Exploring Machine Learning Models for Sarcasm Detection in Online Discussion Forums

Yulu Pan
ypan1@unc.edu

Ryan Brown
brownr46@unc.edu

Yutong Wang
ayu047@unc.edu

Kevin Zhang
zhangk23@unc.edu

***Abstract*—Sarcasm poses significant challenges for machine understanding and systematic sentiment analysis, and the ability to detect it is crucial for interpreting human language in both spoken and written contexts. This work explores the problem of sarcasm detection as a classification problem using various Machine Learning models. The goal is to find the best model that can be trained with easily accessible resources, such as public datasets and Google Colab. The models were assessed based on their accuracy and source usage, and popular machine learning classification algorithms were surveyed and tested. The findings of this study could contribute to the development of more robust and accurate sarcasm detection models, which can improve our ability to interpret human language in a variety of contexts.**

I. Introduction

Sarcasm is a unique form of language that has gained widespread popularity on social media platforms such as Twitter and Reddit. It allows people to express their contempt for certain events and individuals in a subtle yet powerful way. However, the figurative nature of this type of expression poses significant challenges for machine understanding and systematic sentiment analysis, especially since written language lacks spoken context clues.

Despite the challenges posed by sarcasm, understanding it is an integral component of interpreting human language, both in spoken and written contexts. It can pose a threat to national security if a sarcastic comment is misinterpreted under a government announcement, and the United States government has previously solicited a system to detect sarcasm on Twitter (Pauli, 2014). Furthermore, the ability to detect sarcasm is important for large language models like those that power ChatGPT to be able to identify users' sarcasm during conversation to improve responses and overall performance.

Given the importance of sarcasm detection, we explore the problem of sarcasm detection in online discussion forums with various machine learning models. In this work, we surveyed and tested popular machine learning classification algorithms and assessed the models based on their accuracy and source usage. We aim to find the best model which can be trained with easily accessible resources, such as public datasets and Google Colab. By doing so, we hope to contribute to the development of more robust and accurate sarcasm detection models, which can improve our ability to interpret human language in a variety of contexts.

II. Related Work

This problem was inspired by a dataset available on Kaggle, which consists of self-labeled sarcastic statements. Other works in the field of sarcasm detection include a corpus by Khodak et al. (2018), which examines self-labeled sarcastic comments on Reddit due to its "frequently-used and standardized annotation for sarcasm." Additionally, other text classifiers, such as the C-LISTM Neural Network for Text classification by Zhou et al. (2015), have been explored.

Previous studies on automated sarcasm detection have focused on identifying logical and pragmatic clues in sentences (Kreuz and Caucci, 2007). Indicators such as interjections, punctuation, and sentimental shifts have been recognized as significant patterns of sarcasm (Joshi et al., 2017). However, social media users frequently use slang and informal language, which makes it difficult to rely on lexical cues and, in turn, complicates sarcasm detection (Poria et al., 2016).

Sarcasm detection involves extracting features from text and processing them through a model to determine if the text is sarcastic or not. Earlier works focused on extracting lexical and pragmatic indicators to identify sarcasm (Tepperman et al., 2006). Linguistic features, such as positive predicates, interjections, and gestural clues such as emojis, are considered strong indicators (Carvalho et al., 2009). Recent works have used logistic regression, support vector machine with sequential minimal optimization (González-Ibáñez et al., 2011), and random forest (Bouazizi et al., 2016) analysis for classification. Other models, such as Convolutional Neural Networks (CNN), have also been utilized after extracting content and context-based information (Hazarika et al., 2018).

Recent breakthroughs in Natural Language Processing (NLP) have resulted in several models that achieve high accuracy in sarcasm detection. For example, Khatri et al. (2020) proposed using machine learning techniques with BERT and GloVe embeddings to detect sarcasm in tweets. Anan et al. (2023) introduced a BERT system that achieved remarkable results in detecting sarcasm in the Bangla language. However, these models are expensive to train, and with limited computing power, it can be challenging to train these models from scratch using platforms such as Google Colab.

III. Dataset

Although there are billions of comments on social media platforms, little data with sarcasm labels is publicly available. In this project, we used a Kaggle dataset (Ofer, 2017) that can

be accessed for free. The dataset contains 1.3 million scraped Reddit comments containing "\s". The "\s" tag is used by Reddit users to indicate their comment is sarcastic, and is a reliable indicator of sarcasm. The data includes the following relevant columns:

- Label: binary indicator of whether the comment is sarcastic or not
- Comment: text content of the comment being analyzed
- Parent Comment: text content of the comment the original comment is replying to

After generating word clouds representing the most common words used in sarcastic and non-sarcastic comments, we can see that there are a few interesting insights to be gleaned from the textual composition of the comments. Figure 1 shows the word clouds. We can see that there is nothing particularly notable about the non-sarcastic word cloud, which is to be expected since that word cloud should simply reflect textual composition of the regular English language (albeit skewed slightly by the topics covered by the communities our data came from). We can see many similar words present in the sarcastic word cloud, although it also includes several notable words not present in prominent positions in the non-sarcastic word cloud. In particular, the presence of "obviously", "totally", "clearly", and "everyone" are quite interesting. Readers may recognize these words as indicators of hyperbole or irony, both common themes in sarcasm. For example, "Obviously." may be used as a response to appear to agree with a sentiment with which the commenter actually disagrees.

Fig. 1: Word clouds demonstrating differences in the most common words that appear in non-sarcastic and sarcastic comments.

IV. Methods

A. Preprocessing

We removed the data with missing comment sections, then converted the sentences to lower case and preserved only the characters that made up words. We also limited the comment length to 30 words. We then randomly split the data into an 80/20 ratio for the purposes of training and validation. Each comment is assigned with a label, indicating whether it is sarcastic or not.

B. Word Embeddings

We used Keras' one_hot preprocessing utility to encode the comments in our dataset. This preserves the order of words in a sentence, since that is clearly quite important in any NLP task. Although there were 1,010,773 unique words in our training data, we limited the depth of the one hot dimension to 5,000 since it is not realistic given our computational constraints to uniquely encode all 1MM+ words. Although this utility's name implies it returns a list of one-hot vectors, it actually returns a list of integers in the range [1, 5000], reducing memory requirements.

C. Models

1) Recurrent Neural Networks: Recurrent Neural Networks (RNNs) have been a popular method for analyzing language for years. RNNs operate over sequences of input vectors with variable size. We constructed a simple RNN model for classification. After word embedding, the input is sent into one RNN layer. The output is flattened, and then a dense layer is used to get a 1D output ranging from 0 to 1, where 1 indicates certain sarcasm and 0 indicates no sarcasm.

2) LSTM: Long-Short Term Memory (LSTM) is a special type of Recurrent Neural Network (RNN) that remembers information from a long time ago. Thus it better captures dependencies over longer distances. It can be used to detect sarcasm that is related to contents several sentences earlier. This is especially beneficial when analyzing larger comments. LSTM focuses on the cell state Ct which will be used to deal with future element Ct+1. The output of Ct is copied to Ct+1, differentiating LSTM from regular RNNs. Thus Ct+1 includes a summary of all previous input information. LSTM solves the long-term dependency problem of RNN by introducing a gate mechanism to control the flow and loss of features. Ct passes through three gates before the result is passed to Ct+1: the input gate, the output gate and the forget gate. All three gates use sigmoid activation functions that generate values between 0 and 1 to describe how many input signals can be passed through. 1 means all pass; 0 means none pass.

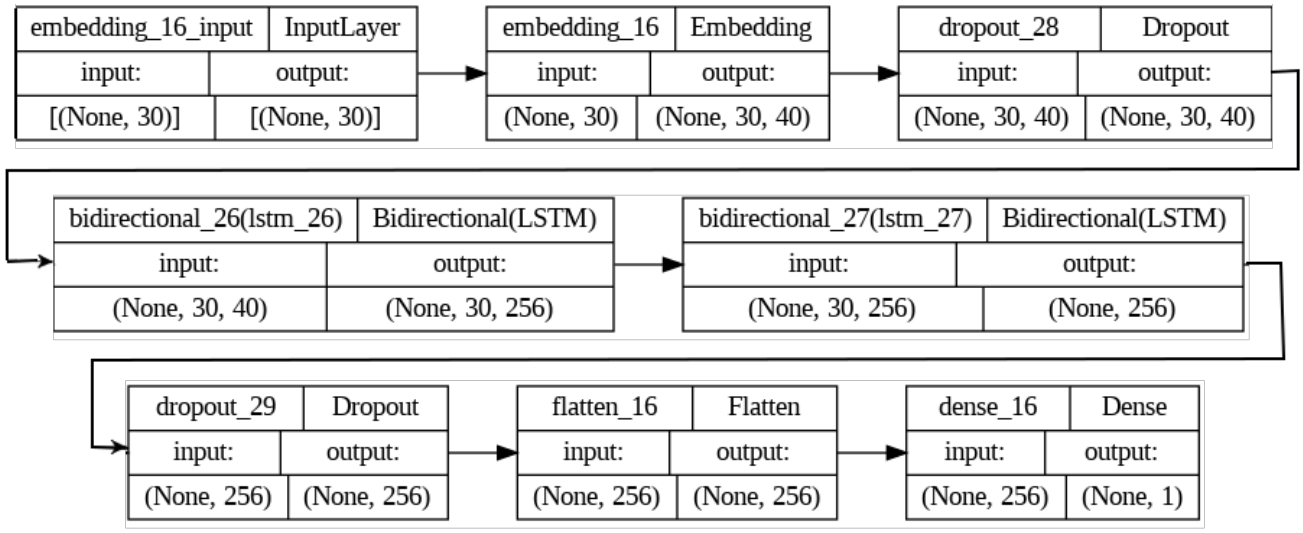

Fig. 2: Two Layer BiLSTM Model Structure

3) GRU: Gated recurrent unit (GRU) is a variant based on LSTM. It has a simpler structure than LSTM networks but still performs well. LSTM is time consuming: Each LSTM cell implies four MLPs. If the time span of LSTM is large and the network is deep, the computation may be prohibitively expensive. GRU helps to address this shortcoming. GRU uses update gates and reset gates to reduce the multiplication of matrices in hidden layers and accelerate training speed. It has two inputs and two outputs while LSTM has three inputs and three outputs. GRU also uses the hidden state ht and input xt. Candidate activation is generated by exploiting reset gate rt. The most important step is the update memory stage. The same update gate vector zt is used to choose and forget memory. That is the reason of GRU's effectiveness.

4) Bidirectional: Bidirectional is a type of neural network architecture that can is used to improve the accuracy of natural language processing tasks. It allows information to flow in both forward and backward directions through the layers of the network, improving the network's ability to capture the context of a sequence. Bidirectional has been applied onto LSTM and GRU modules to create Bidirectional Long Short-Term Memory (BiLSTM) and Bidirectional Gated Recurrent Unit (BiGRU).

By processing the input sequence in both forward and backward directions, BiLSTM and BiGRU are able to take into account both past and future context when producing output at any time step. This can improve the performance in natural language tasks such as language translation, sentiment analysis, and speech recognition. We explore how Bidirectional architecture can affect models in the sarcasm detection task.

D. Process

In constructing our model, we followed a specific sequence that involved adding an embedding layer to the input, applying the corresponding RNN modules, flattening the output, and then feeding it into a dense layer with a sigmoid activation function. Dropout layers are used to prevent overfitting. We used a binary cross-entropy loss function and the Adam optimizer during training, with each model trained for ten epochs. We tested the regular LTSM and GRU architectures, as well as Bidirectional versions of both architectures. We also tested all architectures in both one and two layer configurations. An example architecture is shown in Figure 2.

V. Results

Our evaluation of the various models using our dataset of sarcastic comments is presented in Table 1. All of our models performed quite similarly, although we can see that the two-layer architectures performed better in all cases. Among the models tested, the two-layer BiLSTM architecture achieved the highest accuracy of 71.92% on the testing dataset with a training time of 31 minutes. These results demonstrate the effectiveness of our proposed model and highlight the potential of using LSTM-based architectures for sarcasm detection tasks.

Model	One Layer		Two Layers	
	Accuracy	Time	Accuracy	Time
LSTM	71.53%	16 min	71.69%	21 min
GRU	71.33%	16 min	71.50%	21 min
Bidirectional LSTM	71.62%	21 min	71.92%	31 min
Bidirectional GRU	71.22%	21 min	71.33%	29 min

TABLE I: Model performance

VI. Conclusion

Sarcasm detection is a crucial task for natural language processing, as it plays a significant role in interpreting human language in various contexts. Although it poses several challenges, recent advancements in machine learning and NLP have shown promise in developing accurate sarcasm detection models. In this paper, we explored various machine learning

models and evaluated their performance on a public dataset of sarcastic comments. Our results demonstrate that the two-layer BiLSTM model achieved the highest accuracy of 71.92% with 31 minutes of training time. Our findings contribute to the development of more robust and efficient sarcasm detection models, which can improve our ability to interpret human language in real-world applications. Further research can focus on exploring more advanced deep learning architectures and training models on larger datasets to achieve even higher accuracy.

REFERENCES

[1] Roger J Kreuz and Gina M Caucci. 2007. Lexical influences on the perception of sarcasm. In Proceedings of the Workshop on computational approaches to Figurative Language, pages 1–4. Association for Computational Linguistics.

[2] Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017. Automatic sarcasm detection: A survey. ACM Computing Surveys (CSUR), 50(5):73.

[3] Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. 2016. A deeper look into sarcastic tweets using deep convolutional neural networks. arXiv preprint arXiv:1610.08815.

[4] Roberto González-Ibáñez, Smaranda Muresan, and Nina Wacholder. Identifying sarcasm in Twitter: A closer look. ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2(2010):581–586, 2011.

[5] Mondher Bouazizi and Tomoaki Otsuki. A Pattern-Based Approach for Sarcasm Detection on Twitter. IEEE Access, 4:5477–5488, 2016.

[6] M. Khodak, N. Saunshi, K. Vodrahalli, "A Large Self-Annotated Corpus for Sarcasm" arXiv:1704.05579v4.

[7] Chunting Zhou1, Chonglin Sun2, Zhiyuan Liu3, Francis C.M. Lau, "A C-LSTM Neural Network for Text Classification" arXiv:1511.08630v2.

[8] D. Hazarika, S. Poria, S. Gorantla, E. Cambria, R. Zimmermann, and R. Mihalcea, "CASCADE: Contextual Sarcasm Detection in Online Discussion Forums," arXiv:1805.06413

[9] D. Pauli. 2014. "Oh, wow. US Secret Service wants a Twitter sarcasm-spotter," The Register® - Biting the hand that feeds IT, https://www.theregister.com/2014/06/04/secret_service_wants_twitter_sarcasm_radar.

[10] Dan Ofer. 2017. Sarcasm on Reddit. Retrieved from https://www.kaggle.com/datasets/danofer/sarcasm?datasetId=1309&sortBy=voteCount.