BertNet : Combining BERT language representation with Attention and CNN for Reading Comprehension

Girish Limaye
glimaye@stanford.edu

Manish Pandit
manish7@stanford.edu

Vinay Sawal
vsawal@stanford.edu

Abstract

This paper presents BertNet, a high-performance question answering system utilizing BERT [2] language representation combined with QANet [10] inspired RNN-free attention and encoder layers which trains faster than baseline BiDAF [7] model. We design the final layer of of our model to work with SQuAD 2.0 [1] to effectively accommodate predictions on questions with no answers. This paper also demonstrates additional improvement to the performance by using back translation based data augmentation techniques. We used English-Arabic-English as back translation languages for increasing training data and show significant performance gain on "difficult" questions which the model otherwise doesn't perform well on. On SQuAD 2.0 data-set, our single model trained with data augmentation produces an F1 Score of 76.35 on test-set.

1 Introduction

With the prevalence of ChatBots and Conversational AI in a variety of businesses and consumer use-cases, there is a growing interest in machine reading comprehension and automated question answering systems. Most of the deep learning models for reading comprehension in the past used shallow static word embedding (e.g. Word2Vec, GloVe [4]) and Recurrent Neural Networks to process the sequential inputs. But with advances in using Attention Mechanism ("Attention is all you need" [8]), newer "RNN free" approaches that take less training time are showing promising results. Over the last year, we have also seen that deep language modeling based representational frameworks such as ELMO [5], BERT [2] and ULMFIT [3] out-do traditional representations on multiple NLP tasks. BERT for example presented state-of-the-art results in a wide variety of NLP tasks, including Question Answering , Natural Language Inference (MNLI), and a few other.

Our approach combines BERT based language representation with QANet inspired Attention and Encoder layers. We further demonstrate improvement in our model performance by incorporating back translated questions in our training set from English to Arabic and back in English. We use SQuAD 2.0 as our database for training and test.

2 Related Work

There has been a lot of work on using bidirectional deep embeddings in general and BERT in particular for the SQuAD Dataset, as can be seen in the leaderboard [2]. In addition to these pretrained deep embeddings, top models utilize some form of attention mechanism, which has been proven useful in improving accuracies in many NLP tasks. Attention mechanism initially became popular in context of Neural Machine Translation [5], where it generate a translation word based on the whole original sentence states, not just the last state. The idea is then applied to reading comprehension,

[1] https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev/

[2] https://rajpurkar.github.io/SQuAD-explorer/

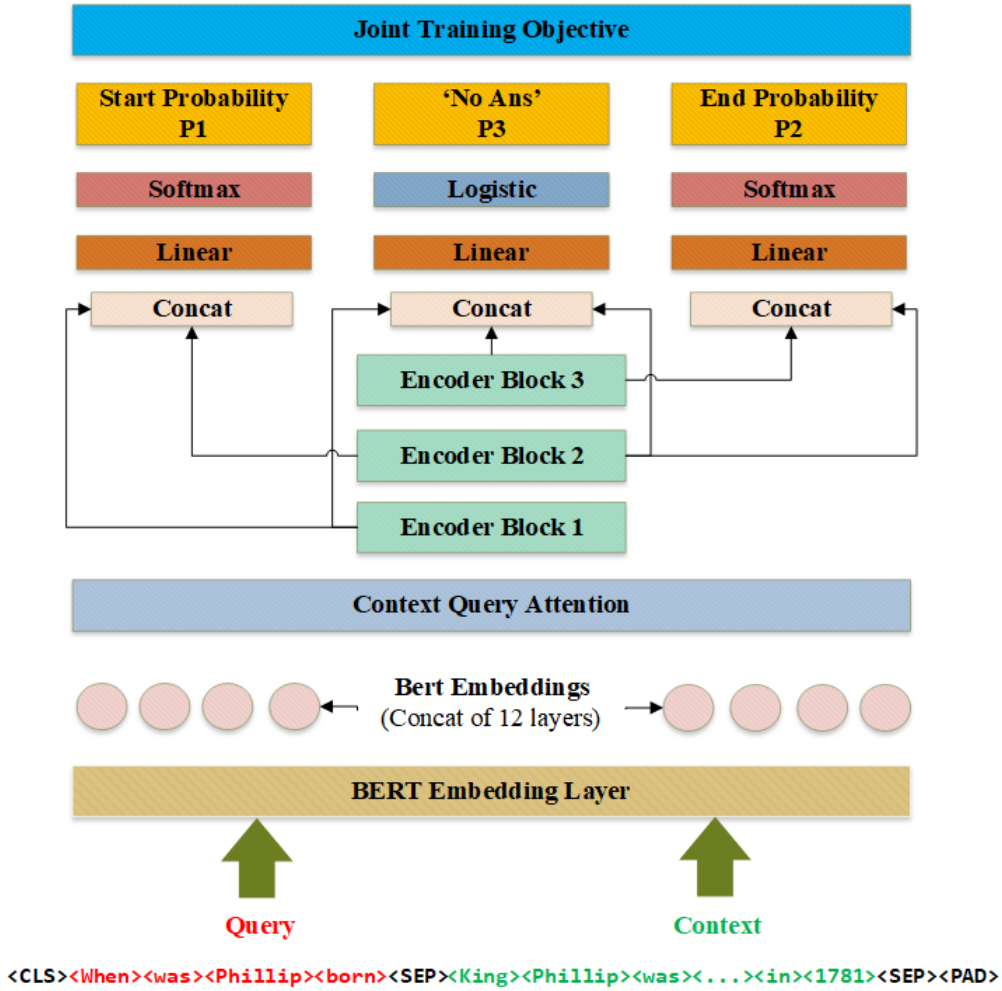

Figure 1: Architecture

allowing the model to select a subset of context paragraph and a subset of question that are most relevant. Data augmentation has also been explored in variety of areas within natural language processing. Zhou et al. [11] improved the diversity of the SQuAD data by generating more questions. Similarly QANet [10] paper shows performance improvement by using back translation to enrich training data.

3 Approach

Here we describe our model's architecture and layers.

3.1 Model

As described in figure 1, our model consists of following layers:

1. Embedding: BERT Embedding Layer for Query and Context sentences
2. Attention: Context Query Attention Layer
3. Encoders: Three Stacked Encoder layers
4. Output: Output pooled from three sub-output layers one each from *StartSpan*, *EndSpan* and *<No Answer>*

3.1.1 Embedding Layer

We use BERT pre-trained model to embed Context and Query. BERT uses positional embeddings to encode the word sequences without a need for Recurrent Architecture. BERT embeddings are generated using unsupervised learning based on Masked Language Models (MLM) and Next Sentence Prediction (NSP). Masked Language Model approach makes it possible for BERT to use bidirectional attention using transformer for a Language Modeling objective.

BERT Model expects sentences in one of two formats. If we have a context of *King Phillip was born in 1711 and ruled Wales from 1729 to 1734* and Query of *Which year was king Phillip born?*, we can either encode it as a single input or two seperate inputs to the BERT Model as shown below.

<CLS><When><was><King><Phillip><born><SEP><King><Phillip>.....<1734><SEP> Or

<CLS><King><Phillip>.....<1734><SEP> and

<CLS><When><was><King><Phillip><born><SEP>

We provide Context and Query together in one sentence with query first followed by the context as described in figure 1. Since BERT is trained on "next sentence prediction" outcome, we believe that using this formulation will provide richer query and context embeddings. We make each sentence of 384 words and Query portion being max of 64 words using padding at the end as required. We experimented using both - only the top BERT layer vs. concatenation of all 12 layers of BERT as our embedding layer. We used the latter in our final model. We also experimented with keeping static BERT frozen embeddings vs. fine-tuning the BERT embedding for the downstream task. Our final model uses fine-tuned BERT embeddings.

3.1.2 Attention Layer

We use a focused, Context-Query attention layer on top of the pre-trained BERT embeddings identical to that of the QANet model. This module is standard in almost every previous reading comprehension models such as Weissenborn et al. [9], Chen et al. [1] and QANet [10]. C is used to denote context and Q for query. The context-to-query attention is constructed as follows:

- Compute similarities between each pair of context and query words that generates a similarity matrix $S \in \mathbf{R}^{n \times m}$
- Normalize each row of S by applying softmax function
- Compute context-to-query attention $A = S \cdot Q^T \in \mathbf{R}^{n \times d}$

The similarity function used here is the trilinear function as mentioned in Seo et al. [7]:

$$f(q, c) = W_0[q, c, q \odot c]$$

3.1.3 Encoder Layer

The encoder layer is a stack of the following basic building blocks of following layers:

[convolution-layer + self-attention-layer + feed-forward-layer]

Similar to QANet and Seo et al. [7], the input of this layer at each position is $[c, a, c \odot a, c \odot b]$, where a and b are respectively a row of attention matrix A and B. For the self-attention-layer, we've adopted multi-head attention mechanism defined in Vaswani et al. [8] which, for each position in the input, called the query, computes a weighted sum of all positions, or keys, in the input based on the similarity between the query and key as measured by the dot product. We experimented with different number of heads from 2 to 8. Each of these basic operations (conv/self-attention/ffn) is placed inside a residual block.

3.1.4 Output Layer

Our output layer is similar to that of QANet with the addition of a stack of sub-output that predict <*No Answer*>. For the training examples which have answers we use the standard strategy of Seo et

al. [7] to predict the probability of each position in the context being the start or end of an answer span. More specifically, the probabilities of the starting and ending position are modeled as:

$$p^1 = softmax(W_1[M_0; M_1])$$

$$p^2 = softmax(W_2[M_0; M_2])$$

where $W1$ and $W2$ are two trainable variables and $M0, M1, M2$ are respectively the outputs of three model encoders, from bottom to top. Additionally we add another sub-output for <*No Answer*> which is calculated as follows:

$$p^3 = sigmoid(W_3[M_0; M_1; M_2])$$

where $W3$ is a trainable matrix.

The final loss is defined as:

$$L(\theta) = -\frac{1}{N}\sum_i^N [(\log(p^1_{y_i^1}) + \log(p^2_{y_i^2})) * (1 - y_i^3)/2 + (y_i^3) * log(p^3_{y_i^3})]$$

Where y^1 and y^2 correspond to actual begin and end span, and y^3 is a boolean flag to indicate whether there is an answer to the question.

3.2 Data Augmentation

Taking inspiration from QANet [10], we used back-translation style data augmentation technique to increase our training-set. We chose Arabic as back-translation language due to the fact that it has wide variation on factors like grammar, syntax, morphology, writing style, phonology, alphabet, and language root to that of English.

Arabic is part of Semitic Language family and has a three consonant root as its basis. Al parts-of-speech are formed by combining the three-root consonants. Arabic does not make the distinction between actions completed in the past with/without a connection to the present. Also, there are no modal verbs in Arabic. These differences add diversity to the paraphrasing effort and produces syntactically variant phrase. Since the task at hand is Span labeling, we chose to augment our training set by back-translating only questions and keeping the context as-is. Here is an example:

Original Question: *is the avian flu a risk only to animals?*

Back-translated Question: *Is avian influenza a threat to animals only?*

4 Experiments

To get benchmark and performance on our model, we conducted experiments on the SQuAD 2.0 dataset (Rajpurkar et al. [6]).

4.1 Data

We use SQuAD 2.0 as our database of choice for training and test. SQuAD 2.0 is a reading comprehension dataset that consists of passages from Wikipedia and associated questions whose answers span in the passage. It also has some questions that are designed to be unanswerable. The task of our model is to do **Span Labeling**: given a passage, identifying a span of text which contains an answer or predict <*no answer*> if the answer is not present. Formally, given a passage P with tokens $p_1, \ldots, p_n$ and a question Q with tokens $q_1, \ldots, q_m$ such that for each token p_i, the goal is to compute the probability of $p_{start}(i)$ and $p_{end}(i)$ marking the start and end of the answer span.

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. As described in section 3.1.1,

Table 1: Model performance on dev set

Dev Set	F1 (%)	EM (%)
Human Performance	89.45	86.83
BERT-NET (Our Final Model with data augmentation)	**77.15**	**74.07**
BERT-NET - no data augmentation	76.27	73.21
BERT Base + output layer	75.94	72.74
BIDAF baseline	59.29	55.99

we feed the question and query together as one input to BERT model. We chose the maximum combined sentence length to 384 and maximum question length to 64. For training we do layer by layer training with finetuning of the network. We first train the bert model for 2 epochs with just the output layer. We then replace the output layer with an attention layer using the pretrained bert model and let it train for one more epoch. We then add the Encoder Layers on top and train for 1 more epochs. For regularization we use dropout between layers setting the values with hyper-parameter optimization. The layer wise training allows us to do quick experiments on top of pretrained bert layer where the F1/EM for the validation set goes up to above 70 quite quickly in quarter to half of an epoch (30 to 60 minutes on NV12) and provides quick feedback on whats working and whats not working.

4.3 Results

We conducted experiments SQuAD 2.0 dataset sequentially starting with BiDAF as base-line and then using BERT model and adding layers as described in section above. Along the way, we recorded EM and F1 scores from our experiments. While we use F1 and Exact Match (EM) are two evaluation metrics of accuracy for our model we have provided analysis of other factors in the next section.

We provide the results in the table 1. As shown, BERT embedding based 'RNN free' model provides a significant increase in F1 and EM metrics over our baseline BIDAF model. Additionally, our model runs in significantly less time than the baseline model. While baseline model took approx. 11 hours to train, the BERT baseline model trains in 100 - 130 minutes per epoch. Also the basic BERT model reaches a performance of 75.9 F1 in just 2 epochs of training. But more importantly, taking BERT model fine-tuned for 2 epochs on SQUaD 2.0 allows us to do rapid experiments with additional layers where the performance of the model quickly reaches above 70+ F1 score on an evaluation set in quarter to half of an epoch (approx. 30 to 60 minutes on NV12). When used with differential learning rates (low learning rate for BERT layer and higher rate for new layers), this provides quick feedback on efficacy of the newly added layers.

Majority of the gain over the baseline model comes from BERT embedding layer with thin output layer. Since we pass the question and context as one line to the embedding layer, BERT which was pretrained with objective of next sentence prediction provide rich representations of Question and Context. RNN Free Attention and Encoder Layers inspired by QANet provide additional performance improvements without need for long training cycles. Data augmentation with back translation provides only a slight incremental improvement in performance but more importantly it demonstrates significant gain in answering a small class of "difficult" questions that the model otherwise does poorly on. This is described further in the section on analysis.

5 Analysis

In this section, we preset quantitative as well as qualitative analysis of our model results. We provide analysis on following aspects.

5.1 Analysis of Length of the answer

As the length of ground truth answer increased, our model's performance decreased but since 95% of answers in the dev set are less than 10 words long, our model's overall performance is still decent. Reduction in performance as a function of word length is an expected result because of the inherent

Table 2: **Model performance on various answer length on dev set**

Ground truth Avg. answer length	EM(%)	F1(%)	No. of questions
$0 < len \leq 5$	79.04	83.54	2472
$5 < len \leq 10$	59.37	74.76	416
$10 < len \leq 15$	51.16	71.46	86
$15 < len \leq 20$	35.71	49.16	14
$20 < len$	0.0	35.71	2

Table 3: **Model performance based on question type**

Question Type	EM(%)	F1(%)	No. of questions
Which	77.38	80.34	274
When	77.36	79.10	561
What	74.61	77.59	3726
Who	74.93	76.99	714
How	71.07	75.37	605
Where	69.45	72.10	275
Why	55.43	68.40	92
Other	52.54	62.38	59

complexity in accurately choosing long span answers even for human evaluators. Our hypothesis is that our approach of choosing the answer span using start and end pointers might be less suited for longer answers. In future we look to experiment with additional ways of choosing spans within a context.

5.2 Analysis on different question types

From a question category point of view, as the sementic complexity of the questions increased, our model's performance decreased. Our model performed best on four out of Five Ws [3] pattern of questions. For what we call "difficult" question types ('Why' type and 'Other' interrogative questions), our model didn't perform as well since these type of questions are inherently hard to comprehend and hence answer even for a human annotator. The model's performance on different golden answer length can be seen in Table 2. Our model performs the best on "when" questions and performs the worst on why question. Also, typically "when" questions have short answer length (2.24 on average), whereas "why" questions have higher answer length (6.87 on average).

5.3 Impact of data-augmentation on Model performance

Given the time and cost constraints of this project we were able to only test data augmentation using back translation of question using only one language(Arabic) with Google translate service. This essentially doubled the size of our training set. Even though our model's performance did not improve significantly using this one set of augmented data, analysis of the errors shows promising results . When trained with back-translation based data augmented training set as compared to non data-augmented training set, Table 4 shows that back-translation improves the model's performance on the "difficult" questions ("Why" and the "Other" type of questions) which our model otherwise performs poorly on. On the "other" type of questions, for example, the F1 score goes up from 62.38 to 72.58. When the questions are back-translated, this provides additional rephrasing of questions into a more Canonical form of the question. Since these type of questions are relatively rare in our data set this does not translate to significantly greater overall performance boost. Following examples shows paraphrased questions[4] and correctly predicted answer.

Context: *Concepts of this network influenced later ARPANET architecture.*

[3] `https://en.wikipedia.org/wiki/Five_Ws`

[4] We back-translated training set for learning. We didn't back-translate dev set. The back-translated question is what we extrapolate our model interpreted this question as.

Table 4: **Model performance when trained with augmented data**

Question Type	EM/F1 no Data-Augment	EM/F1 Data Augment	Difference (%)
Which	77.38 / 80.34	76.27 / 79.26	-1.43 / -1.34
When	77.36 / 79.10	75.40 / 77.32	-2.53 / -2.25
What	74.61 / 77.59	74.12 / 77.15	-0.65 / -0.56
Who	74.93 / 76.99	73.10 / 75.18	-2.44 / -2.35
How	71.07 / 75.37	69.75 / 73.46	-1.85 / -2.53
Where	69.45 / 72.10	66.18 / 69.76	-4.70 / -3.24
Why	**55.43 / 68.40**	**57.60 / 69.87**	**+3.91 / +2.15**
Other	**52.54 / 62.38**	**64.40 / 72.58**	**+22.57 / +16.35**

Original Question: *Telenet was sold to*

Back-translated Question: *Who was Telenet sold to?*

Ground truth: *GTE*

Prediction (without Data-Augmentation): <*Null*>

Prediction (with Data-Augmentation): *GTE*

Context: *Concepts of this network influenced later ARPANET architecture.*

Original Question: *The 4 sales and service centers are viewed as*

Back-translated Question: *4 sales and service centers are viewed as what?*

Ground truth: *the world's first commercial online service*

Prediction (without Data-Augmentation): <*Null*>

Prediction (with Data-Augmentation): *the world's first commercial online service*

5.4 Qualitative Analysis

5.4.1 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:

Question: *What are the two simple word responses to a decision problem?*

Ground truth: *yes or no*

Prediction: *yes or no, or alternately either 1 or 0*

5.4.2 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: *Each of these models can be converted to another without providing any extra computational power. The time and memory consumption of these alternate models may vary.*

Question: *In considering Turing machines and alternate variables, what measurement left unaffected by conversion between machine models?*

Ground truth: *computational power*

Prediction: *time and memory consumption*

6 Conclusion and Future Work

In this report, we proposed a model architecture that uses language representation provided pre-trained BERT [2] model with a top layers and data augmentation strategy inspired by QANet [10]

to produce competitive results on SQUaD 2.0 datasets in much less training time than the baseline BIDAF approach.We find that doing layerwise training on top of fine-tuned BERT layer allows for rapid experimentation. Additionally we show how the back translation based data augmentation has the potential of standardizing text representation which helps in the difficult("Why" and "Other" type) questions which have low F1 and EM scores otherwise.

Based on this work and analysis of the results we believe that pretrained deep embeddings along with "RNN Free" attention and CNN based methods provide a strong starting point for Question Answering task. The analysis also suggests two avenues to improve the results further. Firstly, experimenting with a final output layer that selects a full span and calculates loss based on full spans as opposed to start span and stop span might potentially help with dealing with the weakness of our model on longer answer spans. Secondly, having additional research on data augmentation using back translation with the goal of standardizing the text representation of context and question at training time has the potential to improve model performance on difficult and non-standard question. Test time data augmentation of question and context using variety of languages has the potential to further improve performance on these type questions and is an important area of future research.

Acknowledgments

We would like to thank Hugging Face [5] for their opens source code to use pretrained BERT embeddings for downstream tasks using PyTorch, the CS224N course staff for the source code for the baseline BIDAF model & excellent support that they've provided throughout the course.

References

[1] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051, 2017.

[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] Jeremy Howard and Sebastian Ruder. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146, 2018.

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[5] 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.

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

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

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[9] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Fastqa: A simple and efficient neural architecture for question answering. *CoRR*, abs/1703.04816, 2017.

[10] 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.

[11] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. Neural question generation from text: A preliminary study. *CoRR*, abs/1704.01792, 2017.

[5] `https://github.com/huggingface`