Reading Comprehension with the SQuAD

Hugo Valdivia
Department of Computer Science
Stanford University
Stanford, CA 94305
hugov65@stanford.edu
Codalab: valdivia4

Miguel Garcia
Department of Computer Science
Stanford University
Stanford, CA 94305
miguel16@stanford.edu
Codalab: miguel16

Abstract

For this reading comprehension task with the SQuAD, we've looked at two high-achieving models from the literature: Bi-Directional Attention Flow (BiDAF) and Coattention. With respect to our two main evaluation metrics, F1 and EM, our best model closely followed the architecture of the original BiDAF model and received a 74.4% F1 score and a 64.7% EM score.

1 Introduction

SQuAD is a reading comprehension dataset [1]. The task is to answer questions about provided documents. This task is difficult and sometimes impossible for developing deep learning models for because of constraints on available question answering datasets, both in quality and quantity. However, SQuAD presents us with a vast, crowd-sourced dataset that has already allowed us to make advances in developing expressive models for this task. We drew inspiration from two state-of-the-art models, coattention and BiDAF, to augment a baseline model and compare results.

2 Background and Related Works

The Dynamic Coattention Network (DCN) is a state-of-the art deep learning model for accomplishing the SQuAD task [2]. It takes advantage of the intrinsic relationship between questions and answers to by fusing them into codependent representations through an encoding process (which is described as one of our approaches). After this step, the attention outputs are passed on to a dynamic pointing encoder, which utilizes an iterative process to recover from local maxima: the dynamic pointing encoder switches between start and end index prediction with a Highway Maxout Network that pools across different model variations, effectively accounting for variations in question and answer types. The process only halts when the estimated indexes no longer change, or upon a maximum number of iterations. We decided to, in a sense, ablatively analyze the effectiveness of the coattention attention layer itself by inserting it into our baseline model.

Another high-performing deep learning model for this machine comprehensions task is known as Bi-Directional Attention Flow (BiDAF). The main idea of the BiDAF model is that it creates a context representation that makes use of different degrees of information (character, word, and context), is aware of the query, and does not summarize the information early [3]. The attention flows from the attention layer to the modeling layer, and the authors believe that this allows for the work to be split between the attention and the modeling layers [3].

3 Approach

3.1 Baseline

This baseline approach is thoroughly described in the assignment handout. The summary of the methods in this section will all be taken from the project handout [1]. The provided baseline model can be broken down into 4 layers, which are as follows:

1. Word Embedding Layer
2. RNN Encoding Layer
3. Basic Attention Layer
4. Output Layer

The word embedding layer takes words and maps them to pre-trained GloVe vectors; these word vectors are meant to capture meaningful syntactic and semantic relationships between words [4]. For each (context, question, answer) 3-tuple which makes up a single example, the context and question are turned into lists of GloVe embeddings. These embeddings then serve as inputs to a bidirectional GRU, which returns a list of forward and backward hidden states. These are then concatenated to give us the context hidden states and the question hidden states. Now, with these as inputs, the basic attention layer uses dot product attention. At the end of this layer, the context hidden states are concatenated with the attention outputs to get blended representations. The last layer takes these blended representations, passes them through a fully connected layer and runs them through a ReLU function. To compute the start position distribution, it then assigns a score to each word in the context and then passes this through a softmax layer to get a probability distribution. The same technique is used to get a distribution for the end position.

3.2 Coattention

This model follows closely the coattention layer described in Xiong et al. (2016) and implements it using the baseline model's architecture. Essentially, we substitute out the baseline attention layer for a coattention layer that involves a second-level attention computation. We call this the Slightly Less Dynamic Coattention Network, or SLDCN.

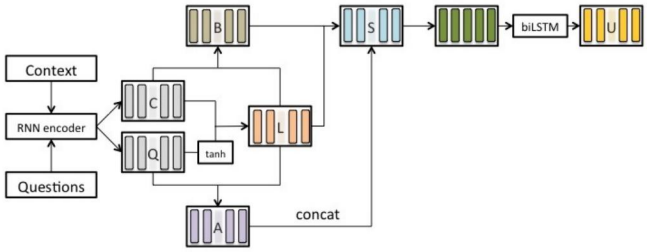

Figure 1: Coattention layer

The first difference between this model and the baseline happens when we apply a linear layer with tanh nonlinearity to the question hidden states to compute projected hidden states:

$$\boldsymbol{q_i'} = tanh(\boldsymbol{W}\boldsymbol{q_i} + \boldsymbol{b}) \in \mathbb{R}^l \qquad \forall i \in \{1, \ldots, M\}$$

After this step, we add trainable sentinel vectors $\boldsymbol{c_\emptyset}, \boldsymbol{q'_\emptyset} \in \mathbb{R}^l$ to each row of context and question hidden state matrices $\boldsymbol{C}$ and $\boldsymbol{Q'}$, respectively. Using these encodings, we compute the affinity matrix $\boldsymbol{L} = \boldsymbol{C}^T\boldsymbol{Q'} \in \mathbb{R}^{(N+1)\times(M+1)}$. Now we compute the Context-to-Question (row-wise) and Question-to-Context (column-wise) attention outputs.

For Context-to-Question (C2Q), we use the attention distribution $\boldsymbol{L_r} = softmax(\boldsymbol{L})_{row-wise}$ to compute the attention output $\boldsymbol{A}$ such that $\boldsymbol{A} = \boldsymbol{L_r}\boldsymbol{Q'} \in \mathbb{R}^{(N+1)\times l}$.

For Question-to-Context (Q2C), we use the attention distribution $\boldsymbol{L_c} = softmax(\boldsymbol{L})_{column-wise}$ to compute the attention output $\boldsymbol{B}$ such that $\boldsymbol{B} = \boldsymbol{L_c^T}\boldsymbol{C} \in \mathbb{R}^{(M+1)\times l}$.

Next, we compute the second-level attention outputs $\boldsymbol{S} = \boldsymbol{L_r}B \in \mathbb{R}^{N \times l}$. Note that the row corresponding to sentinels has been masked out. Finally, we concatenate $[S; A]$ (row-wise) and feed this input into a biLSTM to obtain the coattention encodings in $\boldsymbol{U} \in \mathbb{R}^{N \times 2l}$. This is then passed onto the rest of the baseline architecture and the model continues training.

3.3 BiDAF

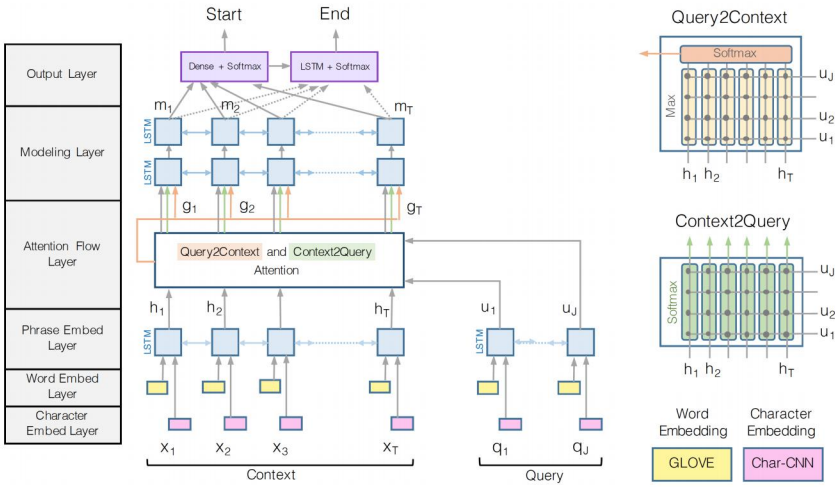

Figure 2: BiDAF Architecture [5]

We very closely follow the approach given in [3]. The summary that makes up this section will be taken from the original BiDAF publication [3] and from the default project handout [1]. For the BiDAF model, we have a 6 layer architecture, which is as follows:

1. Character Embedding Layer
2. Word Embedding Layer
3. Contextual Embedding Layer
4. Attention Flow Layer
5. Modeling Layer
6. Output Layer

For this project, we've implemented all of these layers, with the exception of the character embedding layer. The word embedding layer is precisely as it is for the baseline. From here, we get two matrices $C \in \mathbb{R}^{2h \times N}$ for the context and $Q \in \mathbb{R}^{2h \times M}$. After this, the attention flow layer tries to capture the interactions between the context and question words. The inputs to this layer are the previously mentioned C and Q. This layers computes two attentions: context-to-question and question-to-context. Given context hidden states $c_1, c_2, ..., c_N \in \mathbb{R}^{2h}$ and question hidden states $q_1, q_2, ..., q_M \in \mathbb{R}^{2h}$, we compute a similarity matrix $S \in \mathbb{R}^{N \times M}$. The elements of S are given by:

$$S_{ij} = w_{sim}^{\top}[c_i; q_j; c_i \circ q_j]$$

where $w_{sim} \in \mathbb{R}^{6h}$ is a trainable weight vector. Once S is computed, we can compute the context-to-question attention in a manner quite like is done in the baseline. The attention distributions α^i are computed by:

$$\alpha^i = softmax(S_{i,:})$$

These are then used as weights to compute the context-to-question attention outputs a_i:

$$a_i = \sum_{j=1}^{M} \alpha_j^i q_j$$

Then, to compute the question-to-context attention output c', we follow the following sequence of equations:

$$m_i = max_j S_{ij}$$
$$\beta = softmax(m)$$
$$c' = \sum_{i=1}^{N} \beta_i q_i$$

In words, we take the max over the rows of the similarity matrix, pass this through a softmax layer, and then use these as weights to compute c'.

To combine these two attentions, we get our blended representations b_i as follows:

$$b_i = [c_i; a_i; c_i \circ a_i; c_i \circ c']$$

We have one for each $i \in \{1, 2, ..., N\}$, and we combine these into a matrix $G \in \mathbb{R}^{8h \times N}$.

This G is the input to the next layer, the modeling layer. In this layer, we run G through two layers of bi-directional LSTMs to obtain $M_1 \in \mathbb{R}^{2h \times N}$. We also pass this M_1 through one last bi-directional LSTM layer to get M_2.

Now, we move to the last layer, the output layer. With M_1, we compute the probability distribution for the start index as follows:

$$p^{start} = softmax(w_{p^{start}}^{\top}[G; M_1])$$

Similarly, with M_2, we compute the probability distribution for the end index:

$$p^{end} = softmax(w_{p^{end}}^{\top}[G; M_2])$$

These are then used to compute start and end positions that are used at test time. In our case, we use the argmax of the distributions to compute the start and end indices.

4 Experiments

The following contains a description about the dataset, our various model configurations, the evaluation metrics used, and the results obtained.

4.1 Dataset

Training and evaluation was done over the SQuAD reading comprehension dataset, released by Rajpukar et al. (2016), which is split into context, question, and answer triples. The contexts are excerpts from Wikipedia, and the answers to the about 100k questions are spans within these contexts. The SQuAD dataset constrains the answer space by requiring answers to be spans from contexts, and it prevents the model from learning a different task (such as named-entity recognition) by asking questions varied in the type of reasoning needed to answer [2].

4.2 Model configurations

Across all models, the maximum gradient norm was 5.0, the learning rate was kept at 0.001, maximum context length was 600, maximum question length was 30, and the word embedding size was 100. All models utilized the Adam optimizer. The coattention layer utilized more memory than our VM provided to compute second level attention outputs. We experimented with a smaller batch size and the smallest embedding size, but eventually had to upgrade our VM to a more powerful machine. The following table shows the hyperparameters that we tuned through trial and error (and for BiDAF, following the advice from the original publication [3] as much as possible).

Model	batch size	hidden size	dropout (drop probability)
Baseline	100	200	0.15
Coattention	50	200	0.15
BiDAF	60	100	0.2

4.3 Evaluation metrics

We used F1 and EM scores to evaluate the performance of our models. As defined in the assignment handout, F1 is the harmonic mean of precision and recall, and EM, or exact match, is a binary measure of whether predicted answers exactly match the ground truth answer [1]. Evaluation is made more forgiving by allowing the model to choose any 1 of 3 correct ground truth answers.

Model	F1 %	EM
Baseline	37.27	27.10
BiDAF (Attention Only)	43.31	31.87
Coattention	66.917	55.009
BiDAF (Our Implementation)	74.437	64.754

The values for the first two models come from our local Tensorboard data and could differ by a few percentage points when pushed onto the leaderboards. The values for the last two models come from the official test leaderboard.

4.4 Results

Show below are the Tensorboard graphs of the model loss and F1/EM scores across our models running on the dev dataset.

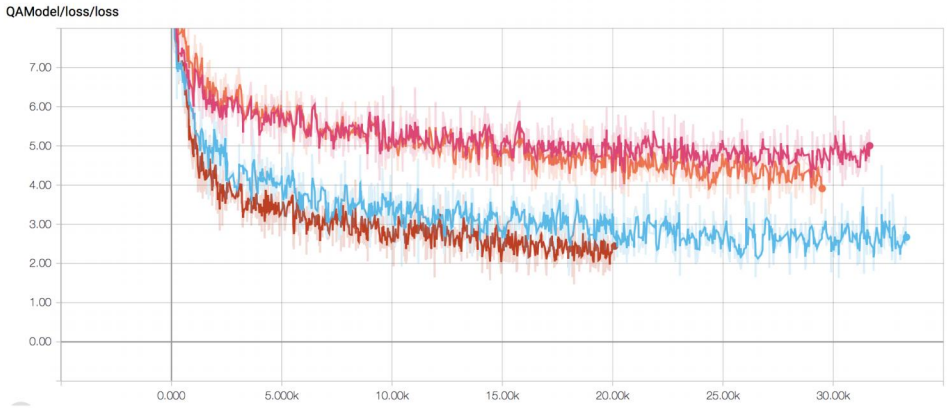

Figure 3: Model loss vs number of iterations for our BiDAF implementation (burgundy), baseline w/coattention (blue), baseline w/ BiDAF attention (orange), and baseline (pink) models.

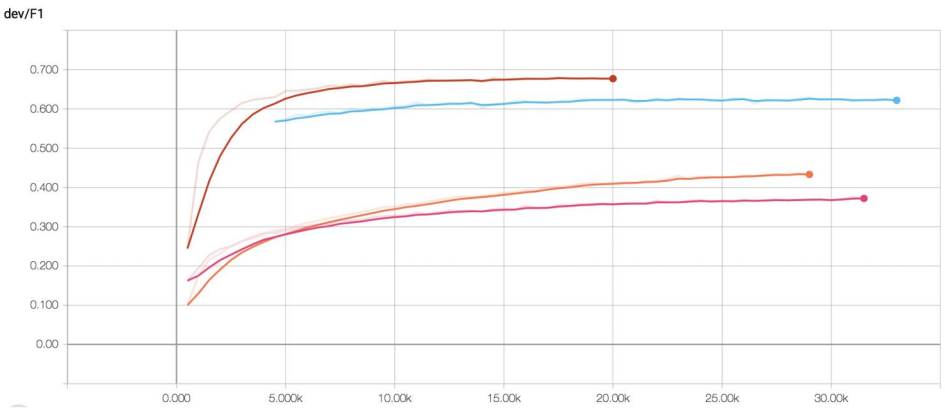

Figure 4: F1 scores for our models across the dev set

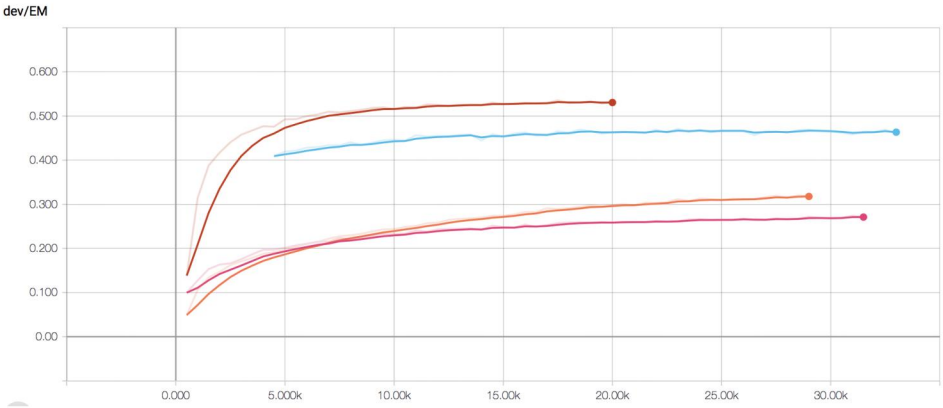

Figure 5: EM scores for our models across the dev set

4.5 Error Analysis

We performed an analysis similar to the one done in [3], with a few categories changed. Given the examples that we saw from our models, we thought the errors could be better classified in the table attached:

Error Type	Baseline w/coattention %	BiDAF %	Examples (25 examples sampled for each model)
Imprecise (true) answer boundaries	52	48	**Context:** "... summing these component forces using vector addition yields the original force . resolving force vectors into components of a set of basis vectors is often a more mathematically clean way to describe forces than using magnitudes and directions . this is because , for orthogonal components , the components of the vector sum are uniquely determined by the scalar addition of the components of the individual vectors ." **Question:** "what are the independant components of a vector sum that has been determined by scalar addition of individual vectors ?" **True Answer:** "orthogonal" **Predicted Answer:** "yields the original force . resolving force vectors into components of a set of basis vectors is often a more mathematically clean way to describe forces than using magnitudes and directions . this is because , for orthogonal components , the components of the vector sum are uniquely determined"
Syntactic complications and ambiguities	36	44	**Context:** "during that year , tesla worked in pittsburgh , helping to create an alternating current system to power the city 's streetcars . he found the time there frustrating because of conflicts between him and the other westinghouse engineers over how best to implement ac power . " **Question:** "what did tesla work on in 1888 ?" **True answer:** "system to power the city 's streetcars" **Predicted answer:** "pittsburgh"
(Plausible) answer, different boundaries	8	4	**Context** "while genghis khan never conquered all of china , his grandson kublai khan completed that conquest and established the *yuan* dynasty that is often credited with reuniting china . there has been much artwork and literature praising genghis as a great military leader and political genius . the years of the mongol-established yuan dynasty left an indelible imprint on chinese political and social structures for subsequent generations with literature during the jin dynasty relatively fewer ." **Question:** "what chinese dynasty did the mongols found ?" **True Answer:** "*yuan*" **Predicted Answer**: "mongol-established yuan dynasty"
Incorrect preprocessing	4	4	**Context**: "on january 7 , 2016 , pepsi confirmed to the associated press that beyonce , who headlined the super bowl xlvii halftime show and collaborated with coldplay on the single " hymn for the weekend " , would be making an appearance . bruno mars , who headlined the super bowl xlviii halftime show , and mark ronson also performed . **Question:** on january 7 , 2016 , it was confirmed that which start would join coldplay for the halftime show ?" **True Answer:** "beyonce" **Predicted Answer:** "pepsi"

4.6 Evaluation of Results

As we see in our results, the baseline model with the coattention layer far surpasses basic attention when we take a look at their F1-EM scores of 66.917%-55.009% and 37.27%-27.10%, respectively. This is expected because the coattention outputs are an improvement over the simple dot-product attention in the baseline. However, baseline model with coattention still trails the high performing full coattention implementation done by Zhong et al. (2016), which reached F1 and EM scores of 75.6% and 65.4% with a pool size of 16 for the Highway Maxout Network.

This approximate 10% gap in scores is probably explained by the inclusion of the HMN in the architecture that follows the coattention layer. We performed a rudimentary test to see whether the difference was in the architecture before the coattention layer, namely in encoding the question and context hidden states that are passed to coattention. The baseline obtains these hidden states with a GRU cell, while [2] accomplished their results with an LSTM. Our results interestingly show that the LSTM is subpar to the GRU implementation, even though it deviates from that in the paper. However, more tests would need to be performed to provide thorough analysis.

Turning to BiDAF, quite differently, we see that swapping out the basic attention layer with the BiDAF attention layer in the baseline implementation does not lead to huge gains in performance. We see a jump from 37.27 to 43.31 in F1 and from 27.10 to 31.87 in EM. This highlights the importance of the rest of the layers that share the work in the BiDAF. Taking out the modeling layer, in particular, led to huge drops in performance. In working on this project, a ton of time was spent solely trying to debug the BiDAF attention layer since we felt as though there had to be a bug if the results were practically the same as the baseline's. The gap of a few performance points from the original paper (F1 of 77.3, EM of 68.0) comes from the things that were left for us to implement; additionally, through more thorough testing and tuning and taking inspiration from the common errors, we could better fit our model to this particular dataset whilst possibly sacrificing performance on similar tasks.

5 Conclusions

For this project, we implemented features from two high-achieving architectures found in the literature: the BiDAF and Coattention models. In the end, our best-performing model achieved an F1 score of 74.437 and an EM score of 64.754.

In terms of future improvements, the next path that we would take is to create an ensemble model. As was shown in the BiDAF paper, this could lead to a big improvement in the EM score and an improvement in the F1 score [3].

Acknowledgments

We would like to thank the teaching staff for making this class a possibility.

References

[1] "Cs 224n default final project: Question answering." `http://web.stanford.edu/class/cs224n/default_project/default_project_v2.pdf`. Accessed: 2018-03-18.

[2] C. Xiong, V. Zhong, and R. Socher, "Dynamic coattention networks for question answering," *CoRR*, vol. abs/1611.01604, 2016.

[3] M. J. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *CoRR*, vol. abs/1611.01603, 2016.

[4] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[5] "Bidaf." `https://allenai.github.io/bi-att-flow/BiDAF.png`. Accessed: 2018-03-18.