Reading Comprehension On SQuAD Using Tensorflow

Chase Brandon QCHASEB@STANFORD.EDU
Department of Computer Science
Stanford University

Michael Holloway MJH17@STANFORD.EDU
Department of Computer Science
Stanford University

Micah Silberstein MICAHS@STANFORD.EDU
Department of Computer Science
Stanford University

Editor: Chase Brandon, CodeLab Name: fooshed

Abstract

This paper describes attempts at developing a high performance reading comprehension model. Reading comprehension, in the current context, is defined as extracting textual answers from context passages given qualitative text questions. To train and test our comprehension model, we use Stanford's SQuAD dataset, a standard dataset for comprehension tasks. We present our initial implementation of the AttentiveReader model as first presented by Herman et al. We also introduce our use of Bi-Directional GRU cell's as an extension to this model. We discuss the advantages of our chosen model in regard to the reading comprehension task. We also discuss some of its shortcomings. In order to overcome these shortcomings, we introduce and analyze several other models presented in various recently published research papers. Many of the discussed papers are currently ranked near the top of SQuAD's leaderboard in regard to the ExactMatch and F1 scoring metrics.

Keywords: Bi-Directional Long Short-Term Memory, Deep Learning, BiLSTM, SQuAD, GRU

1. Introduction

Before we can dive into our model development, we must first introduce the reading comprehension task and its various complexities. The task of training a machine to develop reading comprehension is formally known as natural language understanding (NLU), a specific task in the field natural language processing (NLP). A reading comprehension task is composed of three elements: context, question, answer.

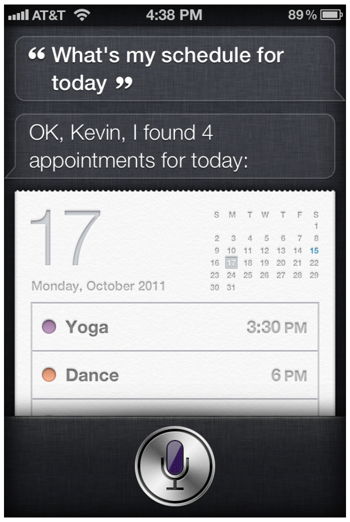
The context component of a natural language understanding task can vary widely depending on the task at hand. For example, the context could be a selected passage from a book or it could be a selection of entries from a person's calendar. Generally, the context refers to the textual situation that the machine is attempting to comprehend.

The question component usually refers to a human supplied query related to the context component. This query is freely formed in the sense that the machine does not provide any mechanism to restrict or refine the user's input; instead, the supplier of the query simply treats the NLU machine as a black box. The unstructured nature of the question component dramatically increases the complexity of NLU tasks.

The answer in a natural language understanding task is a strict subset of the context component that best addresses the query from the question component. The answer to a query is highly dependent on the context aspect of the task. For example, if the context of the task is a calendar, and the query is "When is my appointment with Dr. Smith?", then a correct answer component would most likely be a date and time combination.

Give modern example's of its usage (Google search, siri)

Natural Language Understanding has become important in industry as companies continue to add NLU features to existing products, as well as introducing NLU reliant products to consumers. Siri, Apple's voice assistant, has a heavy reliance on NLU. The context space of one of the voice assistants is quite large because the question component can refer to an array of different contexts, such as a phone book, calendar, and email.



" What's my schedule for today "

OK, Kevin, I found 4 appointments for today:

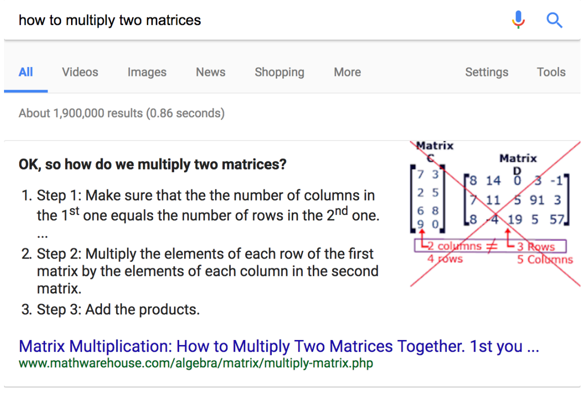
17
Monday, October 2011

Yoga 3:30 PM

Dance 6 PM

Figure 1: An example of Siri's NLU functionality given a calendar context and question pair.

Google utilizes NLU in a multitude of products. Google Search will often use natural language understanding on interrogative based queries in order to surface and highlight potential answers. While Google's system often yields correct and informative results, even it is prone to failure. Google's inability to develop an error free NLU system demonstrates the difficult of the task.



how to multiply two matrices

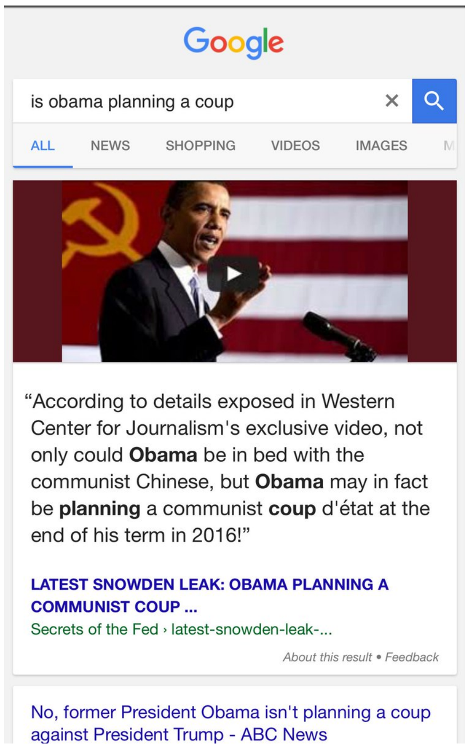
About 1,900,000 results (0.86 seconds)

OK, so how do we multiply two matrices?

1. Step 1: Make sure that the the number of columns in the 1st one equals the number of rows in the 2nd one.
...
2. Step 2: Multiply the elements of each row of the first matrix by the elements of each column in the second matrix.
3. Step 3: Add the products.

Matrix Multiplication: How to Multiply Two Matrices Together. 1st you ...
www.mathwarehouse.com/algebra/matrix/multiply-matrix.php

(a) An example of Google Search's NLU system providing a good answer given a query asking how to multiply two matrices.



is obama planning a coup

"According to details exposed in Western Center for Journalism's exclusive video, not only could **Obama** be in bed with the communist Chinese, but **Obama** may in fact be **planning** a communist **coup** d'état at the end of his term in 2016!"

LATEST SNOWDEN LEAK: OBAMA PLANNING A COMMUNIST COUP ...
Secrets of the Fed › latest-snowden-leak-...

About this result • Feedback

No, former President Obama isn't planning a coup against President Trump - ABC News

(b) Google Search's NLU system providing answer annotation from an unreliable source.

Figure 2: Examples of Google Search's NLU functionality.

Relate the three components to our reading comprehension task.

For our NLU task, a reading comprehension task, it is important that we define our context, question, and answer components. Our context are selected text passages, no longer than a paragraph. The question component is similar to the question component specified above. We assume that the question is relevant to the given context passage. We define our answers to be a single token or a series of sequential tokens found within the context. The number of possible answers for any given context is quadratic with respect to the number of tokens in said context. We can calculate this possible number of answers for a given passage with N passage tokens as shown:

$$O(\frac{N^2}{2} + N) = O(N^2) \qquad (1)$$

To test and train our data, we use a dataset that is tailored for our reading comprehension task, Stanford's SQuAD dataset. SQuAD, or Stanford Question Answering

Dataset, is a reading comprehension focused dataset that consists of over 100,000 question-answer pairs. The dataset spans over 500 different articles (contexts). This is one, if not the largest reading comprehension focused datasets available, and therefore it is ideal for our use.

2. Background/Related Work

We began our model implementation by developing a baseline Bi-Directional Long-Short Term Memory, or BILSTM, model. In order to understand what a BILSTM is, we must understand LSTMs, and before then we must first understand what an RNN is.

An RNN, or Recurrent Neural Network, is a type of neural network that is optimized for the sequential nature of certain datasets. Unlike traditional neural networks, RNNs do not treat all inputs independently; instead, RNNs recurrently compute the same thing for every element in a sequence, using previous values as additional inputs. As a result, RNNs maintain a memory of what they had encountered before, making them ideal to model datasets where the sequential ordering of the data is relevant.

However, RNNs often fail to maintain memory for long term dependencies. For example, consider a passage that contains the phrase "I grew up in Lisbon." followed by the phrase "..., fortunately, I knew [unkown]". We would want to be able to predict that the unknown token is most likely "Portugese", but the separation between the two phrases would make this highly unlikely because of an RNN's lack of long term memory. LSTMs are a specific type of RNN that introduces several complexities that allow it to better maintain a longer memory. The main features of LSTMs that increase their long term memory are the cell state and gate structures. Cell state is like a highway running linearly through the network that maintains information over long periods. Information is added to or removed from the cell state in a regulated way through gates. Gates regulate the degree of change that is let through by employing a layer with a sigmoid function.

Although LSTMs are able to maintain long term dependencies, as they stand, they are only able to make predictions based off previous elements in a sequence. This is the reason for the introduction of the bi-directional aspect to our baseline neural network model. Consider our example passage above regarding "Lisbon" and "Portugese". It is reasonable to assume that after our unknown token, their may be other tokens which would help us predict that unknown token more accurately. For example, if we knew that the word "fluently" occurred directly after the unknown token, the model should have a better notion that the unknown token most likely refers to a language. To add this functionality, we simply stack two RNN/LSTMs on top of each other, yielding a neural network that can make predictions by looking at both the left and right sides of a given context in a sequence. These stacked RNN/LSTMs yield a network known as a Bi-Directional RNN/LSTM.

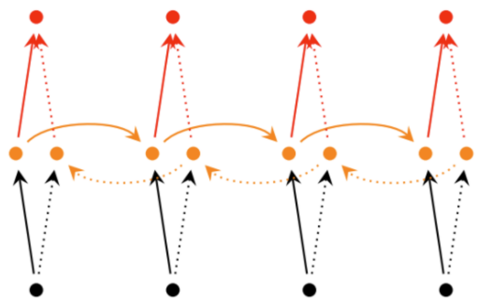


Figure 3: A abstract visualization of a Bi-Directional RNN.

One major concern to deal with when developing models that rely on neural networks is that of overfitting. In the context of a neural network, overfitting occurs when an excessively complex model describes random error as opposed to the underlying statistical relationships. An overfit model often performs poorly on test sets as it is tailored to minor fluctuations in the training dataset. In order to avoid overfitting, we can employ a regularization strategy called dropout. Once implemented, dropout eliminates individual nodes in the net with a certain probability. By eliminating certain nodes during various training steps, dropout ensures that these units do not "co-adapt" too much.

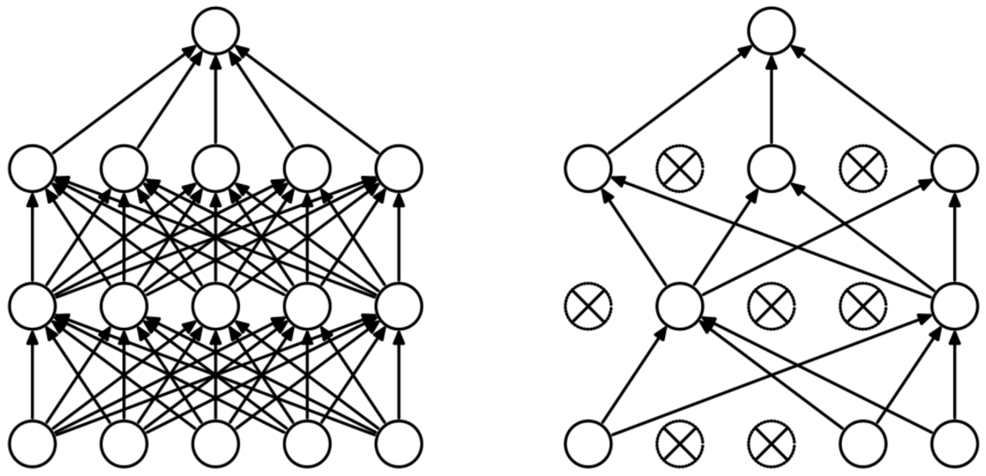


Figure 4: A visualization of a network before and after dropout is applied.

Now that we have a solid understanding of the engineering behind our baseline, we can explore ideas presented in research papers that have resulted in competitive models as assessed by the SQuAD leaderboard. First, we will explore the ideas presented in the paper entitled "Multi-Perspective Context Matching for Machine Comprehension" (Wang et al.). This paper revolves around the premise idea that a question and its associated answer are likely to have similar contexts. For example, for a question such as “Where in Spain is Barcelona located in?”, the context is clearly location-based; therefore, the correct answer, in this case “Eastern Spain”, would also have a location-based context. To model this, Wang et al. used a specific model called a Multi-Perspective Context Matching model that encoded the context similarity to the given question at each point (possible answer) in the given passage. By doing this, possible answer spans were weighted according to context overlap, resulting in competitive SQuAD results.

Next, we investigate the Dynamic Chunk Reader model as presented by Yu et al. One of the challenges of NLU tasks is the fact that answer spans are of variable length. Factoid based question, answer pairs such as “Who won SuperBowl 50?” — “Denver Broncos”. are easier for models to comprehend because their answers are generally named entities or single token facts. However, a robust NLU system must be able to perform well for non-factoid based questions, such as those that ask for explanations. As a way to allow for non-factoid NLU, Yu et al. present a way to generate potential answer spans (chunks), and then rank those answer spans accordingly. The important aspect that we were interested in regarding this paper was the method by which the chunking was performed. First, all of the part-of-speech (POS) patterns from all of the answer spans in the training set were captured. After this was performed, the model iterated through all of these patterns, extracting matching sub-sequences from a given context paragraph. We were very interested in this method because it would be able to significantly increase our models ability to capture factoid based question-answer pairs. Next, for non-factoid based questions, a chunking approach was taken that generated up to N number of chunks of length L from the given context paragraph. This two-fold method for potential answer span generation appeared to greatly increase Yu et al.’s performance on the SQuAD dataset.

Next, we explore “Learning Recurrent Span Representations For Extractive Question Answering” by Lee et al. As has been previously discussed, in a context paragraph with N tokens, the number of possible answer spans is bounded by $O(N^2)$.

The previous two discussed papers have attempted to avoid this computational complexity by developing various mechanisms by which the possible answer space could be reduced from this quadratic space. However, Lee et al. demonstrate that it is beneficial to generate and test all possible answer spans. In order to overcome the quadratic answer space issue, these researchers constructed a neural architecture they called RaSoR, which constructs all span representations, but re-uses previously computed span substructures. For example, consider the context phrase “...in the Dominican Republic’s capital city of Santo Domingo”. If we were to enumerate all possible answer spans just from this excerpt, we would see that there would be many common substructures. By reusing recurrent computations from these substructures, the space of answer span embeddings is reduced from cubic to quadratic. The almost dynamic programming approach used by this neural architecture allows the network to benefit from training on all potential answer spans, while overcoming the computational pit falls of exposing the model to cubic space.

This RNN, LSTM, and BiLSTM knowlege in conjunction with the various approaches presented in the previous three research papers lays the foundation for our experimental process detailed below.

3. Approach

We decided to proceed with developing a baseline model based off of the AttentiveReader model proposed by Herman et al. and further explored by Chen et al. In the AttentiveReader model, the context paragraph and question query are each individually encoded using single layer Bi-Directional LSTMs. The major difference between this model and basic Deep LSTM models, as discussed by Herman et al., is the manner in which attention is employed. In the basic LSTM model, attention is focused on the sentence level; every sentence is expressed as a bag of embeddings. With AttentiveReader, however, the attention is more focused, using token embeddings based of their entire future and past context as seen in the entirety of the context paragraph.

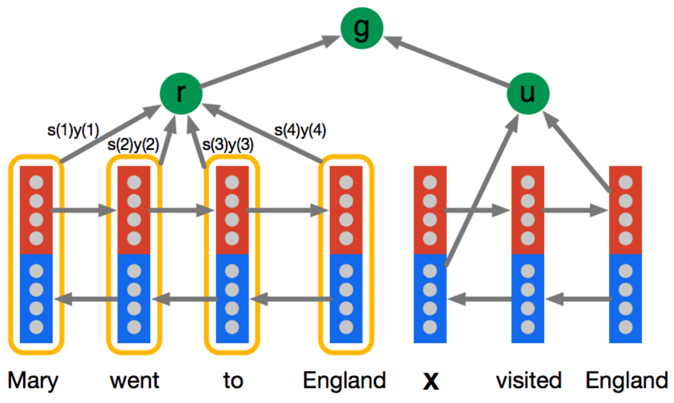


Figure 5: Herman et al.'s vizualization of the AttentiveReader model's encoding of sub-span of a given context paragraph.

On top of the AttentiveReader model, we introduced Bi-Directional Gated Reccurent Units. The motivation for this addition was the fact that we want the context surrounding each token embedding to have a more persisten memory. We explained the motivation for the Bi-Directional aspect of the GRU model previously in greater detail.

4. Experiments

Inspired by Wang et al., we first became interested in exploring different ways to increase our models performance via part of speech tagging. As previously discussed, Yu et al., significantly increased model performance by utilizing part of speech tagging for factoid based questions. We experimented using the Natural Language Toolkit (NLTK) python package to try to identify potential factoid answers. However, we struggled to be able to develop a model that coped well with presence of both factoid and non-factoid answer types. As a result, we abandoned our attempts to introduce a POS aspect to our model.

We also attempted to incorporate the ideas presented in "Multi-Perspective Context Matching" by Wang et al. Once again, the assumption from that paper was that question-answer pairs are more likely to have similar contexts than not. We attempted to incorporate a context similarity of proposed question-answer pairs into our model; however, once again we were unable to successfully do so. We do think that this would give our model a significant performance bump, but we were simply unable to develop a working implementation.

5. Conclusion

In conclusion, our group was very humbled by this experience. The transition from reading and understanding the theory behind cutting edge models to actually implementing them in code was much harder than we anticipated. It was rewarding to finally develop a solid baseline model, but also frustrating that we had a harder than expected time introducing additional complexities. We feel as though the 80+ hours dedicated to this project gave us a much better appreciation of the difficulty in implementing the models discussed in the research papers we read.

6. References

References

[1] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espholt, Will Kay, Mustafa Suleyman, Phil Blunsom *Teaching Machines to Read and Comprehend.* , November 19, 2015.

[2] Zhiguo Wang, Haitao Mi, Wael Hamza, Radu Florian. *Multi-Perspective Context Matching for Machine Comprehension.* December 13, 2016.

[3] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, Bowen Zhou. *End-to-End Answer Chunk Extraction and Ranking for Reading Comprehension.* October 31, 2016.

[4] Kenton Lee, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das. *Learning Recurrent Span Representations for Extractive Question Answering*. November 14, 2016.