Question Answering on the SQuAD Dataset

Brad Girardeau
Department of Computer Science
Stanford University
bgirarde@cs.stanford.edu

Abstract

This project explores approaches to learning the SQuAD dataset. It introduces a simple baseline model using an encoder-decoder architecture and shows how pointer networks and coattention techniques result in significant improvements over the baseline. The best model, combining coattention encoding with a pointer network decoder, reaches an F1 score of 65.036% and EM score of 53.205% on the SQuAD test set.

1 Introduction

Answering questions is a key challenge in natural language processing. It involves both understanding language as well as knowledge about the physical world, and many NLP tasks can be mapped to question-answering tasks. Recently Rajpurkar et. al. released the Stanford Question Answering Dataset (SQuAD) dataset to facilitate research in this area. SQuAD consists of over 100,000 question-answer pairs from context paragraphs found on Wikipedia. It is one of the first high-quality, large scale reading comprehension datasets, ideal for creating and testing new deep learning models.

This project explores the process of building a deep learning model that performs well on SQuAD. It uses the sequence-to-sequence encoder-decoder architecture to show the effectiveness of key techniques in this area. A pointer network encoder, a simple question coattention mechanism, and a full coattention encoder are added incrementally. With the coattention encoder, performance reaches 65.036% F1 and 53.205% EM on the SQuAD test set.

2 Background

The SQuAD dataset consists of questions, context paragraphs, and answer spans. The answer spans are a start and end index into the context paragraph, which indicate the start and end of the answer to the question as found in that paragraph. This task is more constrained than open-ended or multiple choice questions as the answer is found directly in the input paragraph, but it still captures a variety of complex question types.

Performance on the dataset is measured by both an F1 score, which captures the precision and recall that words chosen as being part of the answer are actually part of the answer, and an exact match EM score, which is the number of answers that are exactly correct (with the same start and end index). Human performance on the dataset is 82.3/91.2% EM/F1, while the current best performance for a single model is 72.4/80.8% EM/F1 and for an ensemble model is 76.9/84.0% EM/F1 (r-net by Microsoft Research Asia).

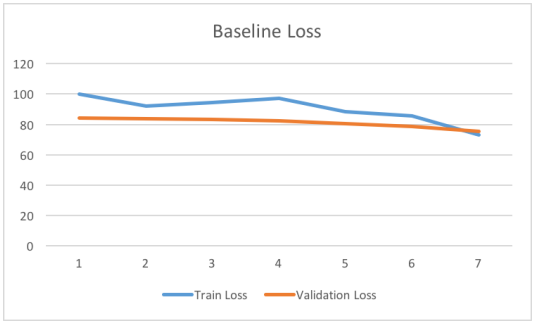

Figure 1: Training and validation loss for baseline model.

3 Approach

Many successful models for the SQuAD dataset use an encoder-decoder architecture. An "encoder" takes as input the question and context, and it feeds its output to the "decoder stage," which then predicts an answer span. The approach of this project is to start with a simple encoder/decoder baseline model, then build up additional components and features of the network. This paper describes each model and evaluates it by looking at its loss curves, F1, and EM scores and training and validation data. The final model of this project is most similar to the Dynamic Coattention Model (which obtained a score of 66.2/75.9% EM/F1).

4 Experiments

This section describes the different models implemented and their performance.

4.1 Baseline

The baseline model uses a Bidirectional RNN based encoder and decoder. The encoder has two steps:

1. Feed the question embeddings $\{q_1, ...q_l\}$ through an LSTM to obtain $q :=$ `final hidden state`
2. Feed the context embeddings $\{c_1, ..., c_n\}$ through another LSTM with initial hidden state q to obtain $c :=$ `final hidden state` and $\{c'_1, .., c'_n\} :=$ `outputs of LSTM`

This encoder represents a "reader" first reading the question, then the context, to obtain the knowledge representation $(q, c, \{c'_1, ..., c'_n\})$. An LSTM is a standard choice for the reader RNN because it can better handle longer dependencies over the entire question and context inputs.

The encoded knowledge representation is then passed to the decoder, which uses another LSTM over the knowledge representation and question, whose outputs are fed through a softmax layer to classify each context token as the start, end, or neither of the answer. The decoder LSTM operates over inputs $[(c'_1, q), (c'_2, q), ..., (c'_n, q)]$, where the question is included with each input to ensure it is available at each step even for long context paragraphs.

The model is then trained using standard cross entropy loss over the decoder softmax outputs (averaged in each sample to reduce the effects of varied context lengths).

4.1.1 Evaluation

The baseline model was not powerful enough to learn this dataset. As shown in Figure 1, the loss decreased slightly overall but was quickly plateauing at a high value (corresponding to 13/17% EM/F1 on the validation set at peak).

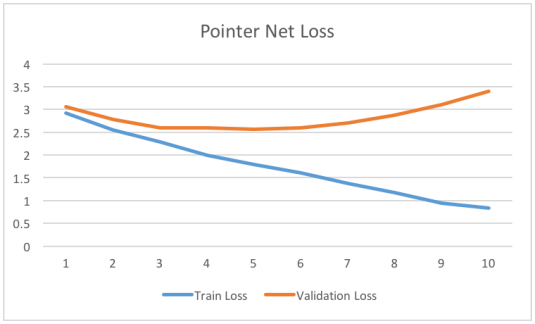

Figure 2: Loss curves for pointer model showing overfitting.

4.2 Pointer Net

The first improvement over the baseline model replaced the LSTM sequence decoder with a pointer network. While decoding can be modeled as a classification task, it would be more natural to predict the answer start and end points directly. This is precisely what pointer networks do. Instead of a fixed dimension output like LSTMs, they learn to output softmax distributions over the input to "point" back into the input. Both the Dynamic Coattention Network and Match-LSTM models use this idea for their models.

The pointer network implementation for this project's model was based on the original pointer network paper. The baseline encoder produces a representation of the context words. In the pointer network, the context word inputs are multiplied by a learned weight matrix, which is mixed using two other learned matrices with the hidden states of a decoder LSTM. This decoder LSTM is run to produce as many hidden states as answer pointers, so in this case over two inputs (which were set to a concatenation of the encoded context and question). The two output softmax pointers from this network are then used as the distributions over the start and end points of the answer.

4.2.1 Evaluation

This model has a significant advantage in training efficiency. Instead of running a decoder LSTM over the entire context to predict where the answer is, the decoder only needs to compute one matrix multiplication over all of the context inputs at once. This made training time per epoch at least three times faster.

In addition, this model was also more powerful than the baseline. Its score was twice the baseline's, with 26/38% EM/F1 on the validation set at peak. It could overfit the training dataset, reaching 68/79% EM/F1, as seen in Figure 2. This shows that the model could learn effectively on the dataset.

To see if the model could generalize better, a dropout layer over the outputs of the context encoder LSTM was added, and the LSTM state size was reduced. This prevented the overfitting previously seen, but it revealed that the model could not generalize to the validation set any more than it already had (achieving a similar EM/F1 score on the validation set as before).

4.3 Attention

A core piece of reading comprehension models still missing is attention, found in many models that perform well on SQuAD. While an LSTM can "remember" dependencies on previous inputs, this ability is limited by the fixed size hidden state, especially over long inputs (like context paragraphs). Attention allows an LSTM to learn to look back at any previous inputs. It works by computing an "attention vector" based on the current state that specifies a distribution over the inputs to include in the current computation.

There are a variety of ways to compute and incorporate attention vectors into a question-answering model. Two related approaches were tried in this project. For both, attention was incorporated into the encoder, since this is what is affected by the LSTM's limitations over the long question and

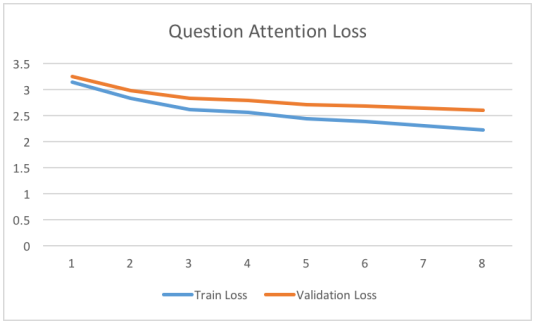

Figure 3: Loss curves for question attention model.

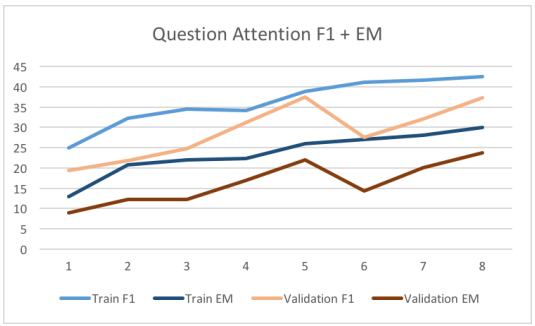

Figure 4: F1 and EM scores each epoch over training, validation data for question attention model.

context input sequences (the pointer network already effectively computes an attention vector as the answer pointer).

4.4 Question Attention

The idea for this model is that the LSTM that produces the context output sequence consumed by the decoder should learn to pay attention to relevant parts of the question while examining each context word. This is accomplished by multiplying the question representation by the context representation to mix the inputs together, converting the question-length vectors to probability distributions using softmax. These distributions then select a weighted average of the original question embeddings to include with the context word input to the LSTM. This specific implementation was inspired by the Dynamic Coattention Network model but is simplified to only include question attention.

4.4.1 Evaluation

Question attention improved performance, with a smooth loss curve and higher F1/EM scores as shown in Figures 3 and 4.

4.5 Coattention

While improving, the model performance still lagged significantly behind the models on the SQuAD leaderboard. A more effective attention mechanism was needed, so the full coattention component (with minor modifications) from the Dynamic Coattention Network model was implemented. This adds an RNN encoding of the inputs and questions (like the baseline) before computing the attention that is fed to the final encoding RNN. In addition, the attention mixes both the question with the context and the context with the question in the RNN inputs. This lets the context and question both influence the "reading" of the other by the model.

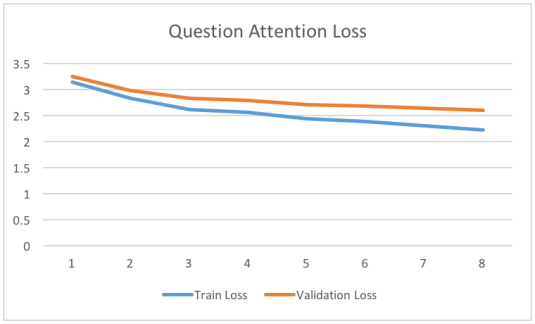

Figure 5: Loss curves for coattention model.

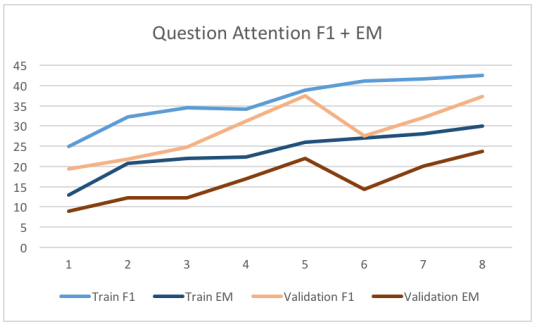

Figure 6: F1 and EM scores each epoch over training, validation data for coattention model.

4.5.1 Evaluation

This model significantly outperformed the others. As Figure 6 shows, the EM/F1 scores on training were 53.3/73.8% and validation were 44/62%. The dev set performance for this model was 51.9/64.5% EM/F1. On the hidden test set, this model scored 53.205/65.036% EM/F1.

5 Conclusion

The SQuAD dataset allows for the exploration of many deep learning systems for the question-answering task. Using a sophisticated attention mechanism like coattention gave the largest increase in performance in this project, showing the importance of using attention to allow other components of the network (like a pointer network decoder) to work effectively. This architecture was relatively simple compared to state of the art systems, using only the core ideas of answer pointers and coattention, but it was still able to perform fairly well, and additional tuning would likely be effective in increasing its performance. For example, the Glove vectors were only dimension 100 and out of vocabulary words were initialized uniformly to zero for simplicity. Using dimension 300 vectors and initializing unknown words to random values (or using more advanced character models) would likely help further increase performance. This project illustrated two of the key concepts of question-answering systems, pointer networks and coattention, and showed how they can contribute to well performing systems.

References

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

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

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