Question Answering with Multi-Perspective Context Matching

Joey Asperger
Department of Computer Science
Stanford University
Stanford, CA 91305
Joey2017@stanford.edu
Codalab username: joeyasperger

Abstract

In this paper, I tackle the challenge of answering questions based on a context paragraph. To solve this problem, I implemented Multi-Perspective Context Matching based on the paper by Wang et. al. It took a very large amount of effort to get this baseline model working, so I didn't have much time to implement my own original ideas in the model. However, this still performed very well relative to most of my classmates, scoring 71% F1 and 60% EM. I would attribute this success largely to the fact that I did not trim the GloVe vectors and that I used character-level embeddings. In future work, I think a useful idea to explore would be to not predict the start and end indices independently, but to first predict the start index and then run an LSTM from this index in order to predict the end index.

1 Introduction

Machine comprehension is a very difficult problem to solve. It can require understanding a wide variety of different words and language constructs, as well as reasoning over significant chunks of text. This project delves into the problem of answering questions based on a context paragraph. There are countless real-world applications of this problem, including making better search engine results and better digital assistants.

1.1 The Data

The Stanford Question Answering Database (SQuAD) was used as the dataset for this project. It contains over 100,000 question-answer pairs covering over 500 different articles. The questions are all asked in natural language, and often have fairly challenging answers. Success on this dataset is measured by

F1 score (the harmonic mean of precision and recall) and percent of exact matches (EM score.) An average human scores 91% and 82%, respectively, while a baseline logistic regression model scores 51% and 40%.

2 Model

My model my is based almost entirely off the model proposed by Wang et. al. in *Multi-Perspective Context Matching for Machine* Comprehension. This model involves 6 main layers shown in figure 1 below. In the following sections, I will describe each layer in more detail.

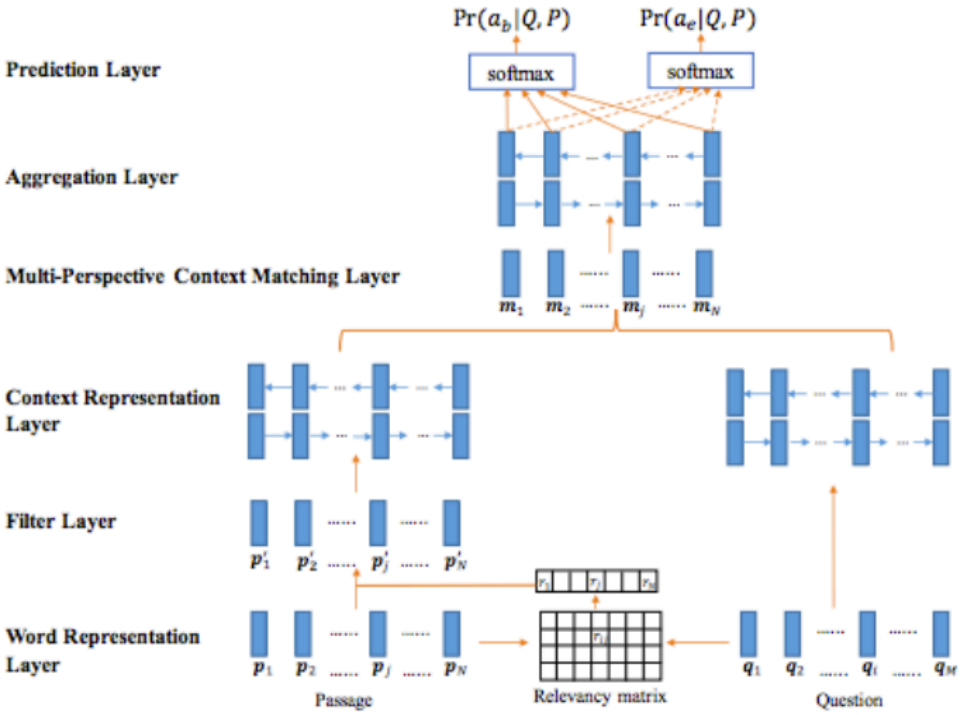

Figure 1: Multi-Perspective Context Matching (Wang et. al.)

2.1 Word Representation Layer

Each word in the context paragraph and the question was represented by a *d*-dimensional vector. This representation consisted of both word-level embeddings and character-level embeddings. The word-level embeddings were taken from GloVe vectors that were pre-trained on the 840-billion-word Common Crawl Corpus. The character-level embeddings were created by using a max-pooled convolutional neural network that was then fed into a 1-layer highway network.

2.2 Filter Layer

The purpose of this layer is to give higher weight to words in the context

paragraph that are more relevant to the question. To do this, it calculates the cosine similarity between each pair of context and question words. It then takes the maximum similarity of each context word with any question word. Each context word representation is then multiplied by this max similarity.

2.3 Contextual Representation Layer

In this layer, Bidirectional Long Short-Term Memory Recurrent Neural Networks (BiLSTMs) are run on the questions and filtered context. This allows contextual information from the surrounding words to be transmitted to each word.

2.4 Multi-Perspective Context Matching Layer

This layer compares each context representation with question representations using a weighted cosine similarity. There are 6 types of comparisons in this layer: taking the max similarity with any question word, the mean similarity with all question words, the similarity with the final LSTM output of the question, and repeating these 3 types in the reverse LSTM direction. For each of these comparison types, multiple perspectives are used, meaning both the context and question representations are multiplied by a learned weight matrix before taking the cosine similarity, so different perspectives will weight features differently. This results in a representation for each context word that is 6 x m where m is the number of perspectives used.

2.5 Aggregation Layer

This layer takes the output from MPCM layer and runs another BiLSTM over it to order to encode information about the surrounding matches.

2.6 Prediction Layer

This layer uses two single-hidden layer feed-forward neural networks to predict the softmax probability of each word independently being the start or end word of the answer.

3 Implementation and Training

I implemented the model using Google's TensorFlow library. For my word representations, I used 300-dimensional GloVe vectors trained on the 840 billion word Common Crawl Corpus. For the character representations, I used 100 max-pooled convolutions of width 5. Every BiLSTM used a state size of 100 for each direction. For the highway network and the prediction network, Relu was used as the nonlinear activation function. Only 10 of each perspective type was used due to memory constraints. In total, this model had 1,005,972 trainable parameters. I used a dropout rate of 0.2 between each layer in order to help regularize the model. I used trained for 9 epochs using an softmax cross-entropy loss and an Adam Optimizer with a learning rate of 0.001. During training, the validation loss approached its minimum after

approximately 6 epochs and then only made very minor improvements after that. Figure 2 below shows cross-entropy loss across epochs.

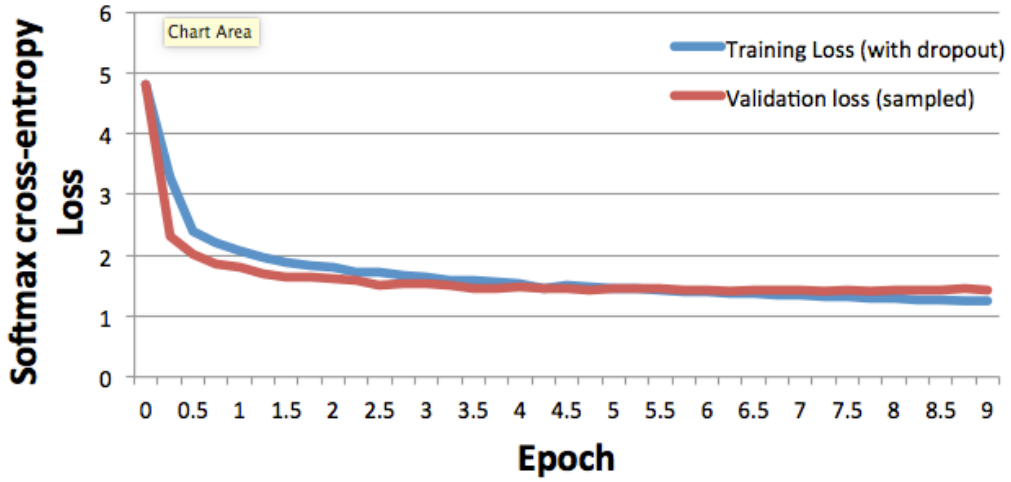

Figure 2: Cross-entropy loss across epochs

4 Results

Dataset	F1 Score	EM Score	Leaderboard Rank
Dev Set	71.3	60.6	#8
Test Set	70.7	60.3	#5

As can be seen in the table above, my model performed quite well. It performed approximately 20 points higher than the logistic regression baseline, but still lagged behind the state-of-the-art models by about 13 percentage points.

4 Discussion

Overall, my model performed very well, achieving high ranks on both the dev and test leaderboards. I think two of the most significant factors in setting my model apart from my classmates' were that I used the 840B Common Crawl GloVe vectors with no trimming and that I also used character embeddings.

In comparison to the original single-model MPCM model designed by Wang et. al., my model scores approximately 5% lower on both F1 and EM score. Based off their ablation study, 2 of those percentage points are likely caused by the fact that I only used 10 perspectives instead of 50 due to GPU memory constraints. The reason for the other 3 percentage points of difference is unclear, but most likely due to hyperparameter tuning. The most significant difference between my model and that of Wang et. al. is that I used a CNN/Highway character embedding instead of an LSTM character embedding. However, I don't think it is likely that this choice made a significant performance difference.

My original goal was to implement MPCM and then try to make original improvements to it. Unfortunately, it took a very significant amount of time to

get this baseline MPCM model working, so I didn't have time to try out any of my own ideas. Hopefully, I'll have the opportunity to experiment more with it in the future. However, I am still very happy with my results with the baseline MPCM model.

4 Future Work

One problem I frequently saw in predictions was that the model would predict the start index correctly, but then predict a completely wrong index for the end of the answer, creating a long, run-on answer. This probably happens so frequently because the start and end indices are predicted completely independently. If the prediction for the end index were to incorporate more information on which index was selected for the start index, it might be able to make better predictions because it would be able to use this information to tell when a sufficient amount of information has been given in the answer and therefore be able to terminate better. Therefore, one idea I would experiment with is to try predicting the start index and then using an LSTM starting from that start index and using the LSTM output for each word to predict the probability of that word being the end of the answer.

Acknowledgments

Special thanks to Microsoft for providing a GPU for training as well as to the entire course staff for their support.

References

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. *Multi-perspective context matching for machine comprehension*. arXiv:1612.04211, 2016.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. *Bi-directional attention flow for machine comprehension*. arXiv:1611.01603, 2016.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. arXiv:1606.05250, 2016.