERT base)	All layers	64.988	68.148
QeRMIT - CNN v2 (+baseline)	All layers	68.970	74.018
QeRMIT - CNN v2 (+baseline, highway)	All layers	77.772	81.047
BERT - TriviaQA (wiki train)	All layers	76.028	78.545
BERT - TriviaQA (wiki + web train)	All layers	75.337	77.892
BERT - TriviaQA (wiki + web train, dev)	All layers	74.548	77.133
BERT (tuned hyperparameters)	All layers	79.138	81.599

Furthermore, when comparing BiDAF with BERT-large PCEs (1024 hidden dimensions) and additional character-level embeddings (64 hidden dimensions) with BiDAF with only GLOVE word embeddings (300 hidden dimensions), we did not see any improvement to BiDAF's performance based on the loss over time while keeping all other hyperparameters the same (See Figure 4). Worse, the change led to a significantly more computationally taxing model with the parameter size almost 100x what it was before. The training time per epoch on NV24 GPU jumped from 5 min/epoch to 1.5 hr/epoch for 30 epochs.

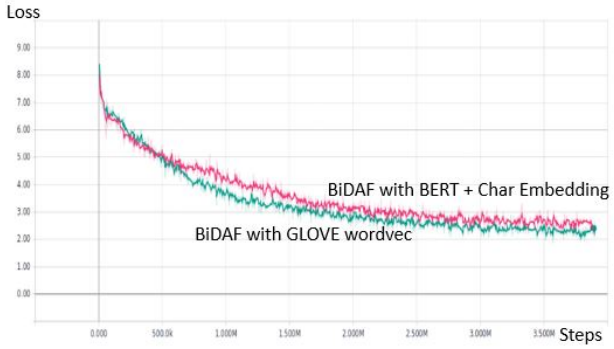


Figure 4: BiDAF (GLOVE wordvec v. BERT + char embed)

It is also interesting to note that we did not see any performance improvement from additional fine-tuning on TriviaQA prior to fine-tuning on SQuAD 2.0. TriviaQA contains context paragraphs drawn from both Wikipedia and web documents. In fact, both F1 and EM scores decreased as we added more TriviaQA data, starting with just the wikipedia training dataset and successively adding the web train, wiki dev, and web dev datasets. It appears that BERT is unable to generalize learnings from TriviaQA to SQuAD, and that fine-tuning on the specific task is extremely important for BERT performance. The TriviaQA dataset also includes more answers per question compared to SQuAD, which only had one answer per question in the training dataset. Our naive approach simply selected the first answer that also appeared in Wikipedia/web documents so that we could calculate the answer span. However, this resulted in some errors in the selected answers, as the below misspelling illustrates. A better approach would have been to take the answer that appeared with the highest frequency.

Q: Who did Claus von Stauffenberg plot to kill?
A: adolf hitle

Our best model through this project was the original BERT-large model with tuned no-answer threshold. This is in fact representative of what is observed in the current SQuAD v2.0 leaderboard.

At the timing of writing, the top models are BERT with synthetic data and modified pretraining methods (See Figure 5 in Appendix).

5 Analysis

When observing QeRMIT's outputs against the provided answers in the dev set, we found that our model still had trouble identifying question-answer sets that had grammar dependencies such as "Edgar's sister Margaret." There's two ways to approach this. The simplest approach is adding more training examples of similar errors for BERT to float the importance of word dependencies to its top layer. A harder approach is to use a diagnostic classifier to understand which layers of BERT excel at which task. A diagnostic classifier is a simple softmax layer on top of each BERT layer that predicts various NLP tasks. Because the diagnostic classifier layer is so simple, it relies heavily on the internal layers below it and is able to determine which layer excels at certain tasks. After identifying which layers are best able to model word dependencies, we can attempt an architectural change to feed forward layers that are more important to the question answering, such as POS tagging and word dependencies.

Furthermore, in many cases, questions with no answers were guessed with an answer. The example below demonstrates that the model still has trouble recognizing obvious dissimilarity between words such as **programmes** and **wreaths of herbs**.

> Context: ...maidens would float their **wreaths of herbs** on the water to predict when they would be married, and to whom...
> Q: What will maidens be able to predict by floating their **programmes** down the Vistula?
> Ground Truth: <no answer>
> Prediction: when they would be married

Nature of BERT is that it is contextual. Having more examples of <no answer> and secondly incorporating word embeddings such as GLOVE wordvec in the original pretraining will help. In addition, creating strong synthetic negative training examples by replacing nouns like "programmes" with other random nouns from the context like "water" will create excellent examples for the model to learn from. The new question in this instance would be: "What will maidens be able to predict by floating their **water** down the Vistula?"

We believe that future work in better characterizing word dependencies such as incorporating constituency parsing would greatly aid in performance. Below is an example where the sentence structure may have confused the model predictor and one where including constituency parsing will help. Notice how the question is asked in an unconventional format.

> Context: The principal role of committees in the Scottish Parliament is to take evidence from witnesses
> Q: Taking evidence from witnesses is one of committees' what?
> Ground Truth: principal role
> Prediction: <no answer>

Finally, given enough computational resources, incorporating character-level embeddings at the initial pretraining of BERT would aid character-level similarities to improve BERT's understanding of words based on word roots and affixes.

6 Conclusion

We propose QeRMIT, a model to better encode the question representation and map it to the context vectors for increased QA task performance. Our experiments with various methods for question representation and projections of those representations to improve performance of the baseline BERT model on SQuAD v2.0 demonstrate the inflexibility of BERT architecture for further layer encoding. In addition, using BERT PCEs as the initial hidden states for BiDAF lowered performance versus the original GLOVE word embeddings.

In fact, our best model through this project was the original BERT-large model with tuned no-answer threshold. This model is currently placed 2nd in the class's PCE-division dev leaderboard. These findings are representative of what we find in the official SQuAD v2.0 leaderboard where the top two

models have the baseline model with synthesized training data and a newly pre-trained BERT model. This inflexibility of the BERT architecture and PCEs seems to be its current weakness.

Future work in understanding how to better ensemble from BERT's architecture and better utilize BERT PCEs is recommended since pretraining is copmutationally expensive. Research in this direction will help make improvements in being able to better utilize the strides made by BERT in a more computationally efficient manner.

Acknowledgments

We want to acknowledge the Stanford CS224N teaching staff for their expertise and guidance in the subject matter and Microsoft Azure for funding this project. We would also like to thank our respective employers (Intuitive Surgical Inc, and Alphabet Inc) for their support in our education.

References

[1] Chute, C. (2019). CS 224N Default Final Project: Question Answering on SQuAD 2.0. *Stanford University.*

[2] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv e-prints.*

[3] Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. (2017). TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. *https://arxiv.org/abs/1705.03551.*

[4] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. (2015). Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603.*

[5] Paulius Micikevicius. (2017). Mixed-Precision Training of Deep Neural Networks. *Nvidia Developer Blog.* https://devblogs.nvidia.com/mixed-precision-training-deep-neural-networks/.

[6] Rico Sennrich, Barry Haddow, and Alexandra Birch. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909.*

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *In the Annual Conference on Neural Information Processing Systems (NIPS).*

[8] Wolf, T. (2019). GitHub repository. https://github.com/huggingface/pytorch-pretrained-BERT. *Huggingface.*

[9] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. (2015). Character-Aware Neural Language Models. *CoRR, abs/1508.06615.*

[10] Joshi, M. (2018). GitHub repository. https://github.com/mandarjoshi90/triviaqa. *Code for the TriviaQA reading comprehension dataset.*

Appendix

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Mar 05, 2019	BERT + N-Gram Masking + Synthetic Self-Training (ensemble) *Google AI Language* https://github.com/google-research/bert	**86.673**	**89.147**
2 Mar 05, 2019	BERT + N-Gram Masking + Synthetic Self-Training (single model) *Google AI Language* https://github.com/google-research/bert	85.150	87.715
3 Jan 15, 2019	BERT + MMFT + ADA (ensemble) *Microsoft Research Asia*	85.082	87.615
4 Jan 10, 2019	BERT + Synthetic Self-Training (ensemble) *Google AI Language* https://github.com/google-research/bert	84.292	86.967

Figure 5: SQuAD v2.0 Leaderboard (March 17th, 2019)