Question Answering Using Hierarchical Attention on Top of BERT Features

Reham Osama, Nagwa El-Makky and Marwan Torki
Computer and Systems Engineering Department
Alexandria University
Alexandria, Egypt
{eng-reham.osama, nagwamakky, mtorki}@alexu.edu.org

Abstract

Machine Comprehension (MC) tests the ability of the machine to answer a question about a given passage. It requires modeling complex interactions between the passage and the question. Recently, attention mechanisms have been successfully extended to machine comprehension. In this work, the question and passage are encoded using BERT language embeddings to better capture the respective representations at a semantic level. Then, attention and fusion are conducted horizontally and vertically across layers at different levels of granularity between question and paragraph. Our experiments were performed on the datasets provided in MRQA shared task 2019 [1]

1 Introduction

The tasks of question answering (QA), especially machine comprehension have gained significant popularity over the past few years within the natural language processing and computer vision communities. Systems trained end-to-end now achieve promising results on a variety of tasks in the text and image domains. The task of machine comprehension is challenging as it requires a comprehensive understanding of natural languages and the ability to do further inference and reasoning. Restricted by the limited volume of the annotated datasets, early studies mainly relied on a pipeline of NLP models to complete this task, such as semantic parsing and linguistic annotation. Benefiting from the availability of large datasets, e.g., SQuAD (Rajpurkar et al., 2016), rapid progress has been made recently.

There have been advancements in multiple variations of the problem including visual question answering and video question answering due to this fast improvement in QA models. Attention mechanisms have a very significant role in increasing the performance of the models as they focus on the targeted area in the passage. In this paper we use BERT (Devlin et al., 2018) to obtain the representation of both the passage and question, then an encoder layer, which consists of recurrent neural networks, is used to build representations for questions and passages, then a co-attention layer and fusion followed by a self-attention layer are used. Finally, an output layer is added to get the index of both the start and end of the answer.

The rapid progress that has been made recently was mainly due to the availability of SQuAD dataset benchmark. The work in (Wang and Jiang, 2016) was one of the first to investigate the dataset. The authors proposed an end-to-end architecture based on match-LSTM and pointer networks. (Seo et al., 2016) introduced the bi-directional attention flow network which captures the question-document context at different levels of granularity. (Chen et al., 2017) introduced a bilinear match function and a few manual features. (Wang et al., 2017) proposed a gated attention-based recurrent network where self-match attention mechanism is first incorporated. In (Liu et al., 2017) and (Shen et al., 2017) the multi-turn memory networks are designed to simulate multi-step reasoning in machine reading comprehension.

(Devlin et al., 2018) introduced a new language representation model called BERT which is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks including question answering. BERT makes use of Transformer (Vaswani et al., 2017) which is an attention mechanism that learns contextual relations between words (or sub-words) in a text. BERT uses the encoder mechanism from the trans-

[1] https://mrqa.github.io/shared

former as the goal is to generate a language model.

BERT is considered a masked language model as the input to the BERT model is masked before entering the model. 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence.

Due to the great effectiveness of the attention mechanism in the performance of the machine comprehension systems, we used two attention mechanisms in this work similar to (Wang et al., 2018), where in addition to the co-attention mechanism proposed in (Seo et al., 2016) we use a self attention for each of the paragraph and question. So, the output layer can use both of them while predicting the start and end index of the answer.

2 Architecture

The model used in this work was inspired by some of the components of (Devlin et al., 2018) and (Wang et al., 2018).

We chose the components from these 2 models due to their effectiveness in solving the task of machine comprehension. So, we expected that merging the strong components from both models will achieve better results than each one of them individually.

The proposed architecture is explained in this section and is shown in Figure 1.

2.1 Embedding Layer

For the input embeddings we used BERT pre-trained models (Devlin et al., 2018) which is based on word piece level tokenization. BERT has two models that have the same architecture with different sizes

1. BERT base: which consists of 12 transformer blocks, 12 attention heads, and 110 million parameters.
2. BERT large: which consists of 24 transformer blocks, 16 attention heads and, 340 million parameters.

BERT can be used in two ways. The first way is to use it as a model and add a task-specific layer on top of it to produce the required output and train the model with the added layer. The second way is to use it as a pre-trained language model while either keeping the pre-trained weights as they are or training them with your model.

In this work we used a pre-trained BERT base model and we used the second way which is using the pre-trained model with training its weights along with the model. Using BERT large model is expected to yield better results when used. We didn't use it in this work due to the limited resources we had, as the machine we had access to couldn't run BERT large model in its memory.

2.2 Encoder layer

The goal of this layer of the model is to transform the discrete word tokens of question and passage to a sequence of continuous vector representations.

In this layer a Bi-LSTM network is used on top of the embeddings provided by the previous layer to model the temporal interactions between words.

2.3 Co-Attention Layer

This layer is similar to the co-attention layer used in (Seo et al., 2016). Given the question and passage representation from the previous layer, a soft-alignment matrix is built to calculate the shallow semantic similarity between question and passage. We use this similarity matrix to compute the attention between question and passage, which is further used to obtain the attended vectors in passage to question and question to passage direction, respectively.

The output here is a passage-aware question representation and a question-aware passage representation. The question-aware passage representation is calculated using the Passage to Question (P2Q) Attention which signifies the question words that are most relevant to each passage word. The passage-aware question representation is calculated using Question to Passage (Q2P) Attention which signifies the passage words that have the closest similarity to one of the question words and are hence critical for answering the question.

After calculating the aligned passage and question representation, a fusion unit is used to combine the original contextual representations and the corresponding attention vectors for question and passage.

There are several ways to perform the fusion according to (Wang et al., 2018) but one of the simplest ways, which we used here, is a concatenation of the two representations. This fusion is performed due to the importance of the original contextual representations in reflecting the semantics at a more global level.

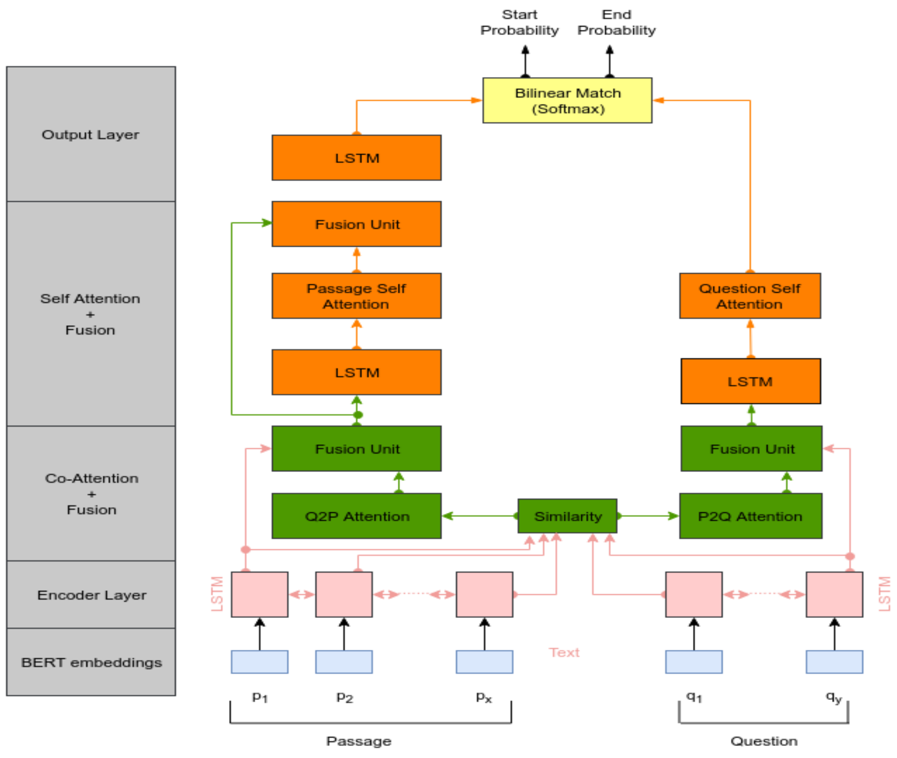

Figure 1: Model Architecture

2.4 Self attention Layer

In this layer, we separately consider the representations of question and passage, and further refine the obtained information from the co-attention layer. Since fusing information among context words allows contextual information to flow close to the correct answer, the self-attention layer is used to further align the question and passage representation against itself, so as to keep the global sequence information in memory.

The idea of benefiting from the advantage of self-alignment attention in addressing the long-distance dependence was taken from (Wang et al., 2017). To allow for more freedom of the aligning process, we used a bilinear self-alignment attention function on the passage representation, introduced in (Wang et al., 2018). We then follow this layer with another fusion unit that combines the question-aware passage representation with the passage self-aware representation. Then, a bidirectional LSTM is used to get the final contextual passage representation.

As for question side we follow the question encoding method used in (Chen et al., 2017) followed by linear transformation to encode the question representation to a single vector. First, another contextual bidirectional LSTM network is applied on top of the fused question representation Then we aggregate the resulting hidden units into one single question vector, with a linear self-alignment.

2.5 Output layer

Instead of predicting the start and end positions based only on the passage representation, a top-level bilinear match function is used to capture the semantic relation between question and passage representation from the previous layer in a matching style.

The top model layer uses a bilinear matching function to capture the interaction between outputs from previous layers and locate the right answer span.

The output layer is application-specific, in Machine comprehension task, we use pointer networks to predict the start and end position of the answer, since it requires the model to find a continuous span of the passage to answer the question.

3 Experiments

In this section, we first present the datasets used for evaluation. Then, we explain the evaluation metrics used, and finally we report the results after training the previously explained model on the given datasets.

3.1 Datasets

In this work we used the datasets provided by the MRQA 2019 shared task. The training datasets included some benchmark datasets such as SQUAD and NewsQA. In-domain and out-of-domain development datasets are also included. Examples of the out-of-domain datasets are DROP and RACE.

The datasets were adapted from several existing datasets from their original formats and settings to conform to the unified extractive setting. The changes made to the datasets to conform to the new settings included:

1. Only a single, length-limited context is provided.
2. There are no unanswerable or non-span answer questions.
3. All questions have at least one accepted answer that is found exactly in the context.

3.2 Evaluation metrics

Performance is measured via two metrics: Exact Match (EM) score and F1 score.

- **Exact Match:** is a binary measure (i.e. true/false) of whether the system output matches the ground truth answer exactly. This is a considered a strict metric.
- **F1:** is a less strict metric. It is the harmonic mean of precision and recall.

A span is judged to be an exact match if it matches the answer string after performing normalization consistent with the SQuAD dataset. Specifically:

1. The text is uncased.
2. All punctuation is stripped.
3. All articles, e.g., a, an ,the, etc. are removed.
4. All consecutive whitespace markers are compressed to just a single normal space ' '.

3.3 Training details

We use the BertAdam optimizer, with a batch size of 6 and initial learning rate of 0.0003. A dropout rate of 0.2 is used for all LSTM layers. We take F1 score as reward with Cross Entropy Loss. We consider the BERT parameters trainable during the training process.

Dataset	**EM**	**F1**
BioASQ	43.02	59.09
DROP	24.38	34.78
DuoRC.ParaphraseRC	38.46	49.64
RACE	24.57	37.38
RelationExtraction	67.87	81.30
TextbookQA	32.10	40.49

Table 1: Development Datasets Results.

Dataset	**EM**	**F1**
BioProcess	44.29	60.83
ComplexWebQuestions	41.87	51.21
MCTest	54.23	67.88
QAMR	47.97	66.01
QAST	50.91	75.51
TREC	27.72	48.71

Table 2: Test Datasets Results.

The training process takes roughly 48 hours on a single Nvidia Tesla K80 GPU when training the whole provided training and validation datasets, and it takes roughly 12 hours when training a sample size of 20000 instances from each of the training datasets and a sample size of 2000 from each of the development datasets.

3.4 Results

The results of our model on all the development datasets are summarized in Table 1 and the results on all the test datasets are summarized in Table 2. The proposed model achieved an average EM of 41.45 and an average F1 of 56.07 on all datasets (development and test sets combined).

The average F1 obtained for the development datasets is 50.45 and the average F1 obtained for the test datasets is 61.69

3.5 Other Experiments

Other experiments were performed either by changing the model parameters or by trying to add new componenets. But they didn't achieve any increase in the performance of the model.

The following is a brief description of each of the tried experiments

1. Making BERT parameters not trainable.
   In this experiment we tried to use the BERT parameters as they are without retraining, but this caused the performance to decrease significantly.

2. Adding CNN character level embeddings with different number of filters(64, 100).
   In this experiment, the input to the model was the concatenation of BERT embeddings and the character level embeddings. At first we used the settings in (Seo et al., 2016) but we couldn't train the model due to the memory limitations.
   When setting the BERT parameters to be trainable, we cannot add the character level embeddings. However, when we set the parameters to be untrainable ,which causes a big decrease in the performance, the maximum number of filters we could use was 100.

3. Adding L2 regularization.
   We expected that adding the L2 regularizer will make the model achieve better results on the validation and not seen datasets but this didn't happen.

4. Using Adamax optimizer instead of BertAdam.
   In this experiment, we used Adamax optimizer instead of the BertAdam optimizer but we didn't achieve better performance.

4 Conclusion

In this work, we described our machine comprehension system which was designed for the MRQA 2019 Shared Task. When supplied a question and a passage it makes use of the BERT embedding along with the hierarchical attention model which consists of 2 parts, the co-attention and the self-attention, to locate a continuous span of the passage that is the answer to the question.

The proposed model achieved an average EM of 41.45 and an average F1 of 56.07. After analyzing our results, we have identified many ways for improving the system in the future. For instance, other features can be added to the passage and question representations such as adding character embeddings with BERT embeddings. Part-Of-Speech (POS) and Named Entity Recognition (NER) features can also be added in the self-attention layer to better capture the information in the passage.

Another way that will probably increase the performance is using the proposed model with the BERT large model to produce the embeddings instead of BERT base in case there are more resources available.

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Rui Liu, Wei Wei, Weiguang Mao, and Maria Chikina. 2017. Phase conductor on multi-layered attentions for machine comprehension. *arXiv preprint arXiv:1710.10504*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Wei Wang, Ming Yan, and Chen Wu. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. *arXiv preprint arXiv:1811.11934*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198.