Multi-hop Question and Answering

Ankush Jain
New York University
Brooklyn, NY
aj2885@nyu.edu

December 16, 2020

1 Introduction

Question and Answering is one of the essential problems in Natural Language Processing with a wide number of applications such as information retrieval systems, visual QA, social media analysis etc. Lately, we have seen an incredible amount of advancement in the research on this problem. The primary reason for this growth can be attributed to the readily available high compute infrastructure, advancements in deep neural networks, availability of large-scale QA datasets, and the emergence of the highly parallelizable transformer models. The availability of all these resources has made it easier to develop the models that can tackle the simple question and answering problem. These simple QA models are single-hop and extractive.

On the basis of the answer generation method, the QA models can be categorized as extractive (where a particular phrase serves as the answer) and abstractive (where the systems generates its own answer by summarizing the information). The extractive QA models can be further categorized as single-hop and multi-hop. Single-hop extractive models have a single phrase within the context comprehension that serves as the answer [1]. Whereas the multi-hop extractive models can gather information from multiple context comprehensions and generate answer with some degree of logical inference or information aggregation [2].

Single-hop	What Netflix series is produced by Joe Swanberg and has Zazie Beetz?	Find series
Multi-hop	What Netflix series, produced by Joe Swanberg, had an actress best known for her role as Vanessa on Atlanta?	Find actress → Find series

Table 1: An example of multi-hop QA

In the above examples, for the single-hop question we will be given the context passage about the question and we would simply zero in on the phrase that answers the question (series name). Whereas for the multi-hop question, we will have to first find the answer to the latter question (actress name – Which actress is best known for her role as Vanessa on Atlanta?) and then we will answer the initial question on the basis of the answer for the latter question (series name - pivoted on the queried actress name).

The go-to model for single-hop QA is BERT (and its derivatives) and it is not suitable for multi-hop QA right out of the box [8]. The idea of multi-hop extractive QA is still in a very nascent stage so it is still being actively researched. In this work, we have trained a model on the Longformer architecture [5] in an attempt to solve the multi-hop QA problem.

2 Related Work

After an initial study of some of the leading research on multi-hop QA, a few common approaches were observed in the model architectures for this problem. One approach is to train the attention mechanism to naturally learn the multi-hopping. [3] The major challenge for this approach is to develop models that have larger attention window as compared to models like BERT (512 tokens). Architectures like Longformer (AllenAI) and ETC (Extended Transformer Construction) (Google) are well equipped to address this problem [5][7].

Next common approach is to break down the complex process of multi-hop QA into simpler steps, then use a single model or an ensemble of models to get intermediate results for these steps and collate them to find the overall answer. The intermediate steps mostly involve tokenizing the input data by using the concepts of graph theory and NLP techniques such as masked language modeling [2][4]. The TAP2 (Translucent Answer Prediction) model by IBM is a good example for such an approach.

Lack of data sets plays another role in the lack of mainstream popularity of research and development of multi-hop QA problem. At the moment there are mainly two datasets for multi-hop QA training. These are namely: HotpotQA[1] and WikiHop[6]. These are reading comprehension datasets where each document is composed of a triplet of the question, answer and context comprehension. Additionally, HotpotQA also provides supporting evidence for the results for explainability.

In this work we are utilizing the Longformer architecture which follows the approach of training the attention mechanism to naturally learn the multi-hopping. And, we are using the HotpotQA dataset to train and benchmark the model performance.

3 Dataset & Model Evaluation

3.1 Dataset Description

The HotpotQA [1] data set is used to benchmark the QA model. There are two versions of the dataset: full-wiki and distractor. We will use the distractor version. The dataset has 112k samples. Each sample has a question, the true answer (label), the supporting evidence (for explaining the answer), and has 10 sets of context sentences. Each question can be answered by using at most four context sets.

These examples are split into a training set of 90k training samples and 22k validation samples. The data is further divided into three levels of difficulty. In this work we have utilized the training samples of easy and medium difficulty (74k samples) to build the QA model. Since, the validation data comprises of only hard level samples, it is not being used. 10% of the training samples are used for model evaluation and testing.

Column	**Value**
_id	5ab28f6c5542993be8fa994a
question	What state is an Iroquois named town with a privately owned public airport in?
answer	New York
context	['Cindy Guntly Memorial Airport (FAA LID: 62C) , originally "Hunt Field", is a privately owned...'] ['"Privately owned public space (POPS), or alternatively, privately...']
type	bridge
level	medium
supporting_facts	['Skaneateles Aerodrome', 0], ['Skaneateles (town), New York', 0], ['Skaneateles (town), New York', 2]

Table 2: An example from the data set

3.2 Evaluation Method

HotpotQA provides its own evaluation criterion that assesses the answer predictions [1]. We have also used a similar criterion to evaluate the model performance. We obtained the character level "exact match score" of the model (similar to accuracy) by determining whether each predicted answer matches strictly with the true answer or not. The scores were calculated in case-insensitive manner. Each sample got a score of 0 or 1 and then we calculated the average accuracy across all the samples. It was the primary metric for the model.

Additionally, we also obtained the metrics like precision, recall, and f1-score by performing a token level match between the prediction and truth. These were also case insensitive but were more relaxed than EM and gave us a more generalized picture of the results. Refer examples below for a better idea.

The HotpotQA authors have also maintained a leaderboard of the various models trained on the dataset (by the community). At the moment, the best exact match score (EM) on the leaderboard ranges from 45.6% to 70.15% and the F1-score range from 59.02% to 83.02%.

	Exact Match Example	**F1, Precision & Recall Example**
truth	United States of America	The United States of America
pred	Pred 1: united states of America Pred 2: united states	THE United Kingdom
score	EM 1 = 1.0	TP = 2 (the, united); FP = 1 (kingdom); FN = 3 (states, of, america)
	EM 2 = 0.0	Precision = 0.67; Recall = 0.40; F1 = 0.5

Table 3: Examples to illustrate the evaluation methods

4 Model Architecture

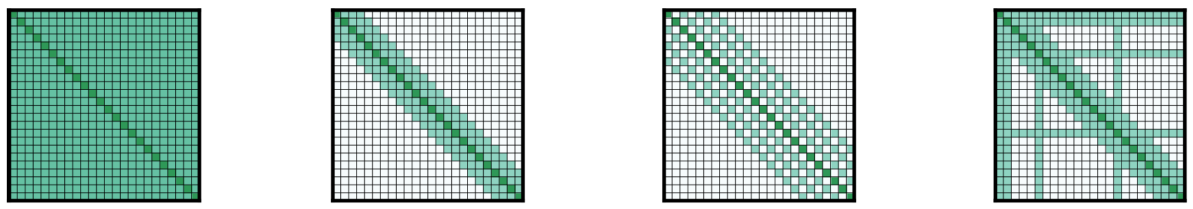
4.1 Problems with BERT-based Transformer models

It is the attention mechanism that lies at the core of the transformer models. It allows the model to enrich each token representation with information from the different parts of the input sequence. The primary limitation of BERT-based Transformer models lies in their inability to process long sequences of text due to the nature of their self-attention mechanism [5]. The maximum sequence size for BERT is 512 tokens and any sequence longer than this will be truncated [8]. So, even if we choose to chunk the long text into sets of 512 tokens, then we lose the interdependent information between the chunks. Moreover, the compute time of the self-attention mechanism scales quadratically with the sequence length.

These limitation causes the major obstacle for multi-hop QA as it relies on much longer sequences (context comprehensions) to get the answers. So, if we don't increase the max sequence size then the information will be lost and if we chose to increase the sequence size, then the compute time increases drastically.

4.2 Model Description

The Longformer is an improved Transformer architecture that is created by Allen AI and was built over the RoBERTa model [5]. It is designed specifically for processing the long documents of text. The Longformer model uses different variations of sliding window for the attention mechanism at different layers. This decreases the quadratic compute time of attention (like in BERT based models with full-attention) to linear time and also allows an increase in the maximum sequence length to 4096 tokens, an eight-fold increase over BERT.



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 1: Comparing full self-attention pattern with sliding window of attention patterns of Longformer [5]

The Longformer uses three different variations of sliding windows. The lower layers of the model utilize the regular sliding windows of size w as the lower layers must attend to all the local information besides a token. The computation cost of attention mechanism for these units is of the linear order $O(n \cdot w) \approx O(n)$.

The higher layers of the model utilize the dilated sliding windows. These dilated windows of size w have a gap of d units between them. This design choice is taken because at a higher level, we want to spread out the attention to more tokens in the vicinity. The computation cost of attention mechanism for these units is of the linear order $O(n \cdot (w + d)) \approx O(n)$.

There is also a special type of sliding window that is called the global + sliding window. These sliding windows are of size w but they also have s number of special units (tokens) in between that can attend to all the units within the sequence. The design of this sliding window allows us to answer the kind of questions that require some sort of logical inference (yes or no type of questions) or aggregation of information (like average or total number of something). The computation cost of attention mechanism for these units is of the linear order $O(n \cdot w + 2 \cdot s \cdot n) \approx O(n)$.

These specially designed sliding windows provide a balance between efficiency (smaller attention window allow for faster computation) and performance (larger attention window allows for greater representational power). And these qualities can be utilized to solve the problem of multi-hop QA.

5 Model Training

The extractive question and answering problem is a multi-label classification problem at its core. We aim to classify the input tokens for the two labels viz. **start token** and **end token**. The start token is the first token of the answer phrase and the end token is the last token of the answer phrase. The diagram below illustrates the model architecture for training a transformer model for question and answering downstream task.

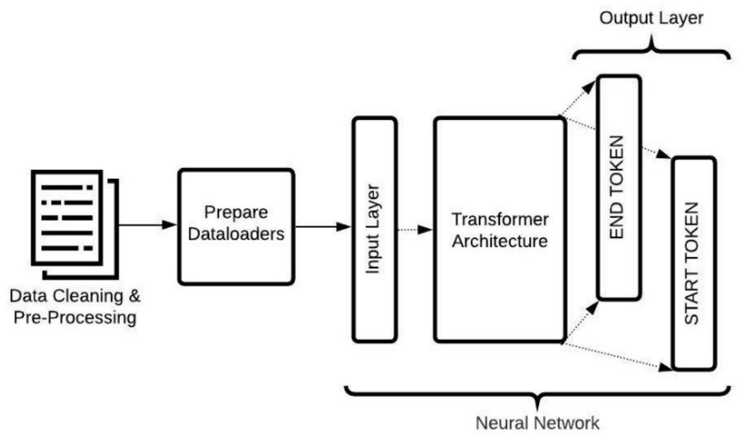


Figure 2: Model pipeline to illustrate data flow

The flow of our model training pipeline is as follows. We first clean and pre-process the JSON of our multi-hop dataset. The text is made case insensitive, whitespaces are normalized, and stop words are removed. We also apply an additional layer of data pre-processing that is discussed in section 5.2. Then, we tokenize the question and context, and concatenate them by inserting a [SEP] token in between. The total number of tokens must not exceed the maximum length of the tokenizer. Any documents that are below this length are padded with [PAD] token and any documents that are above the **max length** are discarded. This strategy is used to avoid the scenarios where the answer is present in tokens beyond the max length limit.

Next, we locate the start and end token index of the answer within the context so that they can act as the two labels for the classification task. Then, we create our data loaders of the selected **batch size** by creating a data set class with the input tokens (context and question), attention mask, start token index of the answer, and end token index of the answer.

Then we define our model architecture. We just define a custom input layer and an output layer according to our problem, and in between we keep the Longformer model. The input layer is created to process a matrix of size **batch_size * max_length**. And the output layer gives two matrices of the same size as the input layer. Each of the start and end matrix is composed of vectors of size **max length** for each of the sample within the batch. Each vector has probability for each token to be the start or the end index of the answer. Lastly, we apply softmax function in order to obtain the one-hot-encoded vectors for the start or end of the answer. We then calculate the cross-entropy loss of the predicted start and end with the true start and end tokens and backpropagate the loss with the Adam optimizer.

5.1 Loss Function

We have used the cross-entropy loss function for this work. Our problem is a classification problem and we need to find the right label (or token in our case) for the start position as well as the end position of the answer. For each of the possible token from the input text (context), we are finding the probability of that token to be our end or the start token for the answer and then we apply softmax for normalizing the probability distribution.

The Sparse Categorical Cross Entropy loss function from Keras is used in particular for this problem. As it is highly optimized for problems with a large number of categories (context tokens in our case). In our MHQA problem, the start token and end token output vectors for a single sample would be very sparse. They would be of the size of tokenizer **max length**, and for a longformer model with 4096 max length, there would be the same number of classes. We have to choose only one token out of 4096 for the start position and another for end position.

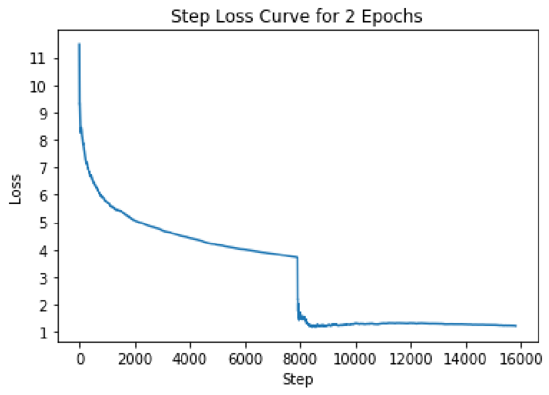


Figure 3: Step Loss Curve for 2 Epochs

In the above plot we have visualized two epochs of training step loss for around 8000 batches of training data with batch size of 8.We can see that in the first epoch (before the steep kink) there was steady decrease in loss and after the kink, the loss became stagnant (or even showed a minor increasing trend). This step loss curve highlights the general practice of limiting the epochs for transformer model training to a range of 2 to 3 epochs.

5.2 Hyperparameters

The Longformer multi-hop QA model was trained with cross entropy loss function and the Adam optimizer. It can be tuned for all the potential hyperparameters such as number of epochs, batch size, weight decay, warmup steps, learning rate etc.

Further, the model specific hyperparameters such as the sliding window size, the dilation gap size, and the number of special units from global + sliding window can also be tuned. And the tokenizer max length are also potential candidates for the hyperparameter tuning.

However, for this work, only the epochs, learning rate, batch size and the tokenizer max length hyper parameters were experimented with. No cross-validation with a separate validation set was performed due to the very high compute requirements of the model, and the long run times for epoch computation ($\approx 3hrs/epoch$). Instead, the training set and the validation set were only used to tune the model because solid differences in the quality were observed when the hyperparameters within the logistically feasible range were experimented with during the process. So, one could tell the right parameter choices even by eyeballing.

Parameter	**Selected Value**	**Experiment Range**
Epochs	2	2; 3; 4
Batch Size	8	4; 8; 16
Learning Rate	3e-5	1e-4; 1e-5; 5e-5
Tokenizer Max Length	1024	512; 1024

Table 4: Hyperparameters

The epoch size of 2 or 3 is generally recommended in finetuning transformer models for downstream tasks such as Question and Answering. And even in practice, a deterioration of the model quality (increase of loss and decrease in accuracy) was observed for epochs greater than 3. So, early stopping was used to determine the epochs. The learning rate 3e-5 and 5e-5 can be cross validated in future with sufficient time and resource as the results for these values were very close.

The real challenge for the training of this model was experienced in balancing the GPU resources for batch size and tokenizer max length. Only a maximum batch size of 8 and the tokenizer max length of 1024 was possible on a 32GB Nvidia Tesla V-100 GPU. Hence, the full range of 4096 tokens max length of the Longformer model was not used and it became the reason for the need of an additional pre-processing step as discussed in the section 5.2. About 30% of raw data was above 1024 tokens.

5.3 Primary Challenge

During the initial trials of model training, it was realized that due to the hardware limitations (32GB GPU), we would have to make a choice between increasing the batch size of model training or the tokenizer max length. It was not possible to increase both of them and train them on the 32GB GPU. So instead, we found the best trade-off between the two at a batch size of 8 and max length of 1024. Any batch size less than 8 wouldn't have converged the weights well and any tokenizer max length of less than 1024 would have defeated the objective of using Longformer.

The next challenge was to increase the number of eligible training samples. An eligible training sample would be the one whose answer lies in the first 1024 tokens of the context comprehension. Any training sample whose answer lies beyond the 1024 tokens limit would be rendered useless for training. The graph below shows the token count distribution for the default dataset. We have sorted the samples in increasing order of the total counts of the context tokens and we can see that, by default, only about 52k samples can fit in a tokenizer max length of 1024. This left about 30% data unused in training.

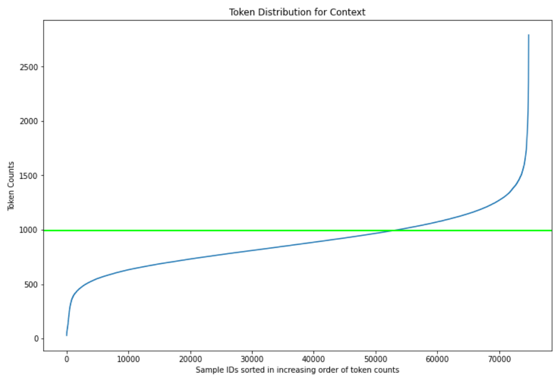


Figure 4: Token distribution before pre-processing

To counter this problem, we added an additional pre-processing step for decreasing the number of context documents and increasing the number of eligible training samples. We performed a tf-idf and character level n-grams based filtering by finding the similarity between the question text and the context documents. We use 4-, 5-, and 6-character window sizes for the n-grams for robustness. Then we keep the top 4 most similar context documents and use this dataset for training (because a maximum of 4 hops are used to answer a question in the data set).

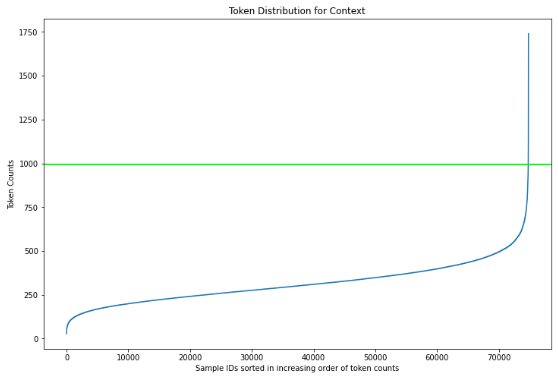


Figure 5: Token distribution after pre-processing

As we can see, after the additional pre-processing step, almost all the data samples can be used for model training (almost 99% data). However, there is a risk involved with this process as well. We can still lose out important documents that are required in QA as we are only using a lexical similarity criterion. Semantic similarity methods like NER or pre-trained word embedding based similarity scores might improve the candidates after context document filtering.

6 Results

We have trained the "allenai/longformer-base-4096" base model from scratch for multi-hop QA on the HotpotQA dataset. It was taken from the Hugging Face model repository. The initial baseline model for the project was created with the official single-hop QA model "allenai/longformer-large-4096-finetuned-triviaqa", trained by Allen AI on the TriviaQA data set. The purpose of selecting a single-hop QA model for baseline was to observe the amount of improvement that we could see when a dedicated model for multi-hop QA was trained.

6.1 Prediction Counts

The following results illustrate whether the model was able to predict any answers at all for a sample or not. This count does not consider the correctness of the answer. It just concerns with whether an answer (correct or incorrect) was generated by the models or not. We can observe that our custom trained model Longformer-mhqa along with the pre-processing steps in section 5.2 was able to produce 189 more answers.

	Easy (Predicted/Total Samples)	**Medium (Predicted/Total Samples)**	**Overall (Predicted/Total Samples)**
Longformer-mhqa	1720/1797	5440/5681	7160/7478
Longformer-baseline	1674/1797	5297/5681	6971/7478

Table 5: Prediction counts

6.2 Exact Match & F1 Score for Difficulty Levels in Dataset

The next set of results show the Exact Match (accuracy) and F1 scores for the different difficulty levels of data samples with the trained Longformer-mhqa model and the untrained Longformer-baseline. We have taken the easy and medium difficulty data samples for training and prediction.

	Easy		**Medium**		**Overall**	
	EM	**F1**	**EM**	**F1**	**EM**	**F1**
Longformer-mhqa	0.4785	0.5807	0.4045	0.4607	0.4223	0.4896
Longformer-baseline	0.3099	0.3949	0.2591	0.3163	0.2713	0.3352

Table 6: EM & F1 comparison for the easy and medium data samples

Upon observing the above results we can see that the custom trained model has seen improvements for both the easy as well as medium data samples in both the metrics. We can observe that for the easy samples (which are mostly composed of single-hop questions), out trained model has seen a lot of improvement. Whereas for medium samples, the improvement is a little less in magnitude, EM Score has seen an improvement of about 15 percentage points.

6.3 Exact Match & F1 Metrics Comparison

This set of results present the Exact Match scores (accuracy) and the F1-score (with Precision and Recall) for the trained Longformer-mhqa model and the non-tuned Longformer-baseline model. Moreover, we are also comparing these longformer models with the baseline model created by the HotpotQA authors, and the top two submissions on the HotpotQA leaderboard.

	Exact Match (Accuracy)	**F1**	**Precision**	**Recall**
Longformer-mhqa	0.4223	0.4896	0.4924	0.5208
Longformer-baseline	0.2713	0.3352	0.3607	0.3507
HotpotQA-author-baseline	0.4444	0.5828	-	-
HotpotQA-leaderboard-1st	0.7024	0.8236	-	-
HotpotQA-leaderboard-2nd	0.7015	0.8302	-	-

Table 7: Comparison of the evaluation metrics for the models

Upon observing the above results, we can see that an improvement in the exact match score is seen with the custom trained longformer multi-hop model over the initial longformer single-hop baseline. We can also observe that the F1, precision, and recall scores are a bit better than the exact match score. This is because the

EM score is much stricter and gives a positive value of 1 only when all the characters for the answer match. Whereas, the f1, precision and recall criterion give a score even for partial matches at token level.

Recall is slightly greater than precision so we can see that our model is able to return more than half (0.52) of the actual correct answers from the whole data set. However, the precision is slightly lower than half (0.49), so out of all the answers generated, the correct answers make up only 0.49 of the retrieved results. Hence, a lot of irrelevant or wrongs answers are also being generated.

However, before starting this project, the aim was to at least cross the exact match score of 0.4444 which was set by the model created by the authors of the HotpotQA dataset (HotpotQA-author-baseline model). And we were unable to achieve this desired goal. Though, our results are pretty close to the author's baseline.

We can also see that even the most competitive models submitted by the major corporations like Google, Allen AI, Samsung, IBM etc. on the HotpotQA dataset have been able to achieve a maximum exact match score at around 0.70. Overall, the model exact match score (accuracy) ranges from 0.4560 to 0.7015 and the F1-score ranges from 0.5902 to 0.8302 on the leaderboard. All these facts highlight the overall complexity of solving the multi-hop question and answering models.

The reason for this inability is due to the small batch size of 8, and the pre-processing step of reducing the number of context passages so as to fit the training data into the tokenizer max length of 1024 tokens.

We have also observed that these hyperparameters do make a difference because even the Longformer authors also submitted their model trained on the data set and are currently placed at 9th position (EM=0.68; F1=0.8125). They used a batch size of 32 and the tokenizer max length of 4096 [5].

7 Conclusion

After working on this project, we can clearly identify all the reasons for a lack of mainstream popularity of this problem in the AI and NLP community. As compared to other NLP tasks, the multi-hop QA problem requires appropriate model architectures for processing long texts, greater amount of computing resources (GPU and RAM), and it also requires more data sets to promote interest in solving this problem.

7.1 Technical Aftermath

With regards to the model trained by us during this work, the following can be concluded. The current model is not sufficiently trained due to the small batch size and small tokenizer max length. Batch size of at least 16 or 32 must be used along with a larger max length (¿1024). So, one of the next steps in improving this model would be to train with a higher batch size and a larger max length of tokenizer.

We should also try to avoid the additional pre-processing step that reduces the number of contexts to four (a number determined by the maximum number of hops for this dataset). It is possible to do this for the Longformer architecture with the HotpotQA dataset as the max token length for raw data is around 2800 as observed from Figure 3. Or, until the time we have resource limitations, we should at least increase this number so as to accommodate more context candidates and avoid missing any crucial text for training as observed from Figure 4.

Also, there are two available variants of the longformer architecture, namely, the longformer base with 148 million tunable parameters and the longformer large with 430 million tunable parameters. The larger model will probably generate better results. So that can also be tried out if we have enough computing resources.

And lastly, we must perform cross-validation to determine the best learning rate for training the model. The accuracies at 3e-5 and 5e-5 were very close so cross validation will give us more stable learning rate out of the two. But, we will also have to reduce the training data size in order to create a dedicated validation set for model evaluation and keep the test set isolated from training.

7.2 Ethical Aftermath

Overall, we have trained a model that performs fairly for the amount of resources invested and is almost neck-to-neck with the model created by the data set authors. But, as this model isn't sufficiently trained, it won't be wise to use it in any real world application.

Ideally, we should also be testing this model for explainability of results. The HotpotQA authors have provided supporting facts with this data set as well, so we can test the model for explainability with them.

We should also determine if there exists any social biases or fairness related problems with this model or the data set. An interesting observation about bias in one of the competing data sets for QA was made by the HotpotQA authors [1]. For the WikihopQA data set, they discovered that the data samples had a bias for the USA and had very few samples that talked about the topics from other countries in the world. A model trained on such a data set will perform poorly on questions about other countries and cultures. We should perform such tests for our model as well so as to determine the right use-cases for its application.

Source Code

GitHub
https://github.com/ankushjain2001/Multi-Hop-Question-Answering

Colab Model Inference
https://colab.research.google.com/drive/10B71qnh9oAkeWJ7x71dTOABJh92KxHGs?usp=sharing

Colab Baseline Inference
https://colab.research.google.com/drive/1Yn7ARYrp3JGKNeBrTnuL2XMMvVkpQwto?usp=sharing

References

[1] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, Christopher D. Manning. (2018). HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering.

[2] Jifan Chen, Shih-ting Lin, Greg Durrett. (2019). Multi-hop Question Answering via Reasoning Chains.

[3] Zhao, C., Xiong, C., Rosset, C., Song, X., Bennett, P., Tiwary, S. (2020). Transformer-XH: Multi-evidence Reasoning with Extra Hop Attention. In The Eighth International Conference on Learning Representations (ICLR 2020).

[4] Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, Douwe Kiela. (2020). Unsupervised Question Decomposition for Question Answering.

[5] Iz Beltagy, Matthew E. Peters, Arman Cohan. (2020). Longformer: The Long-Document Transformer.

[6] Zhilin Yang and Peng Qi and Saizheng Zhang and Yoshua Bengio and William W. Cohen and Ruslan Salakhutdinov and Christopher D. Manning (2018). HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question AnsweringCoRR, abs/1809.09600.

[7] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, Li Yang. (2020). ETC: Encoding Long and Structured Inputs in Transformers.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.