Question Answering Classification With Expanded Datasets

Stanford CS224N custom Project

Rain Juhl
Department of Computer Science
Stanford University
rjuhl@stanford.edu

Eric Feng
Department of Computer Science
Stanford University
ericfeng@stanford.edu

1 Key Information to include

- Mentor: Angelica Sun
- External Collaborators (if you have any): None
- Sharing project: No

Abstract

As machine learning models become larger and more complex, the time and resources spent on creating and forming these systems is becoming more of an issue. Ensuring that large models that take large amounts of time and resources to train are achieving the best possible results and generalizes well is more important than ever. Many times when solving this issue, it can come down to choosing the correct data. Previous studies have shown that models that perform well on question answering tasks on the baseline SQuAD dataset have a poor capability to generalize to other datasets. In our project, we explore one possible reason that this may occur. By artificially padding SQuAD contexts with additional unnecessary but relevant data and training a range of models on the original data and the modified data, we show an intuition that contexts that are not concise and remain unpruned generally diminish the capability of question answering models.

2 Introduction

According to a study conducted by Cognylitica in 2020, data preparation engineering tasks represent over 80 percent of the time consumed on most AI/ML projects. With so many resources being spent on gathering and processing data for down stream tasks, it is important that the data presented to a model's pipeline is optimal to avoid complications and spending additional resources on data processing. At the heart of the problem, we wish to explore the importance of choosing contexts when collecting data and inputs for a question answering task: Is it is more beneficial to allow models to pick up larger amounts of potentially relevant information from an expanded context, or is it important to prune contexts to only get what is necessary? In this paper, we utilize state of the art retrieval methods in order to artificially extend question answer contexts with additional relevant data from Wikipedia. We then train different models to evaluate the general trends of how these different models perform on the two contexts.

3 Related Work

In a paper presented in 2020 by Miller et al, question answering models trained on SQUAD had been shown to have poor generalization when fitting to data where distributions come from different domains than the original squad data. In general, the models would do much worse when fitted onto

the different data domains. In this paper, three additional question answering datasets are created from different sources: NYT, Amazon reviews, and the Reddit, and across the 3 domains there was an average drop of 3.8, 14.0, and 17.4 F1 points respectively. [1]. The paper presents that the out of distribution issue generalizability issue exists, however, it is unable to find a conclusive possibility for why the models do so poorly across the domains shifts, especially to Amazon and Reddit [1]. The paper leaves an open question to explore on these domain shifts. We explore one possibility of poor results across these domain shifts, specifically that the regular Wikipedia domain and the NYT domains are strictly regulated by editors which strictly prune data to be relevant to specific subjects, however Reddit and Amazon reviews are largely unregulated with contexts. Our work could give insight into a possible explanation as to why these models perform poorly across these two separate domain shifts.

4 Approach

The general pipeline of our experiment starts with the creation of extended dataset with the following retrieval method:

4.1 GRNN: Learning Retrieval Reasoning Paths

In the paper Learning to Retrieve Reasoning Paths Over Wikipedia Graph For Question Answering by Akari Asai et al they created a procedure for acquiring reasoning path for context utilizing the graph like link structure of Wikipedia [2]. They use a GRNN to find high probability reasoning paths given a question and some surrounding context, without the answer. A high level overview of their system an be seen in figure 1 below.

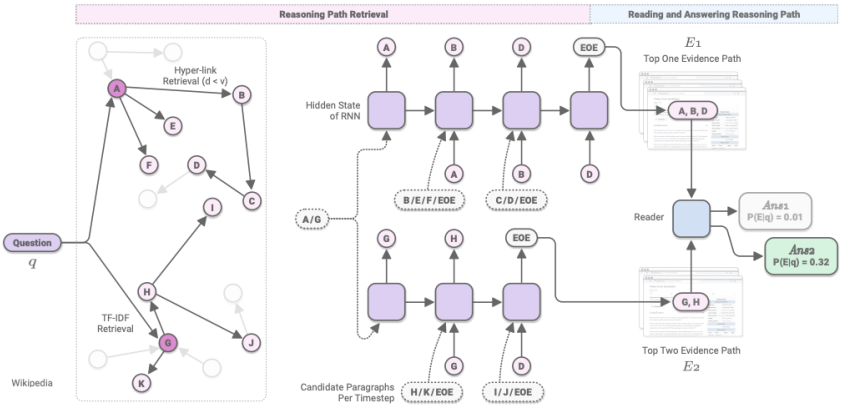


Figure 1: Overview of Retrieval Framework

The figure is broken down into two sections. The reasoning path retrieval and reading and answering reasoning paths. For the purposes of this project we only needed to focus on the reasoning path retrieval to acquire additional contextual information about a question. Reading the figure left to right you see that the first step in the process is setting up a graph of the Wikipedia link structure starting from the question. Afterwards, it then uses a RNN to choose the next path to be taken in the graph. The model additionally uses beam search to find candidate paragraphs and then returns the highest probability paths found by beam search [2].

Our analysis of the new data starts with 3 different models. Since our prepossessing leads to longer datasets, the three models we chose for evaluation are models that have the capability to work with varying amounts of context. As a starting point, we utilize the Bidirectional Attention Flow (BiDAF) model as described in Bidirectional Attention Flow for Machine Comprehension [3]. This model is a LSTM model, which is constrained by the linear interaction distance due to how RNN's are unrolled in linear structures. This restriction theoretically decreases the ability for the BiDAF to be able work with larger. We then explore pretrained Transformers as an alternative to the restrictive RNN model.

We start with the Bidirectional Transformer For Language Understanding as defined in BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, as it utilizes bidirectional self attention which allows for all words in a context to simultaneously attend to all words of a previous layer. This removes the restriction of linearity, allowing for more distant tokens in contexts to interact with each other [4].

We can see that self attention has a major bottleneck. When calculating the self attention, it comes from 3 separate vectors the Query vector Q the Key vector K and the value vector V from activation's. We can see,as per the paper, there is a term in the calculation of attention requires a matrix term that is of the size sequence length by sequence length [4]. The memory needed for large sequences due to this bottleneck restrains the length of the context that the model can work with which lead us to the final model which is the Reformer:

4.2 Reformer: The Efficient Transformer

The Reformer was pioneered in the paper Reformer: The Efficient Transformer and is a modification of traditional Transformers to achieve higher computationally efficiencies. It is able to both decrease memory requirements, such that long sequences (hundreds of thousands of tokens) can be trained on a single 16GB memory GPU, and achieves a faster time complexity of $O(L\log(L))$ than the tradition Transformer which is $O(L^2)$ where L is the sequence length [5].

To achieve reduced memory requirement the Reformer uses reversible residual connections. These connection allow for activation from a layer to be recovered by the following layer. A typical residual connection operates on one input and one output where $x \to y$ such that $y = x + F(x)$. However, reversible residual layers works with two inputs $(x_1, x_2) \to (y_1, y_2)$ and outputs such that $y_1 = x_1 + F(x_2)$ and $y_2 = x_2 + G(x_1)$. This configuration allows a previous layer to be recovered by doing the simple $x_2 = y_2 - G(x_1)$ and $x_1 = y_1 - F(x_2)$ operation.

To decrease time complexity Local Sensitivity Hashing (LSH) is implemented in the Reformer. The goal of LSH is to achieve a similar time complexity to local attention while getting a strong global attention approximation. LSH is motivated by the idea that softmaxes are dominated by their largest inputs. Therefore, in the equation $\textit{softmax}(QK^T)$ we will only need to attend to a a small subset of keys in k_i that are the closet to each query q_i. We can do this by using locally sensitive hashing. One way to do this through a hashing technique shown in figure 2. This figure is projected into 2 dimension and thus highly simplified but it gives a intuition of how this hashing technique works. It shows that after a random rotation the projected point are unlikely to share a hash unless the vectors that are being hashed are close to each other.

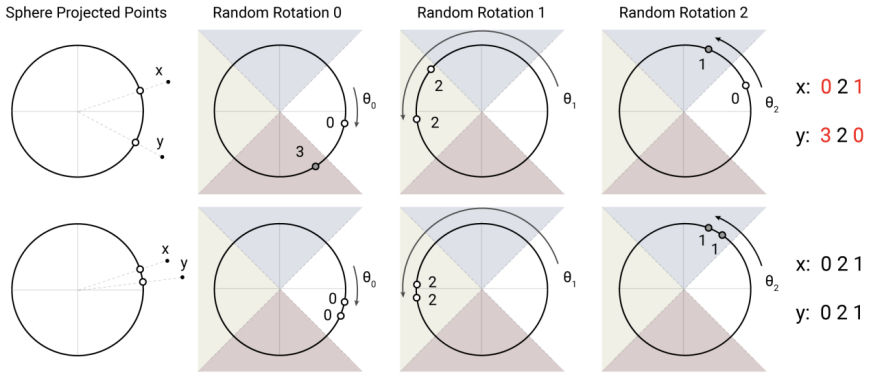


Figure 2: Local Sensitivity Hashing

Once hashed, each output is then sorted and chunked. Then each chunk performs local attention on each query in chunks. An overview of the whole process can be seen in figure 3.

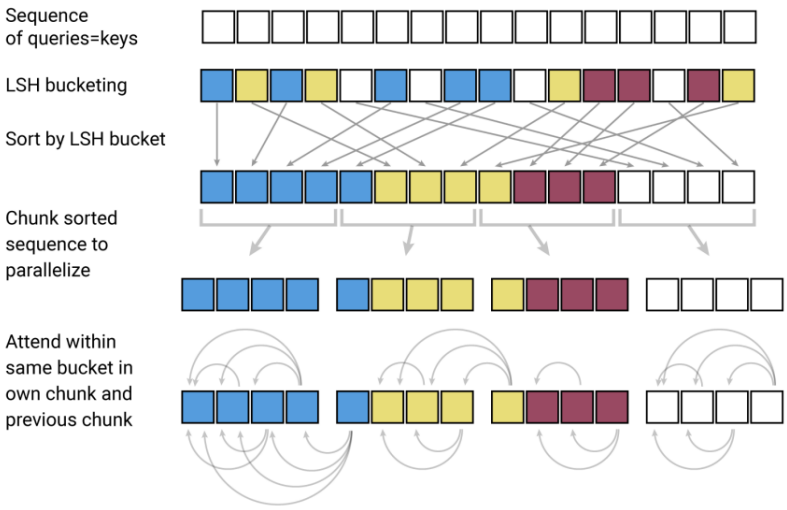


Figure 3: Local Sensitivity Hashing Overview

5 Experiments

5.1 Data

The base of our dataset is SquAD 2 which is composed of questions, answers, and their contexts. The paragraphs in SQuAD are from Wikipedia and the questions and answers were crowdsourced using Amazon Mechanical Turk. There are around 150k questions in total, and roughly half of the questions cannot be answered using the provided paragraph [6]. Since the additional contexts might generate the answers to the questions we did not include any samples where the answer was not contained in the context to be fair to both the baselines and experimental models. From there we further cut the dataset to 7200 train instances and 1500 test instances due to constraints on resources when generating additional contexts.

Then we ran this reduced dataset through the first half of the GRNN (the part that recovers reasoning paths) allowing the model to generate reasoning paths of articles that are very similar to the original context. With the new generated contexts, we add the paragraphs onto the original contexts and then randomly shuffle the data so that the models we train on cannot arbitrarily learn where the original context is located and arbitrarily make a decision off of the original context. When adding the data, we also scan the additional context for exact matches with the original labeled answers, and if any exact matches are found we add the newly found examples to the labels. For example, consider the following question and context. Note that the red text indicates the answer.

Question: In what country is Normandy located?
Context: The Normans (Norman: Nourmands; French; Normands; Latin Nomann) were the people who in the 10th and 11th centuries gave their name to Normandy, a region in France. They were defended from Norse ("Norman" comes from "Norseman") reader and pirate for Denmark, Iceland and Normandy who, under their leader Tollo, agreed to swear fealty . . .

After going the procedure to expand the context we get the following new context.

Context: [Additional paragraph 1 start] The Channel Islands are located in the English Channel, by Normandy France . . . [Additional paragraph 1 end][Additional paragraph 2 start] The western allies launched the largest amphibious invasion in history when they assaulted Normandy, located one the northern coast of France, on . . . [Additional paragraph 2 end][original context]

5.2 Evaluation method

To evaluate our results we used two qualitative metrics: E EM(Exact Match) and F1 scores. The EM score is starting metric that is computed by giving a prediction a score of one if is they are the exact

same as the ground truth and zero otherwise.

F1 scores are computed using the formula $F1 = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$ where TP is the true positive rate of words in the prediction span versus the ground truth span and FP and FN are false negative and false positive rates respectively.

5.3 Experimental details

5.3.1 BiDAF

The BiDAF was trained on both datasets for 25 epochs with the hyper parameters described in the original provided baseline implementation, the maximum context length was changed to allow all additional contexts to be added.

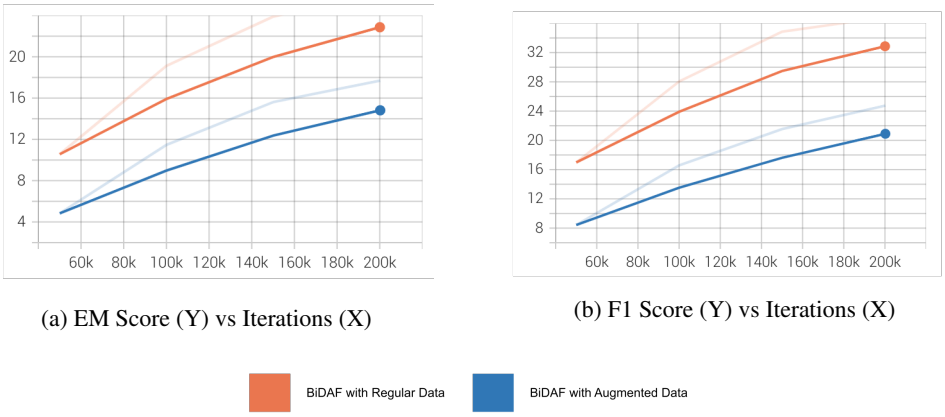
5.3.2 DistilBERT

DistilBERT was fine-tuned on both datasets for 15 epochs. It used adamW as an optimizer and had a learning rate of $5 \cdot 10^{-5}$ and weight decay rate of 0.01. Additionally, the dropout rate was 0.01.

5.3.3 Reformer

The Reformer was fine-tuned on both datasets for 15 epochs with adamW. The learning rate of $5 \cdot 10^{-5}$ and weight decay was set to $1 \cdot 10^{-4}$. While dropout was set to 0.6. These parameters seemed to combat the Reformer's tendency to over fit the best from the ones we tested.

5.4 Results



(a) EM Score (Y) vs Iterations (X)

(b) F1 Score (Y) vs Iterations (X)

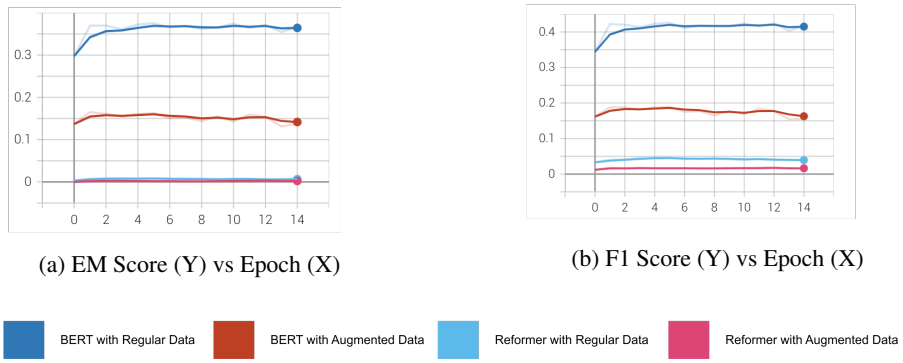
Figure 4: BiDaf Eval Metrics

Table 1: Validation Evaulation Metrics

Metric	BiDAF	BiDAF Aug	distilBERT	distilBERT Aug	Reformer	Reformer Aug
EM Score	0.227	0.148	0.366	0.138	0.007	0.002
F1 Score	0.368	0.248	0.418	0.155	0.039	0.016

6 Analysis

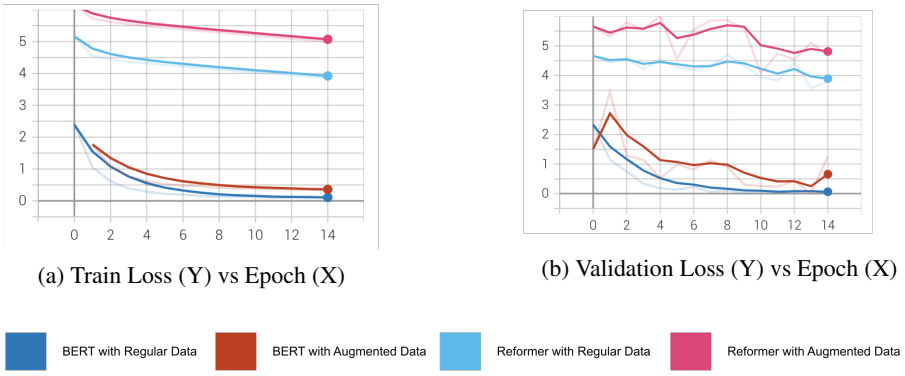
Numerically, based on figure 4 and 5 it is clear that additional context did not improve any of the models since both EM and F1 were consistently lower for the models with the additional context. However, looking at the train and validation loss plots from figure 6 it is clear that each model is



(a) EM Score (Y) vs Epoch (X)

(b) F1 Score (Y) vs Epoch (X)

Figure 5: Validation Plots



(a) Train Loss (Y) vs Epoch (X)

(b) Validation Loss (Y) vs Epoch (X)

Figure 6: Loss Plots

over-fitting. To help with this we played around with learning rate and dropout to help combat this. For the Reformer increasing the drop-rate helped but the plot shows that it is still over-fitting. The culprit is likely our small dataset. Because of this, it is hard to tell whether or not a large dataset with additional context would be helpful or not. However, we can conclude that for a diverse range of models and attentions mechanisms using additional context data with a small dataset is not beneficial.

Proposed by Miller et al [1], a way of determining a loss for a model's ability to generalize can be defined as the sum of the following three terms

$$\mathbf{L} = (\mathbf{L}_S - \mathbf{L}_D) + (\mathbf{L}_D - \mathbf{L}_{D'}) + (\mathbf{L}_{D'} - \mathbf{L}_{S'})$$

We define $\mathbf{L}_S$ and $\mathbf{L}_D$ to be the train loss and the dev loss for the original SQaUD dataset and we define $\mathbf{L}_{S'}$ and $\mathbf{L}_{D'}$ to be the train loss and dev loss for the expanded data set.

The first term is described as the Adjectivity Gap, which is a measure of test and dev loss loss for SQaUD data and the third term is known as the Generalization Gap which shows the difference between expanded dataset and the original dataset. With these two terms, we expect that if the test and train were selected properly these values should be close to 0. Lastly and most importantly the middle term is the Distribution Gap, which is the difference between training loss between models of different distributions, the difference shows how models act differently when natural distribution shifts occurs

Observing the graphs we can see that throughout the epochs, the train and validation losses remain relatively similar through out training and evaluation, However we see that for all 3 models, there is a substantial difference between the two validation losses. We can observe that the process of which we train question answering tasks for the original squad dataset does not generalize well when applying the same process to the expanded dataset, which implies that when having contexts that are unpruned and contain additional data will perform worse unless there are changes in how the model approaches the problem.

We also observed that many of the additional contexts were adding answers that were plausible solutions, we can see for example that a plausible solution in the examples provided in the sample section of the experiment would be "the northern coast of France", however the only solution labeled would be explicitly "France". Occurrences like these happened throughout our observation of data, thus these undetected false negatives in the data may have artificially deflated our evalutation metrics.

Now lets turn to specific models and their outputs to see if there are some qualitative indications about how this would with a larger dataset.

BERT performed the best on questioning and answering out of all the models but like the others it did better without the additional context. Although this difference in F1 scores is poor, there are still some promising results. BERT has sequence max of 512 tokens, so anything over that is truncated. This means often the original context get truncated. There are examples when the model still gets an F1 score over 0.5 despite this. indicating that additional context can be useful. In order for the transformer to approach a question answering task in large unpruned contexts, we see that additional methods would need to be used to fully interpret them.

From our results, it is clear that overall, the Reformer did very poorly. This is likely due to the fact the pretrained model is trained on a corpus of only 208,016 words as compared to other pretrained transformers such as BERT which has been trained on billions of words. Additionally, looking at baseline Reformer it is clear that it has a problem understanding numbers. Since a large portion of the dataset answers are numeric instead of learning how to pick out the correct ones the Reformer instead picks high numeric density spans. Here is example of an output from the Reformer, (38 °C). In the warmer months, the dew point, a measure of atmospheric moisture, ranges from 57.3 °F (14.1 °C) in June to 62.0 °F (16.7 °C) in August. Extreme temperatures have ranged from 15 °F (26 °C), recorded on February 9, 1934. For this example the answer was indeed a number (1936) but not within this context. One upside to the augmented Reformer it doesn't suffer as severely from this problem. One explanation for this is that the new contexts are not as numerically dense. It appears that the additional data added provided more variability in the train set and may have increased robustness.

7 Conclusion

We see that generally across the board, the control models outperform the experimental models across for all the different metrics. Although there are many factors that we are not accounting for, we see can build a general intuition that the pruning of contexts to only the most relevant text is important for question answering tasks. It appears that the different attention mechanisms did not substantial effect the models ability to pick up on the additional context, We hypothesise this to generalize to other models because of our diverse selection.

In further work we would like to explore a few possible avenues of research.

First of which would be to utilize semi supervised learning in order to generate new labels in the added contexts. This would allow us to decrease on the total amount of false negative classifications as described in the analysis.

We also see that both of the transformers quickly overfit on the data, reaching optimal validation metrics within the first two epochs. We would be interested in if and how these metrics change if the data didn't overfit so quickly

Finally, In order to get a better understanding of the trend on BERT, we could utilize a sliding window or chunking prepossessing technique to get a better understanding on how the BERT model acts on longer sequences.

References

[1] Benjamin Recht Ludwig Schmidt John Miller, Karl Krauth. The effect of natural distribution shift on question answering models. https://arxiv.org/abs/2004.14444, 2020.

[2] Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. Learning to retrieve reasoning paths over wikipedia graph for question answering, 2020.

[3] Ali Farhadi Hannaneh Hajishirzi Minjoon Seo, Aniruddha Kembhavi. Bidirectional attention flow for machine comprehension. https://arxiv.org/pdf/1611.01603.pdf, 2020.

[4] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/pdf/1810.04805.pdf, 2020.

[5] Anselm Levskaya Nikita Kitaev, Łukasz Kaiser. Reformer: The efficient transformer, 2020.

[6] The stanford question answering dataset. https://rajpurkar.github.io/SQuAD-explorer/.