Question Answering

Peeyush Agarwal
Department of Computer Science
Stanford University
Stanford, CA 94305
peeyush@stanford.edu

Abstract

In this work, we explore attention based deep neural networks for the task of question answering. More specifically, we look at the task of finding the answer span from the context paragraph for a given question. The model is trained and tested using the recently released Stanford Question Answering Dataset (SQuAD).

1 Introduction

Question Answering (QA) is one of the core natural language processing task requiring machine comprehention of text. One way of framing the problem is to require the machine to read a context paragraph and answer questions based on this context.

The recently released SQuAD dataset [1] comprises of approximately 100,000 instances of question-answer-context triplets. These triplets were generated by humans by creating questions and corresponding answer spans for context paragraphs extracted from Wikipedia.

A sample triplet from the dataset is presented below:

Context paragraph: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 2410 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.

Question: Which NFL team represented the AFC at Super Bowl 50?

Answer: Denver Broncos

The answer is always a continuous span from the context paragraph and hence can be represented by a start index and an end index denoting the start and end of the answer span respectively.

Thus, the reading comprehension task can be formulated as predicting the start and end index for a given pair of question and context paragraph.

2 Background

After huge success of attention based models in computer vision tasks, they have been successfully applied to many natural language processing problems as well. In particular, they have shown very promising results on this task of reading comprehension. The state-of-the-art models have proposed various mechanisms of introducing attention such as Dynamic Coattention Networks [4], Bidirectional Attention Flow [5] etc. The basic idea here it to look at the paragragraph in light of the question and only attend to parts that are relevant to the question.

Previously, reading comprehention models were evaluated on cloze datasets such as CNN / Daily News articles [2] and Children's Book Test (CBT) [3] where the goal was to fill in a missing entity based on a given context. However, the recently introduced SQuAD dataset has proved to be a much better evalation metric for reading comprehention tasks.

3 Approach

In this work, we learn a simple attention based model for the task of reading comprehention (and question answering) and evaluate it's performance on SQuAD.

The SQuAD dataset consists of 100K question-answer-context triplets. These texts vary quite a lot in their length. The histogram of context paragraph lengths shows that most paragraphs are smaller than 300 words long. In a similay way, the histogram of question lengths shows that most questions contain fewer than 25 tokens. We will use this information to cap the question and paragraph lengths for faster training.

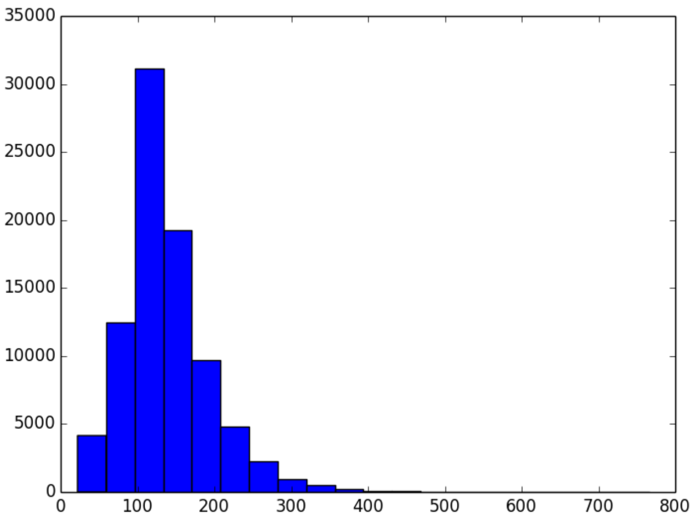

Figure 1: Histogram of paragraph lengths.

In this work, we have implemented a very basic version of attention inspired from the Dynamic Coattention Networks. This version was suggested as one of the baseline models on Piazza.

A description of the model is as follows:

- Tokenize the question and context paragraph using NLTK.
- Use pretrained GLoVe vectors to tranform words into an embedding space.
- Encode the question using a Recurrent Neural Network with LSTM cell. Let Q_E represent the encoded question.
- Encode the context paragraph using a Recurrent Neural Network with LSTM cell. Let P_E represent the encoded paragraph.
- Compute the "affinity matrix" which contains affinity scores corresponding to all context word and question word pairs. Normalize these values to produce attention weights across the question for each word in the document.

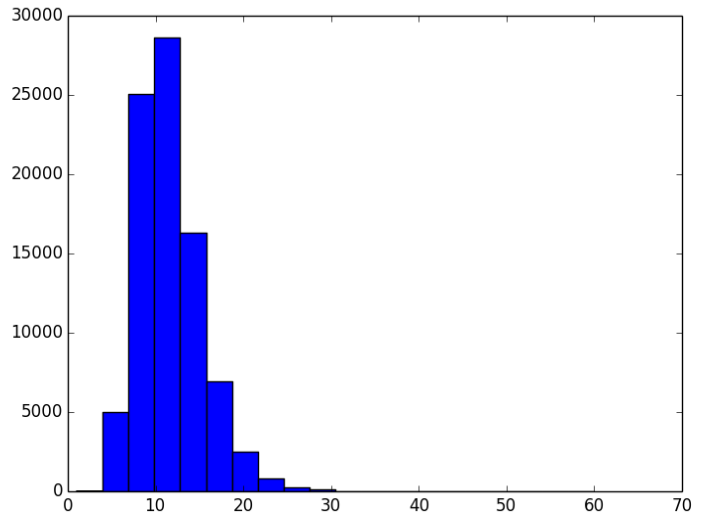

Figure 2: Histogram of question lengths.

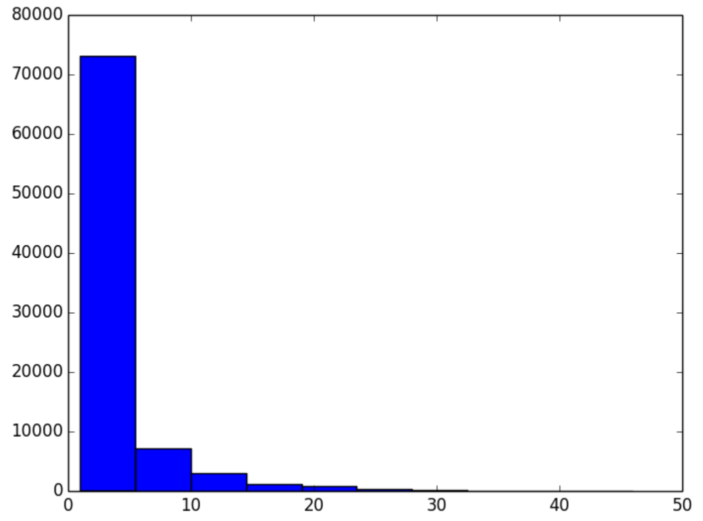

Figure 3: Histogram of answer lengths.

$$A = softmax(P_E, Q_E^T)$$

- Use this affinity to compute the summaries, C_P, of the question in light of each word of the document.

$$C_P = A * Q_E$$

- Concatenate the compued context vector (for question to paragraph) with the paragraph encoding and apply a linear transformation to mix the attention information with paragraph encoding.

$$P_F = concat(C_P, P_E) * W + b$$

- Classify the resultant tensor, P_F, at each point to give probability of the given word being the answer start.
- Pass the same tensor, P_F, through a Recurrent Neural Network and classify the resultnt values at each point to give probability of given word being the answer end.

We used Adam Optimizer for training with loss being the sum of softwax cross entropy loss of start inddex and end index predictions.

4 Experiments

I started by implementing the baseline as suggested in the Piazza post and released presentation. However, my first implementation did not perform as expected - the loss did not decrease very much at the end of first epoch.

To debug issues with the model, I started prining the gradient norm and loss after every fixed number of iterations (10K). I identified that the gradient was overshooting because of incorrect formulation of error. I fixed the bug and also implemented clipping to clip gradients similar to assignment 3. Then, I tried to overfit on 5 examples in order to identify and fix any remaining bugs in the code. Once all these issues were fixed (and the model was able to perfectly fit on the small dataset), running it on complete data led to a significant decrease in loss during the first epoch itsef showing that it had infact started learning.

On training, the loss decreases drastically from 11 to 6 by the end of first epoch. The gradient norm typically stays between 1 and 2.5. From second epoch onwards, the loss stays around 6 (sometimes going to 5) and the gradient norm typically stays below 2. However, the gains are minuscule after second epoch and the model seems to be starting to converge by the end of fifth epoch as can be seen in the figure. There are also some signs of model overfitting the data (like gradient norm upto 1 even after the fifth epoch) and I think that adding dropout will be definitely useful.

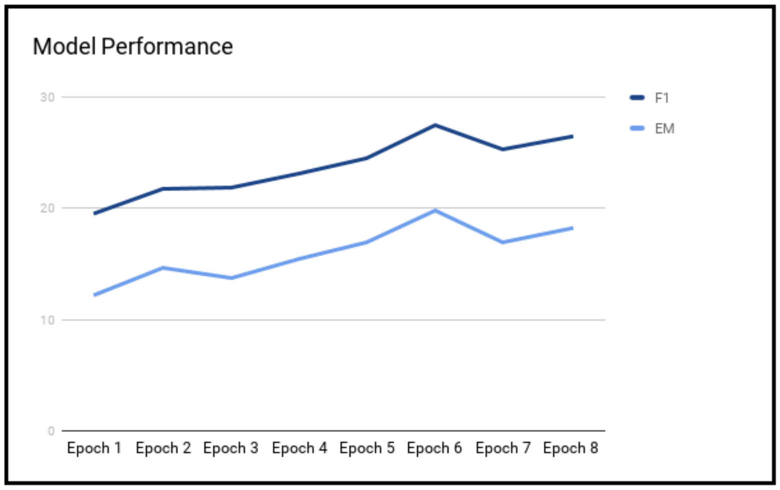

Figure 4: Model performance at the end of each epoch while training.

The trained model does not seem to be giving comparable results on Dev and Test dataset. On test data, it gives an F1 score of 4.7 with an exact match score of 0.5. This is possible because of multiple reasons. The model might have overfitted the train data too much by the time I evaluated it's performance on Codalab. Secondly, the way qa_answer has been implemented also leads to decrease in scores - words/tokens previously unseen (and therefore not present in vocabulary) are dereferenced as $< unk >$. It would be useful to modify the method to also pass the document string and use it to backfill these $< unk >$s for a higher score.

5 Conclusion

In this work, we explored a simple attention based model for the task of question answering. We also identified some of the challenges that this simple model faces and how it can be further improved.

Some possible next steps I would have tried (on the current model) if I had more time:

- Improving the word tokenization by using more recent methods & tools such as Stanford CoreNLP instead of NLTK.
- Increasing the word embedding size or tuning the embedding for the task of reading comprehension.
- Using bi-directional (instead of the current uni-directional) RNNs for encoding the context paragraphs and questions.
- Introducing dropout at various model layers to prevent overfitting.
- Annealing the learning rate (possibly an exponential decay) as the model starts to converge.
- Using the validation dataset to tune various model parameters (like hidden state size, question & paragraph length clipping).

The model itself is not very powerful. The attention mechanism is also very basic. It basically just computed the affinity/relevance for each question word - paragraph word pair and encodes this information while making the prediction on start and end index. It would be useful to try some more complex attention mechanisms such as dynamic co-attention etc. Also, it can get stuck on a local maxima while predicting the start and end index. It would be useful to go back and forth between the two predictions in order to arrive at the optimal start-end combination for a given context-question pair.

Acknowledgments

I would like to thank the CS224n instructors and teaching staff for all the effort they put it - from providing the starter code to baseline model ideas and helping with all the questions on Piazza. I would also like to thank Microsoft for allowing us to use their GPUs for training the models.

References

[1] Rajpurkar, Pranav, et al. “Squad: 100,000+ questions for machine comprehension of text.” arXiv preprint arXiv:1606.05250 (2016).

[2] Chen, Danqi, Jason Bolton, and Christopher D. Manning. “A thorough examination of the cnn/daily mail reading comprehension task.” arXiv preprint arXiv:1606.02858 (2016).

[3] Hill, Felix, et al. “The Goldilocks Principle: Reading Children’s Books with Explicit Memory Representations.” arXiv preprint arXiv:1511.02301 (2015).

[4] Xiong, Caiming, Victor Zhong, and Richard Socher. “Dynamic Coattention Networks For Question Answering.” arXiv preprint arXiv:1611.01604 (2016).

[5] Wang, Zhiguo, et al. “Multi-Perspective Context Matching for Machine Comprehension.” arXiv preprint arXiv:1612.04211 (2016).