Michael Znidarsic

3/27/20

NLP

**Textual Entailment Classification**

In recent years, data scientists have sought to enable computers to understand human languages in the forms of text and speech. While computers may be able to discretize text into words, sentences, and paragraphs, they have a truly difficult time understanding its meaning. Grammar, homonyms, slang, and missing context are some of the most difficult obstacles to overcome. As part of the effort to reach past simple parsing of speech, NLP experts have sought to teach computers the task of Textual Entailment: when given two sentences, a machine is asked whether the first entails, contradicts, or has no relationship with the second. To provides a large, relevant dataset to serve as a benchmark for future Textual Entailment research, the Stanford NLP group put together the SNLI dataset consisting of hundreds of thousands of pairs of sentences labelled “entailment”, “contradiction”, or “neutral”. This study will use the SNLI dataset to attempt to build a neural network model that can classify this difficult dataset at a high degree of accuracy.

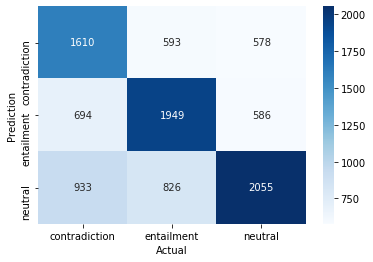
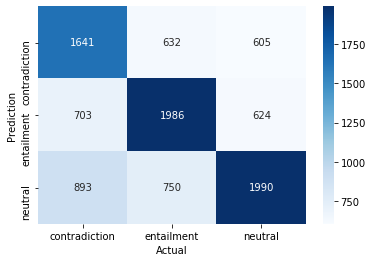
**Pre-Processing and Tokenization (Steps 1 & 2)**

The pairs of sentences and their labels were read in using Pandas. Each label type (e.g. “Contradiction” becomes 2) was assigned an integer, and from this format the labels were convert to one-hot encoding (e.g. 2 becomes [0,0,1]). This format allows neural networks (which do not accept text or strings) to perform categorical cross-entropy to develop a loss function and arrive at predictions. NLTK’s word\_tokenize function was then used on each sentence to convert it from a string to a list of strings. A dictionary was created wherein each unique word used in the training set was assigned an integer ID. The sentences were then run through the dictionary to yield sentences that looked like the following: [12265, 3517, 11577, 12162, 1866, 7271, 7007, 10515, 2452, 2966, 5871, 7633, 2462, 10034, 6575, 2448]. These sentences were then set to a single maximum length. All sentences with more words than the maximum length were truncated to the maximum length. All sentences with fewer words than the maximum length were “padded” to that maximum length (indicators for a padding spot, which happened to be an integer ID of 0, were added until the maximum length was reached). This whole process ensured inputs of a uniform length to the neural network. The intention of padding is for the neural network to learn that the padding indicator (0) means the absence of a word, and thus to avoid undue noise in the data. Any words present in the validation (testing) dataset that were not added to the dictionary by the training data were assigned an “Unknown” indicator (an integer of 3), the purpose of which is similar in concept to padding.

**Experiment #1**

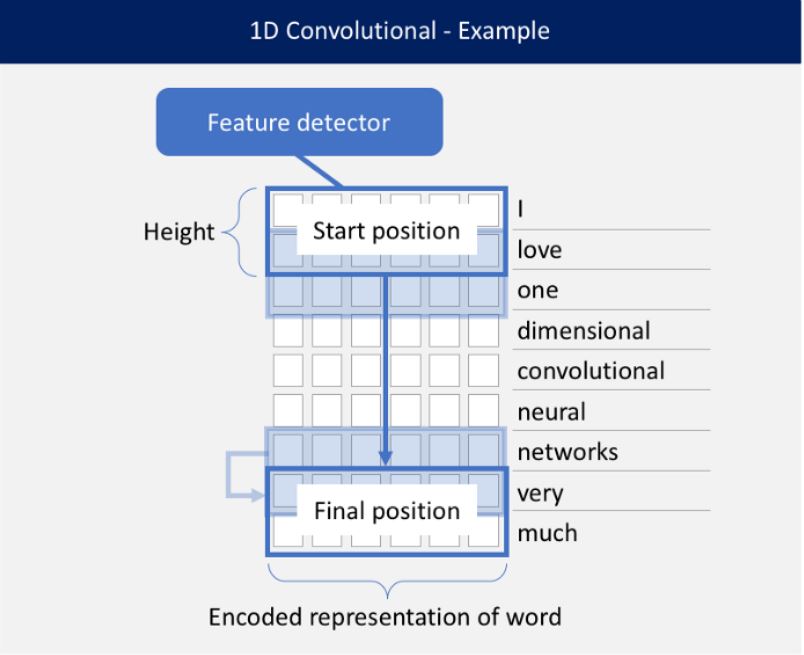
Since the contrast between two (generally similar) sentences relies a great deal on the semantics and parts of speech involved in each, it is necessary to see whether distinguishing word tokens by parts of speech would help accuracy. For example, the word “lead” could either be a noun or a verb, serving completely different roles in a sentence. Two models were built to compare whether tokenizing by word *and* part of speech would be better than simple word tokenization. For the parts-of-speech inclusive model, the following uses of the word “lead” were all tokenized separately: ('lead', 'VBP'), ('lead', 'NN'), ('lead', 'VBN'), ('lead', 'JJ'). This was accomplished using the NLTK pos\_tag parser. For the simple model parts of speech were not distinguished, and all the following uses would simply be tokenized as “lead”.

The model used to test these token configurations is an early version of the one described in Experiment #3 as “Model #1”. That is to say: a convolutional neural network with word embeddings. The confusion matrix on the left shows the model’s results without parts of speech and the confusion matrix on the right shows the model’s results with parts of speech. The simple model without parts of speech tagging achieved a 57.176% cross-validation accuracy, while the model that included parts of speech tagging achieved a 57.146% cross-validation accuracy. While the error clustering is not the same (the parts of speech model predicted Neutral far more than the other model), the accuracies are surprisingly similar. Including part of speech tags in tokenization seemed to have no real effect on a model’s accuracy. There are a couple possible explanations for this result. Firstly, the word embeddings are meant to provide an abstract measure of a word’s relationship with other words in high-dimensional space; they are probably capable of portraying several different meanings of a word. Secondly, since each data example has only two sentences of text, words probably do not occur very often in two different roles within one sentence pair. Whatever the reason, there was no discernable benefit to including parts of speech tagging in the word tokenization process.

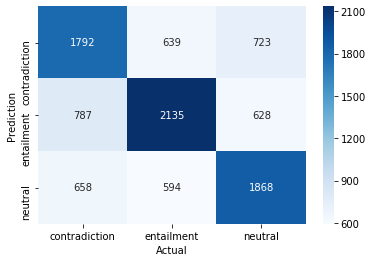
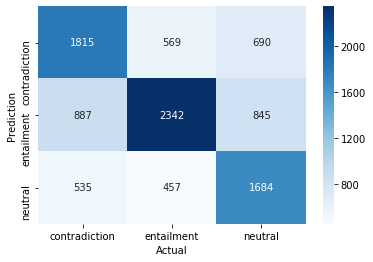


**Experiment #2**

In the convolutional neural network described in Experiment #3, the kernel size is an important parameter that determines how many adjacent words make up one convolution. Although there are many options, the most pressing comparison for such a small set of words as a single sentence is a kernel size of 2 versus a kernel size of 3. The concern: would a kernel size of 2 miss the larger patterns in a sentence, and would a kernel size of 3 be too obfuscated to catch the minute details in a sentence? The kernel size is represented as “height” in the following illustration ( taken from <https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf> ).



The results were not dramatic, but there was a consistent discrepancy. The kernel size of 3 (left) outperformed the kernel size of 2 (right). The 3 model had a cross-validation accuracy of 59.456% and the 2 model had a cross-validation accuracy of 58.988%. While this difference does not appear large, it could have more serious effects as the sampling size is scaled up in the final models (which take many hours to run). It seems that a kernel size of two is not quite able to pick up the context that three can.

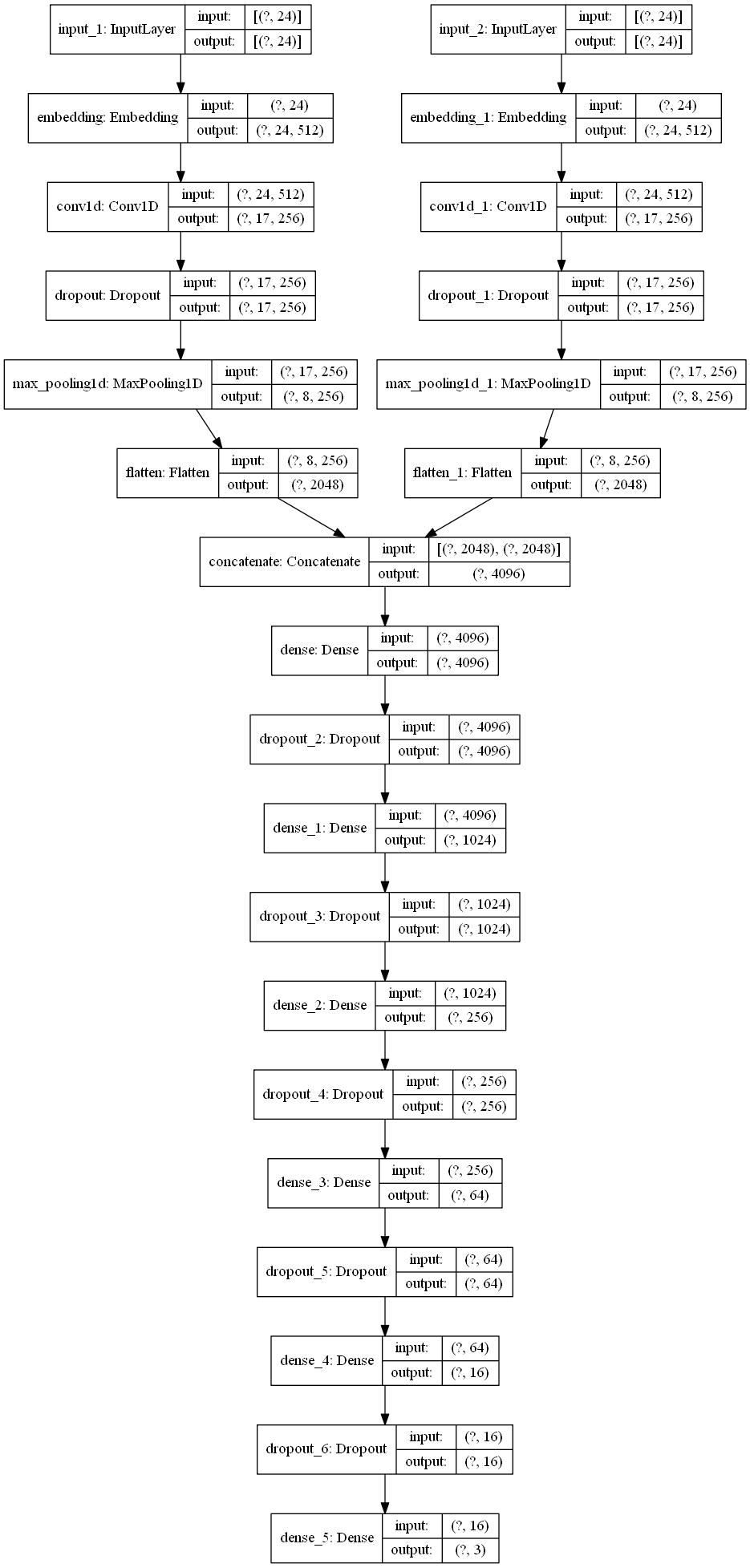


**Experiment #3**

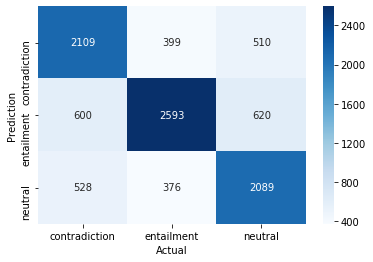
In the final experiment, two neural network architectures will be pitted against one another to determine which is best for classification of Textual Entailment.

**Model #1: 2-Input 1-Dimensional Convolutional Neural Network 512D Embeddings**

This neural network model uses the best pre-processing results of the previous experiments. It is original to this study and comes from no academic literature regarding Textual Entailment. The graph of the neural network’s architecture can be seen on the next page. Each sentence is input separately to its own 512-dimension word embedding layer. These embeddings are then convolved by a 1-dimensional convolutional layer consisting of 256 filters. These convolutions are then max-pooled 1-dimensionally, flattened, and concatenated together to form a single input to the fully connected dense layers of the neural network.

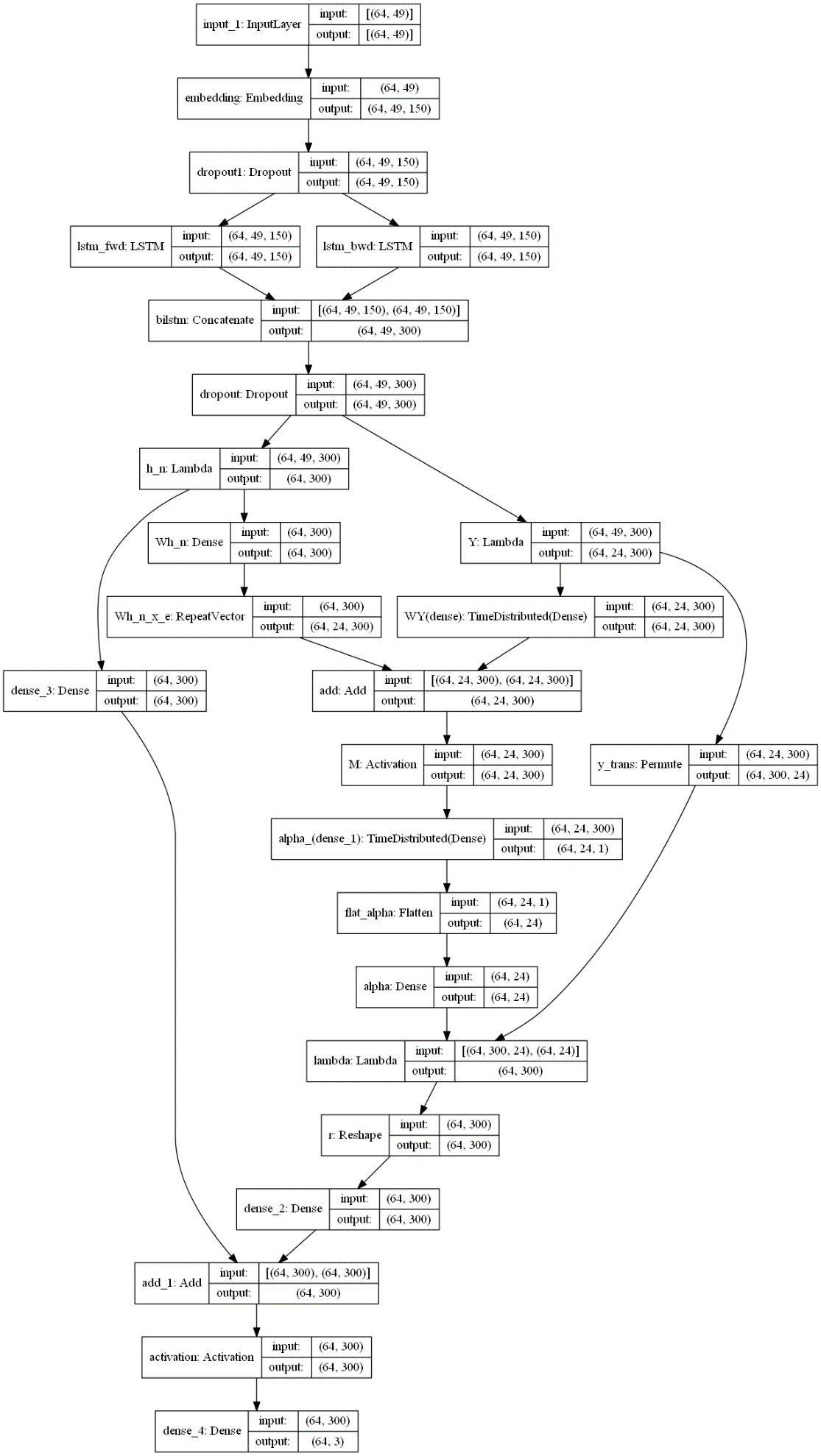


The results of this model were impressive despite the model’s (relative) simplicity. However, they were significantly worse than the lowest results on the Stanford NLP leaderboard that tracks the state-of-the-art models for the dataset over time. The overall accuracy was 69.13%. Contradiction had a precision of 70%, a recall of 65%, and an F1 score of 67%. Entailment had a precision of 68%, a recall of 77%, and an F1 score of 72%. Neutral had a precision of 70%, a recall of 65%, and an F1 score of 67%. Entailment was predicted significantly more often than the other two classes, but there was an otherwise even spreading of error and a balance in precisions. These findings suggest that the model does not suffer from undue bias.

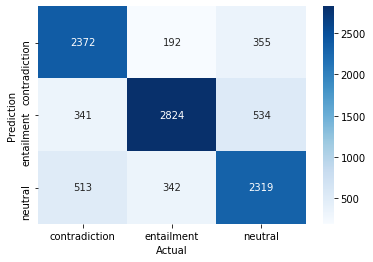


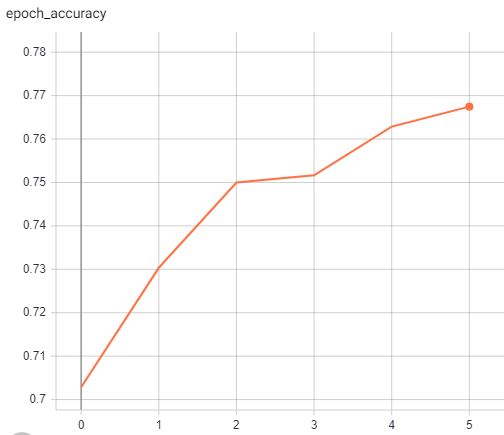
**Model #2: Bidirectional LSTM Neural Network 300D Embeddings With Attention**

The architecture for the second model took heavy inspiration from Shyam Upadhyay’s adaptation (<https://github.com/shyamupa/snli-entailment>) of Rocktaschel et al.’s work in “Reasoning about Entailment with Neural Attention” (<https://arxiv.org/abs/1509.06664>). Each sentence is embedded in the same layer, but split for a Bidirectional LSTM layer, the output of which is then given neural “attention”. This concept is best outlined in Rocktaschel’s paper, but in this application it broadly serves as a way to quantify differences between words (or positions, depending on your point of view) in the first sentence and word in the second sentence. The neural network’s architecture as it is applied in this study is illustrated on the next page.



This model achieved an overall accuracy of 76.75%, very close to the earliest models on Stanford NLP’s state-of-the-art leaderboard. Contradiction had a precision of 81%, a recall of 74%, and an F1 score of 77%. Entailment had a precision of 76%, a recall of 84%, and an F1 score of 80%. Neutral had a precision of 73%, a recall of 72%, and an F1 score of 73%. Entailment was again predicted more than the other two classes, but it was, in fact, the Neutral class that had the lowest precision and F1 score. These results indicate that this model reaches some limitation in identifying neutral sentences, which is intuitive given its greater ambiguity as a class and conceptual positioning “between” contradiction and entailment. Across the board, this model was much more effective than the first.





**Conclusion**

Given the difficulty of Textual Entailment and the nuanced nature of the SNLI dataset, this study achieved impressive accuracies. Many of the feature engineering decisions made during preprocessing did not prove to be as important as minor architectural changes in the neural networks. Inclusion of parts of speech tagging during tokenization did not supply any significant improvement in performance. A CNN kernel size of 3 proved more effective than a kernel size of 2. In the final, major experiment between a Dual CNN architecture and a Bidirectional LSTM with Attention architecture, the LSTM was clearly more effective. However, the CNN performed well despite no well-known corollary for it existing in the SNLI literature. It should also be noted that for both models, a large increase in training data size had a significant positive effect on the model’s accuracy. Although more successful models have tackled the SNLI dataset in recent years, this study should serve as a source for those seeking new approaches to the field and evidence to understand how and why computers can prove effective at Textual Entailment.