Question Answering on SQuAD with BERT

Zhangning Hu
hzn@stanford.edu

Abstract

In the project, I explore three models for question answering on SQuAD 2.0[10]. The models use BERT[2] as contextual representation of input question-passage pairs, and combine ideas from popular systems used in SQuAD. The best single model gets 76.5 F1, 73.2 EM on the test set; the final ensemble model gets 77.6 F1, 74.8 EM.

1 Introduction

Machine Comprehension is a popular format of Question Answering task. Models are asked to provide answer to a question given a context passage. Stanford NLP group released the Stanford Question Answering Dataset (SQuAD)[11] in 2016, and upgraded it to SQuAD 2.0[10] with unanswerable questions. It largely facilitates the progress in the machine comprehension field and has become one of the most important benchmarks in NLP research. The dataset now have more than 150K questions. Each question is either unanswerable with the given passage, or can be answered with a continuous span of text in the passage.

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a major breakthrough in NLP in 2018. It is a pre-trained deep language representation model. It has pushed many NLP benchmarks to new state-of-arts.

BERT consists of deep Transformer layers, enabling it to capture dependencies across long sequences. Its bidirectional representation is conditioned on both left and right context at the same time in all layers. Those features make it an excellent model to produce contextual representations for texts. Most successful models of SQuAD generate contextual representations that incorporate information from both passage and query.

In this project, I explore replacing the contextual representation structures in BiDAF and SAN with BERT; I also build a model of simpler structure on top of the representation produced by BERT. Those models are described in details in Section 3. The experiment details and results can be found in Section 4.

2 Related Work

Many models for SQuAD first build contextual encodings of passage and query; then fuse them together using attentions to create a passage-query representation; after that, use additional layers to further process the representation and predict answer. BiDAF[12] proposed a bidirectional attention flow that combines both passage-to-query and query-to-passage attentions and produces a query-aware representation of the passage. QANet[16], SAN[8] and Unet[13] have similar bidirectional attentions, and further fuse information with self-attentions.

At the time of writing this report, all of the top 20 models on SQuAD 2.0 leaderboard are either BERT or BERT plus some other model (BERT + X). Since all of BERT + X works are still very new, few papers have published about them. MT-DNN[6] paper uses BERT as the pre-trained contextual representation layer, builds models on top of it and trains them in a multi-task settings. The paper shows it obtained new state-of-art results on eight out of nine GLUE[14] tasks.

Prior to BERT, ELMo[9] is a popular pre-trained deep contextualized word representation. ELMo stands for Embeddings from Language Models. It is LSTM-based. It can be used as word embedding to improve model performance.

3 Approach

3.1 Baseline

The BERT paper describes how to adapt it for SQuAD. The PyTorch implementation of BERT from HuggingFace[1] includes that. It takes the last hidden layer of BERT, feeds that into a dense layer and softmax to generate distributions for start and end positions over the input text sequence. The probabilities corresponding the special token [CLS], which is added to the head of every input text sequence, give the probability of the input passage-query being unanswerable.

3.2 Model Architectures

The models to be discussed are extensions of the baseline model. They use BERT to produce representation of the input query-passage pair, and add additional layers on top of it. Specifically, the first two models are inspired by two popular SQuAD models, BiDAF[12] and SAN[7]. Both of them have layers that create contextual representation of the input query-passage pair. It is a natural extension to try replacing those layers with BERT and see how they work.

The models will start with pre-trained BERT weights, and fine-tune with SQuAD 2.0 training data.

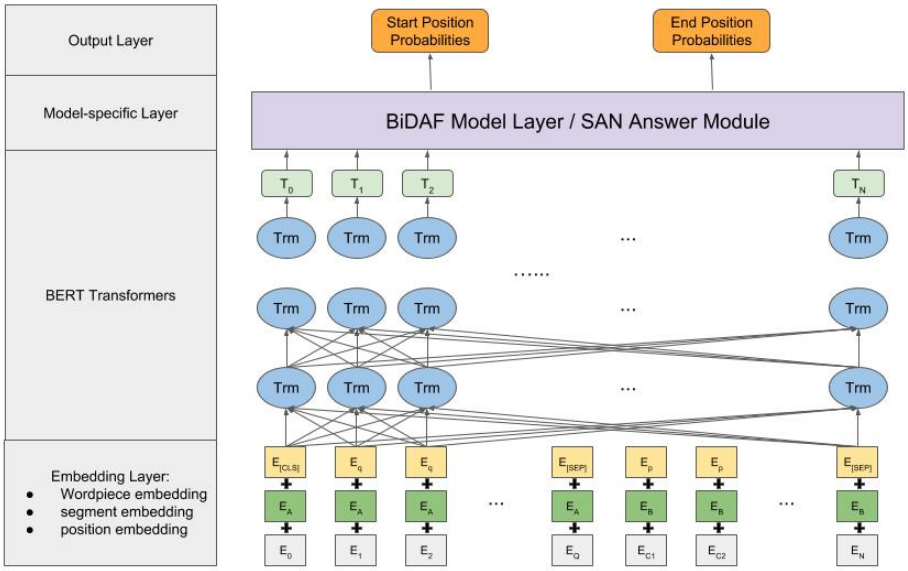

Figure 1: Overall architecture of the models

3.2.1 BERT as Representation Layer

The BERT layer follows the method presented in the BERT paper[2]. The input query and passage pair are tokenized with WordPiece embeddings[15], and then packed into a sequence. A special classification token embedding, denoted as [CLS] is always added to the beginning of the sequence. It is used as an aggregated representation of the entire sequence. In our case, the output corresponding to this token will be used to represent the probability of query being unanswerable. A separation token, denoted as [SEP] is used to separate passage and query. It is added between query and passage, and at the end of passage. A pair of learned sentence segment embeddings are used to further differentiate query and passage. Each query token will be paired with the segment A embedding,

[1] https://github.com/huggingface/pytorch-pretrained-BERT

including [CLS]; each passage token will be paired with the segment B embedding. Lastly, position embeddings are added to each token. The final input to BERT will be the sum of text embeddings, segment embeddings and position embeddings. The input then goes through layers of bidirectional Transformers to produce hidden states that can be used as contextual representation of input query-passage pair. We denote the hidden state size of BERT as *H*, the maximum length of input sequence (including query and passage tokens as well as [CLS] and [SEP]) as *N*. Let $\boldsymbol{G}$ denotes the final layer hidden states output by BERT. It has size $\boldsymbol{G} \in \mathbb{R}^{N \times H}$. The following models add additional layers on top of BERT. Those layers will take $\boldsymbol{G}$ as input.

3.2.2 BERT + BiDAF

The first model replaces BiDAF's embedding layers and attention flow layer with BERT, i.e. use BiDAF's modeling layer and output layer on top of BERT.

Modeling Layer. The input to the modeling layer is the final hidden states of BERT. They are passed to two layers of bidirectional LSTM with hidden size *h*. Thus we obtain output $\boldsymbol{M} \in \mathbb{R}^{N \times 2h}$. This layer should further capture the interaction of the input query-passage sequence.

Output Layer. The output is adapted from CS224n default project handout[1]. It will produce probability distributions of start and end positions for the answer span. To get the distribution of start position, we compute:

$$\boldsymbol{p}_{start} = softmax(\boldsymbol{W}_{start}[\boldsymbol{G}; \boldsymbol{M}]) \tag{1}$$

To get the distribution of end position, the output of modeling layer, $\boldsymbol{M}$, is first fed into a one-layer bidirectional LSTM and obtain $\boldsymbol{M'} \in \mathbb{R}^{N \times 2h}$, then we compute:

$$\boldsymbol{p}_{end} = softmax(\boldsymbol{W}_{end}[\boldsymbol{G}; \boldsymbol{M'}]) \tag{2}$$

Here $\boldsymbol{W}_{start}, \boldsymbol{W}_{end} \in \mathbb{R}^{1 \times (H+2h)}$ are linear layers with learnable parameters.

Loss Function. The loss function is the sum of cross-entropy loss for start and end positions. We weight losses of unanswerable query-passage pairs a little more, i.e. when start and end positions are both 0. This is because to predict a query to be unanswerable, we need both start and end positions point to the first token with high probability. This is harder than predicting normal span. We want the model to put more effort on doing it right. In summary, if the start and end positions are $i, j \in 1, ..., N$:

$$loss = -log\boldsymbol{p}_{start}(i) - log\boldsymbol{p}_{end}(j) - \frac{1}{2}(log\boldsymbol{p}_{start}(i) + log\boldsymbol{p}_{end}(j))_{[i=j=0]} \tag{3}$$

3.2.3 BERT + SAN

SAN stands for stochastic answer network. Its prediction generation module features multi-step reasoning, simulating the process of going through texts for multiple passes and refine predictions. This model replaces the lexicon encoding layer, the contextual layer and the memory generation layer described in the original SAN paper[8] with BERT. This model is also inspired by the MT-DNN paper[6]. It presents using SAN's answer module on top of BERT for natural language inference tasks. Following SAN for SQuAD 2.0 paper[7], our answer module will jointly train an answer span detector and an unanswerable question classifier.

Span Detector. In this section, we denote the final layer hidden states of BERT as $\boldsymbol{G} \in \mathbb{R}^{H \times N}$. Denote input query length as N^q, input passage length as N^p. Denote the part of BERT's final hidden states that correspond to query as $\boldsymbol{G}^q \in \mathbb{R}^{H \times N^q}$; those correspond to passage as $\boldsymbol{G}^p \in \mathbb{R}^{H \times N^p}$. Denote the total step of reasoning as *T*. For each step $t \in \{1, 2, ..., T\text{-}1\}$, the reasoning state is defined as:

$$\boldsymbol{s}_t = GRU(\boldsymbol{s}_{t-1}, \boldsymbol{x}_t) \tag{4}$$

The initial state $\boldsymbol{s}_0$ is defined as:

$$\boldsymbol{a} = softmax(\boldsymbol{w}_0 \boldsymbol{G}^q) \tag{5}$$

$$\boldsymbol{s}_0 = \boldsymbol{G}^q \boldsymbol{a}^T \tag{6}$$

The input to GRU, $\boldsymbol{x}_t$, is computed as:

$$\boldsymbol{b} = softmax(\boldsymbol{s}_{t-1}^T \boldsymbol{W}_1 \boldsymbol{G}^p) \tag{7}$$

$$\boldsymbol{x}_t = \boldsymbol{G}^p \boldsymbol{b}^T \tag{8}$$

The start and end position predictions at step $t \in \{1, 2, ..., T\text{-}1\}$ are:

$$\boldsymbol{p}_t^{start} = softmax(\mathbf{s}_t^T \boldsymbol{W}_2 \boldsymbol{G}) \tag{9}$$

$$\boldsymbol{p}_t^{end} = softmax(\mathbf{s}_t^T \boldsymbol{W}_3 \boldsymbol{G}) \tag{10}$$

$\boldsymbol{w}_0 \in \mathbb{R}^{1\times H}, \boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3 \in \mathbb{R}^{H\times H}$ are all vector/matrix with learnable parameters.

The final prediction is an average of each step:

$$\boldsymbol{p}^{start} = \frac{1}{T}\sum_t \boldsymbol{p}_t^{start} \tag{11}$$

$$\boldsymbol{p}^{end} = \frac{1}{T}\sum_t \boldsymbol{p}_t^{end} \tag{12}$$

At the training time, a random dropout is applied to the step predictions so that the model will not reply on a particularly step. Dropout is applied at each $\boldsymbol{s}_t$ as well.

Unanswerable Classifier. We train a simple unanswerable query classifier jointly with the above span detector. It is computed as follow:

$$\boldsymbol{c} = softmax(\boldsymbol{w}_4 \boldsymbol{G}) \tag{13}$$

$$\boldsymbol{m} = \boldsymbol{G}\boldsymbol{c}^T \tag{14}$$

$$p = sigmoid(\boldsymbol{w}_5 \boldsymbol{m}) \tag{15}$$

$\boldsymbol{w}_4, \boldsymbol{w}_5 \in \mathbb{R}^{1\times H}$ are both vectors with learnable parameters.

Loss Function. We combine the loss of the span detector and the classifier. The loss of the classifier is weighted more for the same reason as the first model. If the label start and end locations are $i, j \in 1, ..., N$, and $y \in \{0, 1\}$ denotes whether the question is unanswerable: $y = 1$ means unanswerable, $y = 0$ means answerable, the loss is computed as:

$$loss = -log\boldsymbol{p}_{start}(i) - log\boldsymbol{p}_{end}(j) - \frac{3}{2}\{ylogp + (1-y)log(1-p)\} \tag{16}$$

3.2.4 BERT + Model3

The last model uses a simpler output layer on top of BERT than the previous two. As it is the third model presented and is light-weight, we will call it Model3. It applies two dense layers to BERT's output, the activation function is gelu[5]. A skip connection is added to the second dense layer:

$$\boldsymbol{U} = glue(\boldsymbol{W}_1 \boldsymbol{G}) \tag{17}$$

$$\boldsymbol{U'} = glue(\boldsymbol{W}_2 \boldsymbol{U}) + \boldsymbol{G} \tag{18}$$

The start and end position probability distributions are computed as:

$$\boldsymbol{p}_{start} = softmax(\boldsymbol{W}_{start} \boldsymbol{U}) \tag{19}$$

$$\boldsymbol{p}_{end} = softmax(\boldsymbol{W}_{end} \boldsymbol{U'}) \tag{20}$$

$\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{H\times H}$, $\boldsymbol{W}_{start}, \boldsymbol{W}_{end} \in \mathbb{R}^{1\times H}$ are linear layers with learnable parameters. Layer normalization is applied after each activation function.

Loss Function. The loss of this model is the same as the BERT+BiDAF model.

3.3 Span Selection

We use a heuristic way to select answer span at the prediction time. First, we get 20 most probable start positions (i.e. positions of highest p_{start}) and 20 most probable end positions. Then for the $20 * 20$ start-end position pairs, we remove invalid ones, i.e., start or end position not within the range of passage; start position after end position; start to end span exceeds the max answer length we set. From the valid pairs, we choose the one with highest $\boldsymbol{p}_{start}(i) * \boldsymbol{p}_{end}(j)$ as the candidate answer. We use $\boldsymbol{p}_{start}(0) * \boldsymbol{p}_{end}(0)$ as the probability of the question being unanswerable. We pre-define a threshold h. If

$$\boldsymbol{p}_{start}(0) * \boldsymbol{p}_{end}(0) - \boldsymbol{p}_{start}(i) * \boldsymbol{p}_{end}(j) > h \tag{21}$$

then we predict this question as unanswerable; otherwise, we predict the candidate answer.

Table 1: Model Performance on SQuAD 2.0 Datasets

	Dev (EM/F1)	**Test (EM/F1)**
BiDAF Baseline (ON official test set)	61.2 / 64.9	59.2 / 62.1
SAN (on official dev and test set)	69.3 / 72.2	68.7 / 71.4
BERT-base Baseline	74.4 / 77.1	- / -
BERT + BiDAF	74.0 / 76.9	- / -
BERT + SAN	74.9 / 77.6	- / -
BERT + Model3 (single)	75.4 / 78.4	73.2 / 76.5
BERT + Model3 (ensemble)	76.2 / 79.0	- / -
BERT + Model3 & BERT + SAN (ensemble)	76.7 / 79.4	74.8 / 77.6

4 Experiments

4.1 Dataset

SQuAD 2.0[10] is a reading comprehension dataset on Wikipedia articles with more than 150K questions. About 1/3 of the questions can not be answered with their corresponding articles. The dataset used in this report is set up as described in CS224n Winter 2019 default project handout[1]. The training set has more than 130K questions; the development and the test sets each have around 6000 questions.

4.2 Experiment details

We use HuggingFace's PyTorch implementation of BERT[2] . The pre-trained BERT model used is BERT-base-uncased, which contains around 110M parameters. The BERT parameters is not frozen in training but fine-tuned together with other layers. The models are trained with BertAdam optimizer included in Huggingface's implementation. Learning rate starts from 5e-5, and decayed with a factor of 0.95 every 10000 training examples. Models are trained for 2 to 3 epochs. Due to GPU memory constraint, the max input sequence length is reduced to 384, batch size is set to 8. The dropout rate for BERT during training is 0.1.

For BERT+BiDAF model, the hidden size of LSTM layers is 100, dropout rate is 0.2.

For BERT+SAN model, the reasoning step is 5, dropout rate for predictions is 0.4, dropout rate for the states of reasoning steps is 0.2.

For BERT+Model3, the dropout rate for output layer is 0.1.

4.3 Results

SQuAD's evalution metrics are F1 and Exact Match(EM). Table 1 shows scores of the models explored in this project and some reference models. The best single model gets 76.5% F1 and 73.2% EM on the test set. The ensembled model gets 77.6% F1 and 74.8% EM on the test set.

One unexpected result is that simpler model (Model3) works better. This could due to complex structures are hard to train with pre-trained BERT. For example, in the original BiDAF model, the entire architecture is randomly initialized and trained for around 30 epochs. When using with BERT, we typical just fine-tune the model for 2 to 3 epochs. More training seems to cause the model to overfit on the training data. There is also a mis-match of best learning rates for fine-tuning BERT versus training complex output layers. This might explain why the performance ranking is opposite to the complexity of the output layer: Model3 > SAN > BiDAF.

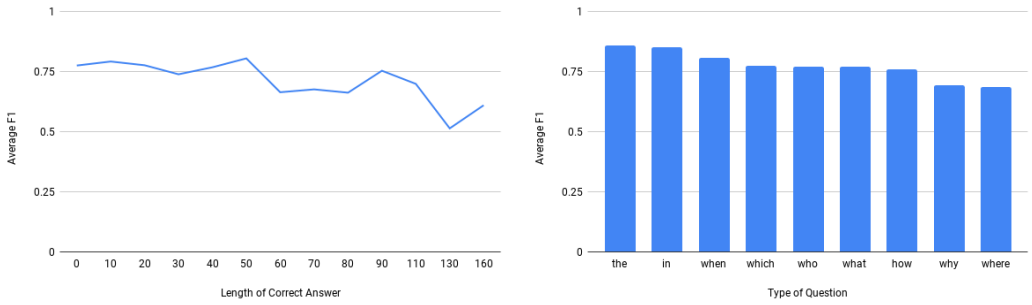

Figure 2: BERT + BiDAF: Average F1 w.r.t. correct answer length and type

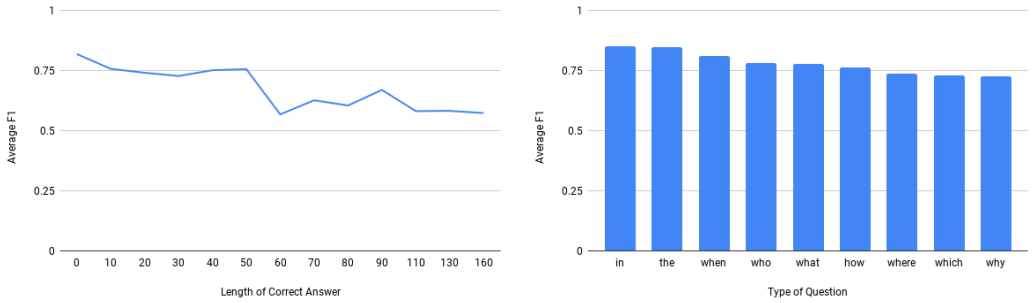

Figure 3: BERT + SAN: Average F1 w.r.t. correct answer length and type

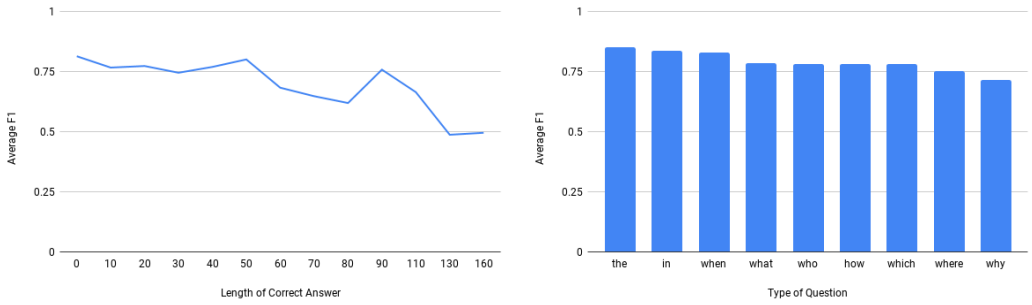

Figure 4: BERT + Model3: Average F1 w.r.t. correct answer length and type

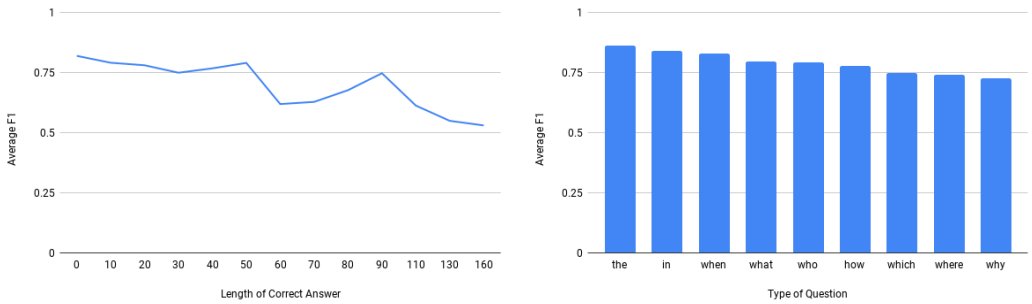

Figure 5: BERT + Model3 & BERT + SAN ensembled: Average F1 w.r.t. correct answer length and type

5 Analysis

5.1 Quantitative analysis

To further understand the models, we break down their performance with respect to the length of correct answers and the type of questions. Generally speaking, the models' performance goes down when the length of correct answers become longer. However, we are not seeing dramatic drop as reported in last year's reports[4][3] with non-BERT models. F1 scores of all models are generally above 0.75 when the correct answer length is less than 50. For longer answers, they can still maintain F1 scores at around 0.6. This shows BERT's ability to capture long distance dependencies with its deep and large-scale self-attention layers. This is one of the major reasons why BERT out-performs previous models. For question types, the models are generally better at "when" questions as time-related patterns are easier to spot; "why" type questions are hardest as expected as they involve more reasoning and typically have longer answers. The figures also show some difference between models. BERT+BiDAF are better when questions have answers, but not so good when they are unanswerable. On the other hand, the simple Model3 struggles when the answers are really long, but does a good job on unanswerable questions. Unanswerable questions count for around half of the dev and the test sets, that is why Model3 out-performs BERT+BiDAF. The models also slightly differ on the type of questions they are good at. Ensembling different kinds should improve performance. Our results also verified this.

5.2 Quanlitative analysis

5.2.1 Paraphrases

In some error cases, the core concept the question cares about is expressed in a different way in the passage. When those paraphrases differ in wording, the models will fail to recognize that they refer to the same thing.

Question: What is the seldom used force unit equal to one thousand newtons?

Passage: Other arcane units of force include the sthène, which is equivalent to 1000 N, and the kip, which is equivalent to 1000 lbf.

Answers: sthène

Prediction: No answer

5.2.2 Partial match

Sometimes, the models tend to find answer in the passage that partially matches the concept talked about in the question, but fails to notice the important words that further describe that concept. For example, in some cases, what question asks about is opposite to what is described in the passage. The model may focus on the similarity in phrases but fail to capture the difference in the direction of meaning.

Question: The successful searches for what showed that the elementary particles are not observable?

Passage: The strong force only acts directly upon elementary particles. However, a residual of the force is observed between hadrons (the best known example being the force that acts between nucleons in atomic nuclei) as the nuclear force. Here the strong force acts indirectly, transmitted as gluons, which form part of the virtual pi and rho mesons, which classically transmit the nuclear force (see this topic for more). The failure of many searches for free quarks has shown that the elementary particles affected are not directly observable. This phenomenon is called color confinement.

Answers: No answer

Prediction: free quarks

Question: How many Protestant Walloons fled to England before the Foreign Protestants Naturalization Act was passed?

[2]https://github.com/huggingface/pytorch-pretrained-BERT

Passage: Both before and after the 1708 passage of the Foreign Protestants Naturalization Act, an estimated 50,000 Protestant Walloons and Huguenots fled to England, with many moving on to Ireland and elsewhere. In relative terms, this was one of the largest waves of immigration ever of a single ethnic community to Britain. Andrew Lortie (born André Lortie), a leading Huguenot theologian and writer who led the exiled community in London, became known for articulating their criticism of the Pope and the doctrine of transubstantiation during Mass.

Answers: No answer

Prediction: 50,000

Those two types of errors show that what the models are doing is still kind of text matching. They need further improvement on its reasoning ability.

6 Conclusion

In this report, I implemented three models that combine BERT and popular models include BiDAF and SAN. They are able to improve performance on SQuAD 2.0 reading comprehension task comparing to BERT baseline. The analysis of the results shows that BERT based models are better at capturing long distant dependencies in texts. It also shows that the models' ability in reasoning still need improvement. For future work, BERT is pre-trained with single word masking task, but often phrases carry more information than single words. This is especially important in reading comprehension tasks. We can try to pre-train BERT with n-gram masking task, so that it better capture the meanings of phrases.

References

[1] Cs224n default final project: Question answering on squad 2.0. *CS224N 2019 Winter*, 2019.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[3] Zihuan Diao, Junjie Dong, and Jiaxing Geng. Question answering on squad dataset. *CS224N 2018 Winter*, 2018.

[4] Beliz Gunel and Cagan Alkan. Question answering on squad. *CS224N 2018 Winter*, 2018.

[5] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.

[6] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504, 2019.

[7] Xiaodong Liu, Wei Li, Yuwei Fang, Aerin Kim, Kevin Duh, and Jianfeng Gao. Stochastic answer networks for squad 2.0. *CoRR*, abs/1809.09194, 2018.

[8] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. Stochastic answer networks for machine reading comprehension. *CoRR*, abs/1712.03556, 2017.

[9] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

[10] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018.

[11] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[12] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[13] Fu Sun, Linyang Li, Xipeng Qiu, and Yang Liu. U-net: Machine reading comprehension with unanswerable questions. *CoRR*, abs/1810.06638, 2018.

[14] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461, 2018.

[15] 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, Lukasz 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, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[16] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018.