CS224n: Homework 4
Reading Comprehension

Leandra Brickson, Ryan Burke, Alexandre Robicquet

1 Overview

To read and comprehend the human languages are challenging tasks for the machines, which requires that the understanding of natural languages and the ability to do reasoning over various clues. Reading comprehension is a general problem in the real world, which aims to read and comprehend a given article or context, and answer the questions based on it.

The point of this project is to implement a neural network architecture for Reading Comprehension using the recently published Stanford Question Answering Dataset (SQuAD) [1].
In the SQuAD task, answering a question is defined as predicting an answer span within a given context paragraph. We implement a neural network architecture for Reading Comprehension using a LSTM cells approach.

2 Dataset

2.1 SQuAD Dataset

As it is mentioned on the SQuAD website[1] and in [4], Stanford Question Answering Dataset (SQuAD) is a new reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. With 100,000+ question-answer pairs on 500+ articles, SQuAD is significantly larger than previous reading comprehension datasets, and was used in [7]. For this project, we use a subset of the SQuAD dataset for training, with 87k samples, organized into triplets of (question, context, answer). The question and context of this set are of varying length, and the answer is stated as a number pair labeling the character index at the start of then answer and the character index at the end of the answer. There is additionally a development set of 10k triplets for validation on the training. The test set will be evaluated online , to give a final valuation of our network.

2.2 Preprocessing

In order to properly adjust our network to the accept the variable length of question and context passages, we need to find the optimal number of words (tokens) from which we either truncate (for the too long elements) or pad (for the too shorts ones) in order to simultaneously limit the loss of information (truncating) and the number of empty tokens (padding) as much as possible.
In order to familiarize ourselves with the SQuAD dataset, we plotted the histograms giving us the number of sentences containing a certain number of tokens. Cf figure 1. Those method are often use for NLP deep learning, such as in [3, 1, 6]

[1] https://rajpurkar.github.io/SQuAD-explorer/

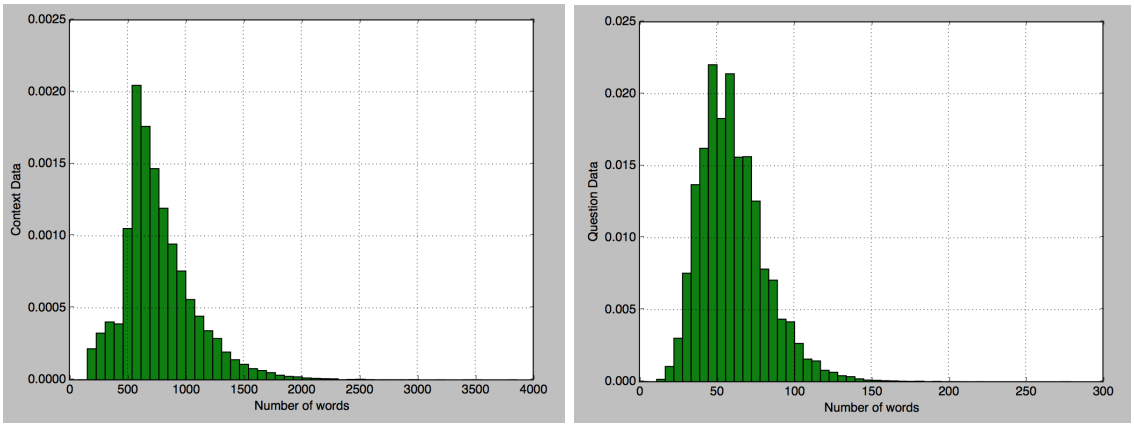

Figure 1: Decoder

These histograms were very usefull to understand the data we've been working with. After analyzing those plots, we deciding to consider the following values:

$$\text{max length context} : 750$$

$$\text{max length questions} : 70$$

The input questions and context were padded to this length if shorter, and cut off if longer.

3 Implementation

In the SQuAD task, the goal was to predict an answer span tuple $\{a_s, a_e\}$ given a question of length $n, q = \{q_1, q_2, ..., q_n\}$, and a supporting context paragraph $p = \{p_1, p_2, ..., p_m\}$ of length m. Thus, the model learns a function that, given a pair of sequences (q, p) returns a sequence of two scalar indices $\{a_s, a_e\}$ indicating the start position and end position of the answer in paragraph p, respectively. In this section, we detail network designs which we thought would be good approaches to this problem.

3.1 Algorithm description

For this specific problem, we implemented our own LSTM cell. $lstm - cell.py$ as a wrapper around our GRU cell implementation that allows us to play nicely with TensorFlow. Long Short Term Memory networks usually just called "LSTMs" are a special kind of RNN, capable of learning long-term dependencies.

3.2 Encoder

The first step of our QA network is the Encoder, which inputs the question and context passages, and converts them to a knowloedge representation of the query and information provided. Conceptually, it would ideally provides a summary of the information portrayed in the question and context, so that it is easier for the network to identify the location of the answer. For our network, a few encoding networks were considered, these are summarized in figure 3.

The first is a simple LSTM encoding network for both the question and the context. The LSTM model is detailed in the lower part of figure 2 [2]. The output hidden state of the question encoding will be the input hidden state for the context encoding. The more sophistocated variant of this

[2] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

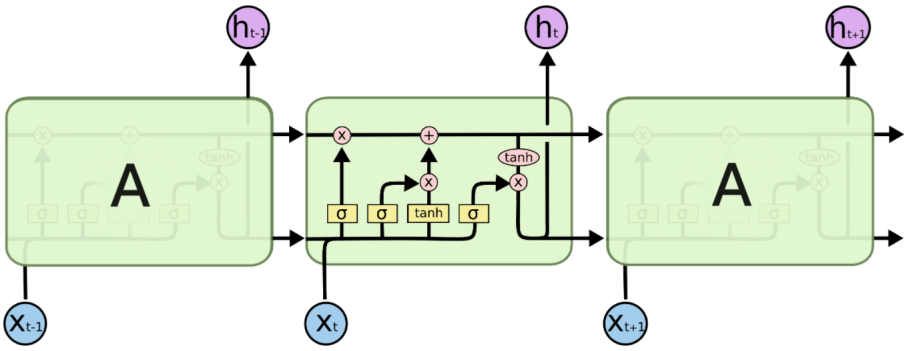

Figure 2: LSTM

encoder would be a Bi-LSTM encoder shown second in figure5. This bi LSTM is detailed third in figure 5 where the input is encoded in a forward and backward LSTM separately, and then concatenated.

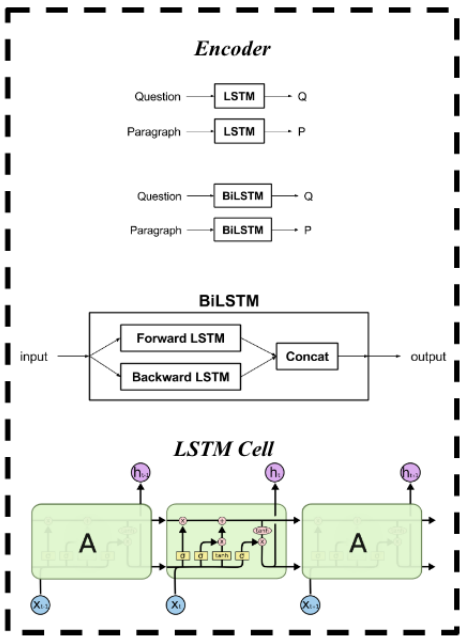

Figure 3: Encoder

3.3 Decoder

The final part of the system is a decoder, which takes the summary of the query and context(called the knowledge Representation, or KR) meaning from the main network and converts it to an actual labeling of the answer words in the context paragraph. The first decoder network attempted was a linear classifier network, which classified the KR into one of three classes: 'Not', 'Ans Start' and 'Ans End'. The next dedcoder is illustrated first in figure 4. This network takes the KR, linearly classifies it to identify the answer start index and then passes the KR into a decoding LSTM beofre linearly classifying to find the answer end index. Another possible decoder is having a separate single-layer neural network hidden layer to determine each answer index, this is shown as the second decoder in figure 4. Finally, the last Decoder in figure 4 combines a couple nonlinear layers with memory, an LSTM and a single layer neural net for start and end point classification

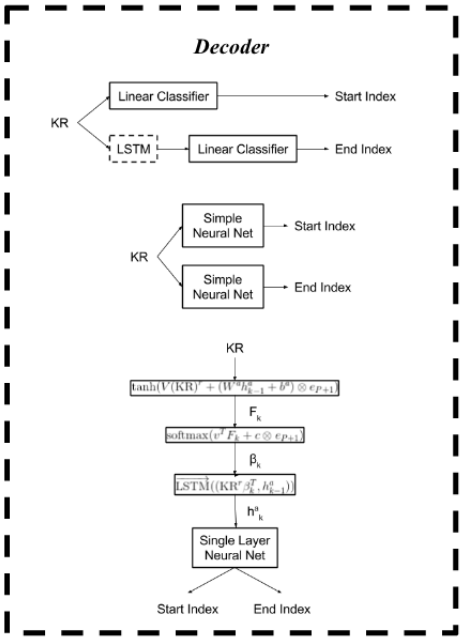

Figure 4: Decoder

3.4 Main Architecture

The goal of the main network is to convert the separate question and answer hidden representations into a single hidden representation. This single representation should preserve the necessary information to correctly identify the correct answer string in the context paragraph. The first architecture considered was a simple concatenation of the last hidden layer of the encoded question and each of the hidden layers of the context paragraph. A more complex architecture is the concatenation of both forward feeding and backward feeding MatchLSTMs. The MatchLSTM itself [5] consists of several nonlinear equations to extract the attention coefficients for each word in the context paragraph, and an LSTM applied to the attention weighted context paragraph. Both parts of the MatchLSTM use the previous MatchLSTM hidden state as input, giving the overall system better memory.

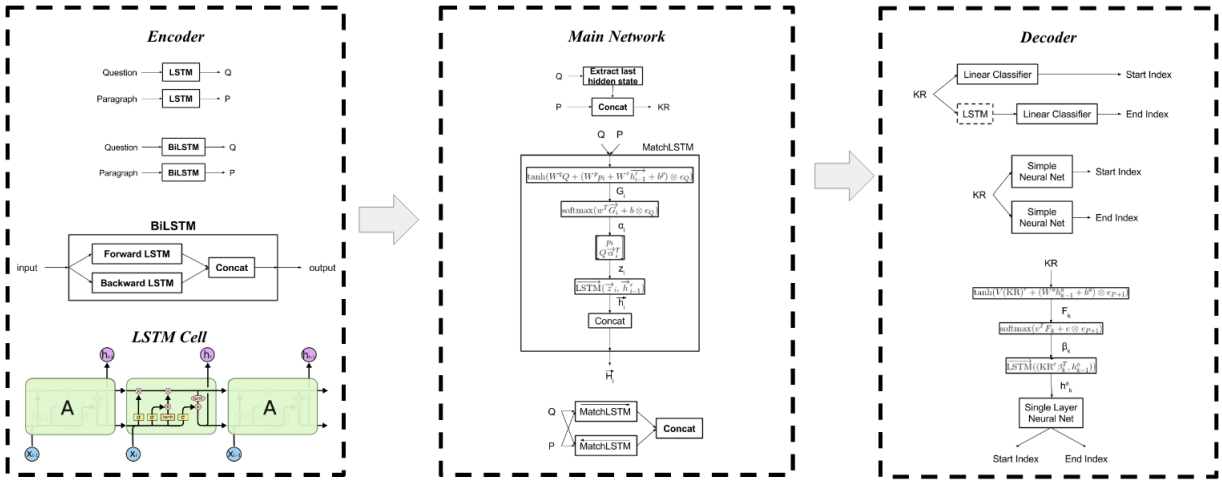

Figure 5: Main Architecture - Encoder / Network / Decoder

4 Experiments

4.1 Problems Encountered

Unfortunately, almost every minute of this project was spent programming the majority of a large-scale neural network training code infrastructure. Each member of the team spent almost a full week debugging and attempting to find the root causes of many cryptic tensorflow errors. In the end, we were unable to try out many of the network configurations we had researched and been interested in.

Most of the base-code were also extremely confusing, leading us to first apply the homework 3 to this project. This approach was then modified last minute after some piazza updates that unlocked the majority of homework 4 for us, too late unfortunately.
Our reduce network (smaller set of training and testing) doen't seem to be training very well , best scores achieved is $F1 = 7.2$, and $EM = 4$. First thing we try at this point is hyper parameter tuning.

4.2 Evaluation

F_1 score is a metric that loosely measures the average overlap between the prediction and ground truth answer. We treat the prediction and ground truth as bags of tokens, and compute their F_1. We take the maximum F_1 over all of the ground truth answers for a given question, and then average over all of the questions.

4.3 Results

The final system which ran was a network which first passed the question into an LSTM, passed the output hidden state from this question as the input hidden state of a context-encoding LSTM. The LSTM cell was made so that the inputs necessary could be fully understood. This output was then passed into a linear classifier , which classified between three states, 'Not', 'Ans Start' and 'Ans End'. This was then passed into a softmax rectifier and the word with highest probability of being 'Ans Start' and 'Ans End' were taken to be the beginning and end of the sentence, respectively. Since this network was finally debugged this morning, hyperparameter tuning wasn't able to be done, and we reached a max f1 score of 13, which is far below the expected baseline.

Next Steps

The next step in this project for better results is to tune hyperparameters and start running more complex architectures. Those architectures would be inspired from papers such as [1] regarding the attention model [2] regarding the encoder/decoder model.

References

[1] Yiming Cui et al. "Attention-over-attention neural networks for reading comprehension." In: *arXiv preprint arXiv:1607.04423* (2016).

[2] Ryan Kiros et al. "Skip-thought vectors." In: *Advances in neural information processing systems*. 2015, pp. 3294–3302.

[3] Ankit Kumar et al. "Ask me anything: Dynamic memory networks for natural language processing." In: *CoRR, abs/1506.07285* (2015).

[4] Pranav Rajpurkar et al. "Squad: 100,000+ questions for machine comprehension of text." In: *arXiv preprint arXiv:1606.05250* (2016).

[5] Shuohang Wang and Jing Jiang. "Machine comprehension using match-lstm and answer pointer." In: *arXiv preprint arXiv:1608.07905* (2016).

[6] Caiming Xiong, Victor Zhong, and Richard Socher. "Dynamic Coattention Networks For Question Answering." In: *arXiv preprint arXiv:1611.01604* (2016).

[7] Junbei Zhang et al. "Exploring Question Understanding and Adaptation in Neural-Network-Based Question Answering." In: *arXiv preprint arXiv:1703.04617* (2017).