CS224N SQuAD Challenge with Bidirectional Attention Flow and Context Features

Adrien Truong
Department of Computer Science
Stanford University
aqtruong@stanford.edu

David Lee-Heidenreich
Department of Computer Science
Stanford University
dleeheid@stanford.edu

Abstract

For our CS224N final project, we tackled the problem of question and answering using the SQuAD dataset as an evaluation of our model. We reimplemented the modeling layer and attention layer from the BiDAF paper as well as incorporated manual features inspired by the DrQA paper. We then experimented with different hyperparameters such as higher dropout rates and learning rates. In the end, on the dev set, we achieved an F1 score of 74.4 and an EM score of 63.7.

1 Background/Related Work

Machine Comprehension is an important research topic in natural language processing that has seen great progress in recent years. Machine Comprehension refers to a computer's ability to find answers to questions about a given text. Specifically, answers to questions are found as substrings of the given context. To facilitate Machine Comprehension research, the Stanford Question Answering Dataset (SQuAD) was created and released and released to the public. SQuAD is a dataset consisting of 100,000+ question-answer pairs on 500+ articles[1] . Additionally, there is a SQuAD leaderboard anyone can submit to track each models' performance. Attention mechanisms have proved remarkably effective in the Machine Comprehension task. The state of the art models have just recently surpassed human performance in the exact match metric using new techniques such as Attention over Attention[2] . In this paper we created our own trained model using the SQuAD dataset for training and evaluation and compared our results to the current state of the art models.

2 Task Definition

Given a context paragraph and a question, our task is to find a substring of the context paragraph that best answers the question. More formally, we define a sequence of words of length M for the context paragraph as $C = \{c_1, c_2, ..., c_M\}$ and a sequence of words of length N for the question as $Q = \{q_1, q_2, ..., q_N\}$. Our model attempts to approximate a function that takes in a context, question pair and outputs a range within the context, $f : (C, Q) \rightarrow (a_{\text{begin}}, a_{\text{end}})$ where $1 \leq a_{\text{begin}} \leq a_{\text{end}} \leq M$.

3 Approach

3.1 Baseline

We started our experiments on the baseline provided by the starter code. The following is a description of the baseline:

RNN Encoder Layer The baseline first converts the question and context tokens into GloVE[3] word embeddings $x_1, x_2, ..., x_M$ and $y_1, y_2, ..., y_N$ respectively. We then feed these embeddings into a bidirectional RNN encoder layer to get $q_i = [\overrightarrow{q_i}; \overleftarrow{q_i}]$ and $c_i = [\overrightarrow{c_i}; \overleftarrow{c_i}]$

Attention layer Then, the encodings go through a basic dot product attention layer. For each context c_i, a question aware context vector b_i is calculated as follows:

$$e^i = [c_i^T q_1, ..., c_i^T q_N] \in \mathcal{R}^N$$
$$\alpha^i = softmax(e^i) \in \mathcal{R}^N$$
$$a_i = \sum_{j=1}^{M} \alpha_j^i q_j \in \mathcal{R}^{2h}$$
$$b_i = [c_i; a_i] \in \mathcal{R}^{4h}$$

Output Layer Each b_i is then fed into a ReLU nonlinearity to get b_i'. Then we obtain logits by passing b_i' through a downprojecting linear layer. Then, we pass our logits through softmax to get a start probability distribution over all the context tokens. We repeat this softmax layer for the end distribution just with different weights.

Prediction We greedily choose:

$$a_{begin} = argmax_{i=1}^N p_i^{begin} \text{ and } a_{end} = argmax_{i=1}^N p_i^{end}$$

3.2 Bidirectional Attention Flow

From this baseline, we augmented the attention layer using bidirectional attention flow. In addition to the context attending to the question, the question should also attend to the context. As explained in the BiDAF paper [4], we compute a similarity matrix $S \in \mathcal{R}^{M \times N}$. Each entry S_{ij} represents the similarity between c_i and q_j. We compute a score like so:

$$S_{ij} = w_{sim}^T [c_i; q_j; c_i \circ q_j] \text{ where } w_{sim} \in \mathcal{R}^{6h}$$

Context to Question Attention To obtain question aware contexts, we find an attention distribution over the questions by passing each row of S into softmax and then taking a weighted sum of the questions:

$$\alpha^i = \text{softmax}(S_{i,:}) \in \mathcal{R}^N \quad \forall i \in \{1, ..., M\}$$
$$a_i = \sum_{j=1}^{M} \alpha_j^i q_j \in \mathcal{R}^{2h} \qquad \forall i \in \{1, ..., M\}$$

Question to Context Attention For each context location i, we take the max over the corresponding row $m_i = max_j S_{ij} \in \mathcal{R}$. We pass these scores into softmax to get an attention distribution over the contexts and take a weighted sum of the contexts:

$$\beta = \text{softmax}(m) \in \mathcal{R}^M$$
$$c' = \sum_{i=1}^{N} \beta_i c_i \in \mathcal{R}^{2h}$$

We then combine all of the above to get a blended representation for our contexts:

$$b_i = [c_i; a_i; c_i \circ a_i; c_i \circ c'] \in \mathcal{R}^{8h} \quad \forall i \in \{1, ..., M\}$$

3.3 Optimal Span Selection

Next, we focused on the prediction layer. In our baseline, we were greedily taking the max over both probability distributions independently. However, this is clearly not the global optimum. To improve span selection, we find the max over the joint probability distribution:

$$\text{argmax}_{ij} p^{begin}(i) p^{end}(j)$$

We then also found that capping the length of the span produced better results:

$$\text{argmax}_{ij} p^{begin}(i) p^{end}(j) \text{ where } i \leq j \leq i + 15$$

3.4 Context Features

Next, we took inspiration from the DrQA paper [5] and did some feature engineering to augment our context embeddings. We added the following features:

1. **Case insensitive match** An indicator feature that is 1 when a context token matches a question token, not considering capitalization.
2. **Lemma match** An indicator feature that is 1 when the lemma of a context token matches the lemma of a question token.
3. **Part of Speech** A 19 dimensional one hot encoded vector that represents a context token's part of speech.
4. **Named Entity Recognition** A 19 dimensional one hot encoded vector that represents a context's NER tag if any.

3.5 Larger Word Vectors

Next, we experimented with larger word vectors. Instead of using 100 dimensional GloVE vectors trained on Wikipedia, we used the 300 dimensional GloVE vectors trained on Common Crawl [3]. These common crawl GloVE embeddings have been pretrained on 840 billion tokens versus only 6 billion for the original GloVE vectors. We hypothesized that the word vectors may be more fine tuned and decrease the number of out of vocabulary tokens we encounter.

3.6 Improved Output Layer

Next, we took a look at our output layer and replaced the ReLU nonlinearity with a bidirectional GRU, similar to the BiDAF paper. This takes advantage of the fact that our blended context representations are a sequence and have an ordering to them. We replaced our original output layer:

$$b_i' = ReLU(Wb_i + v)$$

with the following from the BiDAF paper:

$$\{\overrightarrow{b_1}; \overleftarrow{b_1}, ..., \overrightarrow{b_M}; \overleftarrow{b_M}\} = biGRU(\{b_1, ..., b_M\})$$
$$b_i' = [\overrightarrow{b_i}; \overleftarrow{b_i}]$$

We do this twice with different weights to get $b_i'^{start}$ and $b_i'^{end}$.

3.7 Shared Weights

Our last architecture modification was to change the output layer described above and share weights instead of using 2 separate RNN's to calculate b_i' from start and end. So instead of having $b_i'^{start}$ and $b_i'^{end}$, we have a single b_i' that we feed into the softmax layers for start and end. We hypothesized this shared weight output layer might perform better because start and end indexes should be represented in a similar manner.

3.8 Hyperparameter Tweaks (go more in depth about specific changes)

In addition to architecture changes, we explored the effects of various hyperparameters. One of the first parameters we tweaked was the max length of the context. After analysis of the dataset, we lowered the max to 400 from 600 to get closer to the mode of the distribution. Later on, we experimented with a dropout value of 0.2 instead of 0.15 and tried adding an L2 regularization constant ($\beta = 0.001$) to our loss to discourage overfitting. However, the addition of a L2 regularization constant with $\beta = 0.001$ performed markedly worse than without.

We also experimented with the hidden size by decreasing the hidden size of our initial RNN encoding layer to 100. Decreasing the hidden size makes the model simpler and perhaps more generalizable. We found 100 to be a better hidden size than 200.

Lastly, we explored the effect of tweaking the learning rate. We changed the learning rate we fed into the Adam Optimizer from 0.001 to 0.005 and 0.048. A learning rate of 0.001 performed best, as the models with learning rates of 0.005 and 0.048 had high train and dev loss that did not decrease much over time compared to models with a 0.001 learning rate. We hypothesize that the higher learning rates are overshooting global minima.

4 Results and Analysis

	Dev	
Model	F1	EM
Baseline	44.0	34.8
BiDAF	51.4	41.6
Optimal span selection	58.8	47.2
Context features	68.8	57.7
Bigger word vecs	70.7	59.9
2 GRU modeling layer	71.0	59.8
Shared GRU modeling layer	74.4	63.7
Shared GRU modeling layer (hidden size 100)	74.4	63.5

Table 1: Different architecture results, each row builds upon previous

	Dev	
Model	F1	EM
Our best model	**74.4**	**63.7**
BiDAF (reference)	77.3	68.0
DrQA (reference)	78.8	69.5
Hybrid AoA Reader	89.3	82.5
QANet	89.0	82.7
Reinforced Mnemonic Reader + A2D	88.8	82.8

Table 2: Best result compared to state of the art

A summary of the results of our experiments can be found in Table 1. As expected, we see that reimplementing key ideas from the BiDAF paper and the DrQA paper improves performance relative to the baseline.

However, one interesting/unexpected result was the effectiveness of sharing weights in the modeling layer. From this simple change, our F1 score increased by 3 points. We hypothesize that the task of encoding the blended representations is similar for estimating both the start and end distributions. By sharing weights, the model is able to learn twice as much and share what it has learned for estimating both start and end.

Overall, we see that our model is unfortunately not competitive with the current state of the art. However, our model approaches the performance of the 2 primary papers we looked at. We can explain the gap due to missing a few small concepts from both papers such as a character level CNN from BiDAF and fine tuning the 1000 most common word vectors from DrQA.

In the rest of this section, we analyze key components of our model and try to understand their weaknesses and strengths.

4.1 Question to Context (Q2C) and Context to Question (C2Q) Attention Analysis

The idea of attention is that it summarizes the context and question to the essential components and ignores words that are superfluous or do not contribute much to the meaning. This way, our fixed size hidden states contain much more important information. Space isn't being used to represent meaningless words. In this section, we perform qualitative analysis by looking at the attention distribution of three examples. We pick out tokens that have large attention weights.

Context: quickbooks sponsored a " small business big game " contest , in which death wish coffee had a 30-second commercial aired free of charge courtesy of quickbooks . death wish coffee beat out nine other contenders from across the united states for the free advertisement .
Question: how many other contestants did the company , that had their ad shown for free , beat out ?
True Answer: nine
Predicted Answer: nine
Q2C: sponsored a "", in which had a 30-second aired free charge. beat out nine other contenders from across for the free.
C2Q: how many other contestants that beat out?

We see here that Q2C does a good job of summarizing the context based on the question. The summary has enough information to answer the question. In addition, we see that C2Q does a good job of slimming down the question to its core meaning.

Context: bskyb utilises the videoguard pay-tv scrambling system owned by nds , a cisco systems company . there are tight controls over use of videoguard decoders ; they are not available as stand-alone dvb cams (conditional-access modules) . bskyb has design authority over all digital satellite receivers capable of receiving their service . the receivers , though designed and built by different manufacturers , must conform to the same user interface look-and-feel as all the others . this extends to the personal video recorder (pvr) offering (branded sky+) .
Question: who has design authority over all of the digital satellite receivers that are capable of using their service ?
True Answer: bskyb
Predicted Answer: bskyb
Q2C: are has design authority over all digital satellite receivers capable of their service the receivers
C2Q: who has design authority over all that

Again, we see here that C2Q provides a somewhat reasonable summarization of the question. Q2C attention seems to have paid attention to the phrase in the context relevant to answering the question. One surprising thing is it does not pay attention to "bskyb" yet still gets the question correct.

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 .
Question: what theorem states that the probability that a number n is prime is inversely proportional to its logarithm ?
True Answer: the prime number theorem
Predicted Answer: digits
Q2C: prime that is is the prime number theorem that the probability that a number n is prime is inversely proportional to its number logarithm
C2Q: what theorem states that probability

This case is interesting because the attended words look decent. The true answer is also in the Q2C attention. Yet the model predicts an irrelevant word "digits". This example highlights that attention can be good yet the prediction can still be wrong. Sometimes attention isn't all you need.

4.2 Error Analysis based on Types of Questions

1st Token of question	F1	EM	# of examples
when	83.3	72.7	676
in	78.7	67.4	436
who	74.7	65.8	1057
how	69.2	50.0	1045
which	68.1	55.2	451
what	65.3	48.7	4704

Table 3: Model performance in different questions categorized by first question token (Most common first tokens shown)

In order to further understand our model, we categorized our model's errors based on the first token of the question for each of our examples. We feel that the first token is a good indicator for what type of question that a question is.

As we see from Table 3, the model performed best on examples where the question started with "when". We hypothesize that this is due to the fact that an answer's NER type is almost always a date. Since we provide the NER type as an additional context feature, the network only has to learn to distinguish between different dates. The search space is a lot smaller than for a "what" question.

"what" questions are on the other end of the accuracy spectrum. There is a 10 point gap between our overall F1 score and our F1 score for "what" questions specifically. The answer to "what" questions can be any possible NER or POS type, possibly rendering our additional input features not as useful. Additionally,"what" questions seem to be more vague and subjective. It is likely that most models, and even humans have worse performance on the "what" questions compared to other types of questions.

5 Conclusion

Overall, we learned a lot of practical deep learning skills. We got practice reading real research papers and reimplementing them to the best of our ability even in the face of ambiguities. We saw that better performance doesn't always come from complex models or even better architectures. Sometimes, all that's needed is tuning of hyperparameters or a better algorithm to turn the network's output into the final answer. Lastly, we experienced first hand the problem of interpretability. Understanding a model's weaknesses and reasons for error is nowhere near straightforward and involves lots of conjectures and trial and error.

One idea for future work is to explore the problem of imprecise answer boundaries. Potentially, we can take the examples with imprecise answer boundaries where the model has included too much information relative to humans. We can then train another network to solve the problem of summarization conditioned on the question. We could then append this network to the end of our current model to get more precise answer boundaries.

Contributions

Both of us contributed equally to this project. We both read papers and tried to interpret their models. We traded off the responsibility of actually implementing models and managing training. We both contributed equally to the writing of this paper.

Acknowledgments

Thank you to the CS224N course staff for being so responsive on Piazza. We especially want to mention Abi See for always providing detailed answers and providing much support throughout this project. Lastly, we want to thank Microsoft Azure for sponsoring this project and providing free compute power to train our models.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[2] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. *CoRR*, abs/1607.04423, 2016.

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[4] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051, 2017.