Sarcastic Text Classification with Deep Neural Networks

Ricson Cheng (ricsonc), Daniel Bork (dbork), Muchen Zhou (muchenz)
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
ricsonc@andrew.cmu.edu, dbork@andrew.cmu.edu, muchenz@andrew.cmu.edu

Abstract

We explore how deep learning methods can be used for automated sarcasm detection from text. In particular, we focused on a comparison between recurrent neural networks, deep feed-forward neural networks, and several baseline methods. We expected recurrent neural networks to be able to effectively extract features from the sequential data. We tested our neural networks on two datasets from the literature, one composed of Amazon reviews [5] and the other derived from Twitter [4], and compared our performance to a benchmarks from the published literature, as well as to human accuracy, and our baseline classifiers. We found that all the baseline classifiers were ineffective, compared to previous work and the deep neural networks that we used, and discovered that recurrent networks performed better than simple feed-forward network on the Amazon dataset but worse than the feed-forward network on Twitter data, and conclude the length of the text to be classified determines the optimal network architecture.

1 Introduction

Our group investigated the use of deep neural networks to assess sarcastic, and non-literal intent in text. This problem is part of the sentiment analysis sub-field of natural language processing, which is concerned with extracting intended emotional content from a given textual passage.

Within sentiment analysis, automated sarcasm detection is a major open problem with potentially wide-ranging applications, because sarcasm can effectively invert meaning. This is most obvious in advertising on platforms, such as Amazon and Yelp, with user-generated product or service reviews, where an inability to detect sarcastic language may cause a customer to be shown related items which they may be disinterested in or even actively opposed to. Similar concerns extend to social media platforms, many of which show users pages of potential interest according to algorithms partially informed by natural language processing of the user's posts.

Automated sarcasm detection is a challenging problem, as it requires identifying scenarios in which the author's intended sentiment is differs from or even is directly antithetical to that usually conveyed by its constituent words. We expect that this might be a challenging problem for many machine learning algorithms like logistic regression, because the sarcasm in a text may not be a linear function of its constituent words, and two words which are not sarcastic in isolation may be sarcastic when put next to each other.

Because of this, we decided that deep learning methods might be relatively well-suited to the problem, an application that our literature review suggested to have not been well-explored in the past. In particular, we focused on deep recurrent neural networks, as their design is especially appropriate for gathering features from sequential inputs, such as text and social media posts.

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2 Related Work

A survey of sarcasm classification has been carried out by Joshi et. al [8]. According to the survey, the majority of classification algorithms were either rule-based, which is based on hand-engineered features, or categorized as deep learning. They say that only a small amount of previous work for using deep learning to detect sarcasm. They mention Silvio et al. who has used convolutional neural networks in order to detect sarcasm [1]. LSTMs have also been used on top of convolutional neural networks with success by Ghosh and Vale.

In Tang et. al. [14], the authors used deep learning framework by concatenating the sentiment-specific word embedding (SSWE) features with the state-of-the-art hand-crafted features, a technique which can be applied to classifying sarcasm.

One example of the systematic generation of a corpus to be used for sarcasm detection was provided by [13], who derived such a corpus from Internet comments and then applied supervised learning to the resulting dataset.

Previous machine learning approaches to sarcasm detection have been shown to be capable of performance that is similar to human performance, as demonstrated in a paper by Gonzalez [7]. Such results foster optimism that a deep learning-centric approach may also match or surpass human classification. In a similar vein, we also hope to compare machine-generated features with those manually identified by humans to be associated with sarcasm, similar to the approach taken by [15].

3 Methods

3.1 Dataset

We used two sources of data. The first dataset was 1,280 Amazon reviews from Filatova [5], of which 35% were sarcastic. The second dataset was 478,000 tweets from Cliche [4], of which 30.5% were sarcastic. Having two different datasets allowed us to investigate the advantages and disadvantages of different machine learning techniques in different settings. The Amazon review dataset had much fewer individual samples, although each sample was composed of more words. On the other hand, there were many more tweets, but each was much shorter, which may make it harder to determine sarcasm.

We compared the performance of our classifiers on these two datasets in order to see how different neural network architectures would perform on different datasets. We expected performance of any network architecture on this problem to increase when given more data, and, because the reviews were much longer on average than the tweets, we expected recurrent neural networks to perform better relative to other architectures on the Amazon dataset than on the Twitter dataset.

3.2 Preprocessing

There are multiple ways of extracting features given labeled sentences. In our case, since we are more interested in determining the sequential relationship between words in a sentence, we will construct our features based on individual words(including punctuation marks) rather than the whole sentence. In order to transform sentence into the input matrix, we first parse the sentence into words and punctuation marks and then used GloVe [12] to find each word's and punctuation's vector representation. GloVe is an unsupervised learning algorithm that learns a vector representation of word. The model performs dimensionality reduction on the co-occurrence counts matrix. In general, this is done by minimizing a "reconstruction loss" which tries to find the lower-dimensional representations which can explain most of the variance in the high-dimensional data.

We have tried to train our own model by using tweets that contains #sarcasm. However, due to twitter's API limit, we couldn't pull enough data to train a proper GloVe model with enough vocabulary. Therefore, we used the pre-trained 25-dimensional GloVe model which is trained on 2 billions tweets with a vocabulary size of 1.2 million.

On top of those 25 dimensions, we add two more. One additional dimension was used to denote whether the input word has appeared in GloVe model, and another dimension is used to mark a column in our sample as padding or non-padding. In both cases, all other 25 dimensions of the vector are set to 0.

For the Amazon reviews, we padded or trimmed all samples to 300 tokens long, and for the Twitter data, we padded to 30 tokens long. These thresholds were chosen to be greater than the length of 95% of the data points, ensuring that we did not lose too much information. Thus, each input to our classifier was a 28 by 300 matrix for Amazon reviews, or 28 by 30 matrix for the tweets.

3.3 Classification

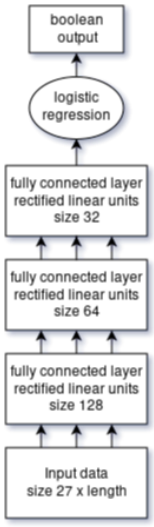

Figure 1: Simple Deep Feed-Forward Network

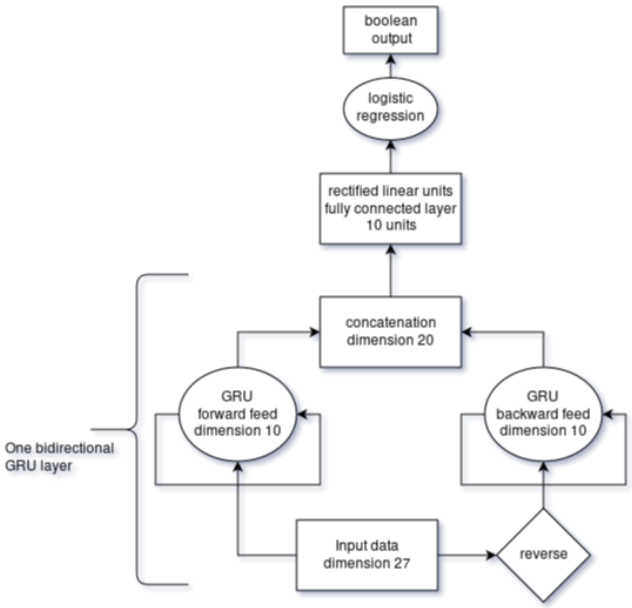

Figure 2: One Layer GRU Network

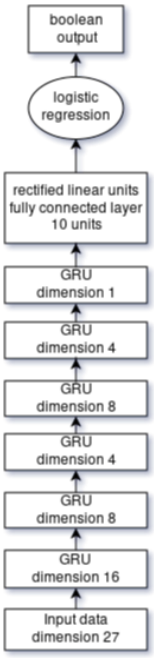

Figure 3: Deep GRU Network

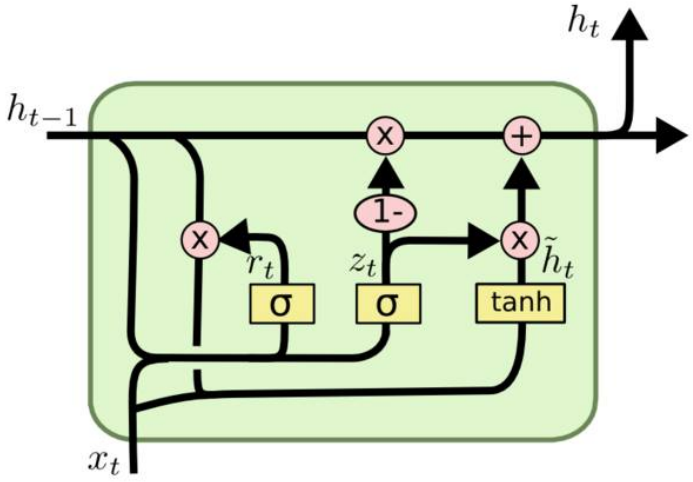

Figure 4: Internal structure of a Gated Recurrent Unit, from [11]

We tried three separate neural network architectures for classification. The first network we used was a simple, densely connected feed-forward network with three hidden layers and rectified linear units, depicted in figure 1. We chose to use rectified linear activation units in order to avoid the vanishing or exploding gradient problem. The exact number of neurons in each layer was found not to have a large effect on the effectiveness of the network, as long as the number of parameters did not grow

large enough to cause problems with overfitting. Because this network had the highest number of parameters (about 1.1 million for the Amazon data and 120000 for the Twitter data), we had to use dropout regularization in order to prevent overfitting [10] .

Because text data is sequential, we thought that a recurrent type neural network might be able to take advantage of patterns in the text data. Therefore, we used a recurrent neural network architecture with Gated Recurrent Units. In the network, GRUs take the place of simple activation functions, and has an additional gates to control the flow of information in and out of the unit. We chose to use GRUs over LSTM networks, another popular architecture which accomplishes the same task, because the performance of GRUs and LSTMs are comparable but GRUs require slightly less computation [3]. Our network is constructed to be bidirectional, so that information is fed in to two separate GRU layers, once in the forward direction, once in the reverse direction. Then, the output of the two layers is concatenated together. Finally, everything is flattened and fed into densely connected layer of 10 units, and then logistic regression is applied. This network is depicted in figure 2.

Our third network utilizes many of the bidirectional layers of gated recurrent units used in the second network, stacked on top of each other. We found that the dimension of each layer did not significantly affect the performance of the network. However, we hoped that by increasing the depth of the network, we would be able to learn higher level features and concepts from the data, just as layers in a convolutional neural network which are higher up tend to learn higher level features. Finally, we fed the output of the network into a logistic regression. This network is shown in figure 3.

In order to train our networks, we used glorot uniform weight initialization as described in a paper by Glorot and Bengio [6]. We trained using the Adam algorithm, which is a extension of the basic stochastic gradient descent [9]. We used mini-batch sizes of 1000 samples. The code for our classifiers is implemented in the Keras machine learning library [2], and is available on GitHub, and can be found at `https://github.com/ricsoncheng/sarcasm_machine`.

4 Results

4.1 Evaluation Procedure

We trained all the three neural networks until convergence. In addition, we also ran some baseline classifiers like logistic regression, support vector machines, and k-nearest neighbors, optimizing the hyper-parameters for SVM and KNN with cross-validation. Because SVM was extremely computationally expensive, we ran SVM on 10% of the Twitter data instead of all of the data.

We divided both the Amazon and the Twitter datasets into a train set, validation set, and test set, using the validation set to select hyper-parameters of the models and to experiment with different neural network architectures, and using the test set only to obtain our final results. For the Amazon data, 1000 of the samples were train, 100 were validation, and 178 were used for test. For the Twitter data, 400000 of the samples were train, 40000 were validation, and about 37000 were used for test.

Because our datasets are unbalanced, with the majority of the labels being negative, we chose to report F-scores rather than accuracy. The F-score is computed as the harmonic mean of the precision and recall of a classifier. This makes it a good measure for unbalanced datasets where the majority of labels are negative because a classifier which classified every label as negative might have a good accuracy, but 0 recall and precision. In addition, F-score is the most popular metric reported by previous work, which makes comparing results easier.

4.2 Comparison with baseline and prior work

	Amazon Reviews F-Score	Tweets F-Score
Logistic Regression	0.33	0.42
SVM	0.0	0.33
KNN	0.30	0.22
Human	0.62	0.69
Previous Results	0.71 to 0.89	0.51 to 0.88
Feed-Forwad Netowrk	0.40	0.86
One Layer GRU Network	0.59	0.74
Deep GRU Network	0.54	0.81

For Amazon reviews, we see that logistic regression, SVM, KNN, and the feed-forward network are ineffective. In particular, the KNN achieves a 0 F-score by labeling all samples as negative. However, both the one layer and deep GRU network give significantly better results. While none of our classifiers does as well as the previous results, the results are not directly comparable because previous authors used larger datasets with over 10,000 reviews while we were only able to train with 1,000 reviews. The small amount of data we had hindered the training of our deep neural networks, which usually need large amounts of data.

For tweets, logistic regression, SVM, and KNN are inadequate as before. Interestingly, we find that the feed-forward network is very effective, and nearly matches the best previous result for classifying tweets. Both the one layer GRU and the deep GRU network that we used performed well, and better than human, but not as well as the feed-forward network.

We compare our work to a similar study by Silvio et. al., who used a convolutional neural network, and did not report an F-score, but reported an accuracy of 87% [1], which is comparable to the accuracy of the one layer GRU network we used (85%) and the deep GRU network (89%), but not as good as the feed-forward network.

5 Discussion and Analysis

We hypothesize that the performance of various architectures of neural networks depends on the length of the text data. For Amazon reviews, the recurrent nature of the one layer and the deep GRU networks enabled the learning of long term dependencies in the text. On the other hand, the average length of the Twitter data was about 10 times shorter, which meant there were not many long term dependencies in the data, so the recurrent nature of the networks did not help. By contrast, the feed-forward network might have struggled with the Amazon dataset because the large number of inputs caused there to be over a million parameters, making training difficult. On the twitter dataset, the feed-forward network did much better because there were far fewer parameters, allowing for effective training without overfitting.

5.1 Model Limitations

In a few potentially important ways, our model was unable to fully take into account all of the information presented in the datasets. For instance, the Amazon dataset from [5] also contained star ratings assigned by each reviewer, which could could also be intended non-literally or used to participate in features associated with sarcasm. However, our method as implemented did not account for these. One approach to fix this would be to include the rating as additional categorial variable encoded as a one-hot vector.

Our method as currently implemented also ignores punctuation and misspellings and, in the Twitter case, treats hashtag in a fairly naive manner, by preprocessing them into the word '#' followed by another word containing the content of the hashtag. All of these features could be potentially significant, and misspellings could be addressed by integrating a spell checker with word replacement into our preprocessing protocol.

Our model is also not currently equipped to perform any statistical analysis of our results, including both the significance of our results in isolation and the significance of any difference between network architectures. In a similar vein, because we only tested one architecture that contained recurrent units, it is difficult to justify sweeping claims about the suitability of recurrent neural networks in general for this task relative to other architectures. This could be addressed by simply testing more architectures, possibly while keeping the structure, other than the recurrent units, constant across tests.

6 Conclusion

Joshi et. al. previously mentioned that deep learning approaches to sarcasm classification were promising because the area was not well studied. In this paper, we demonstrate that deep neural networks are capable of achieving competitive performance even without the help of hand-engineered features.

In addition, we discovered that the appropriate architecture for sarcasm detection and potentially many other tasks in sentiment analysis may be dependent on the length of text to be handled. For longer pieces of text, a recurrent architecture is better at classification because it has fewer parameters and is able to take advantage of the sequential structure of natural language. On the other hand, when dealing with small pieces of text such as tweets, a simple feed-forward network is often sufficient, and performs even better, because there is a lack of a need to remember long term dependencies.

6.1 Future Work

We have also considered using bi-gram and tri-gram vectorization features instead of uni-gram vectorization features. By adding two or three neighboring words' vector together, we will get a vector that represents similar semantic meaning as the bi-gram or the tri-gram. This may be useful because features of sarcasm may be present in a bi-gram or tri-gram form. Furthermore, it may be interesting to see if deep neural networks perform better when the input data is augmented with handpicked word and sentence level features.

While the datasets we studied provided a contrast between sample size and data length that potentially illuminates the functioning of recurrent neural networks, our classification of the Amazon dataset was hindered by the small amount of data available to us. A larger dataset comprising of long pieces of text would make our results more rigorous, and allow our deep neural networks to achieve f-scores on par with previous papers.

Our analysis did not focus on the interpretation of the actual features learned by our architectures; considering these explicitly may prove interesting, and may further inform our understanding of the relative utility of recurrent neural networks as applied to this problem. A future direction might be to compare frequently used handpicked features for sarcasm classification with those found in the deep neural networks.

References

[1] Silvio Amir, Byron C. Wallace, Hao Lyu, Paula Carvalho, and Mário J. Silva. Modelling context with user embeddings for sarcasm detection in social media. *CoRR*, abs/1607.00976, 2016.

[2] François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[3] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[4] Mathieu Cliche. The sarcasm detector. `https://github.com/MathieuCliche/Sarcasm_detector`, 2014.

[5] Elena Filatova. Irony and sarcasm: Corpus generation and analysis using crowdsourcing. In *LREC*, pages 392–398, 2012.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks.

[7] Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 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*, pages 581–586. Association for Computational Linguistics, 2011.

[8] Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. Automatic sarcasm detection: A survey. *arXiv preprint arXiv:1602.03426*, 2016.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[11] Christopher Olah. Understanding lstm networks.

[12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[13] Oraby Shereen, Vrindavan Harrison, Hernandez Ernesto Reed, Lena and, Ellen Riloff, and Marilyn Walker. Creating and characterizing a diverse corpus of sarcasm in dialogue. In *SIGDial*, 2016.

[14] Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 208–212, Dublin, Ireland, August 2014. Association for Computational Linguistics and Dublin City University.

[15] Oren Tsur, Dmitry Davidov, and Ari Rappoport. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *ICWSM*, 2010.