What Complements Coattention

Stanford CS224N Default Project

Thomas Mayer and Olivia Weiner
Department of Computer Science
Stanford University
`mayert@stanford.edu`
`oweiner@stanford.edu`

Abstract

In this project, we aim to reproduce a coattention layer on the Stanford Question Answering dataset (SQuAD) baseline model, and investigate its relationship with other common SQuAD techniques. We start by testing how a coattention layer improves the baseline model and find that when it is not paired with any other techniques, it lowers performance. Next, we implement the Dynamic Decoder and Highway Network, as well as Character Embeddings, and find that both increase the performance of the coattention baseline, but still underperform the standard baseline. In fact, the baseline only improves in the case where we use character embeddings with the standard BiDirectional Attention Flow (BiDAF) layer and single-pass decoder.

1 Key Information to include

- TA mentor: Ben Newman
- External collaborators (if no, indicate "No"): No
- External mentor (if no, indicate "No"): No
- Sharing project (if no, indicate "No"): No

2 Introduction

Question Answering systems query a large database of knowledge to construct and return answers to input questions. The Stanford Question Answering Dataset (SQuAD) is unique in its large volume and quality of data to train on. It also provides a Bi-directional Attention Flow (BiDAF) baseline model that can be used to test other SQuAD techniques on. We train our model on SQuAD to output the span in our context document in which the answer to our input questions lie.

Coattention was introduced to the SQuAD dataset in Caiming Xiong's paper and found to significantly improve upon the results of state of the art models when implemented together with a dynamic pointing decoder to form a Dynamic Coattention Network (DCN). However, beyond this implementation, there has not been a lot of research on the Coattention Network's application to Question Answering systems, and to our knowledge, there has not been an investigation into the effect of coattention on a simple model, and how it interacts with other common QA techniques. These questions are of importance as they can act as guidelines to whether coattention is a useful technique on simpler models, and can act as an indicator to which techniques should be recommended or avoided in conjunction with using coattention. Our report aims to investigate these questions.

The coattention layer is similar to BiDAF attention (used in the 2022 SQuAD IID track baseline) in that it contains two-way attention between the question and the context. It differs in that it also has a second-level attention computation which attends over the attention output representation. This results in a more sophisticated and potentially expressive representation of each word in the document. We elaborate on some distinguishing details of how the coattention layer is constructed compared to BiDAF in the Approach section.

We implement the Coattention Network from scratch on the standard SQuAD baseline model and use this as our second baseline. Next, we implement three other common SQuAD techniques to investigate how they affect the coattention baseline compared to how they affect the standard baseline. The first techniques we implement are the Dynamic Decoder and Highway Network. We choose these because our reference paper by Caiming Xiong implemented their coattention layer alongside these techniques and emphasised their combined importance in forming a Dynamic Coattention Network. The other technique we implement is the use of character embeddings. We chose this because it is a common and often successful question answering model technique, and it was advised by our mentor Ben Newman.

We found in our experiments that implementing the coattention layer on our SQuAD baseline model lowers performance across all metrics. The Dynamic Decoder improves the coattention baseline but not the BiDAF baseline, although the DCN still falls short of the BiDAF baseline. Character embeddings improve the baseline across all models, but had the biggest effect in the BiDAF baseline.

3 Related Work

The idea of a coattention network was initially introduced for visual question-answering. Visual question-answering is similar to the verbal question-answering used for SQuAD, except instead of selecting a span of words within a document to answer a question, the model was trained to highlight the relevant portions of an image in order to answer the question. In order to do this, a novel coattention network was introduced, in which the attention calculation across the words in the question and the attention calculation across the features in the image are computed in tandem, and the computation of one impacts the computation of the other (Lu et al., 2017). At the time, attention calculations had been computed solely on the image features, meaning that the key information stored in the question was did not effect these computations.

This idea of coattention was further expanded upon and put to work in verbal question-answering for the SQuAD dataset in the form of the Dynamic Coattention Network. The model employs a coattention network early on, ultimately outputting a coattention embedding for each word in the document, based on several calculations performed on both the question encodings and the document encodings, including two similarity matrices with the softmax taken across the question words in one, and across the document words in the other.

The primary innovation of the Dynamic Coattention Network was actually the Dynamic Decoder. Previous models had generally used single-pass decoders, that is, decoders where a single calculation was performed to predict the span of the answer. However, the Dynamic Coattention Network used an iterative model, whereby at each step, the decoder uses an LSTM to keep track of all the previous predictions made for the start word and the end word, and makes its new guess based on the hidden state of the LSTM. This was said to avoid the pitfalls of previous models in which they would get stuck at local maxima and be unable to move past them to find the globally optimal start-end pair.

The model also combined two previous ideas: Highway Networks and Maxout Networks. Highway Networks are networks in which early layers pass on their results not just to subsequent layers but also directly to the final layers. Maxout Networks are networks in which a number of calculations (a number referred to as the size of the maxout pool) are computed in parallel at each layer, but only the maximum value achieved across the maxout pool is passed on. Due to their strong empirical performance in previous experiments, for the Dynamic Coattention Network, they were combined into a single Highway Maxout Network (HMN). The primary justification given for using such a model is that there are a great variety of question-document pairs, and they may thus require a number of different strategies to produce satisfactory answers. In this vain, the HMN allows the model to try out many different strategies simultaneously and find which one produces the most promising results overall (Xiong et al., 2018).

Additionally relevant to our implementation are character-level embeddings. Character embeddings have previously been implemented as a supplement to the Bi-Directional Attention Flow (BiDAF) model, using a Convolutional Neural Network to achieve a character-level embedding for the word vectors used alongside the pretrained GLoVE word vectors. This has achieved noteworthy results in the past (Seo et al., 2018).

4 Approach

4.1 Baselines

As mentioned in our Introduction, we use two baselines in our project. The first is 2022 SQuAD IID track provided starter code. The second is the same baseline with a coattention layer added that we have implemented. The coattention layer was reproduced using the description in the conference paper "Dynamic Coattention Networks for Question Answering" by Caiming Xiong. Our approach will be to compare the effect of several common SQuAD techniques on the two baselines to investigate which combination of features creates the highest relative improvement for the coattention vs standard baseline. The SQuAD techniques we look into are the following: the Dynamic Decoder, which includes a Highway Maxout Network, and character level embeddings. They were picked under guidance of Stanford SQuAD handout recommendations.

4.2 Implementation

We start by implementing the coattention layer on our SQuAD baseline model to form our second baseline. As mentioned, the coattention layer is similar to BiDAF attention (of our baseline) in that it contains two-way attention between the question in context. Its main difference is that it contains a second-level attention computation which attends over the attention output representation. Below, we expand upon the noteworthy implementation details of adding this to our SQuAD baseline model.

Given question representation matrix $[x_1^Q, x_2^Q, ..., x_n^Q] \in R^{nx2h}$ and document representation matrix $[x_1^D, x_2^D, ..., x_m^D] \in R^{mx2h}$ we add a sentinel vector $q_\emptyset, c_\emptyset \in R^{2h}$ to the end of the question and document representation matrix respectively to form matrices $[x_1^Q, x_2^Q, ..., x_n^Q, q_\emptyset] \in R^{n+1x2h}$ and $[x_1^D, x_2^D, ..., x_m^D, c_\emptyset] \in R^{m+1x2h}$. Here, h represents the size of the hidden layer, n, the number of words in the question and m, the number of words in the document. The addition of one extra vector allows our model to avoid attending to any particular word in the input. Another noteworthy part of the coattention layer is that it adds a non-linear projection layer to the question encoding. This allows for some variation between the document and question encoding space. We apply the projection layer through the following equation: $x_j^Q = \tanh(Wq_j + b) \in R^{2h}$.

Following our coattention implementation, we implemented the Dynamic Decoder in accordance with the original implementation by Xiong et al. We feed into the decoder the coattention encodings, $C \in R^{2h}$. At each iteration of the decoder, we start by concatenating the coattention encodings corresponding to the previous start-word and end-word predictions, s_{i-1}, e_{i-1} into one vector $[u_{s_{i-1}}; u_{e_{i-1}}] \in R^2h$. We feed this, along with the previous hidden state and cell state, $h_{i-1}, c_{i-1} \in R^h$, into an LSTM, which returns the next hidden and cell states, h_i, c_i. As the standard, we set s_0 to be 0 and e_0 to be the index of the last word in the document.

Then, for each index t in the document, we compute a score for the start, $\alpha_t \in R$ and a score for the end $\beta_t \in R$. These are computed using two separate HMNs with the following architecture, passing in the coattention encoding $u_t \in R^{2h}$ the current hidden state $h_i \in R^h$, and the aforementioned coattention encodings $u_{s_{i-1}}, u_{e_{i-1}} \in R^{2h}$:

$$r = \tanh(W_0[h_i; u_{s_{i-1}}; u_{e_{i-1}}]),\ r \in R^h, W_0 \in R^{h \times 5h}$$

$$m_1 = \max(W_1[u_t; r] + b_1),\ m_1 \in R^h, W_1 \in R^{p \times 3h \times h}, b_1 \in R^{p \times h}$$

$$m_2 = \max(W_2 m_1 + b_2),\ m_2 \in R^h, W_2 \in R^{p \times h \times h}, b_2 \in R^{p \times h}$$

$$\mathbf{HMN}_{out} = \max(W_3[m_1; m_2] + b_3),\ \mathbf{HMN}_{out} \in R, W_3 \in R^{p \times h \times 2h}, b_2 \in R^{p \times h}$$

Note that the use of ; denotes concatenation, and that p is the size of our maxout pool, with each use of the maximum function taken across the first non-batch dimension. We can then take a softmax across the scores for each index to generate a probability distribution for each word being the start token and the end token.

Lastly, we implement character embeddings. This allows our model to condition on the morphology of words in order to predict the meaning of out-of-volcabulary (OOV) words. GloVE (which our SQuAD baseline model uses) assigns random vector values to OOV words. The character embeddings, instead, find a vector representation of the OOV words by looking at their character level compositions, which improves accuracy by reducing randomness in our predictions.

We implement character embeddings by implementing a randomly initialised vector for each character index in our regular Embedding class. Next, we apply a dropout layer and 2D Convolutional Neural Network to the embeddings. This helps us determine the vector representations of words by looking at their character compositions, and extracting meaning from segments of a word. We apply the max function over the width of these embeddings to ensure a fixed-size vector for every word, and concatenate our resulting embeddings with our word embeddings. Next, we continue as the BiDAF model does with its word embeddings by applying a linear projection layer and highway network to our concatenated character and word embeddings.

Since all three of our implementations were implemented in different parts of our model (the Attention class, Decoder class, and Embeddings class), it is easy to "turn" the techniques on and off at different times, depending on the experiment we are running, without affecting the other parts. We do this by using different git branches for the character embeddings, and by calling different choices of attention and decoder classes into our model depending on the on the test.

5 Experiments

5.1 Data

We trained our model on a portion of the publicly available SQuAD training dataset. The dev set and test set are non-overlapping subsets of the publicly available SQuAD evaluation dataset. These are all elaborated on further in the default project handout.

5.2 Evaluation method

We evaluated our test set on the test data, as described above and in the project handout, and the specific metrics used are the negative log likelihood (NLL), F1 (the harmonic mean of precision and recall), EM (rate of matching exactly the provided answer), and AvNA (rate of correctly predicting whether there was an answer in the context or not). We compared these metrics following the same implementation of techniques across the two different baselines.

5.3 Experimental details

We ran 8 different experiments. First, we ran our two baselines (SQuAD standard baseline, and SQuAD standard with coattention baseline) on their own. Next, we ran three tests on each baseline: one with the dynamic decoder and highway network, one with the dynamic decoder, highway network and character embeddings, and one with only character level embeddings.

We also ran a few experiments varying the specifics of the dynamic decoder: specifically the number of iterations for which the decoder ran and the computation of the loss function across the different iterations.

We consistently used a learning rate of .5 and the batch size varied from 32 to 80 depending on the number of iterations we were running the decoder for and the associated memory constraints.

5.4 Results

Attention	Decoder	Char Embeds.	NLL	F1	EM	AvNA
BiDAF	Standard	No	3.06	61.54	58.16	68.31
Coattention	Standard	No	3.61	54.76	51.57	63.50
BiDAF	Dynamic	No	4.06	53.95	51.67	62.31
Coattention	Dynamic	No	3.69	53.65	50.36	61.40
BiDAF	Standard	Yes	3.02	63.65	60.39	69.58
Coattention	Standard	Yes	3.57	55.31	51.42	64.80
BiDAF	Dynamic	Yes	3.89	54.03	51.84	62.33
Coattention	Dynamic	Yes	3.48	57.25	54.06	64.59

Table 1: Numerical Results

Note: Each instance where it is indicated we use the dynamic decoder, we also use the highway network. On the test leaderboard we submit our best performing model, which was BiDAF with Character Embeddings. This obtained F1 score of 63.90 and EM score of 60.36 on the test leaderboard.

Our results on models using coattention are worse than we expected. The fact that dynamic decoder and highway network compliment coattention to improve performance was excpected, and the positive effect across all models of character embeddings was not expected. This tells us that our approach in having both a BiDAF baseline and a coattention baseline was useful, as it allowed us to compare effects across baselines and see e.g. that the dynamic decoder only improves the baseline when matched with coattention.

6 Analysis

We found considerable degradations in performance with both the coattention network and the Dynamic Decoder. Our two best results came from BiDAF attention and the standard, single-pass decoder, with considerable improvement using additional character embeddings instead of only the pre-trained GLoVE vectors.

There are a number of possible explanations for this. One is access to resources. The paper from which we borrowed our primary model recommended that the model run the decoder for 4 iterations before settling on its predictions for the span. However, with the resources that we had access to, running the decoder for 4 iterations would have used up so much memory as to require reducing the batch size to 8, which would require us to spend a full day each time we trained the model, which would have simply unfeasible given the constraints of the project. This perhaps indicates that coattention is more effective on larger-scale projects with more access to memory or with more training time.

Additionally, the original published paper that first introduced coattention introduced as a novel way to calculate attention between two separate vectors – in their case, between a feature vector for the image and a vector of word representations. However, the baseline model we worked from already utilized BiDAF, which also computes attention based on the relationships between the words in the document and the words in the question, although it does so in fewer layers and with a more simplistic model than the coattention calculations. Thus, coattention is less of a novel improvement than it once was.

Perhaps the final reason we were not able to match their positive results with coattention or with the Dynamic Decoder is the sheer number of factors that come into play, including but not limited to: the learning rate, the choice of optimizer, the batch size, the number of epochs, and the default start and end predictions initially fed into the LSTM. There are only so many trials that can be run and so many tweaks that can be made, so it is possible that some combination of the above parameters would have resulted in a successful improvement on the baseline.

In examining the effects of the other parameters, we see that coattention either improves or maintains the performance of the dynamic decoder, while it majorly reduces the performance of the standard single-pass decoder. This makes a lot of sense, as the dynamic decoder was designed to work in tandem with a coattention layer, but the single-pass decoder was designed to take the BiDAF as input. On the other hand, the dynamic decoder had a more neutral effect on coattention overall and a distinctly negative impact on models that utilized BiDAF. As mentioned above, this is perhaps the result of inoptimal parameters for the decoder, such as the number of passes or the default start and end predictions.

The final result worth noting is that character embeddings were a consistent improvement regardless of the combination of attention model and decoder, with the most drastic improvements occurring in the case of BiDAF with the single-pass decoder and coattention with the dynamic decoder. This result should not be surprising, as the advantage provided when encountering out-of-vocabulary words is considerable.

As mentioned above, we also ran a number of tests varying the number of iterations for the dynamic decoder or the dynamic decoder's loss function, but these ultimately did not lead to significant variations in results, except for a few extreme examples where an error in implementation resulted in especially low metric results.

7 Conclusion

We find that implementing coattention on a simple BiDAF model significantly decreases performance across all metrics; NLL, F1, EM and AvNA. Implementing the Dynamic Decoder alongside coattention improves its performance, however, not enough to reach BiDAF baseline performance. We also find that the dynamic decoder and highway network lower performance on a BiDAF model without coattention, which indicates that the function of the dynamic decoder and highway network complement coattention but not BiDAF attention. We suspected this result, as the work of Caiming Xiong on coattention on the state of the art SQuAD model emphasised the importance of their combined implementation.

We find that implementing character embeddings improves performance significantly on all models we tested. It had the largest effect on F1 scores in the BiDAF model without the dynamic decoder and highway network, whose score it increased by over 2 points. On all other models it increased performance by less than 1 point, which indicates that coattention networks and dynamic decoders implemented with highway networks may decrease the effectiveness of character embeddings on a simple model.

Our chief achievements are discovering the extent to which coattention is detrimental on a simple BiDAF network, as well as showing that increase in performance by character embeddings implementation is lower on a simple model with coattention than it is on one with BiDAF.

One of our primary limitations is the lack of variations in models that we test coattention on. Although we suspect that coattention is ineffective on across simple SQuAD models, we were only able to test it on one standard baseline, and only investigated three common Question Answering model techniques. Future work could go through the techniques applied in the previous state of the art model that coattention improved in order to find out for which techniques coattention improves a model when implemented alongside them. This may create an avenue for more concise recommendations when implementing a coattention layer.

We were also limited by the amount of iterations that the Dynamic Decoder was able to accommodate on our virtual machine. This is due to limited storage on our virtual machine. Our models in the results section were run with a Dynamic Decoder using 2 iterations. We also ran tests with one iterations and with three iterations and noticed minimal effects. An interesting area for

for future work would be to run the dynamic decoder together with coattention on higher iterations. Since the Dynamic Decoder improved the results of coattention, it is possible that higher iterations of the Dynamic Decoder would yield results in line or higher than the BiDAF baseline.

References

"CS 224N Default Final Project: Building a QA system (IID SQuAD track)." (2022).

Lu, Jiasen, Jianwei Yang, Dhruv Batra, and Devi Parikh. "Hierarchical Question-Image Co-Attention for Visual Question Answering." arXiv preprint arXiv:1606.00061 (2017).

Seo, Minjoon, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. "Bi-Directional Attention Flow for Machine Comprehension." arXiv preprint arXiv:1611.01603 (2018).

Xiong, Caiming, Victor Zhong, and Richard Socher. "Dynamic coattention networks for question answering." arXiv preprint arXiv:1611.01604 (2018).