Detection Of Sarcasm In Online Comments

Tanmay Khokle*, Qiye Chen*, and Samuel Engida*

Khoury College of Computer and Information Sciences, Northeastern University

***Abstract*—Sarcasm is wired into a lot of human conversations. A sarcastic remark may seem to indicate a positive outlook, while implying a negative sentiment. It is much easier to discern when a person is being sarcastic when one can see their expressions or hear their tone. With social media becoming an increasingly popular and preferred means of communication, most conversation are text based. This has made it difficult to detect sarcasm even for humans. This paper aims to develop machine learning models that detect sarcasm using the Self-Annotated Reddit Corpus (SARC) dataset. [1] Sarcasm is largely affected by context, and a solution is implemented to experiment with models that do not consider text context such as logistic regression and a feed forward deep neural network, and models that incorporate some contextual information including a model based on LSTMs [2]. For evaluating the results on this classification task, F-1 score was used as a metric.**

1. Introduction

Sarcasm is almost everywhere in daily conversations. Sometimes, sarcasm can make the atmosphere enjoyable and lighten the mood, and at other times can provoke people to become hostile towards each other. Often, sarcasm is not easy to spot, because at first glance, it looks like a friendly comment. There are no clear rules or formulas that determine whether a comment is sarcastic. Additionally, the sequence of the words themselves may change the implication entirely. For example, the statement "Great! More work is just what I needed today" appears to be sarcastic, while the statement "Great! What I just needed today is more work" appears to be more of a regular remark.

Nowadays people use social media a lot to enjoy interacting with each other, and most of the platforms are already able to detect explicitly contents like bullying, sexual, violent, etc. However, a sarcastic comment may look like a positive statement, while actually being insulting, offensive or derogatory. This could potentially provoke audiences and ruin experiences. Additionally, sarcasm may change the sentiment of a text. This is critical in applications such as market analysis and recommending to users based on sentiment analysis to have a reliable detection of the user's emotion. What a network may think is a positive outlook could actually be a completely negative critique instead and skew the results of the model with an inaccurate prediction.

In this project, we propose to build models to detect sarcasm in online conversations that are entirely text based. Specifically, we explore a machine learning approach to determine sarcasm, which is to learn examples of sarcastic and non-sarcastic comments and be able to predict whether an unseen comment is sarcastic in the future. Moreover, as sarcasm is highly affected by context it was also our aim to compare and contrast the use of machine learning models that don't incorporate context (logistic regression and Feed Forward Neural Networks) and those that do (LSTMs). In short, build a model that can relatively do very well in classifying sarcastic and non sarcastic texts.

The code for our project can be found at https://github.com/TanmayPK/CS6140-Project

2. Related Work

Some previous work had been done for detecting sarcasm. Both Kolchinski [3] and Hazarika [4] focused on embedding the authors and the forums in addition to texts themselves to form a complex relationships that could infer sarcasm, but they achieved it by various methods: RNN and CNN networks respectively. Although the results were somehow fancy, they suffered from an unavoidable problem that the author or the forum could be new to the models at test time, making the models hard to predict, especially if the models rely heavily on the authors and the forums.

Our project is largely inspired by the Stanford CS224N custom project Context-Based Models for Sarcasm Detection by Nicholas Benavides and Angelo Ramos. Whether a sentence is sarcastic depends on the context, such as the person who wrote the sentence, the other sentence this sentence replies to, the forum where the sentence appears, and so on. However, given recent advancements in NLP technologies to extract more information from texts, the authors assumed that many other sarcasm detection work did not utilize well enough the context from sentences, so switching to sequential models like LSTM or GRU could be more beneficial. In many instances, a sentence would not look like sarcastic until the reader reads the very end of it, and also in many times, the order of the same words could make a sentence sarcastic or not. By incorporating with such sequential models, the detector could become aware of such dependencies and improve accuracy. Additionally, previous works have generally focused on only the reply comments. We propose that considering the parent comment in sequence along with the reply is critical in getting the necessary context. Even though a single statement may be sarcastic by itself, a sarcastic comment is often in reply or retort to another statement.

3. Technical Approach

The goal of this project is simple: given a sentence, along with other useful context, as feature vectors, predict a binary value that indicates whether that sentence is sarcastic. However, the feature vectors are not in the same format for all the models, because the baseline, non-sequential ones expect a fixed-size sentence vector for any sentence, whereas sequential ones expect a list of word vectors for a sentence, which could be of different length across sentences.

3.1. Dataset

We use the Self-Annotated Reddit Corpus (SARC) [1] dataset to train models. The complete version contains around 1.3 million labeled samples, which each of them has the comment itself as text, name of author, name of subreddit, its parent comment (if any, the comment being replied by current comment), a binary label where 0 indicates not sarcastic and 1 indicates sarcastic, and other fields such as post dates. Although the entire distribution of the dataset is skewed, where most of the comments are not sarcastic, the authors of the dataset provided a subset of SARC that the labels are completely balanced, which contains around 1 million samples in total (505k samples for each label). For realistic reasons such as computing power and time limits as well as the difficulty of handling skewed data, we simply use the subset version. Based on the discussion in the related work section, we only use the comment and its parent comment.

3.2. Data Preprocessing

The preprocessing has a common stage and two other independent stages for non-sequential models and sequential models respectively.

The common stage converts all comments, including parent comments, into a list of tokens, and prepare the Glove [5] feature vectors for the tokens. First, we expand contractions so that words like "can't" and "wouldn't" will be converted to "can not" and "would not" respectively. Second, we lemmatize, or stem, each word in all comments, so that any word that is not in the original tense or in plural form will become its original form. Till now the tokenization is complete. In many NLP tasks, removing stop words is almost a standard, but since sometimes it is the "the" or the "a" or other similar word that makes a comment sarcastic, we decided not to remove stop words. For Glove feature vectors, we use a pre-trained version from official sites.

The stage for non-sequential models is to compute a sentence vector for each comment, which is already tokenized. First we use the portion of the dataset for training to learn TF-IDF parameters, and then compute TF-IDF values for each token in each comment. Second, for each comment, multiply each token's Glove feature vector by that token's TF-IDF value, and average the result for all tokens to serve as the sentence vector.

The stage for sequential models is to compute a sequence of Glove feature vectors for each comment, where the order of the feature vectors correspond to the original order of that comment's words. This is done by simply mapping each token in the token list to its corresponding feature vector. Since each Glove feature vector has at least few hundreds of dimensions and there are hundreds of thousands of comments including the parent comments, simply converting the entire dataset into sequences of Glove feature vectors is not feasible, so we instead do the conversion on-the-fly when taking mini-batches to train models.

3.3. Logistic Regression

Since the problem is essentially a binary classification, it is natural to use logistic regression, the simplest classification model, as our baseline. The model expects a vector that is a concatenation of two sentence vectors, one for the original comment and the other for the parent comment, as the input, and outputs a vector of two dimensions, each of which indicates the probability the original comment being sarcastic or not.

3.4. Feed Forward Network

A feed forward network can be thought of as an extension of a logistic regression model, where the input vector to the logistic unit, which originally was a concatenation of two sentence vectors, will be encoded by some non-linear function first to produce another feature vector, and then be fed to the logistic unit. Since additional non-linearity is applied to the input, feed forward network is expected to be able to capture more information from the comments and hence be able to make slightly more accurate predictions.

3.5. Long Short-Term Memory

Both logistic regression and feed forward network are not sequential models. Since the comments are by themselves sequences of words, these models are likely to perform less accurate than sequential models. LSTM [2], a variant of recurrent feed forward network, is one kind of sequential model that is expected to fit better into the problem, as it can naturally work on sequential data and has potential to consider long dependencies within a sequence. While the output of LSTM is still a 2D vector having the same meaning, the input is now a sequence formed by concatenating two sequences of Glove feature vectors, representing the parent comment and the original comment respectively.

There are some variants within LSTM. It can be unidirectional or bi-directional, as well as optionally having attention mechanism. In this project, we consider bi-directional one, with and without attention, because usually the characteristics making a sentence sarcastic or not relate to the ordering of words.

4. Experimental Results

The general progression of the project's experimental course of action included: Data Preprocessing, Logistic Regression, construction of a Feed Forward Neural Network and LSTM. For each machine learning approach, testing for goodness of the model was carried out using the F-1 score. For the neural networks, training the model consisted of hyperparameter tuning based on F-1 score performance on the validation set.

4.1. Data Processing

Before training, we split the dataset into training set, validation set, and test set by 60 / 20 / 20 ratios. After tokenize all sets of data, we use the training set to learn TF-IDF parameters. For the Glove feature vectors, we simply download a pre-trained version "glove-wiki-gigaword-300" [5], which almost every token learned from the SARC dataset has its unique feature vector of 300 dimensions. Since a sentence vector is just a TF-IDF weighted average of tokens' Glove feature vectors, a sentence vector also has 300 dimensions as well. For logistic regression and feed forward network, after stacking two sentence vectors representing the original comment and parent comment, they will receive a 600-dimensional vector as input. For LSTMs, each comment will be represented as a sequence of 300-dimensional Glove vectors, and since each comment has different length, each sequence has different lengths as well. The input to LSTMs would be a concatenation of the parent comment and the original comment sequences. However, since all sequences within a mini-batch must have the same number of time steps, we pad any shorter sequence by zeros to the maximum length within that mini-batch. For sequences longer than the sequence length, we truncate the sequence to the first tokens up till the sequence length.

4.2. Logistic Regression

Our Logistic Regression classifier is a simple classifier regularized with l2 penalty. The max iterations were set at 500. As we can see from Table 1, We got an F-1 score of about 0.60, which is respectable for a model which does not consider any context at all, other than the natural location of the feature vectors of parent and reply comments. This classifier gave us a good baseline performance to compare the other more involved models.

4.3. Feed Forward Neural Network

For the feedforward network, we had 2 hidden layers with a ReLu activation. The number of neurons in the hidden layer was found to work best at 256. For training the parameters, the Adam optimizer [6] was used, and CrossEntropyLoss was used to calculate the training loss. Through experimentation, we determined a learning rate of 0.001 was best for training the network. The batch size was set at 128. The network started to overfit the training set in about 3 epochs, as we can see from Figure 1. The best model was saved and loaded for the inference on the test data. The F-1 Score for the feedforward network came to about 0.66, as we can see from Table 1.

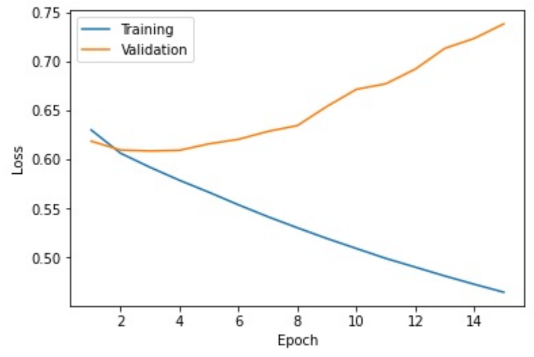

Figure 1. Training and Validation Losses: Feedforward

4.4. LSTM

For the contextual models based on LSTM [2], we implemented two models.

4.4.1. Bi-LSTMs. The first model comprised of two Bi-directional LSTMs stacked one after the other. After each layer, there was a dropout with a probability of 0.2. The input of the LSTM model was encoded with the embnedding layer, which loaded the same pre-trained GloVe embeddings. The output from the Bi-LSTMs was concatenated by connecting the final hidden states from the bi-lstm layers. This combined hidden state was then fed as an input to a linear layer with a single output. The output was then fed through a sigmoid activation. Predictions over the sigmoid value were made by using the default threshold of 0.5. Through experimentation, learning rate of 0.001 seemed to perform well. We used the Adam [6] optimizer and Binary Cross Entropy Loss for this network as well. The hidden units were set at 128. The model started overfitting the training set after about 5 epochs, so the best model was saved and used for inference. The validation and training losses can be seen in Figure 2. As we can see from Table 1, we got a significantly better F-1 score since LSTMs consider the context of sentences. The F-1 Score was about 0.73 in this case, which is in line with similar models for detecting sarcasm in text, and in fact performs marginally better than some models which only used a single directional LSTM without parent context [4].

4.4.2. Bi-LSTMs with Attention. The second model based on bi-LSTMs was essentially mostly the same. The major difference here was that we used a simple multiplicative

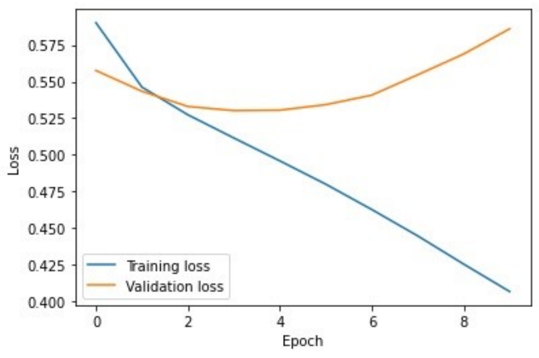

Figure 2. Training and Validation Losses: Bi-LSTM

attention layer to get weights for the sequences [7]. The hidden states were first fed into a linear layer with a single neuron. The output was then passed through a TanH activation layer, and the output was given a Softmax distribution. The original final hidden states were then weighted by these attention weights before feeding into the same linear layer as the plain bi-LSTM model in the previous subsection. The advantage of attention is theoretically that it tells the network which words may impact the classification more. In one sense, it basically assumes some words are more impactful to determining if a statement is sarcastic, and the simple weights train this very information automatically. With the same settings as the Bi-LSTM model, We noticed it achieved a marginally lower validation loss, as can be seen from Figure 3. The F-1 score was the same at 0.73, but it had significantly better recall for class 1, with 0.71 versus 0.67 for the Bi-LSTM without attention. The precision and recall scores were more consistent in a sense, even though the F-1 score in average was the same.

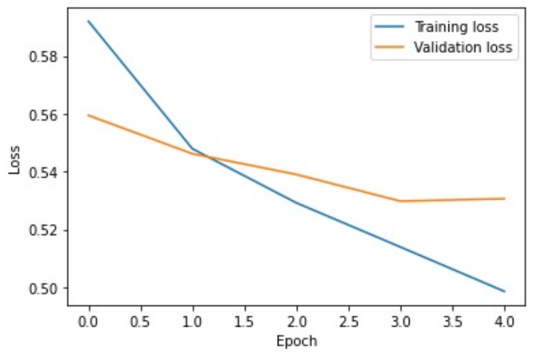

Figure 3. Training and Validation Losses: Bi-LSTM with Attention

5. Conclusion

As the experiments show, feed forward network performs better than logistic regression, and bi-directional LSTM performs even better. This provides an evidence that sequential models like LSTM can indeed fit more naturally with sequential data than non-sequential models. Nevertheless, the results of this experiment is far from perfect, because the accuracy of best performing model is only around 0.73, which means on average, one of the four comments will be mis-classified. To achieve production-grade results, some creative work needs to be done. However, it is certainly a respectable score considering how challengint it is to identify something that can be innately contradictory.

We have only tried a small set of hyperparameters for each type of model, which means there could be a lot of room for potential improvement by simply exploring different sets of hyperparameters. Besides, there are different sequential architectures we could have tried, if time permits.

Last but not least, this project serves as a foundation for future investigation about better sarcasm detection, as well as other relevant NLP tasks or even other tasks involving sequential data. Current future plan is to explore more advanced and pre-trained sequential models, which can be fine-tuned to perform transfer learning, so that better representations of data can be obtained to perform relevant tasks.

Model		**Label 0**			**Label 1**		**Accuracy**
	Precision	*Recall*	*F1-Score*	*Precision*	*Recall*	*F1-Score*	
Logistic Regression	0.56	0.60	0.58	0.64	0.59	0.60	0.60
Feedforward	0.67	0.64	0.65	0.65	0.69	0.67	0.66
Bi-LSTM	0.71	0.79	0.75	0.76	0.67	0.71	0.73
B-LSTM w/ attention	0.72	0.76	0.74	0.75	0.71	0.72	0.73

TABLE 1. F-1 Score Results

References

[1] M. Khodak, N. Saunshi, and K. Vodrahalli, “A large self-annotated corpus for sarcasm,” 2017. [Online]. Available: https://arxiv.org/abs/1704.05579

[2] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, nov 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[3] Y. A. Kolchinski and C. Potts, “Representing social media users for sarcasm detection,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 1115–1121. [Online]. Available: https://aclanthology.org/D18-1140

[4] D. Hazarika, S. Poria, S. Gorantla, E. Cambria, R. Zimmermann, and R. Mihalcea, “CASCADE: Contextual sarcasm detection in online discussion forums,” in *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1837–1848. [Online]. Available: https://aclanthology.org/C18-1156

[5] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[6] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[7] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical attention networks for document classification,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 1480–1489. [Online]. Available: https://aclanthology.org/N16-1174