Reading Comprehension with Deep Learning

Atticus Geiger
Department of Computer Science
Stanford University
Stanford, CA 94305
atticusg@stanford.edu

Dylan Liu
Department of Chemistry
Stanford University
Stanford, CA 94305
dliu3@stanford.edu

Abstract

We train a model that combines attention with multi-perspective matching to perform question answering. For each question and context pair in SQuAD, we perform an attention calculation over each context before extracting features of the question and context, matching them from multiple perspectives. Whilst we did not have time to perform a hyper-parameter search or incorporate other features into our model, this joint model obtained an F1 score of 22.5 and an EM score of 13.0, outperforming our baseline model without attention, which attained an F1 score of 19.4 and an EM score of 10.0. In the future, we would like to implement an early stopping feature and a co-dependent representation of each question and context pair.

1 Introduction

While natural language processing has historically focused on reading comprehension tasks for the sake of machine translation, a new challenge that has recently arisen is the task of question answering, which concerns itself with answering questions about a given passage. Traditional approaches to the task involve parsing the contexts and questions under a highly structured syntax and grammar, requiring part of speech-tagging, co-reference parsing, and dependency parsing, examples of which are expensive to generate. Consider the following story.

Atticus went to the toilet.
Apollo walked to the computer cluster.
Dylan brought the pizza there.

A reasonable question based on this story is ”Where is the pizza?” However, to correctly answer this question using traditional models, one would have to correctly parse the dependencies between the people and the locations, along with the co-reference of ”there” and ”the computer cluster”. To forgo the necessity for the numerous manually designed features that these traditional models require, recent models implement recurrent neural networks that allow the models to learn the features automatically. With this in mind, Stanford Question Answering Dataset (SQuAD) was recently compiled by crowdworkers that consists of excerpts from Wikipedia articles, questions about these excerpts, and the locations in the excerpts of the answers to these questions. Our task is to train a reading comprehension model based on deep learning that is able to accurately predict the locations of these answers given the corresponding excerpts and questions. To do so, we use a neural network-based approach implemented through TensorFlow. Each word in the contexts and questions is represented as a word vector, and these representations are encoded through a recurrent neural network (RNN) composed of bi-directional long-short term memory cells (LSTMs). An attention model is used to identify candidate words in the contexts that could form part of the answer, weighing these word vectors heavily. Features were extracted from these question and context encodings, and matched with each other to generate a matching vector. The start and end indices were then predicted by applying a linear transformation to the matching vector, forming a scores vector, followed by a softmax function to convert these scores to a probability distribution.

2 Background

Reading comprehension has long been a prized goal of natural language processing, and with the recent availability of large datasets and machines with immense processing power, there have been many recent gains in this area of study. Many of these gains are the results of artificial networks, particularly neural networks, which have the capability to be fed input and output data to then create a black box function that captures general relationships between the two. The architecture of such networks determines the potential the network has to perform, and as such a variety of architectures have been developed for reading comprehension and here we will overview some the architectures developed for the specific task of SQuAD.

First, let us touch on something shared by all relevant work on this problem: the use of GloVe vectors [1]. GloVe vectors are dense vector representation of words that encode their semantic similarities using a global context co-occurrence matrix, and as such are useful for nearly every natural language processing task that works with words as tokens. Another similarity between many approaches is the use of LSTMs to encode the questions and contexts, before comparing the two. This allows for a richer representation of the inputs that captures temporal relationships between words. While this initial step is quite standard, the methods of comparing the questions and contexts vary widely. One approach is to use co-attention models, which directly identify similarities between the contexts and the questions, creating a co-dependent representation of the two [3]. Some models extract features of the question words and context words, using those to identify context words that are likely candidates to contain the answer [6]. Still others treat the contexts as a live stream, terminating the stream early when the models believe that the answer span has already been completely read [5]. There are also multiple methods to predicting the answer from the mixed context and question representations. One approach is to approach SQuAD as a classification problem and classify every word in the context as either an answer word or a non-answer word. A second approach is to predict the start index and the end index of the answer directly from the context and question mixture representation. Of course, there are many other architectures and approaches that have been used, but not all of them can surveyed here.

A model that is currently the third best performing model on the SQuAD dataset is the multi-perspective matching model. For every context word, this model creates a set of vectors that are a mixture of the given context word with the full question representation and the representation of each word in the question. In our model is a modification on the multi-perspective matching model and shares much of its structure with some novel changes to the architecture.

3 Approach

3.1 Problem statement

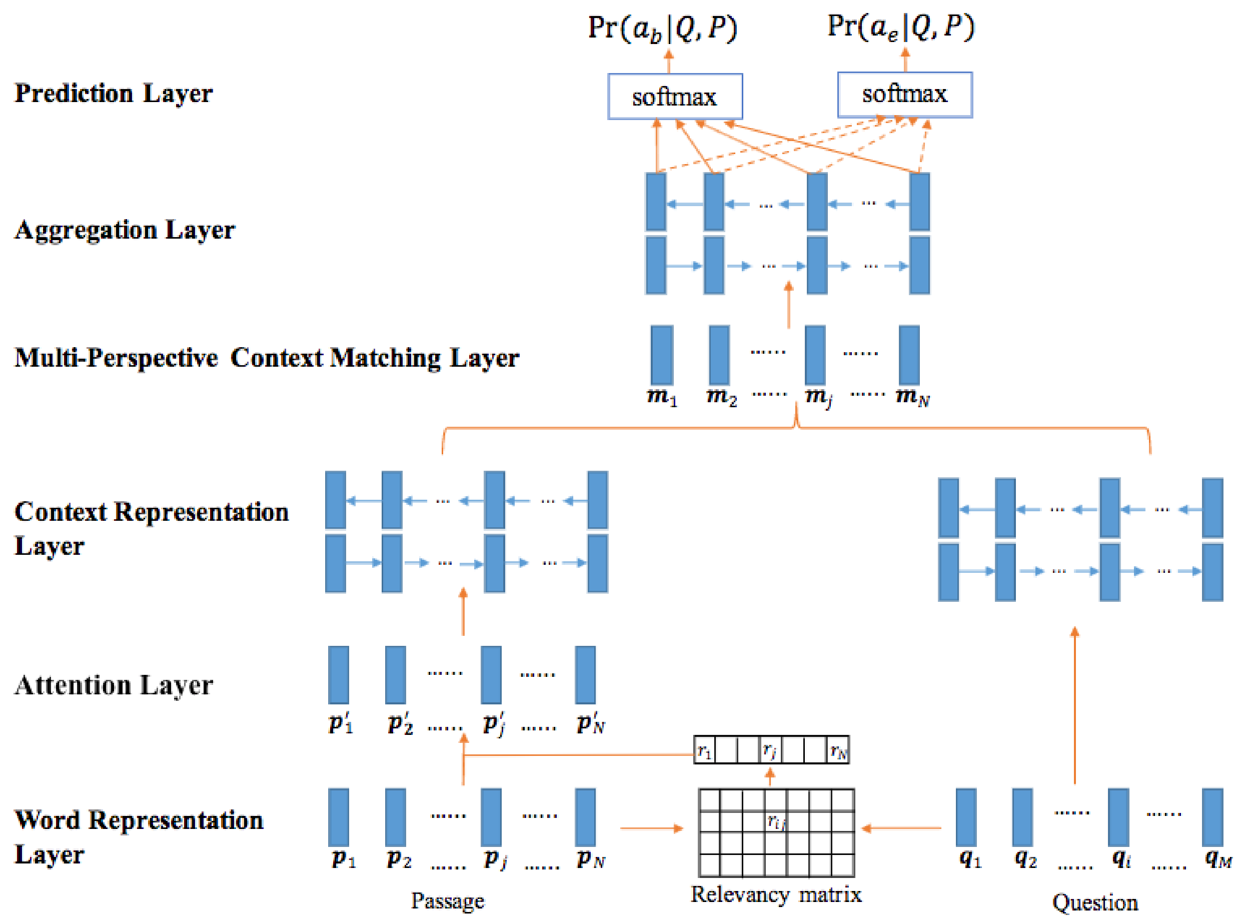
The SQuAD dataset consists of over 100,000 tuples (C, Q, A), where C is a context, Q is a question, and A is an answer. The question is represented as $Q = \{q_i\}_{i=1}^{N}$, where each q_i is a word and N is the length of the question. Similarly, the context is represented as $C = \{c_i\}_{i=1}^{M}$, where each c_i is a word and M is the length of the context. The answer is represented as $A = (a_0, a_1)$, where a_0 is the index of the first word of the answer in the context, and a_1 is the index of the last word of the answer in the context. The answer is thus treated as the substring of the context represented as $\{c_i\}_{i=a_0}^{a_1}$. We then generate probability distributions p, q over the context words to predict the answer. That is, $p(c_i) = \mathbb{P}[a_0 = c_i | Q, C]$ and $q(c_i) = \mathbb{P}[a_1 = c_i | Q, C]$. We generate predictions $a_0 = \arg\max_i p(c_i)$ and $a_1 = \arg\max_i q(c_i)$.

Two metrics are useful in evaluating model performance on this task: F1 and exact match (EM). Let p denote the precision of the predictions, and let r denote the recall of the predictions. Then the F1 score is defined to be $\frac{2pr}{p+r}$. The EM score is simply the fraction of the data on which the predicted answer exactly matches the actual answer. To combine these into a single metric, we simply take their geometric mean.

3.2 Preprocessing

From SQuAD, 87,599 data were allocated as a training set for our model. In these data, the maximum length of any context was 766 words, and the maximum length of any question was 60 words. Over 95% of the data had context lengths that were less than $\frac{1}{4}$ the maximum context length and question lengths that were less than $\frac{1}{3}$ the maximum question length. We truncated the data at these lengths since any recurrent neural network training on these data would have very few training examples for the cell units that occur after the truncation length, and therefore the model would train poorly on these later cell units. Furthermore, this would reduce our model size significantly and allow for other parameters elsewhere. For data in the training set whose answers lie beyond this truncation length, we simply removed them from our training set since the cross-entropy loss would not be defined.

We embedded each of the words in the contexts and the questions using 100-dimensional Global Vectors (GloVe), denoted as the word representation layer in figure 1, and the questions were then encoded through a bi-directional RNN of LSTM cells with 100-dimensional hidden states and a dropout rate of 0.2, denoted as the context representation layer. As a baseline model, we leave the context word representations as are. In our advanced model, we calculate attention over these context word representations and re-weight them according to their attention, denoted as the attention layer. For each question, we have a forward representation $q^{(f)}$ and a backward representation $q^{(b)}$, whose concatenation we denote $q = q^{(f)}||q^{(b)}$. The attention over each context word c_i was then computed to be $a_i = q^T W c_i$, where W is a weights matrix to be learned. These raw attentions were converted to a probability distribution by applying the softmax function, and the context word representations were re-weighted by this quantity $C := \text{softmax}(a) \circ C$, where $\circ$ denotes the Hadamard product. These context word representations were then encoded through the same RNN with a new set of weights, again denoted as the context representation layer.



$\Pr(a_b|Q,P)$
$\Pr(a_e|Q,P)$
Prediction Layer
softmax
softmax
Aggregation Layer
Multi-Perspective Context Matching Layer
m_1 m_2 ...... m_j ...... m_N
Context Representation Layer
Attention Layer
p'_1 p'_2 ...... p'_j ...... p'_N
r_1 r_j r_N
r_{ij}
Word Representation Layer
p_1 p_2 ...... p_j ...... p_N
q_1 q_2 ...... q_i ...... q_M
Passage
Relevancy matrix
Question

Figure 1: A visual representation of our question answering model, adapted from Wang, et al [2].

3.3 Matching

For a given question word encoding q and a given context word encoding c, we define a function

$$f_i(c, q) = \cos(w_i \circ c, w_i \circ q). \tag{1}$$

This extracts the i-th feature of the context encoding and the question encoding by taking the Hadamard product of each with a weight vector that acts as a feature extractor. The cosine similarity of the i-th feature of the question encoding and the i-th feature of the context encoding then defines a component of our matching vector. To accommodate different styles of contexts, different matching strategies were used. Consider the following three pairs of questions and passages.

Question: For whom does the bell toll?
Passage: I am involved in mankind, and therefore never send to know for whom the bell tolls; it tolls for thee.

Question: Who has gathered in Norway a group of lawless resolutes to reclaim the lands that his father lost?
Passage: Now, sir, young Fortinbras, of unimproved mettle hot and full, hath in the skirts of Norway here and there shark'd up a list of lawless resolutes for food and diet to some enterprise that hath a stomach in 't; which is no other (as it doth well appear unto our state) but to recover of us, by strong hand and terms compulsatory, those foresaid lands so by his father lost.

Question: Which U.S. president accused Obama of wiretapping his microwave?
Passage: U.S. president Donald Trump has recently accused Obama of wiretapping his microwave.

One matching strategy is full matching, which compares the forward and backward representations of the entire question with the forward and backward representations of each word in the context, respectively. That is, we have matching vectors $m_i^{(f)} = f_i(c_j^{(f)}, q^{(f)})$ and $m_i^{(b)} = f_i(c_j^{(b)}, q^{(b)})$ for each $c_j \in C$. The forward full matching vector is useful in answering the first question above since the answer in the context follows the phrase that strongly matches the question, thus identifying the end of the context as a viable candidate for the answer. Similarly, the backward full matching vector is useful in answering the second question above since the answer in the context precedes the phrases that strongly match the question, thus identifying the beginning of the context as a viable candidate for the answer.

A second matching strategy is max-pool matching, which compares the forward and backward representations of each word in the question with the forward and backward representations of each word in the context, respectively. For each word in the context, the maximum value of the matching over all words in the question is taken to be the corresponding component of the matching vectors $m_i^{(f)} = \max_k f_i(c_j^{(f)}, q_k^{(f)})$ and $m_i^{(b)} = \max_k f_i(c_j^{(b)}, q_k^{(b)})$ for each $c_j \in C$. This strategy identifies locations in the context that correlate strongly to parts of the question.

A third matching strategy is mean-pool matching, which again compares the forward and backward representations of each word in the question with the forward and backward representations of each word in the context, respectively. For each word in the context, the mean value of the matching over all words in the question is taken to be the corresponding component of the matching vectors $m_i^{(f)} = \text{avg}_k f_i(c_j^{(f)}, q_k^{(f)})$ and $m_i^{(b)} = \text{avg}_k f_i(c_j^{(b)}, q_k^{(b)})$ for each $c_j \in C$. This strategy is useful in answering the third question above since the answer in the context is in between key phrases, so neither the full forward representation of the context nor the full backward representation of the context would suffice to answer the question.

Utilizing these strategies, multi-perspective matching was then performed between the question embeddings and the context embeddings, denoted as the multi-perspective context matching layer in the figure. Different weights were used for each matching strategy and for each direction. Five features were extracted from each question word and each context word, and these features were used to generate matching vectors. These matching vectors were then concatenated to generate a final matching vector, denoted as the aggregation layer. The start and end indices were then predicted by applying a linear transformation to the matching vector, followed by a softmax function to convert these scores to a probability distribution, denoted as the prediction layer, with different weights for the start index and the end index. The cross-entropy loss of these predictions was then the objective that we minimize with our model. The ADAM optimizer was used to train the RNN, using a learning rate of 0.0005 and a batch size of 10 data. The gradients were clipped at a magnitude of 5.

Table 1: Results of our baseline and advanced model on the development set

MODEL	F1 SCORE	EM SCORE
Baseline	19.4	10.0
Advanced	22.5	13.0

4 Experiments

We evaluated our model on the development set of 10,570 data that is provided by SQuAD, and the results are summarized in table 1. Unfortunately, we were not able to perform a successful submission to CodaLab to obtain results on the test set, which we have a note about at the end of this paper. Due to time constraints, we were only able to run the baseline model and advanced model with one set of hyper-parameters to achieve the results in the table above and as such the conclusions we can draw from the table are limited. That being said, with our single set of hyper-parameters, we can see that the novel architecture we improved both the F1 and EM scores.

While we have already noted that this is weak evidence, we can outline some reasoning here on why our modification would result in better performance. When the attention layer is applied in place of the filter layer, the model is given parameters to capture what makes a context word more or less relevant to the question. The filter layer provides no parameters for the model to learn, scaling all context vectors by the cosine similarity. The bilinear attention layer gives the model the ability to adjust parameters, and account for the fact that a context word simply having a high or low cosine similarity with the question representation does not conclusively determine that the context word is useful or not useful for the answer. This allows the model to better focus in on the appropriate section of the context representation before further computation is done.

Something that must be noted is that our results are drastically lower than those in the original multi-perspective matching paper. This could literally be due to hyperparameters being off during the single time we trained the baseline and advanced models, as we had no time to make any adjustments at all. This possibility is further evidenced by the fact that our model successfully over-fits training data over a couple epochs, and has no problem completely over-fitting small samples. We believe these numbers would see drastic improvement given more time for tuning.

5 Conclusion

Much of our time was spent on debugging the baseline model, and we unfortunately didn't have a functional model until the night before the poster session. Therefore, we had very little time to experiment with modifying our model, and the only innovation we were able to apply was to modify the paper by Wang, et al to replace their filter layer with a more powerful attention layer [2]. With more time and resources, we would have liked to incorporate features of other successful models into ours. One feature we would have liked to incorporate was the early stopping feature that Yu, et al implemented that would allow for faster computation time since the model doesn't need to examine irrelevant parts of the passage that occur after the answer span. Another feature we would have liked to incorporate was a co-dependent representation of each question and answer pair. Instead of our current model, wherein we have separate representations of each question and each context, then attempt to match the two, we would like to have a single co-dependent representation of the two that would facilitate matching. Given time and resources to implement these two features and to better tune our hyper-parameters, we believe that we can attain significantly better results.

6 Closing remarks and an appeal to humanity

We are including this section at the end of our paper to explain why we do not have a successful submission to the test board. After working constantly on this project for multiple weeks, we finally got our model to work, not with incredible results, but at least with the results that we have reported in this paper. For the last 15 hours straight we have been working to fix a bug on coda lab where our process is killed:

```
bash: line 1: 8 Killed ( python codeq/qa_answer.py --dev_path dev.json )
 > stdout 2> stderr.
```

There is nothing in stderr or std out. This is a generic OOM error and it continues to occur as I have decreased our batch size. At 7:30 AM, we put forth one final submission with a batch size of 1, but we are assuming that it is going to run into the same error, and in fact while writing this the submission failed with the same error. We have used all 3 of our submissions to the test board, and while failed submissions don't count in principle, they must be manually removed, and Stephen Koo appears to have departed, leaving us the following message.

"Alas I am going to bed now for a final tomorrow morning. Hope all goes well with your final CodaLab submissions and nothing catches on fire. If your test submission fails, since I cannot give you your stacktrace in my sleep (if only I could!), you will have to rely on guesswork."

Whilst nothing has caught fire yet, we are unable to continue debugging or submitting to CodaLab as our submission attempts have expired. This all being said, it would be a great great disappointment to have all our work come to naught, and we are wondering if there is some way the work we have done can be counted towards our grade despite our lack of final submission on the leaderboard. We understand that lacking a final submission is going to result in a grade drop, but this assignment has frankly been a hellish time commitment, and it would be utterly heartbreaking to have so much work culminate to a working model, but a failing or near failing grade on the assignment. We hope our final grade can be a reflection of the work we have done and the successful model we built, and not a reflection of a 15 hour battle with the submission processes. That being said, we understand whatever decision you come to and we know all the chaos around this assignment cannot be fun for you guys either.

Thank you and best of luck.

Atticus and Dylan

7 References

[1] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.
[2] Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. Multi-perspective context matching for machine comprehension.
[3] Caiming Xiong, Victor Zhong, Richard Socher Dynamic Coattention Networks For Question Answering
[4] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2016.
[5] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end answer chunk extraction and ranking for reading comprehension. arXiv preprint arXiv:1610.09996, 2016
[6] Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. CoRR, abs/1608.07905.