Bert
Default CS224N Project

Mohith Rajendra
mohithr@stanford.edu

Abstract

Question answering is one the challenging problems in natural language processing. This paper presents adaption of a high-performance question answering system Bidirectional Encoder Representations from Transformers (BERT) which outperforms BiDAF model which is opted as a baseline model. In this paper we compare BERT with fine-tuned BERT model by additional layers to evaluate the performance.

1 Introduction

Question Answering is one of the challenging tasks in natural language processing. In this task a passage and question will be provided for the model to answer the given question based on the passage. QA task has seen a rapid growth in research area due to its application in chat bots to simulate human conversation. This paper is concentrated to increase the performance towards Stanford Question Answering Dataset 2.0 (SQuAD 2.0) which is an addition of new 50k unanswerable questions to SQuAD 1.1 which consist of 100k+ questions. Until recent years there was use of many RNN-based units which was consuming a high training time mainly because of sequential inputs but in recent years we have seen Attention based model which eliminated the use of RNN.

Bidirectional Encoder Representations from Transformer (BERT) [2], has produced state-of-the-art results in many NLP task which includes Stanford Question Answering Dataset. Bert is a pretrained general purpose language representation model, which is fine tuned for downstream NLP tasks also provides comparatively easier platform for fine-tuning to any downstream task. In this project I have experimented by adding auxiliary task on BERT to evaluate the variation of performance based on the auxiliary tasks.

2 Related Work

Solution to QA tasks can be majorly segregated into two section: pretrained contextual embedding (PCE) models and non-pretrained contextual embedding. In recent language model pretrain has proved to be more effective on improving many tasks of natural language. In PCE (pre-trained contextual embedding) word embedding is achieved by pretrained weights of a large-scale language modeling dataset. Google released BERT is one among the well know PCE which is conceptually simple but empirically powerful. A lot of derived model from BERT are leading the leaderboard such A Lite BERT(ALBERT) [9] which is improved version of BERT with parameter reduction and increase of training speed is now leading in leaderboard by overcoming humans. These models are briefly described below

2.1 ALBERT

A Lite BERT (ALBERT)[9] was published recently by Google Research and Toyota Technological Institute at Chicago established new state-of-the-art results on various benchmarks including SQuAD while having fewer parameters compared to BERT-large to have a lower memory consumption and also to increase the training speed.
There are three main contributions that ALBERT makes over BERT with respect to design such as

1. Cross-layer parameter sharing - which enables the model to improve the parameter efficiency
2. Factorized embedding parameterization – decomposing of embedding parameter to two small matrices by first projecting as lower dimension embedding space and then project it to the hidden size
3. Inter-sentence coherence loss - Use of sentence-order prediction (SOP) loss which avoids the prediction of topic but focuses on modelling inter-sentence coherence

3 Approach

In this section we explain the model’s architecture and layers used in the project

3.1 Baseline Model

Bi-Directional Attention Flow (BiDAF)[3] model is opted as the baseline model, which is defined in [3]. In contrast to the original model, word embeddings are used for input layers. In brief, the baseline BIDAF model consist of these following layers:

• Embedding layer: embeddings are done for both question and context by using GloVe pre-trained word vectors, then projected and passed through a two-layer Highway network to refine the embedded representation.
• Encoder layer: output obtained from the embedding layer is passed via a bidirectional LSTM.
• Attention layer: bi-directional attention flow layer, attention should flow both ways from context to question and question to context.
• Modeling layer: two-layer bidirectional LSTM to refine the sequence of vectors after attention layer.
• Output layer: takes input from attention layer and modelling layer to apply bidirectional LSTM to produce a vector of probabilities for each position in the context.

3.2 BERT Model

BERT pre-trains on the concatenation of BooksCorpus (800M words, Zhu et al., 2015), and English Wikipedia (2,500M words). Each input sequence is generated by sampling two spans of text, the first of which receives the sentence A embedding, and the second of which receives the sentence B embedding. The input representation sums up token embeddings (WordPiece embeddings), segment embeddings (representing whether the word belongs to sentence A or B), and position embeddings (see Figure 1).

![Figure 1: BERT input embeddings](image)

BERT is based on the transformer encoder architecture [5] and is currently used as part of most state-of-the-art models in many natural language processing tasks. The inputs are embedded through three different components. First, they are tokenized as usual and embedded using pre-trained token embeddings. Then, a positional embedding is added in order to give the model positional information, since it is attention based. Lastly, a sentence embedding is added, which differs from sentence to sentence but is the same for every word in a given sentence. There are also
two minor additions: we add a token at the beginning of the input, and a token between the context and the question (Figure 2)

During the fine tuning process the model learns a parameter start vector $S \in \mathbb{R}^h$ and end vector $E \in \mathbb{R}^h$ which is later consumed to calculate the probability of the $i$th word as the start of the sentence and similarly to find the end of the answer span.

$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$$

In this project huggingface repository is used which provides an example to run on squad dataset, that allows us to finetune an instance of BertForQuestionAnswering, the BERT model adapted for SQuAD. BertForQuestionAnswering is a BERT Transformer with a token classification head on top, and has a from_pretrained class method that allows us to load BERT weights that are pretrained specifically for the SQuAD task. This runs the output from the generic BertModel through a Linear Layer to obtain start and end logits. Cross Entropy Loss is used during training, and softmax is used in the write_predictions function to select the most likely start and end positions for each answer span. To fine-tune BERT, we experiment with different hyperparameters, and run run_squad.py

### 3.3 BERT + Highway

Another approach that we tried was running the final hidden states through a highway network before computing logits. Highway networks can help improve performance because they contain gating mechanisms that can filter out potentially irrelevant information. In this model, we ran the final hidden states through a highway network using the following equations:

$$x_{proj} = ReLU(W_{proj}x + b_{proj})$$
$$x_{gate} = \sigma(W_{gate}x + b_{gate})$$

Next, we utilized a gate to combine the highway projection with the skip connection.

$$x_{highway} = x_{gate} \odot x_{proj} + (1 - x_{gate}) \odot x$$

And then finally, we passed the output of the highway network through a dropout layer and linear layer to obtain the overall representations of our hidden states.

### 4 Experiments

#### 4.1 Data
Stanford Question Answering Dataset (SQuAD) 2.0 [6] is a dataset comprising of is a reading comprehension dataset that consists of passages from Wikipedia and associated questions whose answers may or may not be present in span in the passage. In this paper we are trying to find a span in the context which answers the question or select <no-answer> if the question is unanswerable.

Our dataset can be split as follows:
- train (129,941 examples): All taken from the official SQuAD 2.0 training set.
- dev (6078 examples): Roughly half of the official dev set, randomly selected.
- test (5915 examples): The remaining examples from the official dev set, plus hand-labeled example.

4.2 Data pre-processing and training

We use 'BERT base uncased' vocabulary and tokenizer provided by the authors of BERT and convert the question and context to lower case before feeding it to the model. We use the huggingface repository that provides PyTorch version of BERT pretrained model.

4.3 Evaluation Method

Performance of each model is evaluated using the two standard metrics: exact match (EM) score and F1 score. EM score measures whether the model output matches the ground truth answers exactly. F1 score is the harmonic mean of precision and recall, and less strict than then EM score. Besides EM and F1, the impact of passage length, question length, answer length on errors are also analyzed. By drawing the distribution of errors across different thresholds, it will help us better understand the model.

4.4 Experiment Details

In this project huggingface implementation of BERT in PyTorch is used under NV6. The pre-trained BERT-base-uncased, which contains around 110M parameters provided by huggingface is used. BERT parameters are provide to auxiliary layers. The set of hyperparameters used are as follows: Number of epochs 2, batch size is set to 6 due, dropout rate is set to 0.1, learning rate starts from 3e-5, and decayed with a factor of 0.95 every 10000 examples. Models are trained for 2 to 3 epochs.

5 Analysis

In this section, we provide analysis on following aspects.

5.1 Qualitative Analysis

5.1.1 Imprecise Span boundary

For most questions, our model was roughly correct but exhibited either too much span as compared to the ground truth answer or not enough span that only partially overlapped the answer. One such example is as follows:

**Context**: The Iroquois sent runners to the manor of William Johnson in upstate New York. The British Superintendent for Indian Affairs in the New York region and beyond, Johnson was known to the Iroquois as Warraghiggey, meaning "He who does great things." He spoke their languages and had become a respected honorary member of the Iroquois Confederacy in the area. In 1746, Johnson was made a colonel of the Iroquois. Later he was commissioned as a colonel of the Western New York Militia. They met at Albany, New York with Governor Clinton and officials from some of the other American colonies. Mohawk Chief Hendrick, Speaker of their tribal council, insisted that the British abide by their obligations and block French expansion. When Clinton did not respond to his satisfaction, Chief Hendrick said that the "Covenant Chain", a long-standing friendly relationship between the Iroquois Confederacy and the British Crown, was broken.

**Question**: Who was the speaker of the tribal council?

**Ground truth**: Mohawk Chief Hendrick
**Prediction:** Hendrick, Speaker of their tribal council,

5.1.2 Incorrect Inference
In some cases, due to the length and semantic complexity of the question, our model missed the mark and was unable to label correct answer span.

**Context:** Moderate and reformist Islamists who accept and work within the democratic process include parties like the Tunisian Ennahda Movement. Jamaat-e-Islami of Pakistan is basically a socio-political and democratic Vanguard party but has also gained political influence through military coup d'etat in past. The Islamist groups like Hezbollah in Lebanon and Hamas in Palestine participate in democratic and political process as well as armed attacks, seeking to abolish the state of Israel. Radical Islamist organizations like al-Qaeda and the Egyptian Islamic Jihad, and groups such as the Taliban, entirely reject democracy, often declaring as kuffar those Muslims who support it (see takfirism), as well as calling for violent/offensive jihad or urging and conducting attacks on a religious basis.

**Question:** Where does Hamas originate?

**Ground truth:** Palestine

**Prediction:** and work

6 Conclusion and Future Work

Acknowledgments
I would like to thank for open-sourcing the source code for BIDAF by CS224N staff and huggingface for open-sourcing the pretrained model of BERT embeddings using PyTorch which eased the implementation of this project.

References


