Intended Sarcasm Detection In English

*

Soumik Deb Niloy(20301207)
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
soumik.deb.niloy@g.bracu.ac.bd

S.M.ABRAR MUSTAKIM TAKI(20301125)
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
sm.abrar.mustakim@g.bracu.ac.bd

A S M TAREQ MAHMOOD(20101073)
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
asm.tareq.mahmood@g.bracu.ac.bd

Md. Sayed Hasan Emon(20201021)
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
sayed.hasan.emon@g.bracu.ac.bd

Dr. Farig Yousuf Sadeque,PhD
Dept of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
farig.sadeque@bracu.ac.bd

***Abstract*—This paper presents a robust approach to detecting intended sarcasm in English text, leveraging the IsarcasmEval [1] dataset. We enhance the state-of-the-art sarcasm detection by incorporating pre-trained Word2Vec embeddings sourced from the "Google News Vector Negatives" [2] dataset and employing Long Short-Term Memory (LSTM) [3] as the central model. Our method involves meticulous feature engineering and preprocessing to harness the nuanced linguistic cues indicative of sarcasm. Extensive experimentation demonstrates the effectiveness of our approach, with notable improvements in accuracy, precision, recall. Practical applications of sarcasm detection, such as sentiment analysis and humor recognition, underscore the real-world relevance of our research. By combining pre-trained embeddings and LSTM [3], this study advances the field of sarcasm detection while highlighting their practical utility in contemporary Natural Language Processing tasks and communication channels.**

***Index Terms*—NLP; Natural Language Processing; Sarcasm Detections; LSTM; Embeddings; Word2Vec; Machine Learning**

I. Introduction

In the realm of Natural Language Processing (NLP), sentiment analysis has been a topic of keen interest. However, detecting the nuanced form of sentiment—sarcasm—still remains a challenging feat. Sarcasm, a form of verbal irony, involves stating something but meaning the opposite, usually to humorously emphasize a point. While sarcasm is a common form of communication in human interactions, machines find it challenging to detect due to its complexity and dependency on context, intonation, and even cultural cues.

Identify applicable funding agency here. If none, delete this.

Sarcasm detection has practical applications in various domains such as customer service, social media analytics, and computational linguistics. For instance, customer feedback or reviews may contain sarcasm, which could be misunderstood as positive sentiment. Similarly, social media posts often employ sarcasm, affecting the accuracy of sentiment analysis tools. Therefore, an effective sarcasm detection model is highly beneficial in these and other contexts.

The primary objective of this research is to develop an accurate and robust model for detecting sarcasm in text data. The focus is on utilizing advanced machine learning techniques, specifically Long Short-Term Memory (LSTM) [3] neural networks, in combination with pretrained word embeddings. The use of LSTMs allows the model to capture long-term dependencies in the text, which is crucial for understanding the context—a key element in sarcasm detection.

The significance of this work lies in its potential to improve the accuracy of sentiment analysis tools, thereby aiding various applications that rely on understanding human sentiment. An accurate sarcasm detector could be integrated into customer relationship management (CRM) systems, social listening tools, and even mental health analysis applications.

This research employs a dataset from the ISarcasm Eval competition for training and validation. The dataset comprises text from diverse sources, labeled as either sarcastic or non-sarcastic. The focus is on English text, and the evaluation metrics include precision, recall, and the confusion matrix.

II. Research Methodology

A. Data Collection and Preprocessing

The dataset for this research is gleaned from the ISarcasm Eval competition, renowned for its rich diversity of text samples, including but not limited to social media, customer reviews, and news articles. After obtaining the dataset, it is loaded into a DataFrame for easier manipulation and analysis. Specific columns that are crucial to the study, namely 'english_task_a', 'english_task_c', and 'train_data', are isolated for further exploration. To maintain uniformity and improve readability, these columns are renamed to more descriptive labels. Subsequently, the text data is tokenized into sequences to prepare it for the machine learning model.

B. Feature Engineering

One of the significant steps in the preprocessing pipeline is the utilization of pretrained word embeddings. In this study, Google News vectors are employed to transform words into 300-dimensional vectors. This embedding matrix initializes the model's embedding layer. To retain the semantic richness of these pretrained vectors, the weights of the embedding layer are set to be non-trainable.

C. Model ArchitectureModel Architecture

The centerpiece of the sarcasm detection methodology is a neural network model based on Long Short-Term Memory (LSTM) units. The model architecture consists of several layers, each serving a specific purpose. The first layer is an Embedding Layer, which uses the pretrained Google News vectors to transform tokenized sequences into 300-dimensional vectors. The second layer comprises LSTM units that capture the temporal dependencies in the text, aided by a dropout mechanism to curb overfitting. Following the LSTM layer is a Dropout Layer for additional regularization. Finally, a Fully Connected (Dense) Layer maps the LSTM outputs to a single output unit, which undergoes a Sigmoid activation function to ensure the output falls within a [0, 1] range, suitable for the binary classification task at hand.

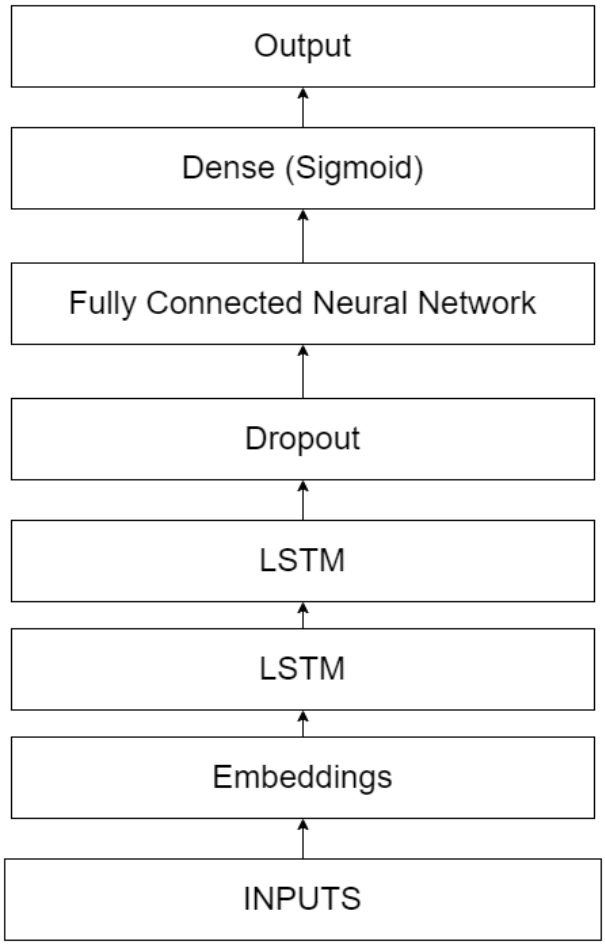

Fig. 1. Model Architecture

III. Implementation

The implementation of the sarcasm detection model involved a series of well-defined steps, each serving a specialized purpose in the construction and fine-tuning of the system. The project was implemented using Python, taking advantage of its rich ecosystem of libraries and frameworks tailored for machine learning and natural language processing tasks.

A. Software and Libraries

The foundational library for constructing and training the neural network model is PyTorch. PyTorch's dynamic computation graph and extensive suite of modules make it ideal for working with Long Short-Term Memory (LSTM) networks. Specifically, PyTorch's nn.Module class was subclassed to create the custom LSTM model for this project. The flexibility and optimization features of PyTorch were instrumental in the successful implementation of the model.

For data manipulation and preparation, the pandas library was utilized. It provided the DataFrame structure, which is particularly useful for structured data manipulation. Additionally, the sklearn library was used for model evaluation, offering robust methods to compute metrics like precision and recall.

B. Word Embeddings

The gensim library facilitated the use of pretrained Google News vectors as word embeddings. These 300-dimensional vectors were imported using gensim's KeyedVectors.load_word2vec_format method and subsequently integrated into the PyTorch-based LSTM model's embedding layer.

C. Model Training

Model training was performed using PyTorch's neural network and optimization libraries. The project used the Adam optimizer and Binary Cross-Entropy Loss (BCELoss), both provided by PyTorch, for weight updates and loss computation, respectively. Custom PyTorch code was written to implement batch training, gradient clipping, and epoch-based training and validation.

D. Validation and Metrics

To assess the model's performance, sklearn was employed to calculate precision and recall after each training epoch. A confusion matrix was also generated and visualized using matplotlib to offer a detailed assessment of the model's classification capabilities.

E. Code Snippets

Key segments of the implementation, such as the LSTM [3] model and the training loop, were modularized into Python classes and functions. The LSTM [3] model, for instance, was encapsulated in a Python class named SarcasmLSTM, which was inherited from PyTorch's 'nn.Module'. The training loop was also organized as a separate Python function, train_model, to maintain code modularity and reusability.

IV. Result & Analysis

After 20 epochs of training, the model's performance was evaluated using several key metrics. The training loss and validation loss were observed to be 0.118 0.118 and 1.79 1.79, respectively, indicating a potential overfitting issue that could be addressed in future work. The model achieved a Precision of 0.394 0.394 and a Recall of 0.335 0.335, resulting in an F1-Score of 0.362 0.362. The Accuracy of the model stood at 0.617 0.617, revealing room for improvement. The Confusion Matrix, which displayed 936 true negatives, 308 false positives, 397 false negatives, and 200 true positives, provided a more granular view of the model's performance. While the model was able to correctly identify a significant number of non-sarcastic comments, it faced challenges in precisely identifying sarcastic comments, as evidenced by the relatively lower Precision and Recall scores.

V. Conclusion

This research ventured to develop a robust model for sarcasm detection utilizing Long Short-Term Memory (LSTM) networks and pretrained Google News word vectors. While the project shows promise in its methodology and choice of evaluation metrics, it was not without challenges. The foremost limitation was the small size of the dataset used for training, which posed two significant setbacks. First, the limited data potentially hindered the model's ability to generalize across a broader range of sarcastic expressions. Second, this data scarcity directly impacted the model's precision, as the training set might not have been diverse enough to capture the complexities of sarcasm. Despite these challenges, the architectural design and training methodology hold promise for effective sarcasm detection. Future work could focus on dataset augmentation, fine-tuning of embeddings, and exploration of more complex models to overcome the limitations encountered in this study.

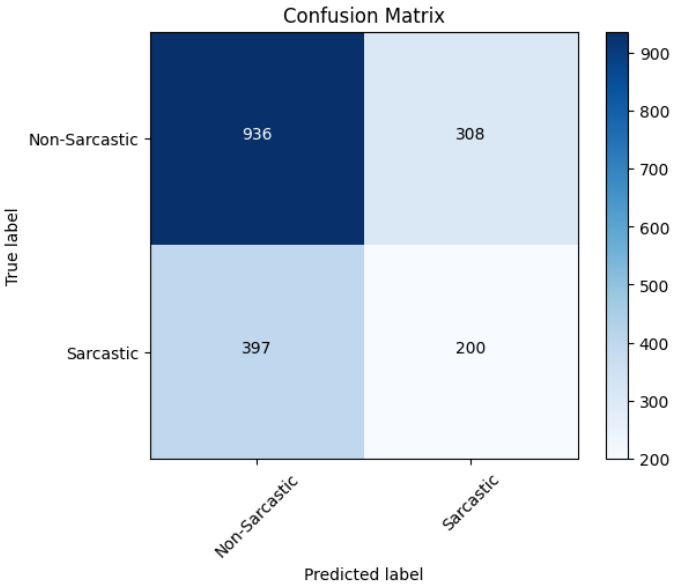

Fig. 2. Confusion Matrix of the Sarcasm Detection Model

VI. Acknowledgements

We would like to extend our heartfelt gratitude to our team members, whose diligent work and invaluable insights significantly contributed to the realization of this research project. Special thanks are also due to ChatGPT, the conversational agent that assisted in various stages of this project, from initial brainstorming to final documentation. Its role in facilitating a smooth research process cannot be overstated.

References

[1] I. Abu Farha, S. V. Oprea, S. Wilson, and W. Magdy, "SemEval-2022 task 6: iSarcasmEval, intended sarcasm detection in English and Arabic," in *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 802–814. [Online]. Available: https://aclanthology.org/2022.semeval-1.111

[2] H. Face, "Word2Vec Google News 300," https://huggingface.co/fse/word2vec-google-news-300, 2023, accessed: September 4, 2023.

[3] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM - a tutorial into long short-term memory recurrent neural networks," *CoRR*, vol. abs/1909.09586, 2019. [Online]. Available: http://arxiv.org/abs/1909.09586