Neural Question Answering

Aneesh Satya Pappu
Department of Computer Science
Stanford University
apappu@cs.stanford.edu

Rohun Saxena
Department of Computer Science
Stanford University
rohun@cs.stanford.edu

Abstract

In this paper we approach the task of question answering on the SQuAD dataset. We built a variety of neural-network based models and compare and contrast the performances of each. Our best performing model was a deep neural network consisting of a bidirectional LSTM encoder, an exact match feature extractor, a bidirectional attention flow (BiDAF) and coattention layer, a modeling layer, and a smarter span selection filter. This model achieved a performance of 74.703 F1 and 63.936 EM on the dev leaderboard set.

1 Introduction

In 2016 the Stanford NLP group released the Stanford Question Answering Dataset (SQuAD), a new reading comprehension dataset consisting of 100,000+ questions. These questions were sourced from Wikipedia articles where the answer to the question is in a passage from the article and the answer was hand-annotated. The dataset was created with the motivation of developing question and answering models for Machine Comprehension to help bridge the gap between the performance of logistic regression (F1-51.0%) and the human performance (F1-86.8%) [1]. The goal of our project is to build a neural network that takes a question and a context and is able to identify the answer to the question in the context with comparable results to the human performance.

Our paper discusses the specifics of the problem, the models we experimented with as well as a discussion of the results and analysis of our best model. We will discuss some of the insights we gained and challenges we ran into in the process of building a Question Answering (QA) model and then some future steps to further explore.

2 Problem and Background

As mentioned earlier the challenge is to identify the span of words in a provided context that contains the answer to a question. We limited the vocabulary size to 400,000 commonly seen tokens and used the GloVe word vector representations trained on 6 billion words from Wikipedia and Gigaword [4]. Our model will be a function that takes the question and returns the scalar indices of the start and end words that mark the answer to the question in the context.

The approaches to question answering that we consider in this paper are the BiDAF [2] and Dynamic Coattention [3] models. BiDAF is a type of attention layer that generates query-aware representations of the context, resulting in state-of-the-art performance on the problem of question answering, having achieved an F1 score of 77.3 on the SQuAD dev dataset (single model).

Coattention is another state-of-the-art approach to attention. Similarly to BiDAF, coattention calculates a two-way attention between the context and query representations. However, coattention also calculates a second level attention computation over the attention representations calculated in

the first level (i.e. attending to attention outputs). This resulted in excellent performance as well, achieving an F1 score of 75.9 on the SquAD dev dataset (single model).

2.1 Dataset

Using the SQuAD dataset from Rajpurkar et al. we trained on sample training set of about 85,000 examples of questions-context pairs and then separated a development set to test our models on about 10,000 examples question-context pairs. The histograms below show the nature of the dataset, specifically highlighting the distribution of lengths of the contexts, questions, and answers

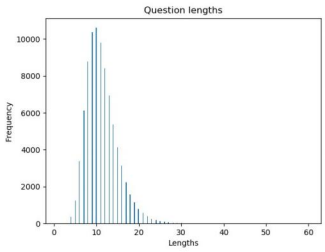

Figure 1: Question lengths in SQuAD

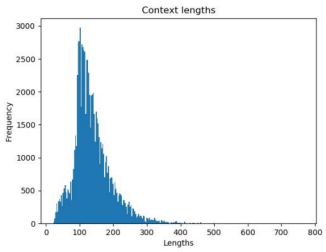

Figure 2: Context lengths in SQuAD

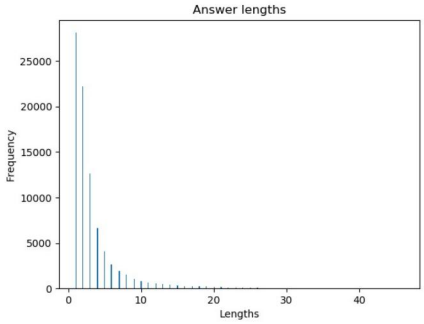

Figure 3: Answer lengths in SQuAD

From these figures we can see the nature of the data we are working with. In general we can see that the majority of questions are in the range of 5-25 words, most contexts are between 75 and 150 words, and that most answers are between 0 and 10 words. We used this information to set the length of the context representation to be at most 450 words and if the context is less to be zero padded. Similar for questions we set the max number of words represented to be 30 and again zero padded if the question was shorter than 30 words.

3 Model

We incrementally implemented variations on the baseline model, tested them individually, and concatenated/swapped out parts to see which models would yield the best results (e.g. switching a BiDAF attention layer for a coattention layer and then ultimately concatenating them).

3.1 Architecture

Starting from the baseline model, we implemented several changes and conducted ablation studies to understand how each part of our implementation helped the overall performance. Our additional implementations to the baseline consisted of using a bidirectional LSTM encoder (as opposed to

the standard GRU encoder), adding a binary exact match feature to indicate whether a word in the context also appeared in the question, a dual coattention and BiDAF attention layer, a two-layer stacked bidirectional LSTM modeling layer and a simple algorithm for smarter span selection. Details of each implementation (in addition to the core remaining parts of the original baseline model) are explained below.

Embedding Layer As mentioned earlier we first feed in the pretrained GloVe embeddings which are 100-dimensional vectors (the default). We decided not to use a convolutional character encoding model after analyzing the ablation results from the original BiDAF paper and seeing relatively low payoff for the large added training time incurred. The embedding layer performs a lookup for both the context and questions, each padded to the their respective uniform lengths (450 and 30 respectively).

Exact Match Features We then added features for context tokens t_i for $i \in \{1, ..., N\}$ that are binary indicator features representing whether token t_i is also found in the question. These binary features are concatenated to the pretrained embeddings that are looked up in the previous layer.

Contextual Layer These new embeddings are then fed through a single layer bidirectional LSTM to obtain hidden representations. The bidirectional LSTM produces a series of forward and backward hidden states for each of the embeddings – we concatenate the forward and backward outputs to produce the layer's output hidden representation for each embedding. This results in matrices $\mathbf{C} \in \mathbb{R}^{Nx2d}$ and $\mathbf{Q} \in \mathbb{R}^{Mx2d}$ where N is the number of contexts, M is the number of questions, and d is the hidden layer size.

Attention Layer The attention layer consists of a BiDAF layer and a Coattention layer. The output of these two attention layers is concatenated to yield the final attention output.

BiDAF The first attention output is calculated using a bidirectional attention flow model to calculate Context-to-Question (C2Q) and Question-to-Context (Q2C) attention. First, a similarity matrix $\mathbf{S}$ is calculated which contains similarity scores for every pair of context and question hidden states where each similarity scores is calculated as $\mathbf{S}_{ij} = \mathbf{w}_{sim}^T[\mathbf{c}_i; \mathbf{q}_j; \mathbf{c}_i \circ \mathbf{q}_j]$ where $\mathbf{w} \in \mathbb{R}^{6d}$ is a trainable weight vector.

To perform C2Q attention we take a row-wise softmax to obtain an attention distribution across all of the questions for each context. This distribution is then used to take a weighted sum of the question hidden states to yield an attention output a_i for each context:

$$\alpha^i = softmax(\mathbf{S}_{i,:}) \in \mathbb{R}^M$$

$$\mathbf{a}_i = \sum_{j=1}^{M} \alpha_j^i \mathbf{q}_j \in \mathbb{R}^2 h$$

Afterwards we perform Q2C attention. For each context location we take the max of the corresponding row in the similarity matrix, take a softmax over the resulting vector of row-wise maximum values, and use this as a weighted sum of the context hidden states.

$$\mathbf{m}_i = \max_j \mathbf{S}_{ij} \in \mathbb{R}$$

$$\beta = softmax(\mathbf{m}) \in \mathbb{R}^N$$

$$\mathbf{c}' = \sum_{i=1}^{N} \beta_i \mathbf{c}_i \in \mathbb{R}^{2h}$$

And finally, we obtain the output $\mathbf{b}_i$ by concatenating the context hidden state $\mathbf{c}_i$, the C2Q attention output $\mathbf{a}_i$, and the Q2C attention output $\mathbf{c}'$ to obtain

$$\mathbf{b}_i = [\mathbf{c}_i; \mathbf{a}_i; \mathbf{c}'] \in \mathbb{R}^{6h}$$

Coattention The second attention output we built was coattention, which is another two-way attention layer between the context and the question [3]. The coattention layer takes the context hidden states $\mathbf{c}_1, ..., \mathbf{c}_N \in \mathbb{R}^l$ and hidden states $\mathbf{q}_1, ..., \mathbf{q}_M \in \mathbb{R}^l$. The first step is to apply a tanh non-linearity to the question hidden states to get the projected hidden states $\mathbf{q}'_1, ..., \mathbf{q}'_M$. Then we add two trainable parameters called sentinel values, one to the context hidden states and the other to the projected question hidden states resulting in the following states, $\{\mathbf{c}_1, ..., \mathbf{c}_N, \mathbf{c}_\emptyset\}$ and $\{\mathbf{q}_1, ..., \mathbf{q}_N, \mathbf{q}_\emptyset\}$ where $c_\emptyset$ and $q_\emptyset$ are the trainable sentinel values. Next we form an affinity matrix $\mathbf{L} \in \mathbb{R}^{(N+1)\times(M+1)}$ and $\mathbf{L}_{ij} = \mathbf{c}_i^T \mathbf{q}'_j \in \mathbb{R}$. At this point we will create the C2Q and Q2C attention outputs. C2Q is calculated as follows:

$$\alpha^i = softmax(\mathbf{L}_{i,:}) \in \mathbb{R}^{M+1}$$

$$\mathbf{a}_i = \sum_{j=1}^{M+1} \alpha_j^i \mathbf{q}'_j \in \mathbb{R}^{N+1}$$

Then, Q2C is calculated as follows:

$$\beta^i = softmax(\mathbf{L}_{:,j}) \in \mathbb{R}^{N+1}$$

$$\mathbf{b}_i = \sum_{i=1}^{N+1} \beta_i^j \mathbf{q}'_i \in \mathbb{R}^{M+1}$$

Using these attention outputs we then calculate the second attention layer, $\mathbf{s}_i$, by using the C2Q attention distributions α_i to take weighted sums of the Q2C attention outputs $\mathbf{b}_j$

$$\mathbf{s}_i = \sum_{j=1}^{M+1} \alpha_j^i \mathbf{b}'_j \in \mathbb{R}^{N+1}$$

Finally, we concatenate the second-level attention outputs $\mathbf{s}_i$ with the first-level C2Q attention outputs $\mathbf{a}_i$ and feed it as input into a bidirectional LSTM, resulting in the coattention encoding represented as $\{\mathbf{u}_1, ...\mathbf{u}_N\}$ This coattention encoding is the final output of the coattention layer.

Modeling Layer The output of the joint BiDAF-Coattention layer is concatenated to the original context hidden representation and then fed into our modeling layer, which is an implementation of the modeling layer described in the original BiDAF paper [2]. The input matrix is the output matrix of the previous step, the query-attended representations of the context. The modeling layer consists of a two-layer stacked bidirectional LSTM. We concatenate the end forward and backward output states for each hidden representation that is fed in, so the output of this layer is a matrix $\mathbf{M} \in \mathbb{R}^{Nx2d}$

Output Layer The output layer is the same as the baseline model, where a linear downprojection layer is used to calculate a score for each context position using a start and end weight with a softmax applied across the logits for start and end.

Smarter span prediction at test time As opposed to the baseline, which takes an argmax over the context positions to determine the start and end positions of the answer, we implemented a smarter span prediction at test time by imposing constraints. The constraint took the simple form of ensuring that the start position $i \leq j$ where i is the start index and j is the end index. We determined this would be worth implementing to avoid erroneous answers where the argmax results in end positions that are before the start position.

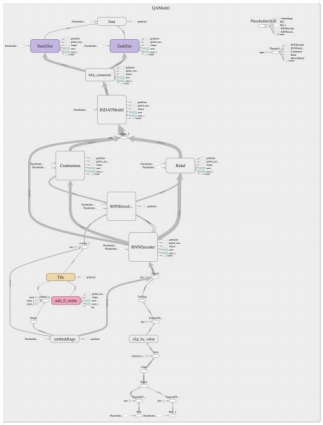

Figure 4: Our Tensorflow Architecture

3.2 Training Details

This section is regarding specific hyperparameters of our model.

Padding We used the standard technique of padding to ensure our context and question lengths are constant when fed to the model. We chose to use a context length of 450 and question length of 30 as shown in figures 1-3.

Optimizer We used the Adam Optimizer with a learning rate set to the default learning rate of 0.001.

Overfitting To counter overfitting, we used dropout and configured the hidden layer size. Our dropout rate was 0.8 in our final models, and we configured our hidden layer size between 100 and 200 before determining that 150 minimized the number of parameters while retaining predictive power.

4 Results and Discussion

Our final model achieved an F1 score of 74.703 and EM score of 63.936 on the dev leaderboard set.

4.1 Ablation Analysis of different components

We implemented our model in 4 parallel sequences from the baseline, iteratively adding parts to each model to understand the improvements in performance that we obtained:

BiDAF – We first implemented the BiDAF attention layer and swapped out the basic attention layer in the baseline model. We then added the modeling layer from the original BiDAF paper [2]. Thereafter, we added on the LSTM encoding, exact match featurizing, and smarter span selection, which we refer to as 'Bidaf+' in the following figure.

After implementing solely the BiDAF attention layer, we obtained a 19 point F1 increase, going from the baseline implementation of 40 F1 to 59.4. After adding modeling, our model's performance boosted to 66.3 F1. Finally, after adding LSTM encoding, exact match, and smarter span, the model improved to 68.4 F1.

Exact Match and LSTM – While implementing BiDAF we also solely implemented exact match and LSTM encoding on top of the baseline. We ran this model to understand the boost we would obtain incorporating LSTM encoding and exact matching.

After implementing LSTM encoding and exact match on top of the baseline, our model's performance increased by 19 points relative to the baseline, improving from 40 to 59.3 F1.

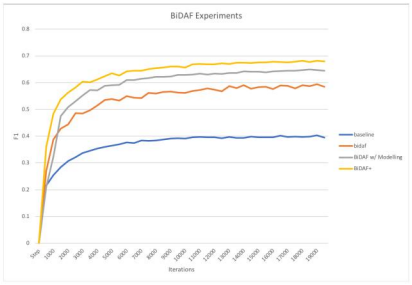

Figure 5: Iterations of BiDAF model and improvements over baseline

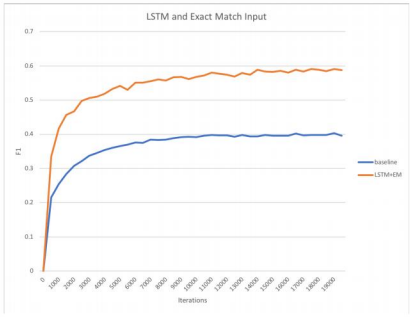

Figure 6: LSTM and Exact Match improvement over baseline

Coattention – We first implemented the Coattention attention layer and swapped out the basic attention layer in the baseline model. We then added on the LSTM encoding ,exact match featurizing, and smarter span selection, which we refer to as 'Coattention+' in the following figure.

After implementing solely the coattention layer, we obtained a 20 point increase in the F1 score over the baseline, going from 40 F1 to 60.1 F1. After adding exact match, LSTM encoding, and smarter span selection, our coattention model's performance improved to 66.5 F1.

Bidaf + Coattention – We then concatenated the BiDAF and Coattention outputs to create our final model. We tuned the hidden layer size, testing at 100, 150, and 200. 200 was too slow to train, and 150 maintained a similar loss trajectory while training slightly faster. Our best performing model was the BiDAF + Coattention implementation with a hidden layer size of 150.

After concatenating the BiDAF and Coattention layers, we ran two experiments, one with hidden size of 100 and another with hidden size of 150. Our 100 hidden size model's performance was equivalent to the performance of our BiDAF+ model at 68.4 Running at a hidden size of 150 got a 1 point F1 increase, improving to our final best dev F1 score of 69.3 (note, this was the score on the local dev set as opposed to the dev leaderboard set).

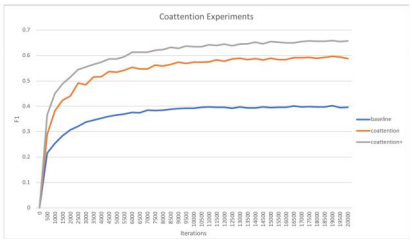

Figure 7: Iterations of Coattention model and improvements over baseline

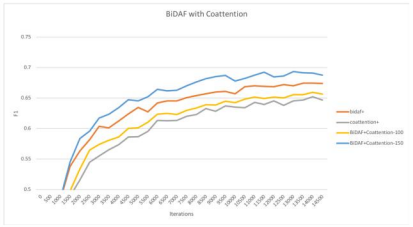

Figure 8: BiDAF + Coattention combined model

4.2 Error Analysis

Looking at the output of our initial BiDAF model we saw that one of the issues was that the answer was weighted towards the end of the context and so we decided to build another attention layer in Coattention with the goal of improving the model's bias towards answers at the end. Here is an example prediction from the first BiDAF model with this issue:

Context: there are infinitely many primes, as demonstrated by euclid around 300 bc. there is no known simple formula that separates prime numbers from composite numbers. however, the distribution of primes, that is to say, the statistical behaviour of primes in the large, can be modelled. the first result in that direction is the prime number theorem, proven at the end of the 19th century, which says that the probability that a given, randomly chosen number n is prime is inversely proportional to its number of digits, or to the logarithm of n.

Predicted Answer: a given, randomly chosen number n
True Answer: the prime number theorem

From this example it is clear that the model chose an answer that was from later in the context as opposed to focusing in on where the answer was earlier on. The Coattention layer we built to help aid this problem on its own performed similar to the BiDAF model but when combined with each other saw a decent improvement in dealing with this attention issue.

This same issue also was the driving factor in switching the RNN Encoder from a bidirectional GRU cell to a bidirectional LSTM cell which are better at bridging longer time gaps between inputs and make earlier parts of the context more relevant.

5 Conclusion and Next Steps

In this paper we implemented a fusion between the attention models of BiDAF and Coattention along with a few other architectural improvements such as exact-matching, LSTM encoding, modeling, and smarter span selection for question answering on the SQuAD dataset. Future work can be done involving deeper models for encoding and use of alternative modeling layers to change the representation of our context-aware question-attended embeddings. Use of deeper alternative modeling layers would allow the embeddings to be more context-aware than the current embeddings. Further hyperparameter tuning of the learning rate, dropout rates, and hidden layer size can also be done, in addition to incorporation of regularization in the loss function to improve generalizability of our model.

Acknowledgments

We'd like to thank the CS224N course staff for working incredibly hard to put together the class and especially the final project. They were amazing at providing quick and helpful feedback on all aspects of debugging and model implementation. Also, a huge shout out to Microsoft for the Azure VM credits – there's no way we would've finished our models without the GPU access.

References

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, SQuAD: 100,000+ Questions for Machine Comprehension of Text, ArXiv e-prints, Jun. 2016.

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

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

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.