Sarcasm Detection in Product Reviews using Sentence Scale Sentiment Change with Recurrent Neural Networks

Submitted in Partial Fulfillment

of the Requirements for CS 886

Dylan Drover, MASc. Candidate

Faculty of Engineering

Department of Systems Design Engineering

August 25, 2015.

Course Instructor: Professor Ming Li

Contents

1 Introduction

Sarcasm detection in text is a particularly difficult problem and has only recently begun to be successfully examined as an automated natural language processing problem. Sarcasm is usually defined as a statement that conveys an intended sentiment while using words or phrases that, taken by themselves, express the opposite sentiment [1, 2, 3]. Sarcasm is habitually associated with negative sentiment. Linguistics considers sarcasm as analogous to verbal irony [4, 3] and the two will be treated as equivalent in the following paper.

The difficulty of detecting sarcasm is inherent in its purpose, that is, to convey a sentiment implicitly. When spoken there are numerous cues that English speakers use to (sometimes) make sarcasm apparent. These include changes in prosody (the pattern or stresses of spoken words) and physical gestures that are associated with sarcasm. Without these features, determining whether sarcasm is being used depends on other aspects. One of the text based features that has been used in previous work [1, 2] is the use of sentiment polarity shifts within a sentence or section of text. This can be used within a sentence by detecting a negative or positive situation described as its opposing sentiment but it can also be conveyed through a longer term context between sentences that changes the overall sentiment polarity.

The hypothesis tested in the following paper is to determine if inter-sentence sentiment change can be used to detect sarcasm in a body of text. Stanford's Recursive Neural Tensor Network (RNTN) (which was trained on the Stanford Sentiment Treebank) [6] was used as a pre-trained model that would determine the sentiment of a sentence and provide a label. This information was then used as a feature vector which was used as input for the model.

Training for the sarcasm classifier was provided by a corpus called the "Sarcasm Corpus" which provides Amazon.com product reviews. The set contains 437 reviews labelled as sarcastic and 817 labelled as regular reviews. Reviews' sentiments were analysed and the resultant data was used to train an Elman recurrent neural network (RNN) to classify sarcasm in a review.

2 | Background Review

Sarcasm Detection

As noted before, there have been a number of attempts to detect sarcasm. One of the main resources for training such models has been the social media site Twitter. Twitter posts can serve as a gold standard by using "hashtags" (tags for a subject or intent that are connected to a Twitter post) such as "#sarcasm". Another resource is Amazon.com product reviews.

Gonzalez-Ibanez et al. noted that sarcasm can originate from from a polarity change in a statement [1]. The goal was to distinguish between sarcastic, non-sarcastic, positive and negative Twitter posts. Their research remarked that humans attempting to detect sarcasm are not the perfect detector. Ground truth for the sarcastic tweets were Twitter posts tagged with the hashtag "sarcasm". The baselines for positive tweets were tweets tagged with positive words and similarly for the negative case. Classification was based on lexical features that were based on unigrams and dictionary-based tags. Pragmatic features were also used, these included "emoticons". Final classification was performed with Support Vector Machines (SVM) and Logistic Regression. The maximum classification accuracy between sarcastic and non-sarcastic tweets was 68.33% using unigrams and a SVM. Interestingly, human accuracy was 66.85%.

A language agnostic system was attempted by Ptacek et al. for detecting sarcasm in tweets in both English and Czech [5]. The paper also noted that changes in sentiment polarity were indicative of sarcasm. The features that were chosen were uni, bi, tri-grams, punctuation, emoticons, capitalization, and skip-grams. The features were then used in Maximum Entropy and SVM classifiers. The final results were much better for English sarcasm detection, most likely due to a larger data set. For Czech tweets the F-score was 58.2% and the English sarcasm detection had a F-score of 0.947.

A 2013 paper by Liebrecht et al. examined at 78000 Dutch tweets with the hashtag "sarcasm"

and 3.3 million that were categorized as not sarcastic [4]. Similar to the current paper, they considered sarcasm to be equivalent to verbal irony and noted there is an added challenge in noting sarcasm in text as it lacks the intonations that come with verbal sarcasm. Another method of detecting sarcasm that was analysed was the use of hyperbole in an otherwise neutral set of text. A balanced Winnow classification was implemented using uni, bi and tri-grams. Their results were favourable with an true positive rate of 0.75 and a area under the curve value of 0.79.

Riloff et al. developed another method for detecting sarcasm that examined the contrast between a positive sentiment that was paired with a traditionally negative situation [2]. The work used tweets tagged with "sarcasm" as a gold standard. The juxtaposition of positive sentiment words with negative situations incorporated the essential idea of world context into detecting sarcasm. The goal was to learn phrases that are implicitly linked with negative sentiment. Their algorithm learns by detecting a positive sentiment word (e.g. "love") as a seed word and find a negative situation that follows the word in the sarcastic tweet. Positive sentiment phrases are then learned by looking at adjacent negative situation phrases. These phrases and situations are then used to detect sarcasm in new tweets. A SVM was used using unigrams and bigrams of the learned phrases resulting in an F-score of 51% (with 23% asea result from guessing).

The final paper that will be discussed relating to sarcasm by Davidov et al. used both Twitter and Amazon as corpora [3]. The work looked at 5.9 million tweets and 66000 product reviews. Their baseline human classification led to a F-score of 0.78 on product reviews from Amazon.com and 0.83 on Twitter. It was attempted to find sarcastic patterns as features for a classifier. The system used a scale that was rated 1-5 for sarcasm (5 being more sarcastic). Templates for sarcasm were used that looked at word frequency and pattern that were applied to new sentences and a similarity measure was applied. Their classification led to results of a F-score of 0.827.

Sentiment Analysis and Time Series Modelling

Review of materials dealing with sarcasm detection have demonstrated that a ubiquitous means of detecting sarcasm is to use either patterns that were associated with sarcasm or polarity shifts in sentiment in or between sentences. Sentiment shifts are the method that is expanded upon in this report. The system demonstrated in this paper expands on work done and adds a novel element of incorporating inter-sentence polarity changes and using these changes as a time series such that context can be remembered and changes in sentiment can be detected as features.

To determine what the sentiment of a sentence is, Socher et al. have created a recursive neu-

ral network architecture that is able to classify negative and positive sentiments in sentences with an accuracy of 85.4% [6]. This was achieved by using the Stanford Sentiment treebank which is comprised of 215154 labelled phrases based on 11855 sentences from movie reviews sourced from "rottentomatoes.com". The phrases were labelled with Amazon Mechanical Turk (supervised labelling of data). The architecture created is called a Recursive Neural Tensor Network (RNTN). The sentiment of a sentence is analysed by parsing the sentence and breaking it into a grammatical hierarchy which is able to successfully integrate negation of words (great versus not great) as well as conditional aspects that alter the meaning of a sentence. It was shown that word order is important in shorter contexts and therefore methods that use the traditional Bag of Words model fail in these areas. The impressive results from this system led to its inclusion in the methodology of this project. Figure 2.1 shows an example of the RNTN performing sentiment analysis. The process is to parse the sentence and determine the relation words have with each other, it then uses the single word sentiment to move up the tree and attain a full sentence sentiment. The details of the full workings of the system are beyond the scope of this project.

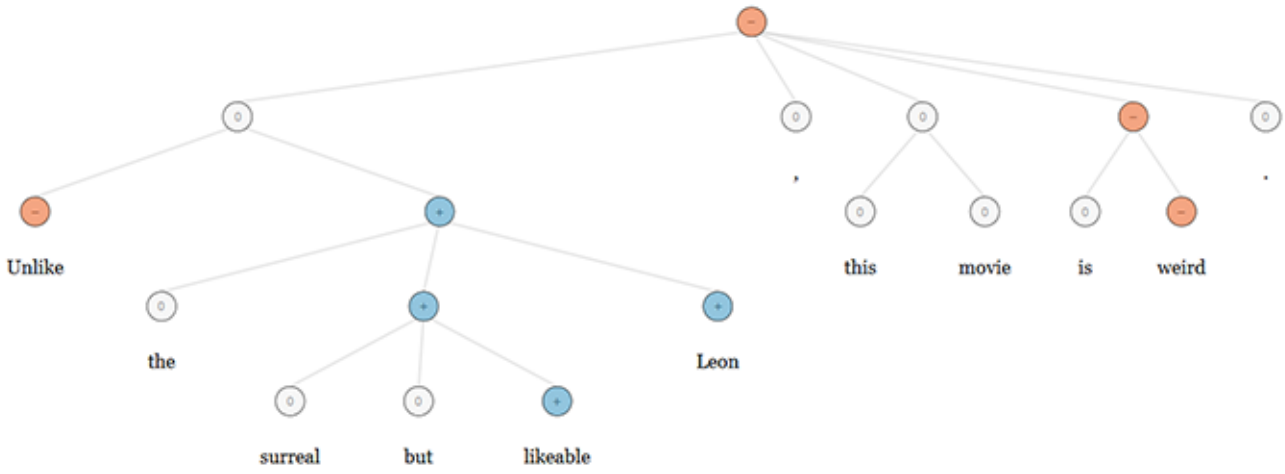

Figure 2.1: Sentiment Analysis of Sentence using RNTN

3 | Methodology and Materials

Hypothesis

The first hypothesis that the paper is intended to examine is whether using inter-sentence sentiment (treated as a feature vector) is a promising tool for sarcasm detection in multi-sentence bodies. The second aspect that will be analysed is whether treating this sentence sentiment vector as time series data is an effective way to discriminate between a sarcastic and non-sarcastic body of text.

Sarcasm Data

The data that was used to train the model is from an open resource known as the "Sarcasm Corpus" [7]. The corpus contains reviews from a multitude of Amazon.com products that are separated into verbally ironic/sarcastic and regular reviews. These reviews were labelled using a two step procedure that is detailed by Filarova [7]. The labelling of sarcasm is not an exact science and the subjectivity of whether the sarcastic reviews are indeed sarcastic should be understood. Regardless, these examples are taken as a gold standard of sarcasm over a paragraph. An example of one of the formatted reviews can be seen in Figure 3.1.

<STARS>5.0</STARS>
<TITLE>These are NOT alive!</TITLE>
<DATE>August 24, 2009</DATE>
<AUTHOR>P. Breakfield IV "Tom Steele"</AUTHOR>
<PRODUCT>Fresh Whole Rabbit (Misc.)</PRODUCT>
<REVIEW>
I'll keep this short and sweet. We ordered one of these rabbits for our children this Easter and boy what a surprise. It is NOT a living rabbit. Someone has killed this rabbit and skinned it, I suppose for eating. Anyway, our children were traumatized and Easter is not the same holiday that it used to be for us. On the upside, we don't have to fill their Easter baskets anymore as we told them the Easter bunny was killed by Amazon.
P.S. The rabbit tasted very good.
</REVIEW>

Figure 3.1: Raw review file from Sarcasm Corpus that was labelled sarcastic

The only data that was to be processed in the experiments was the main text of the review. This data was extracted and used as input for the Stanford Sentiment Analyser [8] which is part of the Stanford Core NLP toolkit. It presents a java based command line program that takes a text file as input and will then output an assigned sentiment to each sentence in the text. The tool gives a sentence one of five sentiments: Very negative, Negative, Neutral, Positive and Very positive. To create a feature vector that could be used as input for a recurrent neural network (RNN) each sentiment was assigned an integer value: Very negative: -2, Negative: -1, Neutral: 0, Positive: 1, Very positive: 2. Each review can then be represented as a variable length vector. Figure 3.2 shows the breakdown and sentiment assignment of the sarcastic review shown in Figure 3.1. The resultant feature vector for this review is $[1, 1, -1, -1, -1, -1, 0, 1]$.

I'll keep this short and sweet.

Positive

We ordered one of these rabbits for our children this Easter and boy what a surprise.

Positive

It is NOT a living rabbit.

Negative

Someone has killed this rabbit and skinned it, I suppose for eating.

Negative

Anyway, our children were traumatized and Easter is not the same holiday that it used to be for us.

Negative

On the upside, we don't have to fill their Easter baskets any more as we told them the Easter bunny was killed by Amazon.

Negative

P.S.

Neutral

The rabbit tasted very good.

Positive

Figure 3.2: Review file text with sentiment tags for each sentence

Figure 3.3 is an example of the change of sentiment in successive sentences in two of the reviews in the Sarcasm Corpus. The sarcastic review (red) is marked by sudden and large changes in sentiment whereas the regular review only changes slightly over the length of the review. This example supports the initial hypothesis that sentiment changes could be used as features that distinguish sarcasm.

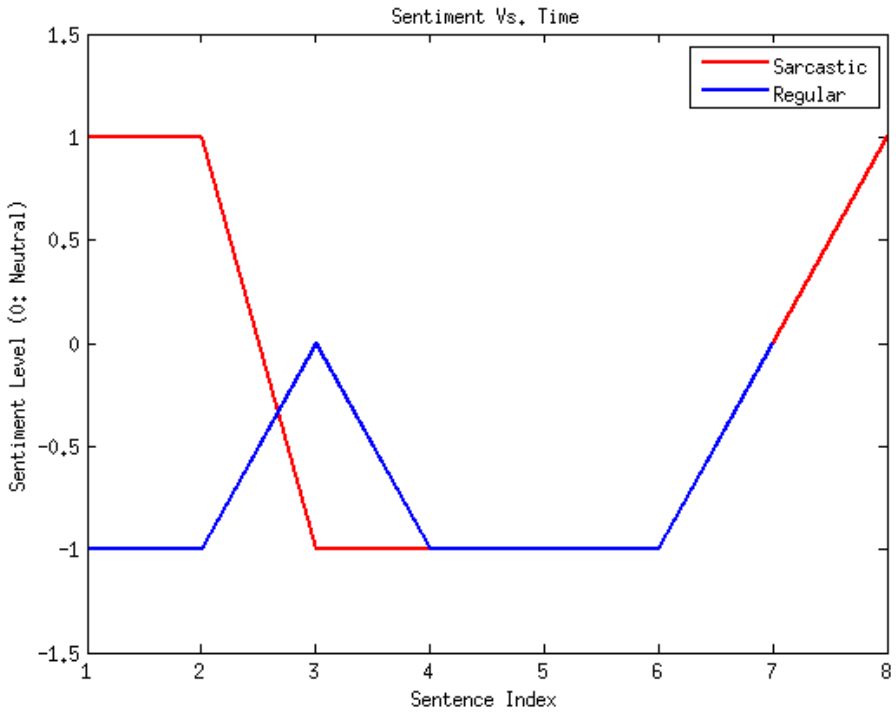

Figure 3.3: Sentiment vector presented as time series (Sentiment versus Sentence number in text

The sentiment vector for each review is paired with a target value, either 1 for sarcastic or 0 for regular, that is repeated for the length of the sentiment vector. The sentiment vectors of all the reviews are concatenated randomly to form a contiguous training vector that has a corresponding target value that is of equal length which indicates whether the input signal is sarcastic or not. In total there are 437 sarcastic reviews and 817 regular reviews. Reviews vary in length from 2 to 156 sentences long. An example the input and target signals can be seen in Figure 3.4. Similar to Figure 3.3 it can be seen that the sentiment for the sarcastic review varies more and contains larger jumps than the regular review.

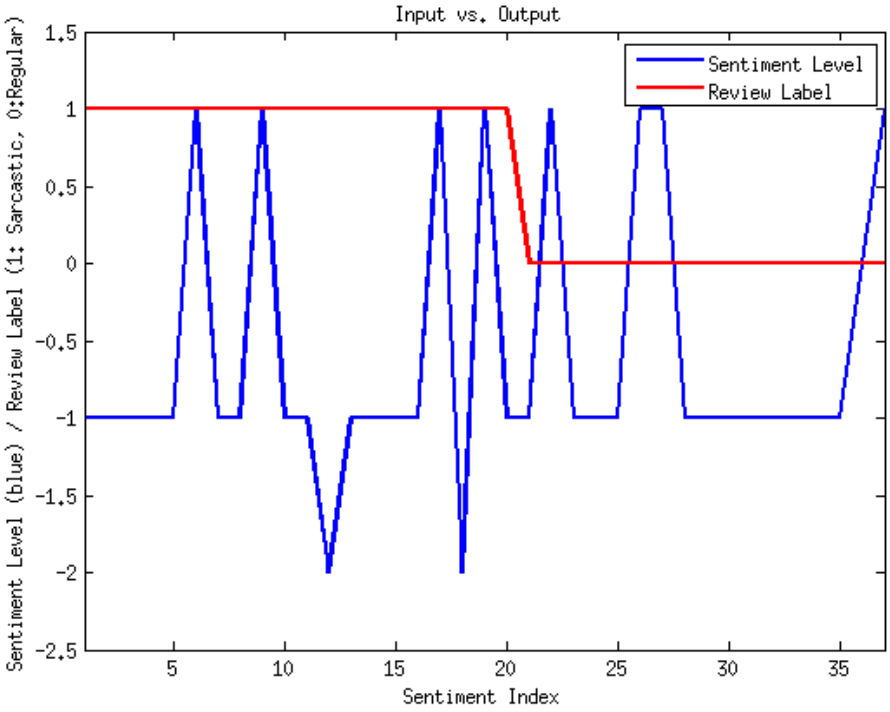

Figure 3.4: Sentiment input versus review label (0 is Regular, 1 is sarcastic

Recurrent Neural Network

With the corpus's review sentiment vectors created it is necessary to categorize them as sarcastic or non-sarcastic. The hypothesis that changing sentiment polarity between sentences is an indication of sarcasm means that the model used to categorize reviews should be time sensitive. This led to the choice of using a simple form of recurrent neural network (RNN) that uses context units that serve as an additional input to the hidden neurons. This model is called an Elman recurrent neural network and it has been used in natural language processing since its conception [9]. The network is set up similarly to a regular feed forward neural network with an input layer, a centre hidden layer of neurons and a output layer, however there is an additional layer that contains so-called context units that store information from previous inputs states and input those activations into the current input activations for the network, therefore the past state has an effect on the current state of the network. The architecture is shown below in Figure 3.5.

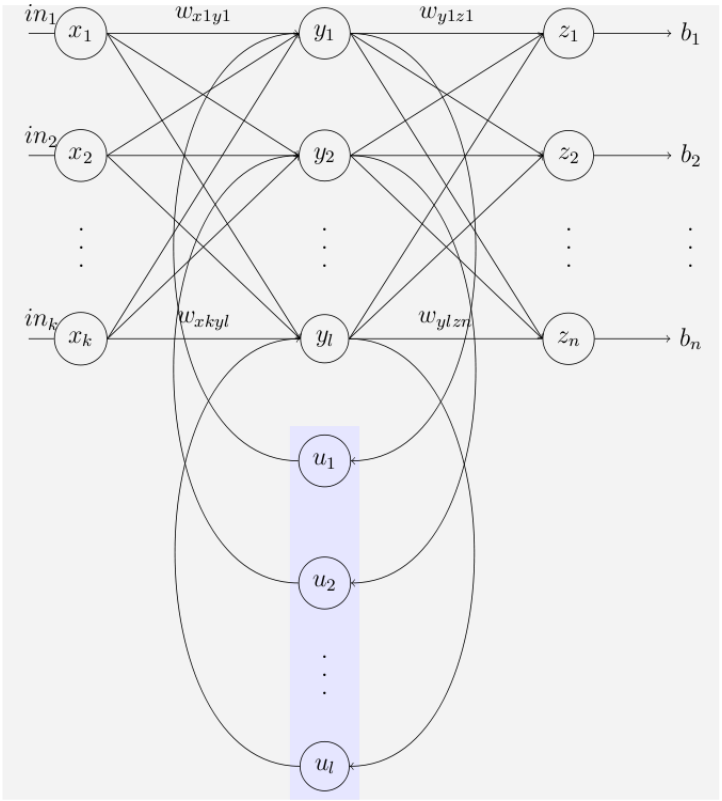

Figure 3.5: Simple example of architecture of Elman RNN [10]

To implement this network, the Matlab neural network toolkit was used. Specifically, a generalized Elamn RNN was used called the "layer recurrent neural network". The number of context layers could be expanded to an arbitrary number. This meant that the hidden layer could have input from an arbitrary number of previous states to obtain an output. The network is trained with the Levenberg - Marquardt algorithm. The recurrent neural network was chosen both for its time sensitivity but also for the fact that it can take an arbitrary length input whereas other neural networks need to have a fixed size input. Since the reviews vary in length and a time sensitive system was desired, this technique was appropriate for the model.

Due to a limitation of resources, the network was limited to four layers of context units as well as 20 hidden neurons. This meant that the current output of the network was dependent on four previous input states of 20 hidden neurons. The same architecture will be trained three times, with the full set of reviews and then with the reviews that are equal to or longer than 10 sentences and the reviews that are shorter than 10 sentences. These experiments serve to verify whether the methodology is more suited for sarcasm detection in larger bodies of text.

4 | Results

Discussion and Analysis

The recurrent neural network (RNN) was trained with 70% of the data set and 30% was used for testing. The best parameters for performance in the network that were able to be achieved with the resources that were available were four context layers and 20 hidden neurons. The network's output was a real valued number between 0 and 1. To get a binary distinction a threshold was applied to the output of the network . Matlab determines performance by taking the binary output signal of the network and comparing it to the target signal and performing a mean squared error calculation. A section of the network response and the target output can be seen in Figure 4.1.

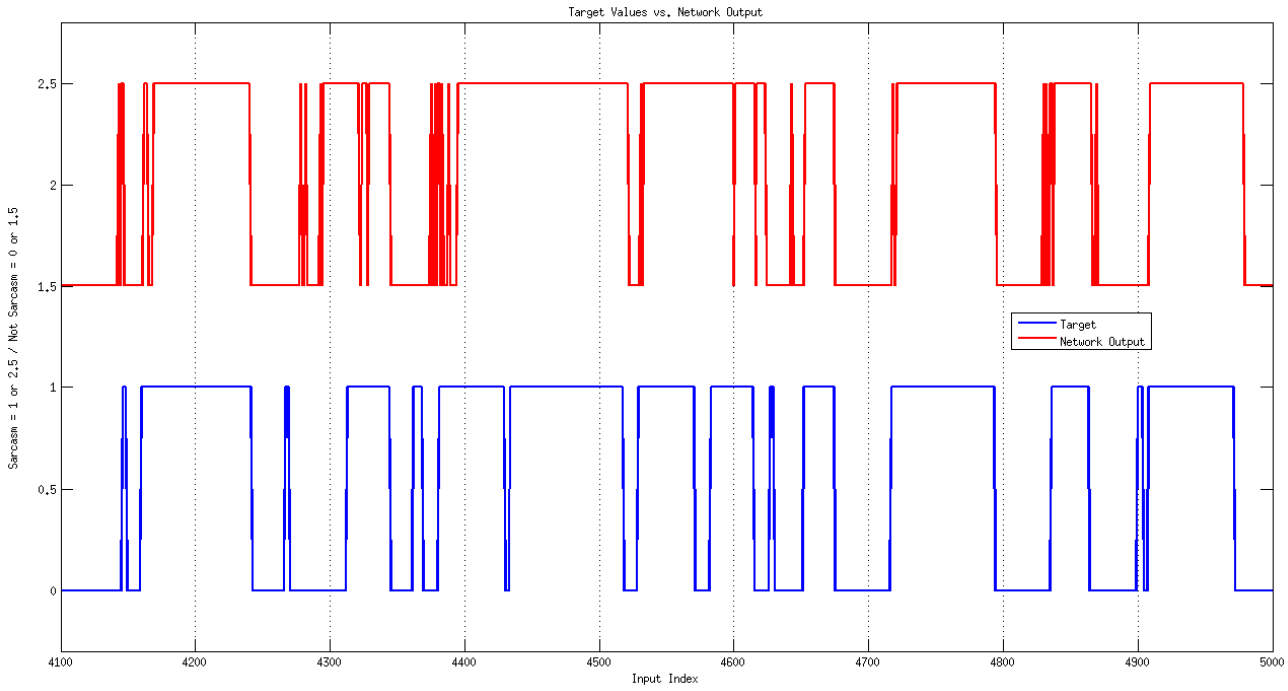

Figure 4.1: Section of full data set trained network output (red) versus target values (blue)

Networks with the same architecture were also trained on a subset of the data that contained reviews that were longer than or equal to 10 sentences and less than 10 sentences. The expectation that the classification would be more successful with the longer reviews was confirmed with a decrease in error rates of almost 4%. A section of the results of the classification of the longer reviews can be seen in Figure 4.2. It can be seen that in some of the shorter sections there appears to be a jumping between classes. These jumps show that the classifier is close to working however there may not have been enough data or a strong enough feature vector to allow for a proper classification. It is, however, an encouraging result.

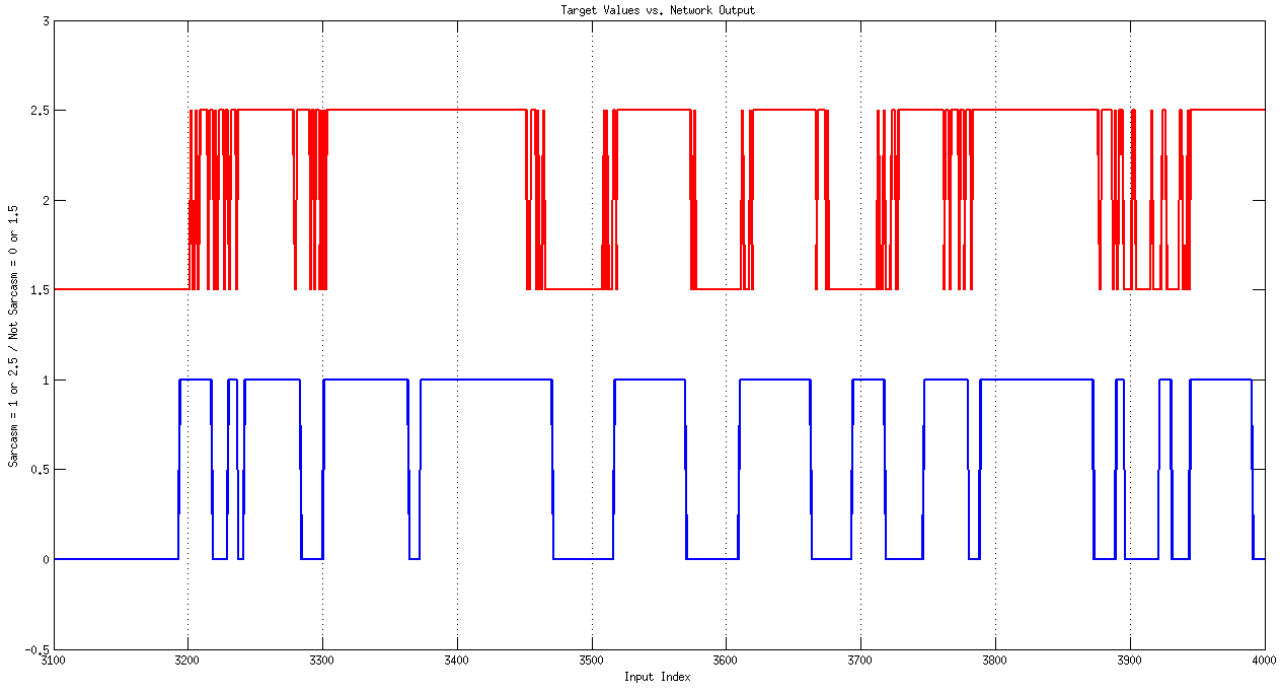

Figure 4.2: Section of long review trained network output (red) versus target values (blue)

It can be noted in Figure 4.1 and less so in Figure 4.2 that the network output appears to have more spikes than the target value. This is present in the entirety of the output of the network. Reasons for the sporadic spikes could be attributed to the fact that the network is dealing with a rather small data set and therefore its prediction rate is less powerful for certain niche circumstances. RNNs also have a response time in some cases and this would result in a slight delay if the input was changed quickly and for a realtively short amount of time (a short review for instance). A prime example of this short coming in Figure 4.1 is near the centre of the plot (just after input index = 4400) where there is an example of regular review that the output misses entirely. There is also a

delayed response to a section after the input index is 4500. It can be seen that the network fares better for longer reviews. The result of performing better with longer reviews is to be expected as this system is designed to look at a longer term context within blocks of text that indicate sarcasm. Figure 4.2 also shows some of the noise that can be explained by the model not being able to make changes quickly enough or potentially having too much "momentum" from previous context. This previous context could overpower the new change in classification since there was a strong signal before it indicating the opposite case. This is a limitation of the RNN model that was implemented and could be solved potentially with a larger network or a different input method.

Figure 4.3 shows the asymptotic decline in the improvement in error rate that occurs after approximately 40 epochs. The halting of improvement held true for the three networks that were trained once the epochs reached over 40 epochs.

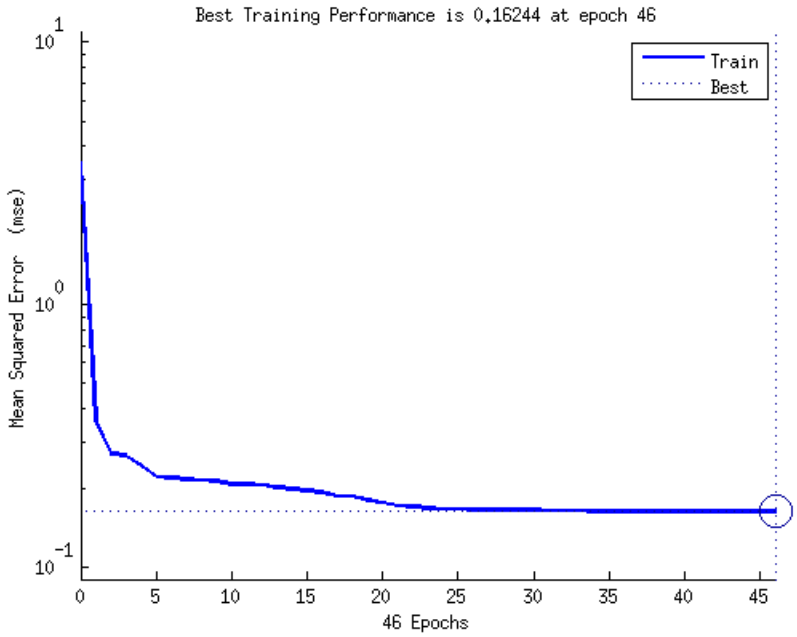

Figure 4.3: Error rate versus training epoch

Table 4.1 shows the error rates of the different networks. The presumption that the classification would fare better with longer reviews is confirmed. It can also be seen that the shorter reviews do slightly worse than using the entirety of the data set. This could be attributed to the fact that there was less data that the network was trained on, however this is contrasted by the fact that the longer reviews also had fewer training examples yet had better performance.

Review Training Set	Error Rate (MSE)
All Reviews	0.2279
Long Reviews (>= 10 sentences)	0.1880
Short Reviews (< 10 sentences)	0.2515

Table 4.1: Error rates of sarcasm classification using RNN trained on variable length reviews and all reviews

Limitations of System

While the idea that the change in sentiment over sentences in a sarcastic paragraph can serve as a promising method for sarcasm detection there are some inherent flaws in the system that was created to implement the technique. The first link in the chain of detection was the Stanford Sentiment Analyser. While the results of this tool are impressive there are still errors that are introduced into the system that the paper presents. An error with the training method is also apparent as the system treated the the reviews as continuous signals that had a desired output signal. This led to the usual issues that are associated with attempting to track a signal, mainly that there are problems with delay responding to the signal. Small changes are also difficult to detect. Due to limitations of the hardware being used to simulate the neural network, there was a practical limit to the amount of neurons and context layers that could be incorporated into the system. A final limitation of any system used to detect sarcasm is that sarcasm and verbal irony are very subjective and even human classification efforts also fail to perform well.

5 | Conclusion and Future Work

The paper that has been presented put forth the hypothesis that changes sentiment polarity (positive to negative) of sentences can be used as a feature for detecting sarcasm within product review length bodies of text. The data used was a labelled corpus of sarcastic and regular product reviews from Amazon.com. The sentiment of the sentences was determined using the Stanford Sentiment Analyser. These features were paired with the time sensitive learning model of a recurrent neural network. Three final models were trained, with all reviews, long reviews and short reviews with error rates of 0.2279, 0.1888 and 0.2515 respectively. The method was proven effective for longer reviews and provides a promising path to follow for more research into sarcasm detection in medium length bodies of text.

Future steps to explore this line of research would include an attempt to access more labelled sarcasm data sets such that the model could be more robust to different cases as the training set that was used provided a relatively small corpus compared to other papers that have explored this subject. A more sophisticated learning model that is time sensitive could be explored. Specifically a Long Short Term Memory (LSTM) [11] network could be used to provide a more robust system that does not suffer from some of the problems that simpler recurrent neural networks suffer from. A Hidden Markov Model could also potentially be used to model and classify the time series information. The presented method could potentially be used to augment existing techniques.

References

[1] Gonzalez-Ibanez, R., Muresan, S., & Wacholder, N. (2011, June). Identifying sarcasm in Twitter: a closer look. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2 (pp. 581-586). Association for Computational Linguistics.

[2] Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N., & Huang, R. (2013). Sarcasm as Contrast between a Positive Sentiment and Negative Situation. In EMNLP (pp. 704-714).

[3] Davidov, D., Tsur, O., & Rappoport, A. (2010, July). Semi-supervised recognition of sarcastic sentences in twitter and amazon. In Proceedings of the Fourteenth Conference on Computational Natural Language Learning (pp. 107-116). Association for Computational Linguistics.

[4] Liebrecht, C. C., Kunneman, F. A., & van den Bosch, A. P. J. (2013). The perfect solution for detecting sarcasm in tweets #not.

[5] Ptacek, T., Habernal, I., & Hong, J. Sarcasm Detection on Czech and English Twitter.

[6] Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013, October). Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the conference on empirical methods in natural language processing (EMNLP) (Vol. 1631, p. 1642).

[7] Filatova, E. (2012, May). Irony and Sarcasm: Corpus Generation and Analysis Using Crowdsourcing. In LREC (pp. 392-398).

[8] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014, June). The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 55-60).

[9] Elman, J. L. (1990). Finding structure in time. Cognitive science, 14(2), 179-211.

[10] "Elman srnn" by Fyedernoggersnodden - Own work. Licensed under CC BY 3.0 via Commons

[11] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. Neural computation, 12(10), 2451-2471.