Applying and Adapting the Reformer as a Computationally Efficient Approach to the SQuAD 2.0 Question-Answering Task

Stanford CS224N Default Project

Hugo Kitano
Department of Computer Science
Stanford University
hkitano@stanford.edu

Taide Ding
Department of Computer Science
Stanford University
tding@stanford.edu

Abstract

For the default project, we adapted the Reformer, an ostensibly more computationally efficient Transformer, for the SQuAD 2.0 task. This is an exciting task, as the Reformer has never been applied to question-answering in the literature. Along with the Reformer's novelty (it was published in January 2020), we are interested in the use of locality-sensitive hashing - the key idea behind the Reformer's memory-saving benefits - which greatly improves the speed of the attention computation. However, after testing two different models to see if the Reformer could outperform the baseline model BiDAF, we found significant flaws with the Reformer, most importantly how long it took to train. Although the Reformer is more memory-efficient, it takes significantly longer to train than a Transformer, which is itself quite slow to train. Ultimately, even per-epoch, the Reformer was not able to match the baseline model's statistics. Our studies do not indicate that the Reformer is a strong model for the SQuAD 2.0 question-answering task. **This project does not include any external collaborators and is not shared with any other class.**

1 Introduction

The Reformer, the Efficient Transformer [1] was published by Google as a conference paper at the ICLR 2020, and has already started conversations about the future of the Transformer model. According to the paper, the Reformer offers improvements to the Transformer's demands for large memory resources and multi-accelerator hardware setups while not significantly affecting overall performance. As a result, the Reformer can compute longer sequences faster than the Transformer with more efficient memory storage.

The classic Transformer model [2] has revolutionized natural language processing in recent years, largely replacing RNNs as the model of choice. Through parallelization of self-attention, it is able to retain memory over long sequences much better than RNNs, allowing it to be trained with larger amounts of data. However, this comes at the cost of immense memory storage. Although the total number of parameters to store per-layer is tractable, Transformers are multilayered in multiple respects, which is a chief culprit of its extreme memory usage. The Reformer implements a variety of fixes to alleviate this problem.

We are testing the ability of the Reformer to accomplish the SQuAD 2.0 task, which takes a question-context pair and outputs either the span within the context that answers the question, or answers "N/A" when the context does not contain the answer to the question. We will be comparing the Reformer to a baseline Bi-Directional Attention Flow (BiDAF) model [4], a model created specifically for the question-answering task incorporating embedding, RNN encoder, and attention layers.

As we proceeded to adapt the Reformer (a sequence-to-sequence model) for the question-answering task, we found some persistent problems that the paper failed to address. First, the sheer amount of

time that the Reformer takes to train is quite long: with default parameters on a standard NV6 GPU, each epoch would take 11 hours! Although the Reformer does save memory, it trains slower than a standard Transformer, mostly because of the time taken hashing, which greatly affected how much training we could do for this project. We tried two models: one that used only one embedding, one Reformer layer, and a masked softmax, and another one where we started with a full BiDAF model that replaced its RNN modeling layer with a Reformer layer. The first model failed more or less immediately. The second model's result was promising, but training was slow, with smaller gradients per-epoch compared to BiDAF.

Overall, we were underwhelmed with how the Reformer model works in practice with the question-answering task. The paper mentions that because of the memory savings, the Reformer can be easily run on a small GPU. Although this is true, the amount of time it takes to train is quite long, and we were unable to get results better than the baseline BiDAF.

2 Related Work

The Reformer is an extension of the Transformer. After years of recurrent neural networks dominating natural language processing, Transformers have become the model of choice by relying only upon attention. The idea is that with two simple stacks of encoders and decoders each with a self-attention and feed-forward layer, transformers can outperform RNNs, especially in cases with longer sequences despite using significant memory and time to train.

The Reformer makes some crucial improvements with regards to memory usage, enabling implementation with only 16 GB of memory and 1 GPU (its improvements will be described in detail in the Approach section). The Reformer has very similar performance compared to a classic Transformer on the imagenet64 and enwik8-64K tasks when evaluated using bpd (bits per dimension) as the metric of interest, despite its approximations during the hashing step.

We feel that though the Reformer paper is a strong contribution with potentially significant impact on the NLP field, it needs more work on the experiments section. Another metric other than bpd would have been a nice addition, and the authors have only tested the model on a limited span of tasks, with imagenet64 being a image classification task, and enwik8 being a text compression task. If it is not useful at other tasks compared to the Transformer, its range of use cases will be narrow. Likely, the niche use of the Reformer will be cases where the inputs are extremely long sequences, the only use case where the Reformer might be fast enough compared to its alternatives to be useful.

3 Approach

The Reformer has three crucial improvements [1] to the Transformer model in the usage of memory, as shown in Figure 1. Within the figure,

1. The black glasses refer to the layering of many encoders and decoders within the Transformer, increasing the number of parameters to store by a factor of N (the number of layers).
2. The red glasses refer to the self-attention computation. To compute dot-product attention on sequences of length L, we need to take the softmax of a matrix of size (L, L), which is an operation of complexity $O(L^2)$ in both time and memory. This causes longer sequences to often be prohibitively expensive.
3. The green glasses refer to the depth of the feed-forward network: since activations and weights must be stored for each layer for backpropagation, this accounts for a significant portion of overall memory use.

The Reformer offers critical improvements on these three inefficiencies of the classic Transformer model. Firstly, the problem of N layers of encoders/decoders is solved by using reversible layers (RevNet), allowing the model to store activations only once instead of N times. The idea is that during backpropagation, RevNet enables each layer's activations to be exactly reconstructed from the activations of the layer immediately before it, eliminating the need for the model to store the activations of every intermediate layer.

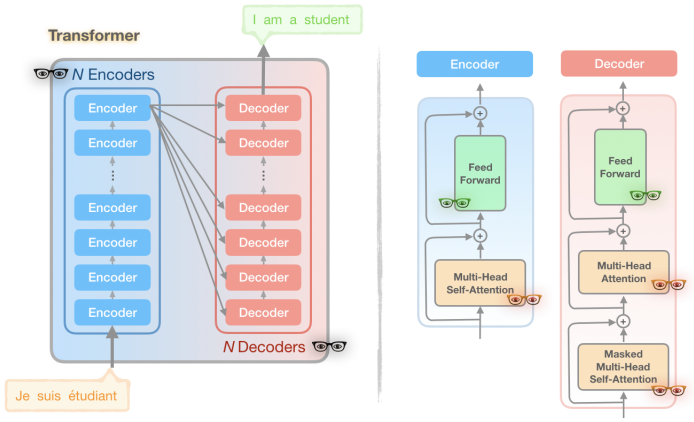

Figure 1: A simplified summary of the standard Transformer model[3]

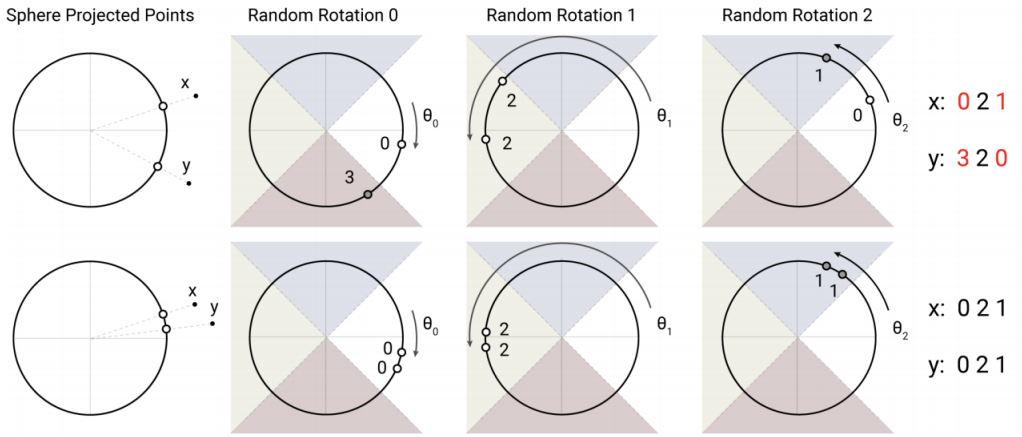

Figure 2: A simplified angular locality-sensitive hashing, in two dimensions and three rotations. Notice the close points have the same hashing. (From the Reformer paper)

The self-attention computation below is by far the most expensive process in a Transformer. The softmax operation over an extremely large matrix takes $O(L^2)$ time for a sequence of length L, since matrices QK^T is a matrix of size LxL.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

The authors of the Reformer point out that since the dot-product matrix is sparse, any softmax computation over it will be dominated by the few large values. Thus, to solve the problem of the gigantic self-attention computation, the authors realized that the softmax computation can be approximated by finding those large values through nearest-neighbors in a high-dimensionality space.

To do so, the authors use locality-sensitive hashing (LSH), one of the techniques that drew us to the Reformer in the first place, to group together the non-zero values. If two points are close in value, we should expect their hashes to be the same. The Reformer uses an angular locality-sensitive hashing, which uses random rotations to put points in a buckets. If two points are close together in dimensional space, they will likely end up in the same buckets. Figure 2 below shows a simplified locality-sensitive hashing in two dimensions with three rotations.

The key ideas behind how the Reformer uses LSH are as follows: In Figure 3 below, subplot (a) describes that the matrix for softmax computation is sparse. If we use nearest neighbors via LSH, we can easily group the nonzero values by swapping some of the axes. Then, to normalize hash bucket

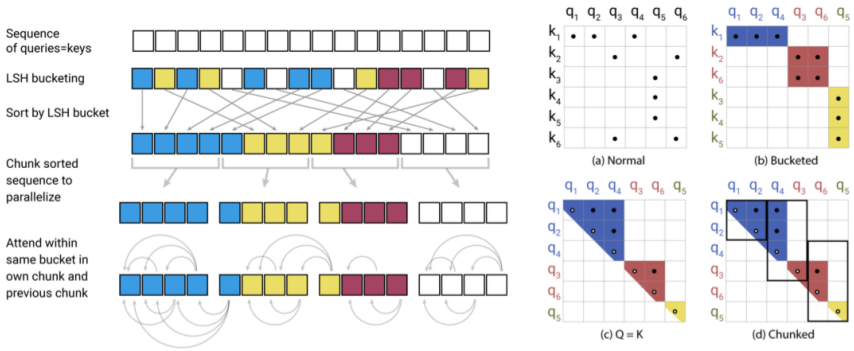

Figure 3: Simplified depiction of LSH Attention showing the hash-bucketing, sorting, and chunking steps and the resulting causal attentions. (a-d): step-by-step of the modified attention computation, which does not need to traverse the entire matrix. (From the Reformer paper)

sizes and ease computation, we replace the K axis with the Q axis so neighbor points are clustered on the diagonal (subplot (c)). Lastly, we process attention in chunks (subplot (d)), dramatically simplifying the calculation by only iterating through a subset of the total values. The complexity is reduced from $O(L^2)$ to $O(L \log L)$, allowing us to operate on longer sequences.

The third fix concerns the layers within the feed-forward network. Since computations in feed-forward layers are done independently across all positions in the sequence, both the forward and back propagation can be split into chunks. Operating one chunk at a time uses less memory.

Our baseline model was BiDAF [4] with word-level embeddings, following the standard parameters given in the default project. We tried two different models using the Reformer layer for the SQuAD 2.0 task to improve on the baseline. Although we were able to obtain an implementation of the Reformer in PyTorch by Phillip Wang [5], we had to make substantial changes to the model to make it compatible to the SQuAD task. The difficulty of incorporating it into the SQuAD task stems from the sequence-to-sequence nature of the Reformer (as well as the Tranformer), specifically that the input and output of the Reformer must be the same shape. SQuAD's (and BiDAF's) inputs of a context and question query (which are of different lengths), and outputs of two probability distributions (defined over the length of the context query) mean that this is not a plug-and-play situation: there has to be some setup to ensure the shapes all match.

Our first model (hereafter referred to as **"Pure Reformer"**) was actually suggested by Phillip Wang himself. We altered the collate function for the dataloader so the length of contexts and queries would always be 401, the length of the longest context in the training set. This was done so the inputs and outputs of the Reformer would have the same shape. Then, we retrieved embeddings of the context and query inputs and concatenated them so their length was 802. We put these into the Reformer, yielding a result of length 802, then split the result into two vectors of size 401. Lastly, we ran the masked softmax function from the baseline model on the two vectors, and returned the probabilities. Unfortunately, the results of this model were suboptimal.

The second model (hereafter referred to as **"BiDAF-Reformer"**) was created to harness the capabilities of the pre-existing BiDAF model in addition to the novel Reformer. Since both the encoding and modeling layers are RNNs, we could replace either or both of them with Reformers. However, given the long training time every time the Reformer is used, and the fact that the encoding layer is used twice (once with the context and once with the query), we decided to only replace the modeling layer with the Reformer. We also need a linear layer after the Reformer to make the last dimension's shape match. In summary, we use a embedding, encoding, attention, reformer (with its linear layer), and output layer for this second model. We were able to tune the hyperparameters of this model to slightly improve its functionality.

4 Experiments

4.1 Data

The models were trained on the default SQuAD 2.0 training set (129,941 examples) and evaluated on the dev set (6,078 examples). Each data point involves a question, a context paragraph, and a correct answer (three different human-sourced answers in the case of the dev and test sets). The task requires the model to identify the span of text within the context paragraph that answers the question, or else indicate that the answer does not exist. Our final model was evaluated on the test set a single time to obtain an unbiased estimate of how well the model would perform on unseen data.

4.2 Evaluation method

To evaluate the performance of our models on the dev set, we used the AvNA, EM and F1 metrics. AvNA simply compares if the predicted value is the same type of answer (a span of the context or "N/A") as the ground truth. EM (Exact Match) is a strict metric that returns true if the predicted answer matches the ground truth exactly, and false otherwise. F1 is the harmonic mean of precision and recall, a common metric for query classification (precision measures whether the output answer is a subset of the correct answer, while recall measures the proportion of the ground truth answer that the model output provides).

4.3 Experimental details

The baseline (BiDAF with word-level embeddings) was trained on an NV6 GPU for 30 epochs with a learning rate of 0.5. Each epoch took approximately 15 minutes for a total of 7.5 hours. Our first model (the Pure Reformer) was trained on an NV12 GPU for 3 epochs, initially with a learning rate of 0.5 and then with a lowered learning rate of 0.1, when it was observed that the NLL was oscillating widely. Even with a decrease in the number of encoders/decoders to 8 from the default of 12, each epoch of training still took about 5.5 hours. Our second model (BiDAF-Reformer) was trained on an NV12 GPU for 24 epochs with a learning rate of 0.7 (later upped to 0.8). Because we saw that the train NLL was decreasing while dev metrics were flatlining, we tuned some of the Reformer hyperparameters, increasing the dropout rate from 0.1 to 0.3 and further decreasing the number of layers from 8 to 4 to reduce overfitting. We ran this smaller BiDAF-Reformer model (BiDAF-Reformer-smaller) for 19 epochs with a learning rate of 1.2, with each epoch taking around 1.5 hours.

4.4 Results

The baseline BiDAF model after 30 epochs of training achieved a maximum F1 of 61.66, maximum EM of 58.34, and maximum AvNA of 68.64 on the dev set. The Pure Reformer model after 3 epochs of training had a maximum F1 of 52.19, maximum EM of 52.19, and maximum AvNA of 52.14 on the dev set. Although the Pure Reformer had higher EM and F1 scores compared to the baseline after 3 epochs, its NLL and its metrics on the dev set were not seeing any improvement. The BiDAF-Reformer model had an EM of 50.06 and F1 of 53.57 on the dev set. After hyperparameter tuning (decreasing number of layers, increasing dropout rate), we submitted the the BiDAF-Reformer-smaller model to the Test Non-PCE Leaderboard, with **EM of 47.63 and F1 of 49.68**. Neither model involving the Reformer was able to outperform the baseline.

The fact that the NLL on the train set was decreasing for the BiDAF-Reformer model makes it clear that the model was indeed learning on the training data: however, the NLL on the dev set failed to decrease to a level similar to the baseline, and EM, F1, and AvNA metrics all flatlined. This strongly suggests overfitting to the training data and a failure to generalize to unseen data: nevertheless, hyperparameter tuning in the form of increasing dropout and decreasing model complexity (number of layers) offered only marginal improvements.

The overfitting problem seen here makes sense, as the Reformer is a much more complicated and expressive layer than a simple RNN. Larger batch sizes and larger amounts of training data may help with this issue, but currently, it seems the Reformer is ill-fitted for the question-answering task. After all, the Reformer's most crucial benefit—that it can efficiently deal with long sequences—does not help here. It is likely too slow and too complex for this task.

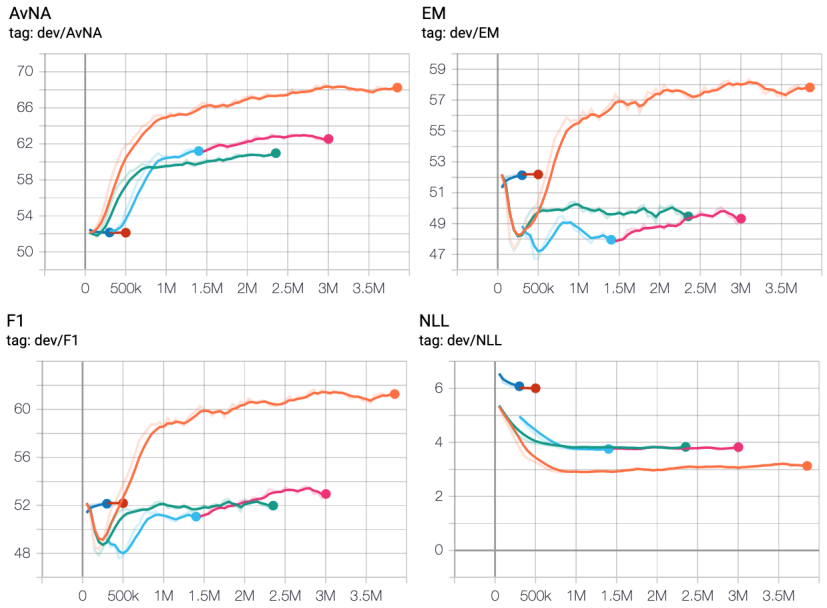

Figure 4: Metrics on the dev set for each model: color changes on the same line indicate learning rate modifications mid-run. Increasing dropout rate and decreasing the number of layers in BiDAF-Reformer resulted in marginally higher EM, but not F1.
Key: {Orange, (Navy, Red), (Light Blue, Magenta), Green} = {Baseline BiDAF (lr=0.5), Pure Reformer (lr=0.5, lr=0.1), BiDAF-Reformer (lr=0.7, lr=0.8), BiDAF-Reformer-smaller (lr=1.2)}

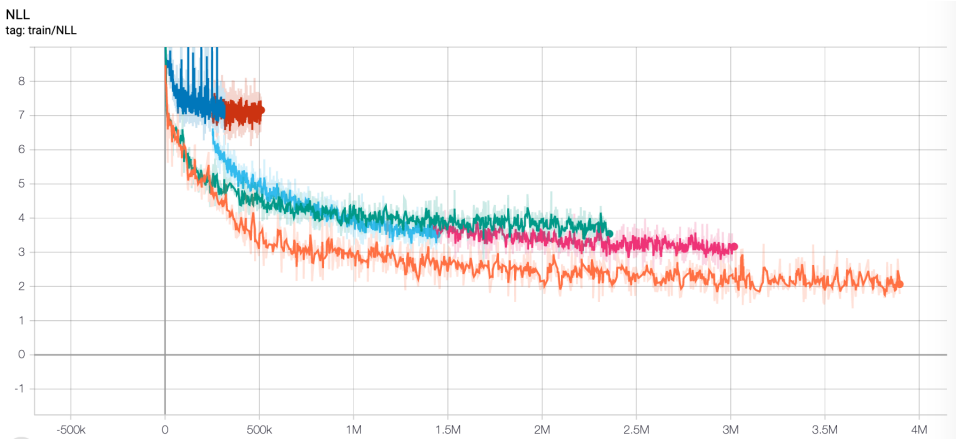

Figure 5: NLL on the training set for each model: the initial run of the Pure Reformer model experienced large oscillations in the NLL, and so the learning rate was decreased. However, the NLL failed to decrease for either the train or the dev sets. For the Bidaf-Reformer, train NLL decreased consistently, similar to the BiDAF baseline, but the metrics on the dev set failed to improve, suggesting overfitting.
Key: {Orange, (Navy, Red), (Light Blue, Magenta), Green} = {Baseline BiDAF (lr=0.5), Pure Reformer (lr=0.5, lr=0.1), BiDAF-Reformer (lr=0.7, lr=0.8), BiDAF-Reformer-smaller (lr=1.2)}

5 Analysis

To truly understand the inner workings of the models in question, we need to do a comprehensive error analysis on their strengths and weaknesses. Here, we will be discussing the baseline model and the best Reformer model (BiDAF-Reformer).

The baseline model is able to answer many of the questions correctly, as shown below in a tricky passage with distractions the model has to parse through. All contexts are truncated for brevity.

- **Question**: Who designed the garden for the University Library?
- **Context**: Another important library – the University Library, founded in 1816, is home to over two million items. The building was designed by architects Marek Budzyński and Zbigniew Badowski and opened on 15 December 1999. It is surrounded by green. The University Library garden, designed by Irena Bajerska, was opened on 12 June 2002...
- **Answer**: Irena Bajerska
- **Prediction**: Irena Bajerska

Here, there are two other architects of the library, and not its garden, yet the baseline model is able to follow through with the guide word "garden" to find the other name. It is a trick the model is able to circumvent.

However, the baseline model does have a significant flaw: the "no answer" option, new to SQuAD 2.0, often gives it trouble. For example,

- **Question**: What is the oldest work of Norman art?
- **Context**: By far the most famous work of Norman art is the Bayeux Tapestry, which is not a tapestry but a work of embroidery...
- **Answer**: N/A
- **Prediction**: Bayeux Tapestry

When the model predicts a span of words instead of N/A when N/A is correct, that implies the model is too sure about its predictions compared to the baseline "no answer." There are also examples where the answer is a span of words, but the model predicts N/A. However, the plurality of incorrect responses follow the structure above.

The Reformer model didn't do as well as the baseline model. Looking at some of the outputs sheds light on why exactly that is. The following is an easier question-context pair that the model nails accurately.

- **Question**: Interest groups and government agencies that were concerned with energy were no match for who?
- **Context**: ...The possibility that the Middle East could become another superpower confrontation with the USSR was of more concern to the US than oil. Further, interest groups and government agencies more worried about energy were no match for Kissinger's dominance...
- **Answer**: Kissinger
- **Prediction**: Kissinger

The Reformer model also has some slight problems nailing down the correct spans, even if it might choose some of the words for the answer correctly. Here, the problem is that the model knows "Protestantism" is a key word, but doesn't realize that the word itself is sufficient for the answer. "in favour of Roman Catholicism" describes the change of religion, but it is not necessary to answer the question of what religion Henry renounced.

- **Question**: What religion did Henry renounce upon ascending the throne?
- **Context**: ...The warfare was definitively quelled in 1598, when Henry of Navarre, having succeeded to the French throne as Henry IV, and having recanted Protestantism in favour of Roman Catholicism, issued the Edict of Nantes...
- **Answer**: Protestantism
- **Prediction**: Protestantism in favour of Roman Catholicism

Sometimes, the Reformer model's predictions are just clearly off-base. In the question-context pair below, the presence of multiple names undoubtedly confuses the model, but the answer should be quite easy to find in the context for a human. It may be difficult, however, for a model to realize that somebody's grandson is that grandson's grandfather.

- **Question**: Who was Kaidu's grandfather?
- **Context**: Instability troubled the early years of Kublai Khan's reign. Ogedei's grandson Kaidu refused to submit to Kublai and threatened the western frontier of Kublai's domain. The hostile but weakened Song dynasty remained an obstacle in the south. Kublai secured the northeast border in 1259 by installing the hostage prince Wonjong as the ruler of Korea, making it a Mongol tributary state...
- **Answer**: Ogedei
- **Prediction**: Wonjong

The Reformer model does have a similar problem with the baseline model regarding no answers. Below is an example of a tricky question that the model misinterprets: nothing in the context happened in Toronto, so the answer should be N/A.

- **Question**: What British General negotiated at Toronto?
- **Context**: In September 1760, and before any hostilities erupted, Governor Vaudreuil negotiated from Montreal a capitulation with General Amherst...
- **Answer**: N/A
- **Prediction**: General Amherst

Overall, both the models suffer from over-guessing: both should be guessing spans less and N/A more. They both are susceptible to tricky questions and complicated contexts that are designed to be elusive, with the Reformer model suffering more.

6 Conclusion

We were initially interested in the Reformer since it eased memory usage and used a intriguing type of hashing that has not been commonly seen in the NLP literature. However, the paper didn't disclose that training generally takes longer with the Reformer than the Transformer, which greatly decreased how much we could train with respect to time and money. The tradeoff between memory usage and training time is less favorable than we thought initially.

Furthermore, the Transformer doesn't do as well as the BiDAF model on a per-epoch level. Our hypothesis is that the Reformer is a much more complicated model layer than an RNN encoder, and thus overfits significantly, despite our efforts to make the model less complex and increase its dropout rate. If we had more time and resources, we could continue to find the sweet spot in terms of overall model complexity and hyperparameters to reduce overfitting.

Overall, we'd like to see whether the Reformer succeeds at other language tasks before we deem it useful in the future. Other than question-answering, testing the Reformer on language translation, part-of-speech tagging, and other NLP tasks would greatly increase confidence in its usefulness.

References

[1] Nikita Kitaev, Łukasz Kaiser, Anselm Levskaya. *Reformer: The Efficient Transformer*. arXiv:2001.04451, 2020.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser and Illia Polosukhin. *Attention Is All You Need* arXiv:1706.03762, 2017.

[3] Alireza Dirafzoon. *Illustrating the Reformer*. https://towardsdatascience.com/illustrating-the-reformer-393575ac6ba0, 2020.

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

[5] Wang, Phillip. "Reformer, the Efficient Transformer, in Pytorch." GitHub, 9 Jan. 2020, github.com/lucidrains/reformer-pytorch.