Efficiency of Dynamic Coattention with Character Level Embeddings

Stanford CS224N Default Project

George Younger
Department of Computer Science
Stanford University
gyounger@stanford.edu

Michael Lin
Department of Computer Science
Stanford University
mlin4@stanford.edu

Semir Shafi
Department of Computer Science
Stanford University
semir@stanford.edu

Abstract

Question answering has long been a difficult task for computers to perform well at, as it requires a deep understanding of language and nuance. However, recent developments in neural networks have yielded significant strides in how well computers are able to answer abstract questions; concepts like dynamic coattention and character level embeddings have helped machines with abstract tasks like reading comprehension. Despite these strides, training models utilizing these techniques remains cumbersome and exceedingly time consuming. Within the scope of this project, we attempt to replicate the techniques mentioned above in conjunction with each other while simultaneously removing aspects of the training process that extend its time frame unnecessarily; our goal was to optimize not only for accuracy of the model but also for the speed of training after discovering our initial baseline model took approximately 15 hours to train on a remote GPU.

1 Key Information to include

- Mentor: Elissa Li
- External Collaborators (if you have any): N/A
- Sharing project: Sure!

2 Introduction

With recent developments in neural networks, question answering has become an increasingly popular methodology of testing how sophisticated a network is and of determining how closely machines can mimic human understanding of language and its nuances. This is an exceptionally difficult problem- question answering requires understanding of both the information that the entity asking the question desires as well as how to glean this information from provided text (as well as determining if the information is even present to begin with).

One problem with these models is that they are sometimes prohibitively computationally expensive to train. As an example, the baseline model provided to us by the starter code for this project took around 13 hours to train; any tweaks or adjustments required a complete retrain of the model. While accuracy in question answering is certainly an important goal, sometimes speed and ease of use can

be equally or even more important goals- some users may be willing to sacrifice some accuracy for the speed at which they can adjust the model.

After identifying this problem, our team decided to implement a couple improvements on top of the baseline model provided to us and attempt to optimize it such that it could be trained more quickly while sacrificing minimal accuracy. Our group chose to implement character embeddings in conjunction with a dynamic coattention model. After doing a literature review, we believed that character embeddings provided a relatively painless and simple means of increasing our accuracy in question answering without sacrificing much, if any, computation time. We determined that the dynamic coattention model would require significantly more computation time, but would also provide a much more robust and accurate question answering model upon which we could experiment with different adjustments to attempt to optimize for speed of training.

3 Related Work

3.1 BiDAF

Our baseline model drew inspiration from the BiDAF model as described in [1]. Seo et. al propose a multi-stage hierarchical process that uses bi-directional attention. Unlike previous work, since they compute attention at every time step and permit the attended vector to flow into the modeling layer, they minimize the loss caused by early summarization. Furthermore, they also leverage a memory-less attention mechanism where the attention at each time step doesn't depend on attention from the previous time step (i.e. avoiding dynamic attention).

3.2 Dynamic Coattention

Xiong and Zhong proposed a two-fold solution to overcome local maxima when answering a question [2]. They propose a "coattention model" to give attention to both the question and the context at the same time, building a codependent representation of them simultaneously which they use to predict the starting and ending point of the answer within the context. Second, they introduced a dynamic point decoder, which iteratively moves through the context to determine which start and endpoints in the context satisfy the question best.

3.3 QANet

Yu et al. developed a model to tackle a similar problem: the latency of training and reference of RNNs used for question answering [3]. Their QANet model eliminates all RNNs and instead consists of convolution and self-attention. They achieve 3-13x faster training time on the SQuAD dataset without compromising accuracy in comparison to recurrent models.

4 Approach

Initially, our approach was simply the vanilla baseline model provided to us by the teaching staff. From there, we constructed a character embedding layer to be appended to the word embeddings we were already utilizing (this was our progress at the milestone). Finally, based on the description of a coattention model[2], we constructed our own implementation of the coattention model, but we did not have time to implement the dynamic decoder also described by the paper as we found it prohibitively difficult to implement.

4.1 Initial Model

Our initial model consisted of two embedding layers, the context and the question embeddings, which both had word embeddings concatenated together. Once we had constructed these embeddings, we passed them through an attention layer, which were just vanilla weight matrices with a bias term. Finally, we passed these through a decoder layer, which was initially just a simple fully connected linear layer followed by an output layer. This was by far our most simplistic model, but it provided the base on which we constructed the rest of our models.

4.2 Character Embeddings

Once we added character embeddings to our model, it was effectively the exact same as the former model (layers were all the same, they had just changed sizes to accommodate character embeddings concatenated on top of the word embeddings). We then passed this through the same decoder model that we had in the initial model, but again with the layer dimensions adjusted to reflect that we had added character embeddings to the input. This was the model we used for the milestone.

4.3 Coattention Model

We used the paper referenced in [2] to build our coattention model, constructing our layers and weight matrices in the same way they did. Our model first constructs the embeddings for the characters and words in both the context and the question. The original paper utilizes an LSTM layer to embed the words in the question and the character, but we decided to just utilize an Embedding layer similar to what we used for the word + character embeddings in the milestone.

We then compute what the paper refers to as the Affinity Matrices between the two; this is the product of the context matrix and the question matrix, with the softmax across the different dimensions to compute the affinities of each with respect to the other; these are referred to by the paper as the normalized attention weights A^D and A^Q. We compute the product of A^Q and D to get the attention contexts of every word in the document given the question; essentially, we are looking to see how important each word in the document could be given the words in the question. We concatenate this with the question embeddings and take a similar product to get the attention contexts of every word in the question given the document; this now tells us which words in the question are particularly important to calculate the answer to the question. Finally, we concatenate this matrix with the document embeddings and feed the final result into an LSTM layer.

Pictured below is a diagram from the paper referenced showing how the coattention model is constructed, which we referenced to build our own coattention model.

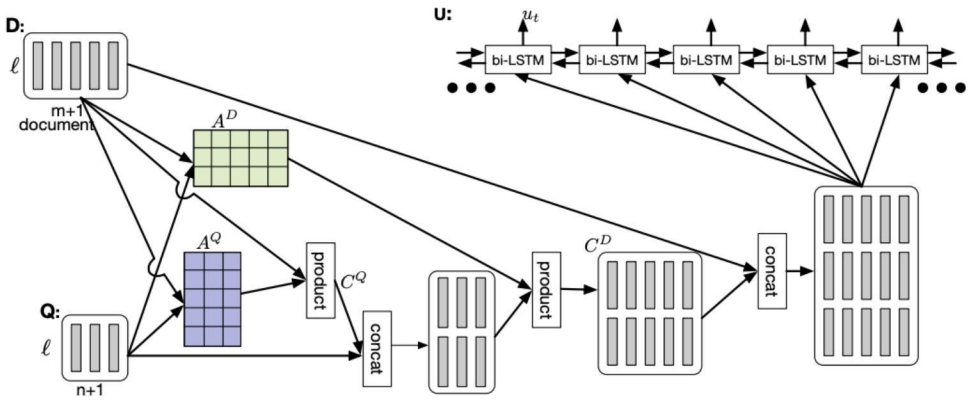


Figure 2: Coattention encoder. The affinity matrix L is not shown here. We instead directly show the normalized attention weights A^D and A^Q.

4.4 Output

The original paper implements a dynamic pointer decoder, iterating across many different potential output points and choosing the best one. Without testing it, we believed that this might introduce a bottleneck to training time, which would counter our goal of optimizing the model for speed; additionally, the coder looked prohibitively difficult to implement, so we opted to utilize the simple output model from the original BiDAF model in the milestone.

We experimented with different layers being added to and subtracted from the output layer (we attempted to add an additional LSTM layer, believing it would help with the coattention encodings by identifying similar patterns of words across the question and the answer like "Where was John

Adams born?" "John Adams was born in..."). We also experimented with removing the modeling layer from the output as we believed the coattention model would have sufficient complexity to be able to identify answers to the questions accurately.

5 Experiments

Our experiments mainly just consisted of training our model and seeing how it performed on unseen dev sets of data; we trained 3 different models after our various adjustments described in our approach to the model and saw their performance on the F1, EM, and AvNA metrics provided by Tensorboard.

5.1 Data

The data we used was paragraphs potentially containing salient information to a question and then the question itself. For example, the question could look something like

"Who won the NFL MVP award in 2016?"

and the paragraph could look something like

The 2016 NFL season began on September 1, 2016, and concluded with the Super Bowl on February 13, 2017. The NFL champions for this season were the New England Patriots, who defeated the Atlanta Falcons 34-28 in the Super Bowl. The NFL MVP for this season was Matt Ryan, the quarterback for the Atlanta Falcons.

If the system were to correctly identify that Matt Ryan was the answer to this question, or give an answer similar to that, it would be rewarded; otherwise, it would receive a score of 0 and need to tweak its parameters. The data for this model consisted of thousands of these question/answer pairs to help the machine develop and understanding of what a question was looking for. Additionally, some provided contexts did not contain the answer to the question asked; in this case, the machine is trained to respond without a response as it does not have the necessary information to be able to answer the question.

5.2 Evaluation method

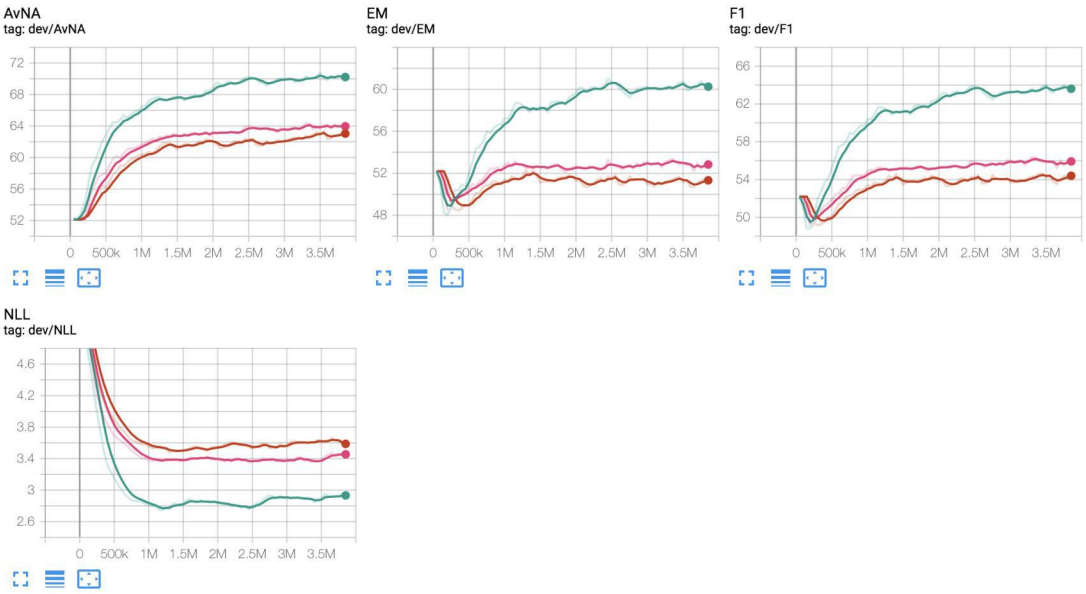
Our evaluation consisted of the previously discussed metrics of the F1, EM, and AvNA scores on the dev set of data, as well as the time taken to train the model as we were seeking to optimize for both the accuracy of the model as well as the speed with which it could train. We wanted to see if we could sacrifice minimal accuracy from the character embedding model and simultaneously train in less time, as the character embedding model took around 13 hours to train.

5.3 Experimental details

Running our experiments was fairly straightforward- we tinkered with the model locally until we believed it was ready to be trained, then copied the code from our local machine onto a remote Azure machine to train on the aforementioned training data set. Every 50,000 iterations, we would pause the model to evaluate its performance on a development set of questions and answers not yet seen, to ensure that it was not overfitting to the training data and was indeed learning correct parameters to

5.4 Results

Visual representation of our models' performance on the dev sets:



	Name	Smoothed	Value	Step	Time	Relative
	train/baseline-20	1.773	1.252	3.898M	Wed Mar 3, 04:10:05	13h 36m 20s
	train/baseline-44	3.332	4.34	3.898M	Sun Mar 14, 03:27:17	6h 40m 43s
	train/baseline-46	2.931	2.946	3.898M	Sun Mar 14, 18:49:13	15h 18m 53s

In the above graphs, the green is our character embedding model, the pink is the model we trained with coattention encoding that had a modeling layer appended to the output layer, and the brown is the model we trained with coattention encoding that did not have a modeling layer appended to the output layer.

Clearly, our coattention models did not perform nearly as well as the pure character embedding model, so the model we chose to use for the final test set was our character embedding model. We go into detail about why we believe this to be the case in the evaluation section, but the main point that we believe we should have considered in hindsight was that applying coattention to word embeddings grants importance to words in the opposite context (question to document and vice versa) based on their presence; when we appended character embeddings, we believe we were introducing false signals to the model that made it believe that the characters were important to answering the questions when in fact it just introduced quite a bit of noise.

In this vein, we discovered that removing the modeling layer from the output of the coattention model did not substantially decrease our accuracy, but it did shave approximately 60% of the training time off the model that did have the modeling layer in its output. We don't believe this is because the modeling layer is unimportant, but rather because combining coattention with character embeddings will not work in general; if we were to have implemented a pure coattention model without character embeddings, we believe the difference between the two models would have been far more stark. Additionally, the pure character embeddings BiDAF model performs far better than the more complex coattention model with the modeling layer in its output at less time; this suggests that combining these two approaches actually makes the model worse along with being more computationally complex, which was the opposite of what we were trying to achieve.

Similarly,

6 Analysis

6.1 Ablation Study

Our baseline word-level embedding BiDAF model achieved an F1 score of 58. Adding character level embeddings had a positive impact on the performance. The F1 score increased from 58 to 63. Because word embeddings can only handle previously seen words, adding a character level embedding appears to perform superiorly since out-of-vocabulary words can also be given an embedding.

Our coattention model with a modeling layer and character level embedding did not perform nearly as well as we would have expected. It acheived an F1 score of about 56. We attribute its low score to the lack of implementation of a dynamic decoder (lack of time) and because we incorporated character level embeddings. Essentially, the coattention model might have performed poorly because the character embeddings introduced more similarities between the question and the context than should have actually been attributed. For example, the coattention model attributes importance to embeddings in the question and document based on their presence in the opposite context; with words this works well to identify important tokens, but once characters are introduced, it may be attributing importance to specific characters, which can end up being nonsensical. We noticed that the time it took to train the model was significant. In an effort to remove the bottleneck, the same coattention model without a modeling layer trained in half the time but still achieved an F1 score greater than 54. Basically, removing the modelling layer did not significantly affect performance but significantly reduced the training time.

6.2 Specific Examples

We looked at several examples in the dev step to see how the question and answering was performing across the three models, and we found some pretty valuable insights from looking at the qualitative results, especially as we consider what about these models led to these different results:

6.2.1 "Why" Questions

Example 1: Why is Warsaw's flora very rich in species?

- BiDAF with character embeddings: The species richness is mainly due to the location of Warsaw within the border region of several big floral regions.
- Coattention without modeling layer, with character embeddings: N/A
- Coattention with modeling layer and character embeddings: N/A
- Correct answer: location of Warsaw

To support our quantitative results of which models performed better, it's clear here that the BiDAF with character embeddings worked better. What we tended to see with most "why" questions was that the first model would be able to answer it, but the next two couldn't find it. Additionally, we found that BiDAF with character embeddings tended to do better with longer answers and with more word variations. Perhaps the coattention models weren't able to understand that "rich in species" correlated with "species richness".

6.2.2 Not Answerable Questions

Who made experimental measurements on a model Rankine cycle?

- BiDAF with character embeddings: N/A
- Coattention without modeling layer, with character embeddings: Watt
- Coattention with modeling layer and character embeddings: Joseph Black
- Correct answer: N/A

Similarly, the BiDAF seemed to perform better than the Coattention models. Interestingly enough, in this section, it was stated that "The experimental measurements made by Watt on a model steam engine led to the development of the separate condenser. Watt independently discovered latent heat, which was confirmed by the original discoverer Joseph Black, who also advised Watt on experimental procedures."

Perhaps the Coattention without modeling layer saw the proximity that Watt was to "experimental measurements" and decided that the answer was correct, while the other Coattention with modeling perhaps noted that Joseph Black was the "original discoverer". Either way, in many of the examples we looked at, they seemed to put more weight in associated names/people than trying to find the exact answer.

6.2.3 "Not" Questions

How many non-Muslims are in Greater London??

- BiDAF with character embeddings: Over 900,000.
- Coattention without modeling layer, with character embeddings: 900,000
- Coattention with modeling layer and character embeddings: 900,000
- Correct answer: N/A

All three models were tricked into returning 900,000 because in the paragraph it stated that "there are over 900,000 Muslims in Greater London." This happened in a lot of other questions where there was a negation. Clearly, this shows that all three models still aren't able to understand how negatives function relative to the sentences.

7 Conclusion

We explored a handful of different approaches on improving the SQuAD evaluation score within the context of coattention models. Immediately, we noticed character-level embeddings increase evaluation metrics by a few points and decided to explore coattention models with character-level embeddings. The performance of our coattention models without a dynamic decoder performed significantly worse than the baseline. We noted how removing the modeling layer reduced the training time in half while achieving a similar performance.

We hypothesized that the coattention model did not perform as well because the character-level embeddings introduced unnecessary and irrelevant similarities between the question and context embedding. Furthermore, we noted that there were some variance in the training runs especially in the F1 score. Some potential avenues for future work can explore removing character-level embeddings, reintroducing a dyamic decoder and observing the performance between a coattention model with and without a modeling layer to see if there are still improvements in training time. Furthermore, it would also be interesting to further explore the QANet model to understand how they intended to improve on training time.

We acknowledge that our model did not perform as well as we would've liked but we note that we've seen promise in the ability to reduce training time.

References

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

[2] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016.

[3] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.