Question Answering on SQuAD with BERT

# Abstract

The complex interactions between context and question makes it difficult for machines to perform good on question answering(QA) tasks. In this study, we explore the performance difference of the task-specific Bi-Directional Attention Flow model(BiDAF) and the pretrained BERT (Bidirectional Encoder Representations from Transformers) QA model on SQuAD 2.0. We propose three self-designed model structures built on the BERT embeddings. The baseline BiDAF model achieves 60.5% F1 and 57.4% EM on the validation set. Our modified fine-tuned model on BERT achieves F1 and EM scores up to 76.6% and 73.6%. The best performed BERT QA + Classifier ensemble model further improves the F1 and EM scores to 78.1% and 75.3%.

# Introduction

From online searching to information retrieval, question answering is becoming ubiquitous and being extensively applied in our daily life. Reading comprehension is a popular instance of Question Answering tasks, where the system tries to provide the correct answer to the query with a given context paragraph. In 2016, Rajpurkar et al.[1] released the the Stanford Question Answering Dataset(SQuAD 1.0) which consists of 100K question-answer pairs each with a given context paragraph and it soon becomes a standard test for the reading comprehension task with public leaderboard available. In 2018, the team further released SQuAD 2.0, which contains over 50,000 unanswerable questions that post a much harder requirement on model development. At the same year, large-scale pre-trained language modes such as OpenAI GPT [2] and BERT [3] have achieved great performance on multiple language tasks using generic model architectures. For BERT, it is pre-trained with two auxiliary tasks(the Mask Language Model task and the Next Sentence Prediction task) with large corpus to encourage the bi-directional prediction on text as well as sentence-level understanding, Since many important down-stream tasks such as Question answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between pair of sentences, BERT can perform well when fine-tuned on these downstream specific tasks without customized network architectures.

# Architecture

Quantifying the success of question answering is a tricky task. When you or I ask a question, the correct answer could take multiple forms. For example, in our previous post, BERT answered the question, "Why is the sky blue?" with "Rayleigh scattering," but another answer would be:

The Earth's atmosphere scatters short-wavelength light more efficiently than that of longer wavelengths. Because its wavelengths are shorter, blue light is more strongly scattered than the longer-wavelength lights, red or green. Hence the result that when looking at the sky away from the direct incident sunlight, the human eye perceives the sky to be blue.

Both of these answers can be found in the Wikipedia article Diffuse Sky Radiation and both are correct. However, we've also had a model answer the same question with "because its wavelengths are shorter," which is close - but not really a correct answer; the sky itself doesn't have a wavelength. This answer is missing too much context to be useful.

What if we'd asked a question that couldn't be answered by the Diffuse Sky Radiation page? For example: "Could the sky ever be green?" If you read that Wiki article you'll see there probably isn't a sure-fire answer to this question. What should the model do in this case?

How should we judge a model’s success when there are multiple correct answers, even more incorrect answers, and potentially no answer available to it at all? To properly assess quality, we need a labeled set of questions and answers. Let's turn back to the SQuAD dataset position are equal to −1.

# Fuctionality

SQuAD2.0 consists of over 150k questions, of which more than 35% are unanswerable in relation to their associated passage. [For our last post](https://qa.fastforwardlabs.com/pytorch/hugging%20face/wikipedia/bert/transformers/2020/05/19/Getting_Started_with_QA.html), we fine-tuned on the train set (130k examples); now we'll focus on the dev set, which contains nearly 12k examples. Only about half of these examples are answerable questions

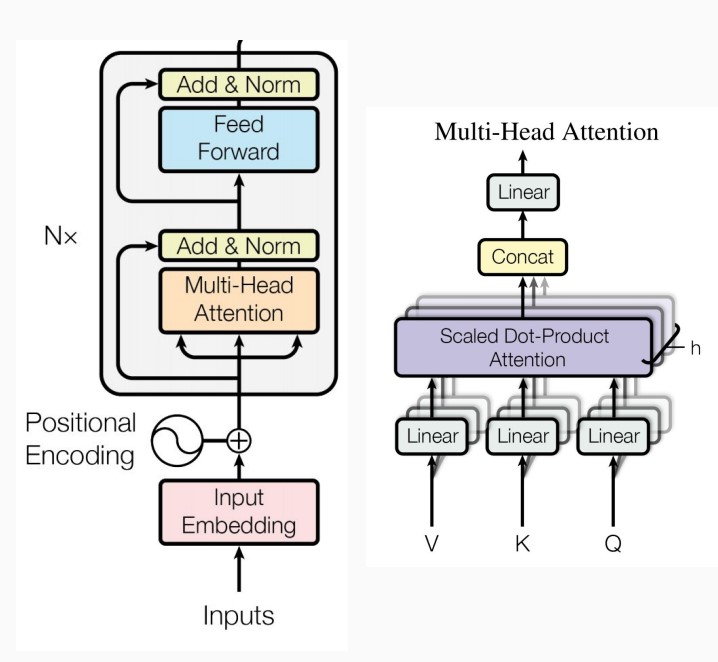
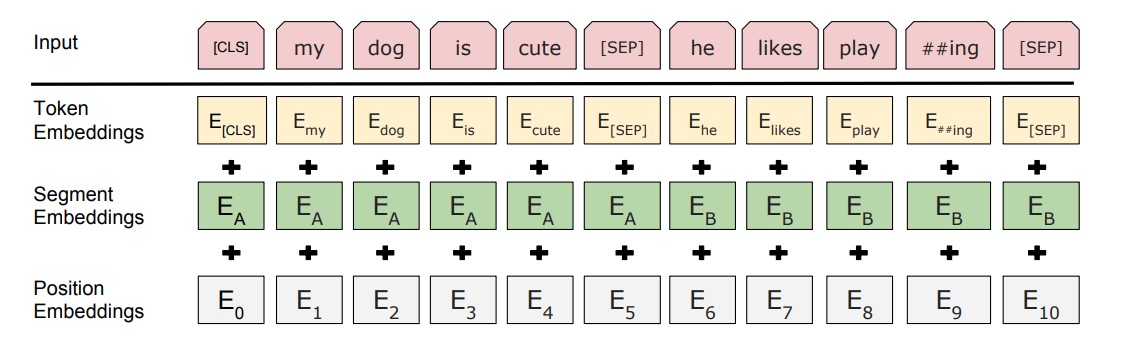


Figure 2: BERT input representation Figure 3: Transformer Struc-

ture

3.2.2 Fine-tuned BERT QA Model with modified output layer

The proposed method is a modified version of the BERT fine-tuned QA model in 3.2.1. In this phase, we aim to improve the model performance by altering the output layers on BERT. As shown in Figure 4, our final best-performed architecture consists of three main layers that are described below in detail.

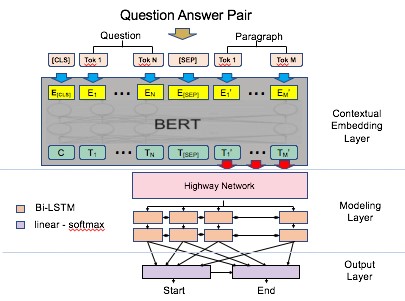


Figure 4: Modified BERT QA model

Embedding Layer

After the pre-training process of BERT, the transformer outputs the full sequence of hidden-states corresponding to the last attention block. The hidden states are the concatenated vector representation of the the context and questions *T*1*,T*2 ···*TN* ∈ *Rh*. Therefore BERT can be viewed as the embedding layer which is fixed and are not updated during training in the model.

Highway Network

We use a two-layer Highway Network[6] to transform each hidden vector *Ti*. The transformation is applied twice and each time with distinct learnable parameters. ,which means we apply the above transformation twice, each time using distinct learnable parameters. By introducing the Highway Network we hope to refine the BERT embedded representation with the gating mechanism and enable to the LSTM structure with bigger step transition depth to optimize the training procedures. [8]

*xp* = ReLU(*WpTi* + *bp*)

‘ *x*gate = *σ*(*W*gate*Ti* + *bgate*)

*x*highway 

where *W*gate*,Wp* ∈ *Rh*×*h* and *bp,bgate* ∈ *Rh*

Modeling layer

Modeling layer consist of two layers of bi-directional LSTMs with hidden size *h* to scan the vector space representations after the highway layer. It is designed to better capture the relation between context and question. The output of the modelling layer is *M* ∈ *R*2*h*×*L*. After the modelling layer, we also perform dropout operation to avoid overfitting.

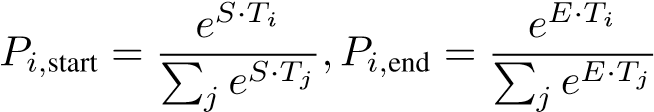
{*g*1*,*··· *,gL*} = biLSTM({*x*1*,*··· *,xL*})

{*m*1*,*··· *,mL*} = biLSTM({*g*1*,*··· *,gL*})

*mi* = [−*m*→*i*;←−*mi*] ∈ *R*2*h*, for *i* = 1*,*···*L*, where L is the maximum sequence length

where

Output Layer For the output layer, we predict the answer start and end probability distributions independently with two heads, each performing a linear down-projection followed by softmax.



3.2.3 Fine-tuned BERT QA Model with Modified Loss

Pooled representation of the firsts token

The third model is aimed at improving the model performance on non-answerable questions. We observe that the first token [CLS] is an indicator to emit logit for "no answer". Therefore, as shown in Figure ??, we take the final hidden state for the first token [CLS] in the input, which by construction corresponds to the the special [CLS] embeddings. The vector is denoted as *Q* ∈ *Rh*. Then the predicted class*Cp* could be constructed with a sigmoid regression layer:

*Cp* = sigmoid(*W* ∗ *Q* + *b*)

where *W* ∈ *RK*×*h,K* = 2*,ki* = {0*,*1} ∈ *R*2, as 0 is an indicator of non-answerable questions and 1 is an indicator of answerable questions.

Modified Training Loss Function In the original scheme, the loss is computed as the sum of the cross entropy loss for start and end positions. The proposed model considers the total loss to be a weighted average of the start and end position losses and the cross entropy loss of the predicted labels*Cp* compared to true labels *Cr*. Let *si,ei* denote the start and end logit outputs, *sr,er* be the true start and end positions, and *CE* be the cross entropy loss. Then the modified loss becomes: *Loss* = (*αCE*(*si,sr*) + *αCE*(*ei,er*) + *βCE*(*Cp,Cr*))*/*(2 ∗ *α* + *β*)

3.2.4 Ensemble BERT QA + Classification Model

Instead of combining the QA and classification tasks as 3.2.3, in the third model, we separately fine-tuned the QA and classification model and ensemble the two models to jointly produce the answer span. In Figure 6., for the BERT classifier on the right, it takes the final hidden state for the first [CLS] token, which by construction corresponds to the special [CLS] word embedding. We denote this vector as *C* ∈ *RH*. The vecor is then input into a classification layer *W* ∈ *RK*×*H* with K being the number of labels. As the same with 3.2.3, *K* = 2.

Then the label probabilities can be computed as

*P* = softmax(*CWT*)

If the predicted lable is 0 as impossible answers, the output answer span from the QA model would be rejected.

4.4 Results

4.4.1 Final Model Results

The experimental results on the validation set for our current findings are listed in the below table: The scores are broken down to the total data, the data with answers and the data that has no answer.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Total Questions | | Answerable Ques | | Non Answerable | |
|  | F1 | EM | F1 | EM | F1 | EM |
| Baseline | 60.5 | 58% |  |  |  |  |
| Distilbert-base-uncased | 66.7 | 66.3 | 75.8 | 68.9 | 63.6 | 63.6 |

Evaluation Metrics captured for the Model performance.

# Conclusion

In this paper, we implement and design three variants on three different types of question-answer modes: single BERT fine-tuned model, single BERT model classified model. ensemble BERT question answer with classifier model. The experimental evaluations show that our model achieves competitive results in SQuAD 2.0. The model can perform well on predicting the correct answer locations for answerable questions and also detecting whether the question is answerable. For future development, we hope to incorporate more additional features to the BERT embedding output, such as name entities, POS tag and the question-context matching features, to further boost its performance.

# References

1. Zhang Rajpurkar and et al Lopyrev. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250, 2016*, 2016.
2. Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI, 2018.