All SQuAD Needs is Self-Attention: Char-QANet

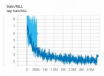
Luis Alcaraz, Troy Lawrence
Department of Computer Science, Department of Electrical Engineering
Yian Zhang

Introduction

Techniques such as Long-Short Term Memory (LSTM) networks were able to improve on previous methods that took on sequence learning, sequence translation, marking improvements in speech and text comprehension. Limitations were identified, however, specifically when mapping the meaning of long sequences of data. **Transformers**, using self attention and a completely new architecture, were able to identify deeper connections between words and characters at distances that LSTMs and RNNs weren't capable of mapping. However, Transformers architecture is unique and requires specific tasks.
Therefore, we wanted to try our hand at building a transformer like architecture. QANet gives a possible solution to this problem. Working on top of a Bidirectional Attention Flow model, **QANet** infuses this LSTM model with self-attention (what makes Transformers so powerful at learning contingencies within language) and convolutional layers.
Although we are not presenting a new technique, we are trying our hand at one of the best approaches that surpasses BiDAF's accuracies. Nevertheless, within this project, we experimented with hyperparameters, allowing us to get a better understanding of the architecture and its limitations.

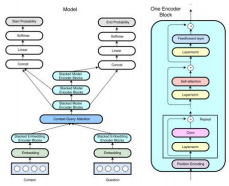
Key Findings

- Although the QANet paper states it is 5x faster than BiDAF, this is only true when you have a large number a GPUs available. QANet is a very computationally expensive model due to the use of Multi-Head Attention and convolutional layers in the encoder blocks.
- Initialization of hyperparameters is key to getting successful results from such architecture. If hyperparameters are not tuned properly, loss stagnates early on or grows exponentially. (below are two QA models, light blue has stagnated due to improper hyperparameters.



- Although Loss plateaus rather quickly compared to BiDAF, learning continues as F1 and EM scores continue to rise. This is due to the fact that the model is learning more complex dependencies.

Methods



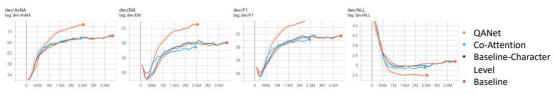
Our implementation started with character embeddings, which utilize CNN's to create the projections of the characters. We also infused co-attention into our baseline model which attends both question and answer simultaneously.

All data came from SQuAD 2.0 with pre-split train, dev, and test sets. GloVE pre-trained embeddings were used in embedding layer

We utilize QANet's architecture, which builds off of the BiDAF model. By using residual encoder blocks, we allow the model to learn complex dependencies. By using embedded character and word embeddings through convolutional neural networks, it captures local structures of text. Self-Attention utilizes multi-head attention which allows the model to learn global interactions between word pairs.

With each layer, there are hyperparameters associated. We experimented on these hyperparameters and identified potential drawbacks from QANet.

Results

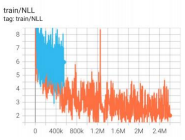


Above are shown our dev set results. The QANet model performance, (visuals above and metrics below), out paces baseline and Co-Attention metrics in a regards.

Dev Results			
	Dev AvNA	Dev EM	Dev F1
Baseline	68.5	57.6	60.9
Baseline+Char-Level	68.3	58.0	61.5
Coattention	68.8	58.1	61.4
QANet	**72.6**	**63.0**	**66.4**

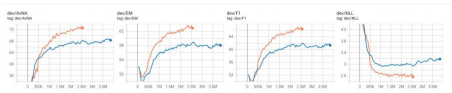
Our model was able to produce the following test set results- F1: 59.425, EM: 62.785. We believe to know why there is a difference between test and dev, which will be explored in the analysis section.

Analysis



One of the first things we identified was the importance of hyperparameters. This was visually apparent when choosing the proper optimizer. The baseline utilizes Adadelta (light blue) in comparison to ADAM (orange). This, we believe, is due to Adam's bias correction towards moments, leading to a faster convergence.

Furthermore, the scheduler was also extremely important as QANet requires an inverse exponential increase. In addition, we found that the baseline learning rate (.5) was inadequate for learning as it pushed the negative log likelihood loss towards exponential increase within 3 epochs. Setting it to QANet's suggested .001 gave intended results.



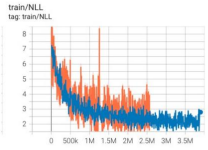
Above we can see a comparison between our implementation of QANet and the Baseline implementation. It is key to point out that QANet was still learning. However, given the computational limitations, we did not want to deallocate our virtual machine and lose all progress. Nonetheless, this is significant since it means we would have trained for longer and possibly gotten better dev results. Furthermore, it could have refined learned weights, which would have had an impact on test results.

We believe the less optimal test set results arise from this issue of training, where we only trained for 20 epochs compared to the baseline's 30. This is due to the fact that each epoch took 34 minutes to train for our QANet implementation in comparison to the 6-minute epochs of the baseline. Although the QANet paper claims speed increases of 3x-9x in comparison to BiDAF, this was something our team did not experience. Again, this is due to the computation complexity of QANet, and its memory hungry encoder blocks.

We found it very interesting how sporadic QANet's negative log likelihood loss was in comparison to the baseline. We tried reducing this through various hyperparameters such as Gamma1 and Gamma2 for the Adam optimizer. However, these test were not fruitful.

Co-attention, although a promising technique, did not preform well under our implementation, only slightly outperforming the baseline. Therefore, we decided not to attempt to implement it within our QANet model.

Given that QANet utilizes character embeddings, our first implementation of these embeddings was helpful in building the QANet model.



Conclusions

QANet and the Self-Attention mechanism are very powerful. Due to the multi-head attention, the model was able to create local and global dependencies that surpassed LSTM's and RNN's attempt. This was also done with one third less epochs. Therefore, it exemplifies how powerful these encoder blocks can be under the right resources.

However, we also learned that QANet has its limitations, which it shares with transformers. Due to these blocks that consist of convolutional layers and self-attention layers, the model becomes memory expensive, which comes to limit which hyperparameters one can choose to obtain the best results. Such challenges forced us to limit our batch size from the recommend 64 too 16 right from the start.

However, we are still hopeful in the use of transformers and models like QANet which utilize transformer building blocks given advancements made towards making transformers more memory efficient. Papers like The Reformer: The efficient Transformer present unique changes to transform architecture like Locality Sensitivity Hashing Attention, which by not preforming a dot product, brings down memory and computational task complexity to *O(nlogn)* instead of *O(n^2)*.

Nevertheless, we were proud with the achievements we made in our project. We learned about the limitations oof QANet, while observing the usefulness of such problematic encoder blocks.

References

Yu, Adams Wei, et al. "Qanet: Combining local convolution with global self-attention for reading comprehension." *arXiv preprint arXiv:1804.09541* (2018).

Xiong, Caiming, Victor Zhong, and Richard Socher. "Dynamic coattention networks for question answering." *arXiv preprint arXiv:1611.01604* (2016).

Seo, Minjoon, et al. "Bidirectional attention flow for machine comprehension." *arXiv preprint arXiv:1611.01603* (2016).

Acknowledgments

CS224N Teaching Staff for an amazing quarter of learning through new experiences and exceptional instruction