Multi-hop Question Answering on HotpotQA

Chris Wang, Yilun Xu, Qingyang Wang
Stanford University
{chrwang, ylxu, iriswang} @ stanford.edu

Abstract

We explore BERT and RNN-based models for multi-hop question answering on HotpotQA. State-of-the-art models, particularly BERT-based ones, have achieved close to human performance on single-hop question answering tasks, while their performance fails to generalize in the multi-hop area. We take two approaches to handle the key challenges of multi-hop QA–complex questions and long, noisy contexts. To de-noise the context, we use a pipeline method which links a sentence classifier with a pre-trained BERT fine-tuned for question answering. We use a RNN-based classifier to select "supporting" sentences first, and then feed the output into the BERT model for question answering. Our result shows that BERT could be a viable model to conduct multi-hop reasoning if we can control the noise properly. However, its performance deteriorates drastically on noisy context. Separately, we experiment on attention structures for RNN-based models without BERT. Our proposed structure manages to outperform the baseline on a 10% subset, but it fails to do so on the full training set. However, based on our error analysis, we still believe that RNN is better at handling noise than BERT with an RNN classifier, and expect our structure to perform better with further hyper-parameter tuning.

Mentor: Our CS224N mentor is Xiaoxue Zang. The project idea is provided and advised by Peng Qi from Stanford AI Lab.

1 Introduction

In this project, we explore effective models for multi-hop question answering on the HotpotQA dataset. As shown in the following example, multi-hop questions have complex question structures and long, noisy contexts.

Single-hop	What is 2018's highest grossing movie?	Locate the movie
Multi-hop	When was 2018's highest grossing movie released?	Locate the movie $\rightarrow$ Find the date

Previous question answering research has been primarily focused on single-hop questions which have relatively simple structures and can be answered using information contained in one paragraph. State-of-the-art models, particularly BERT-based ones, manage to achieve close to human performance on multiple datasets. When generalizing such methods into the multi-hop area, two major challenges impede the their performance:

- Complex questions, particularly those with multiple question words, challenge the model's ability to understand the query correctly.
- Long context, which contains relevant information and noise, requires the model to identify noise, establish inference among multiple relevant paragraphs, and process long sequences effectively.

Unlike the single-hop question sphere, for multi-hop question answering there is no evidence showing that BERT is a dominant model structure. As a result, we work on both BERT and RNN-based structures to tackle

this problem. As we will detail in section 3.2, we adopt a pipeline method for BERT. To begin with, we train a RNN-based classifier for context selection, where we use the answer as extra supervision. Then, we feed the output of our classifier into a fine tuned BERT for question answering with our training set. In parallel, we modify the baseline model proposed by Yang, Qi, Zhang, et al. (2018) [5]. As we will detail in section 3.3, we experiment on contextualized word representations, sentence-level attention structure and additional attention layers to marginalize the noise and enhance question/context understanding.

2 Related work

Our project idea is based on the new multi-hop question answering dataset, HotpotQA, introduced by Yang, Qi, Zhang, et al. (2018) [5]. Compared to existing works, HotpotQA features answers that depend on explainable hops across multiple contexts, and provides supporting sentence labels which can be used as extra supervision. The authors establish a baseline model and the evaluation method in their work, which we will detail in section 3.1 and 4.2, respectively.

The task of Question Answering has gained significant popularity over the past years. One key factor that has advanced this task is the use of attention mechanisms, which has allowed models to focus on targeted parts of sentences. [4] Several variants to improve attention structures have been proposed. One is to use a dynamically allocated attention weight matrix. [1] The hierarchical multi-stage process that represents context at different levels has been proven to improve the performance on the SQuAD dataset significantly. [4] Other than model structure, other parts of the end-to-end question answering system have been improved as well. Due to their ability to capture syntactic and semantic information from large scale unlabelled data, pre-trained word embeddings play a vital role in context understanding and machine translation. The deep contextualized representation of the word, exposing all signals captured by the embedding, has been proven to be highly beneficial to downstream tasks. [3]

3 Approach

We start from the baseline model. We want to investigate the performance of BERT on multi-hop question answering tasks, but we are also not sure if this is an ideal structure.

3.1 Baseline

The baseline model, provided by Yang, Qi, Zhang, et al. (2018) [5], takes a list of question-context pairs, together with a few relevant features as input, and predicts the answer span or "yes"/"no".

To begin, the model encodes the paragraphs and questions using GloVe word embeddings and CharCNN. Then, word-level bi-attention and self-attention are applied to each context-question pair. Furthermore, the attented vector is used in three RNN layers to predict start token, end token and "yes"/"no" type answers, respectively (which override start/end). A supervised layer is added before the prediction layers to choose the supporting sentences, which also becomes a part of the loss.

We use the authors' code for baseline model on Github: `https://github.com/hotpotqa/hotpot`. The original data import structure occupies a large amount of memory. We change the preprocessing and import structure before training the model. For fast iteration, we benchmark the baseline performance when training/testing on 10 percent of the data.

3.2 BERT with classifier

Setup

With existing sequential models and transformers, training occurs in a single direction (left-to-right or right-to-left). BERT, or Bidirectional Encoder Representations from Transformers, is designed to have deep bidirectional representations from both the left and right context simultaneously. BERT accomplishes this by introducing pre-trained representations trained with a bidirectional trasnformer trained on "masked words" and a binarized next sentence prediction task.[2]

Because of its deep embedded pre-trained representations, BERT is easily fine-tuned for a wide range of tasks, including question answering. Normally, fine tuning by applying two linear layers to the BERT outputs in order to predict the start and end token to the answer in context is sufficient to fine-tune the model for question answering. However, since the pretrained BERT model has a maximum context length of 512, which is smaller than HotpotQA's contexts, we are left with two choices–either to retrain a BERT model from scratch, a very computationally challenging task for such a long context, or to choose a subset of the original context to put into and fine-tune our BERT model. We choose the latter option, due to our computing limitations. Thus, we must first design a classifier to determine what sentences to input to BERT.

Classifier

The goal for our classifier is to classify each sentence in the context based on whether or not it should be fed into the BERT model. We want our overall input to the BERT to be less than the maximum context size, yet still have the necessary supporting facts to predict the answer. We construct a classifier based on the baseline model by Yang, Qi, Zhang et al., in order to instruct the BERT model to only pay attention to the most relevant sentences. We tune the objective of the classifier to best predict sentences for the downstream task of answer selection in the BERT–regardless of sentence EM or F1.

In our modified classifier, we adjust the baseline model to focus on predicting the relevance of each sentence in answering the question. Because we believe that predicting whether each sentence is helpful for answering the question is more of a high-level semantic problem than predicting the answer itself, we switch the layers of the baseline to predict the start and end token of the answer before the supporting facts. We keep the supervision of the answer in predicting supporting facts because our experiments show that this supervision is imperative to guiding the model in the right direction for the supporting facts. As in the baseline, our final classification for each sentence is based on a linear projection of the concatenated first and last word contextualized representations. We propose a few combinations of the losses to co-train the prediction of the supporting fact and the start/end token, trying to optimize for the supporting input to the BERT model. Another problem we face is that the labels for the classifier are unbalanced, since each example has approximately 70 sentences, but only 2 of them are labeled as relevant. We use a weighted cross-entropy loss to penalize false negative predictions and choose the best weight combination balancing precision and recall for the downstream task–BERT answer prediction.

BERT

We fine-tune our BERT model for question answering in three different ways–training it on gold standard sentences only, training it on gold standard paragraphs only, and training it on the output of the classifier. In each case, we take the original training set and modify the context in one of the three ways, and then train the BERT with the modified contexts and corresponding answers. The BERT model will concatenate the question and the answer tokens and train a deep contextualized representation of the input. We then apply a linear layer to this representation to predict a start and end token. Through this way, we fine tune the BERT representations for question answering with multi-hop supporting facts, which have varying levels of noise.

3.3 RNN-based model

Data preparation

Compared to single-hop question answering tasks, HotpotQA provides more context (10 paragraphs) for each question. The baseline model concatenates all paragraphs, together with their titles (labels), as the input. We believe including the titles creates extra noise for the model. As a result, we remove all paragraph titles when preprocessing the data.

ELMo

In order to capture the contextual features, we experiment with replacing GloVe with ELMo. ELMo, as proposed by Peter et al., is a contextualized representation of the entire sentence, with a weighted stack of internal states from a biLM. [3] The high-level LSTM states are expected to capture context-level information, while lower-level states should model syntax-level meaning. Incorporating all states in the downstream model will allow it to utilize the contextual information. [4]

We use the pretrained ELMo from Allennlp [3] in place of GloVe. To fine-tune the embedding, we add two linear layers on top of the API, and update the weights in the training process. However, such structure doesn't

perform well on our downstream task. Meanwhile, including ELMo increases our training time and memory usage significantly. To make the best use of our time, we decide not to include ELMo in our final model.

Attention

For multi-hop question answering, longer context requires the model to identify noise and conduct multi-hop reasoning. Complex structure of questions requires the model to locate the key words properly. To overcome the hurdles, we propose a few attention changes to achieve our goal.

1) Original structure of the bi-attention layer in baseline model

Let $\boldsymbol{h}_i$ and $\boldsymbol{q}_j$ be the embedding vector for context word i and question word j. Denote n_q and n_c as the lengths of the question and context respectively.

$$a_{ij} = \boldsymbol{w}_1 \boldsymbol{h}_i + \boldsymbol{w}_2 \boldsymbol{q}_j + \boldsymbol{w}_3 (\boldsymbol{h}_i \circ \boldsymbol{q}_j) \tag{1}$$

Then we have attented token for each context word i as:

$$\boldsymbol{c}_i = \sum_{j=1}^{n_q} \boldsymbol{q}_j \text{SIGMOID}(a_{ij}) \tag{2}$$

And the query-to-context vector $\boldsymbol{q}_c$ as:

$$\boldsymbol{q}_c = \sum_{i=1}^{n_q} \boldsymbol{h}_i \text{SIGMOID}(\max_j a_{ij}) \tag{3}$$

The concatenated vector $[\boldsymbol{h}_i, \boldsymbol{c}_i, \boldsymbol{h}_i \circ \boldsymbol{c}_i, \boldsymbol{q}_c \circ \boldsymbol{c}_i]$ is exported as the output.

2) Sentence-level bi-attention

To deal with the long context, we propose to include sentence-level bi-attention in our model and expect such practice to highlight the relevant sentences and marginalize the noises. Most existing work [6] on sentence-level attention introduces a RNN as sentence encoder. As shown in figure 4 (see appendix), each word encoder, multiplied by its attention, is passed into a sentence encoder to generate a sentence-level attention. However, we don't think such an approach is applicable to our issue. The sentence encoder (RNN) brings more parameters, which may cause the overfitting problem. Also, we don't need the sentence encoder in the following layers and thus may not tune its parameters properly.

Bearing that in mind, we consider using the simple average, 2-norm or max attention score of the word-level attentions, for each sentence. Then, we multiply the word-level attention with our sentence-level score to get an "augmented" score.

Denote $a_{ij}^{(n)}$ as the word-level attention score for the n-th word in sentence S, where a_{ij} is the score computed in equation (1). We can get the sentence-level attention score as:

$$\textit{Simple avg.: } \bar{a}_S = \frac{1}{|S|} \sum_{n \in S} a_{ij}^{(n)} \;||\; \textit{2-norm: } \bar{a}_S = \sqrt{\frac{1}{|S|} \sum_{n \in S} \left(a_{ij}^{(n)}\right)^2} \;||\; \textit{Max: } \bar{a}_S = \max_{n \in S} a_{ij}^{(n)}$$

Furthermore, we can get the augmented word-level attention score as:

$$\tilde{a}_{ij}^{(n)} = \bar{a}_S * a_{ij}^{(n)}, n \in S$$

The augmented score is then used to compute (2) and (3), which are combined as output of the attention layer. Our small scale test shows that simple avg. (see appendix, figure 5) works best in getting the sentence level score. However, it fails to outperform the baseline model.

3) Paragraph-level bi-attention

We come up with two hypotheses to explain the failure of our sentence-level attention. On one hand, our structure may skew the gradient computed by the optimization algorithm, and thus impede the back propagation. On the other hand, our structure may overstate the key words and fail to consider links between sentences. Consider the following example:

"Charlie ... . And he ... [answer key]."

If we have "Charlie" in the question, our sentence attention will stress more on the first sentence while the true answer is contained in the second one.

To test our hypotheses, we further propose to include a paragraph-level bi-attention, following the same structure of sentence-level bi-attention. However, we find that the training loss diverges after a few thousand steps. As a result, we believe that the sentence-level/ paragraph-level attention may hurt the back propagation, and decide not to include them in our final model.

4) Question/ context self-attention

To locate the question words properly, we propose adding a self-attention layer on question, before computing the context-question bi-attention. Furthermore, we add a self-attention layer on context before the bi-attention. We expect such a structure to improve the multi-hop inference.

Finally, we propose our RNN model structure as shown in figure 6 (see appendix).

Fine-tuning hyper-parameters

In addition, we fine-tune the following hyper-parameters of our model to achieve better performance. We test both SGD and Adam on our dataset. We finally decide to use Adam with $\beta = (0.9, 0.999)$ for its fast convergence. Accordingly, we initiate our learning rate as $lr = 0.001$ and set learning rate decay by a factor of 0.5, with patience of 2. In order to regularize the model, we set the dropout probability as 0.2 for all dropout layers in our model.

4 Experiments

4.1 Data

We are using the HotpotQA dataset for multi-hop question answering task. The HotpotQA dataset contains 112,779 valid examples of question and answers, over 90% of which is multi-hop ones.

The training set, provided by HotpotQA, contains around 90K samples, each with 10 paragraphs as context for the question answering task (only two of which are used to answer the question). We estimate that it will take about a day to train the model on full dataset. To make best use of our time, we conduct most experiments using 10% of the training set and only train the final model on the full dataset. There are two variants of the val and test sets. In the distractor setting, there are 10 context paragraphs, just as in the train set. In the full-wiki setting, there are no guarantees to the number and relevance of the context paragraphs. We focus our efforts on the distractor setting, though we also analyze our results on the full-wiki setting.

4.2 Evaluation method

HotpotQA suggests three sets of metrics to evaluate the performance-normal EM and F1 scores to exam answer quality, supervised EM and F1 scores to check whether supporting facts are used properly, and joint EM and F1 which is a combination of the two. We use all three sets of metrics to report our performance.

4.3 Experimental results and analysis

BERT with classifier

The classifier is trained for 10 epochs with SGD, learning rate decay by a factor of 0.5, and a batch size of 12. It takes about 6 GPU hours to finish the training. Our experiment shows that a wide range of precisions and recalls can be achieved by changing the class weight of cross entropy loss, loss weight (supporting fact loss vs. supervised loss), and the threshold. The optimal input to the BERT model should have a balance between

low false negative rate and low false positive rate, since the former ensures that enough supporting sentences get selected and the latter controls the noise. We put more efforts into limiting the false negatives, because missing a positive supporting fact will bottleneck the performance of the model. We finally decide to use class weight of $0:1=1:11$, loss weight of $1:2$ and the threshold of 0.65. Our optimal classifier model has a precision and recall of 0.4646 and 0.8259. While the model may not optimize the supporting fact EM score, we will ensure it optimizes the answer EM through BERT.

Using 10 $\%$ of the training data, we first train three separate BERT models (along with a separate classifier model) in order to determine which BERT training procedure is the most effective. The classifier prediction itself is also tuned with different thresholds for inclusion of a sentence in the BERT input. Across different classifier thresholds, BERT trained on the gold sentences achieved a maximum answer EM score of 32.84 and F1 score of 42.27, while BERT trained on the gold paragraphs achieved a maximum EM score of 33.64 and F1 score of 43.51. BERT trained on the output of the sentence classifier model achieved a maximum answer EM score of 31.21 and F1 score of 45.43. At first, this result seems counter-intuitive because the classifier attempts to predict the context sentences, not paragraphs. However, BERT trained with the gold standard paragraphs achieves a higher score likely because the paragraphs train the BERT to deal with longer contexts with relevant, but not directly helpful sentences. BERT trained with the supporting fact sentences only is not robust to sentences which are not actually supporting facts, while BERT trained with the output of the classifier takes in too much noise. All three of these models outperform the baseline on 10 percent of the data, which has an EM of 29 and F1 of 39.

After selecting the paragraph BERT, we fine-tune a BERT model on the entire training set (with gold standard paragraphs as context). We then evaluate answer and supporting fact EM/F1 using our two-part model consisting of the optimal context classifier and the full train set paragraph trained BERT (for both full-wiki and distractor). To test the BERT, we also filter out the gold standard paragraphs from the dev set and evaluate only the BERT portion of the model on the dev input. The following table displays our results.

Dev set results, BERT models		Ans		Sup		Joint	
Model	Setting	EM	F1	EM	F1	EM	F1
Baseline model	distractor	44.44	58.28	21.95	66.66	11.56	40.86
Our classifier and BERT	distractor	42.13	53.63	22.58	62.14	12.44	36.64
Perfect paragraph classifier and BERT	distractor	53.67	66.97				
Baseline model	full-wiki	24.68	34.36	5.28	40.98	2.54	17.73
Our classifier and BERT	full-wiki	24.11	32.29	5.55	39.91	3.35	17.31
Perfect paragraph classifier and BERT	full-wiki	29.13	37.43				

As we can see, although the two-part BERT+Classifier model outperforms the baseline significantly with 10 percent data, it performs slightly worse than baseline (at least on answer) on the full dataset. We believe this is caused by the BERT being already pretrained on a large corpus of data, and thus can be effectively fine-tuned data with li. Thus, the BERT model may be effective to use when presented with limited data.

RNN-based model

We use 10 $\%$ of the training data to test our proposed structures. To begin with, we test the sentence-level attention structure (using simple average) and achieve answer EM score of 25.17 and answer F1 score of 33.96, which is worse than the baseline performance. Separately, we test the paragraph-level attention structure and find that the training loss diverges after a few thousand steps. Based on the experiments, we believe that our proposed sentence-level/ paragraph-level attention may impede the back propagation and thus hurt the performance. We decide not to include those changes in our final model.

We add a self-attention module for question and context, before the bi-attention layer. The model manages to achieve answer EM score of 29.50 and answer F1 score of 39.58, which is slightly better than the baseline. As a result, we believe additional self-attentions layers may enhance the model's capacity in question/ context understanding, and decide to include that in our final model.

After deciding the model structure, we train our model on the full training set with Adam, initial learning rate of 0.001, learning rate dacay by a factor of 0.5 and a batch size of 12. It takes about 20 GPU hours for our model to finish training. Our results are detailed as follows.

Dev set results, RNN-based model		Ans		Sup		Joint	
Model	Setting	EM	F1	EM	F1	EM	F1
Baseline model	distractor	44.44	58.28	21.95	66.66	11.56	40.86
Our RNN (w/ self-attention)	distractor	43.17	56.90	19.82	65.44	10.14	39.58
Baseline model	full-wiki	24.68	34.36	5.28	40.98	2.54	17.73
Our RNN (w/ self-attention)	full-wiki	22.96	31.97	4.27	36.35	2.04	16.23

As shown above, our observation on 10% training date fails to generalize to the full dataset–our structure and model configuration under-performs the baseline. On one hand, we conduct hyper-parameter fine-tuning on the 10% dataset, whose optimal setting may be different from the full training set. On the other hand, we conduct the training procedure differently from Yang, Qi, Zhang, et al. (2018) [5], due to memory and storage constraints. For example, we use much smaller batch size than that reported by the authors, and that could deteriorate the result.

Comparison of BERT and RNN-based models

Furthermore, we sample 200 examples out of the dev distractor set, which contains about 7,400 examples in total. We manually label the 200 questions into 7 categories, and conduct human evaluation of the predictions made by RNN, BERT with classifier and BERT with gold paragraphs models.

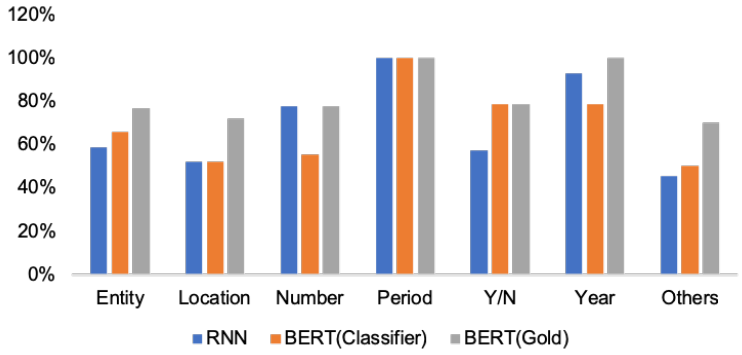

Figure 1: Subset error analysis

Our analysis shows that BERT with gold paragraphs performs predominately well on all 7 categories. BERT with classifier outperforms RNN on "Entity", "Y/N" and "Others". Particularly, we notice that BERT with classifier achieves the same performance as BERT with gold paragraph on "Y/N" type questions, which indicates that our classifier has been successful in selecting supporting sentences for comparison questions. Also, we find that the classifier improves BERT's performance on a few questions, which indicates that de-noising the input for BERT may generate positive effect in the setting of long context.

BERT is well-known for being able to detect long-range dependencies. The stack of bidirectional self-attention layers largely reduces the operation time for sequentially executed operations and makes it easier to learn long-range dependencies because each token is connected to each other token. This is seen when looking at the BERT with gold paragraphs (or in tasks like SQuAD), as it outperforms an RNN-based model in recognizing the bridge entities and figuring out the semantic relationship. However, without the goal standard paragraphs, the BERT may struggle in dealing with noise. Because of the large number of connections, which are not all sequential, it will be thrown off by irrelavent data more strongly than a sequential model. Thus, when are classifier works suboptimally, the performance of the BERT model declines significantly. Thus, in general, BERT may not be optimal to multi-hop reasoning tasks because these tasks usually involve hopping over portions of data that are noisy and irrelavent–including their attention is detrimental to the model.

RNN performs well on "Number" and "Year" categories, whose answers are considered to be relatively "unique" in the context, which indicates the model's ability in building links among multiple sentences/ entities. As we look closer at the predictions, we find that the model is able to make reasonable guess even if the answer is inaccurate. However, the performance deteriorates on "Entity" questions. Compared to BERT, the RNN

model is good at handling noise, while it is not robust enough for all type of questions. Meanwhile, the RNN is not able to answer "Y/N" questions effectively. One possible reason is that our loss weight in prediction layer among "Yes", "No" and answer span is sub-optimal. Further fine-tuning the loss weight may improve the result.

For example, a snippet is shown as *Blackfin is a family of processors developed by the company that is headquartered in what city?*

Arm Holdings (Arm) is a British multinational semiconductor and software design company ... Headquartered in Cambridge, United Kingdom, its primary business is in the design of Arm processors (CPUs) ... Analog Devices ... is an American multinational semiconductor company specializing in data conversion and signal processing technology, headquartered in Norwood, Massachusetts ... The Blackfin is a family of 16- or 32-bit microprocessors developed, manufactured and marketed by Analog Devices

The answer to this question is Norwood. The result from the RNN is Cambridge, and the result from BERT is Norwood. In the example given, there are more than five companies and corresponding headquarters, and it is quite reasonable for the model to pick up Cambridge because it appears a few times and it headquarters a semi-conductor company. In order to pick up Norwood as the correct answer, the model needs to use "Analog Devices" as a bridge to link Blackfin and Norwood. Presumably, the multi-head attention structure of BERT will be able to link the underlying information with the bridge when necessary information is provided by the classifier. On the other hand, the question will impossible to answer by the BERT even for humans if any of the necessary information is absent, which, unfortunately, is often the case currently.

5 Conclusion

Overall, through our experiments, we have been unable to significantly improve the baseline through either the BERT model or our RNN structure. In the BERT case, we believe this is caused by the inherent setup of the context, which contains large amounts of noise that confuse the BERT model's bidirectional self-attention that connects all tokens. The connection of multiple related sentences is not exactly where BERT struggles; rather, it is having to use a context which has noise. Without the noise, BERT could be a viable model to implement–it is able to connect related sentences to each other. For the RNN, we have found that more tuning of the model is needed. Additionally, although it may be more effective at handling noise than BERT, without noise, its maximum capability is probably lower than BERT because of its inability to make long distance connections. Thus, there is an inherent tradeoff between RNN's ability to handle noise and the BERT's ability to answer questions given a context. Ideally, we can train a perfect classifier for BERT, but given that impossibility, more work needs to be done on both pipelines.

6 Future Work

Co-train classifier and BERT: At this stage, we use a pipeline method for our BERT-based model, where we select relevant sentences using our classifier first, and then feed them into the pretrained BERT. The precision and recall of our classifier cap our overall performance. As the next step, we believe linking the models together and co-training the two components may improve our performance.

BERT model configuration: We choose the most basic BERT model (BERT-small-uncased) for our experiments and only fine-tune it for 3 epochs. We believe that replacing the basic BERT with a large one and fine-tuning it for more epochs will further improve the performance. It may also be helpful to re-train BERT from scratch for a larger context length, so that the classifier would not be needed. However, this is not guaranteed to help because as we have shown, BERT has trouble dealing with noise.

Hyperparameter fine-tuning: In this project, we spend majority of our time trying different methods, and not all of our models are fine-tuned properly. Also, we conduct most test on the 10% dataset, whose results fail to generalize well on the full dataset. In the future, we believe fine-tuning the models and using a larger subset for tests may improve the results.

Appendix

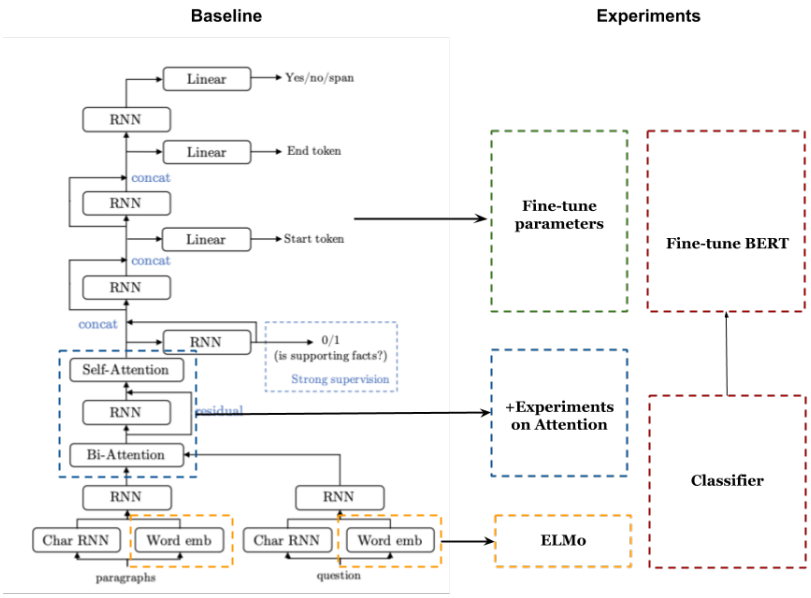

Figure 2: Model structure

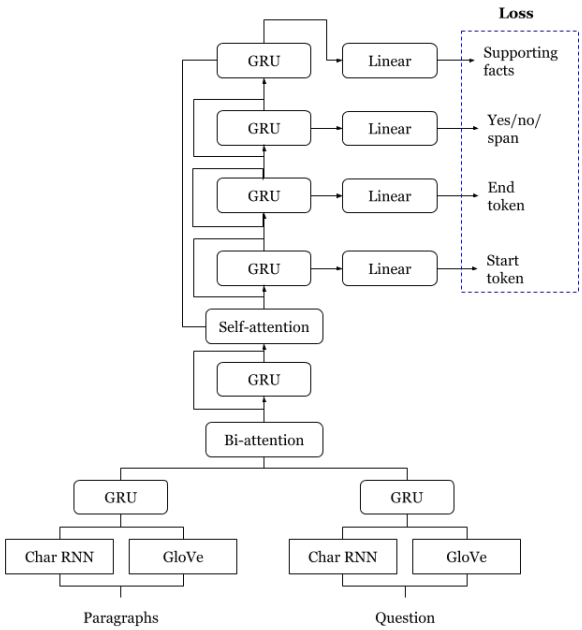

Figure 3: Classifier

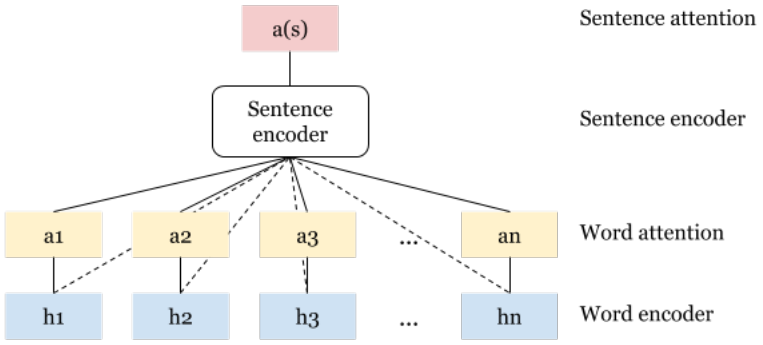

Figure 4: Existing work on sentence level attention

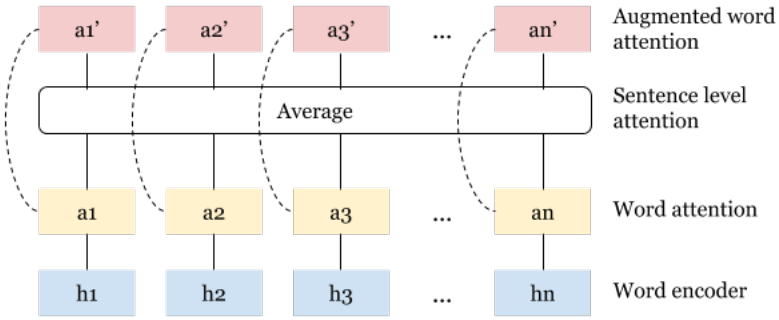

Figure 5: Our approach to incorporate sentence level attention

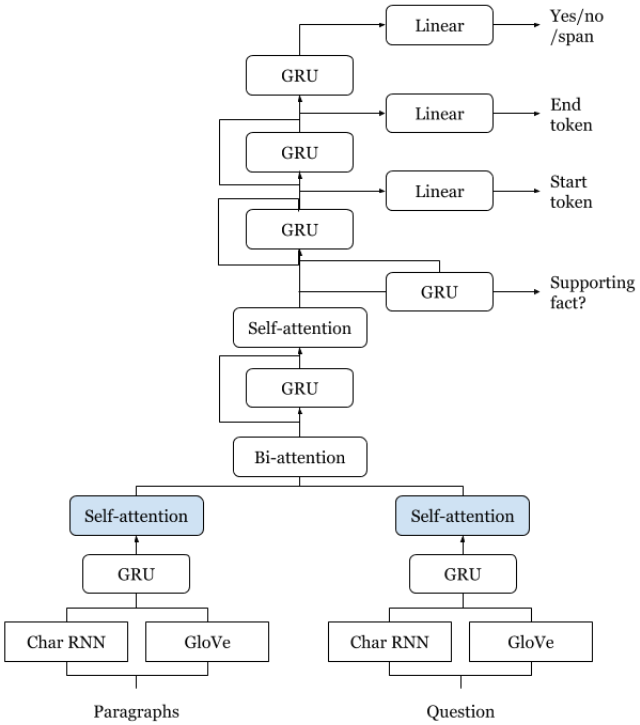

Figure 6: Model structure-RNN

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2014). arXiv: 1409.0473. URL: http://arxiv.org/abs/1409.0473.

[2] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[3] Matthew E. Peters et al. "Deep contextualized word representations". In: *CoRR* abs/1802.05365 (2018). arXiv: 1802.05365. URL: http://arxiv.org/abs/1802.05365.

[4] Min Joon Seo et al. "Bidirectional Attention Flow for Machine Comprehension". In: *CoRR* abs/1611.01603 (2016). arXiv: 1611.01603. URL: http://arxiv.org/abs/1611.01603.

[5] Zhilin Yang et al. "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.

[6] Zichao Yang et al. "Hierarchical Attention Networks for Document Classification". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, 2016, pp. 1480–1489. DOI: 10.18653/v1/N16-1174. URL: http://aclweb.org/anthology/N16-1174.