Saracasm Detection Project

Leqi Liu
Université Paris-Saclay,
leqi.liu@etu-upsaclay.fr

Jiren Ren
Université Paris-Saclay,
renjiren120@gmail.com

Abstract

Sarcasm detection is a challenging task in Natural Language Processing (NLP) due to its implicit nature, where the intended meaning often contrasts with the literal text. Accurately identifying sarcasm is crucial for sentiment analysis, opinion mining, and conversational AI. In this study, we employ FastText and SkipGram for word vectorization, leveraging its ability to capture subword information and enhance word representations. These embeddings are then applied to CNN and LSTM to evaluate their effectiveness in sarcasm classification. Besides, we use SVM as a baseline model for comparison to exemplify the ability of neural networks in distinguishing sarcastic texts. The study hypothesizes that LSTM will excel in detecting sarcasm due to its ability to capture long-range dependencies, while CNN may be effective in identifying localized sarcasm patterns. The results are expected to shed light on the strengths and limitations of each model, offering insights to improve sarcasm detection techniques.

1 Introduction

The ability to accurately detect sarcasm can enhance AI's interpretative skills in various applications, such as social media analysis, chatbot interactions, and humor recognition.

With the advancement of deep learning and large-scale language models, sarcasm detection has gained significant research attention. Recent approaches leverage transformer-based architectures, such as BERT and GPT, to better grasp the implicit nature of sarcasm. Additionally, AI-generated sarcastic expressions can be used to improve training datasets, helping models better understand and generate nuanced responses. This can lead to more sophisticated conversational AI, capable of engaging in humor-driven interactions and understanding complex emotional contexts.

By improving sarcasm detection, NLP models can become more adept at reasoning about implicit information, ultimately enhancing their adaptability to fields like literary analysis, customer sentiment analysis, and humor recognition. This project explores various techniques to build a robust sarcasm detection model and investigates how AI can be trained to recognize and even generate sarcastic expressions for improved language understanding.

2 Methods

In this study, we build upon the work of *Fracking Sarcasm using Neural Networks* (Ghosh et al., 2016), who explored sarcasm detection using neural networks without explicit word vectorization techniques. Unlike their approach, we utilize FastText and SkipGram to vectorize the words and compare its effectiveness in sarcasm detection. FastText enables our models to capture subword information, making it particularly useful for handling informal and creative sarcastic expressions. We apply these embeddings to CNN and LSTM. Besides, we use SVM as a baseline model for comparison to exemplify the ability of neural networks in distinguishing sarcastic texts.

By leveraging word vectorization, our study seeks to determine whether explicitly learned embeddings enhance sarcasm detection compared to previous neural network approaches. Through empirical evaluation, we aim to provide insights into the advantages and limitations of different architectures and contribute to the development of more effective sarcasm detection models.

3 Dataset

For this sarcasm detection task, we used a dataset sourced from the reference *Fracking Sarcasm us-*

ing Neural Networks (Ghosh et al., 2016). The dataset consists of labeled Twitter data, where each tweet is categorized as either sarcastic or non-sarcastic. This dataset is specifically curated for sarcasm detection in informal, social media-based text, making it highly suitable for our task, given the nature of sarcasm in such contexts.

The dataset contains a total of 54,931 samples, each of which is a tweet. The labels are binary, with 1 indicating sarcasm and 0 indicating non-sarcasm. This provides a clear and well-defined classification task, where the model learns to distinguish between sarcastic and non-sarcastic expressions based on the language patterns observed in the tweets.

Train Set	
Sarcastic	Non-sarcastic
24,453	26,736
Test Set	
Sarcastic	Non-sarcastic
1,419	2,323

Table 1: Dataset Split

In detail, the dataset is constructed by collecting tweets that explicitly contain the `#sarcasm` hashtag as a retrieval cue. Since relying solely on this heuristic might overlook sarcastic tweets without such explicit markers, the list of indicative hashtags is expanded using an **LSA-based approach**, incorporating tags like `#sarcastic`, `#yeahright`, and other related terms. Additionally, tweets from users with a strong inclination toward either sincerity or sarcasm (e.g., professional comedians) are included to enhance dataset quality.

Unlike previous works, we used **SVM** as our baseline classification model and used SkipGram and FastText for word embeddings. Initially, we conducted experiments on smaller subsets (`train_sample` and `test_sample`) to validate our approach before applying it to the full `train` and `test` sets.

4 Data Preprocessing

In **data preprocessing**, we made several works:

1. All `@mentions` were replaced with `@user` to anonymize user references.

2. Any hashtag matching the pattern `#sarca*` (e.g., `#sarcasm`, `#sarcastic`) was removed to prevent direct labeling bias.

3. **Stopwords** and **URLs** were filtered out to improve model focus on key linguistic features.

4. All **emojis** were replaced with its meaning in English.

5. All verbs and nouns are **lemmatized**.

These preprocessing steps ensured a cleaner, more generalized input for sarcasm detection while maintaining the essential context of the tweets.

5 SVM

Support Vector Machines (SVM) are a well-established machine learning approach that is effective in high-dimensional spaces, making it suitable for text classification tasks. In our experiment, we used SVM with a linear kernel, leveraging the Term Frequency-Inverse Document Frequency (TF-IDF) representation of text. The TF-IDF features were extracted to capture the importance of words relative to the entire corpus, and the linear kernel was chosen for its simplicity and efficiency.

Mathematically, the SVM objective is to find a hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ that maximizes the margin between the two classes. The optimization problem is formulated as:

$$\min_{\mathbf{w},b,\xi_i} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i \quad (1)$$
$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0$$

where ξ_i are slack variables that allow some margin violations, and C is the regularization parameter. The goal is to separate sarcastic and non-sarcastic classes in the feature space, ensuring the best possible margin between them.

6 Neural Network

6.1 CNN

Convolutional Neural Networks (CNNs) have demonstrated great success in capturing local dependencies in data, particularly in tasks where spatial hierarchies or local feature patterns are significant. In the context of sarcasm detection, CNNs

are effective at identifying the short-range relationships between words and phrases, which are often key to understanding the sarcastic tone. The CNN model was designed to learn spatial patterns in the text that reflect sarcastic sentiments, such as reversal of sentiment or ironic expressions, through local contextual cues.

6.2 LSTM

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) that are particularly effective at modeling sequential data with long-range dependencies.(Hochreiter et al., 1997) Given an input sequence $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$, where $\mathbf{x}_t$ is the word embedding at time step t, the LSTM uses gating mechanisms to control the flow of information.

- Forget Gate: Decides what information to discard from the cell state:

$$f_t = \sigma(W_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_f) \tag{2}$$

- Input Gate: Determines which values to update in the cell state:

$$i_t = \sigma(W_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_i) \tag{3}$$

$$\tilde{C}_t = \tanh(W_C \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_C) \tag{4}$$

- Cell State Update: Updates the cell state based on the forget and input gates:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5}$$

- Output Gate: Decides what the next hidden state should be:

$$o_t = \sigma(W_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_o) \tag{6}$$

$$\mathbf{h}_t = o_t * \tanh(C_t) \tag{7}$$

In sarcasm detection, the understanding of sarcastic intent often requires considering the broader context, which can span multiple sentences or phrases. LSTMs, with their gating mechanisms, are able to retain long-term information and mitigate issues like vanishing gradients in traditional RNNs.

We used pre-trained word embeddings to represent the text input, which were then processed by the LSTM layers to capture the sequential relationships between words in the context of sarcasm. A fully connected layer at the output stage was used to classify the final representation into sarcastic or non-sarcastic categories. The LSTM model is capable of learning complex patterns that involve shifts in meaning and sentiment, which are characteristic of sarcastic statements.

7 Experimental Setup

We employed an SVM with a linear kernel as our baseline model. The linear kernel was chosen for its simplicity and computational efficiency. No hyperparameter optimization was performed for the SVM; it was trained using default settings. The SVM model serves as a straightforward comparison to more complex models, helping to highlight the improvements gained from using deep learning architectures.

For the CNN, we designed a multi-layer architecture with one-dimensional convolutional layers. The input to the network consisted of word embeddings (either SkipGram or FastText) fed into an embedding layer. The CNN was trained using the Adam optimizer with a learning rate of 0.001, and we used dropout regularization to mitigate overfitting.

The LSTM model aimed to capture long-range dependencies within the text. Similar to the CNN, the input to the LSTM was word embeddings, which were passed through an embedding layer before entering the LSTM network. The LSTM consisted of a single layer with 128 units. We used the Adam optimizer with a learning rate of 0.001 and applied dropout (0.5) to prevent overfitting.

The code of SVM and neural network was implemented using Sklearn and Pytorch library, respectively. And the code of word vectorization was implemented using Gensim.

8 Experimental Analysis

The models were evaluated using standard classification metrics, including accuracy, precision, recall, and F1-score. Since sarcasm detection requires a balance between precision and recall, we primarily focused on maximizing the F1-score.

8.1 SVM Performance

As shown in Figure 1 and 2, the SVM model demonstrates reasonable performance, particularly in detecting non-sarcastic tweets (F1-score:

0.67). However, it exhibits a higher precision (0.71) for non-sarcastic tweets but lower recall (0.57) for sarcastic tweets. This suggests that the model tends to misclassify sarcastic tweets as non-sarcastic, likely due to its reliance on surface-level text features rather than contextual relationships.

SVM - Classification Report:

	precision	recall	f1-score	support
Non-Sarcastic	0.71	0.64	0.67	2323
Sarcastic	0.49	0.57	0.52	1419
accuracy			0.61	3742
macro avg	0.60	0.60	0.60	3742
weighted avg	0.62	0.61	0.61	3742

Figure 1: SVM Classification Report

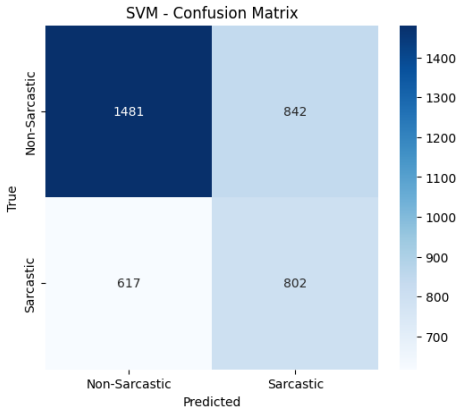

Figure 2: Confusion Matrix of SVM

8.2 CNN Performance

8.2.1 Skip-Gram

As shown in Figures 3 and 4, the CNN model with Skip-Gram embeddings outperforms all other models, achieving an accuracy of 0.79. Its ability to capture local semantic features through convolutional operations allows it to detect subtle linguistic patterns associated with sarcasm.

Classification Report for CNN-SkipGram:

	precision	recall	f1-score	support
Non-Sarcastic	0.89	0.76	0.82	2323
Sarcastic	0.69	0.84	0.76	1419
accuracy			0.79	3742
macro avg	0.79	0.80	0.79	3742
weighted avg	0.81	0.79	0.80	3742

Figure 3: SkipGram & CNN Classification Report

8.2.2 FastText

Similarly, Figures 5 and 6 show that CNN with FastText embeddings performs slightly worse than

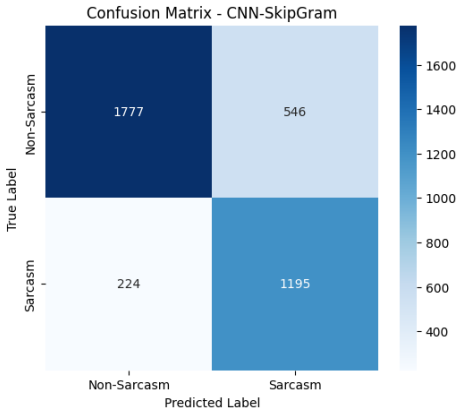

Figure 4: Confusion Matrix of SkipGram & CNN

CNN with SkipGram. Unlike SkipGram, which focuses on individual word relationships, FastText considers subword information, which may not be as effective for sarcasm detection. Since sarcasm often depends on phrase-level rather than word-level nuances, FastText does not provide a significant advantage.

Classification Report for CNN-FastText:

	precision	recall	f1-score	support
Non-Sarcastic	0.87	0.74	0.80	2323
Sarcastic	0.66	0.82	0.73	1419
accuracy			0.77	3742
macro avg	0.77	0.78	0.77	3742
weighted avg	0.79	0.77	0.78	3742

Figure 5: FastText & CNN Classification Report

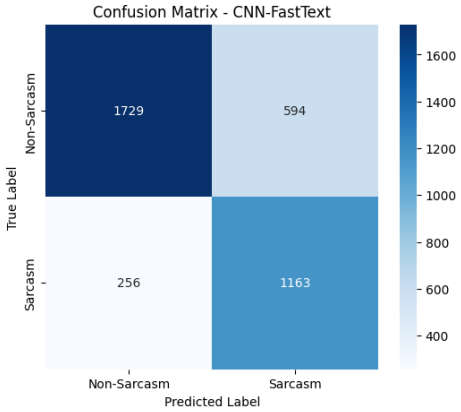

Figure 6: Confusion Matrix of FastText & CNN

8.3 LSTM Performance and Possible Issues

Figures 7 and 8 show the results of the LSTM model with SkipGram embeddings, while Figures 9 and 10 present the results with FastText embeddings. Surprisingly, both LSTM models

fail to distinguish between sarcastic and non-sarcastic tweets, predicting almost all tweets as non-sarcastic.

Classification Report for LSTM-SkipGram:

	precision	recall	f1-score	support
Non-Sarcastic	0.62	1.00	0.77	2323
Sarcastic	0.00	0.00	0.00	1419
accuracy			0.62	3742
macro avg	0.31	0.50	0.38	3742
weighted avg	0.39	0.62	0.48	3742

Figure 7: SkipGram & LSTM Classification Report

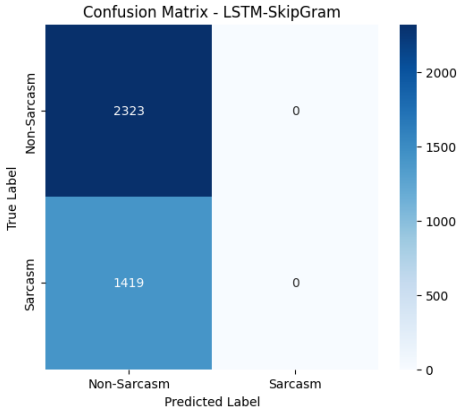

Figure 8: Confusion Matrix of SkipGram & LSTM

Classification Report for LSTM-FastText:

	precision	recall	f1-score	support
Non-Sarcastic	0.62	1.00	0.77	2323
Sarcastic	0.00	0.00	0.00	1419
accuracy			0.62	3742
macro avg	0.31	0.50	0.38	3742
weighted avg	0.39	0.62	0.48	3742

Figure 9: FastText & LSTM Classification Report

One possible explanation for this failure is the **vanishing gradient problem**. LSTMs are designed to capture long-range dependencies, but sarcasm detection in short texts (such as tweets) may not benefit from this characteristic. Instead of leveraging contextual dependencies, the LSTM might struggle with effective learning due to the following factors:

- **Hidden layer size**: We experimented with both increasing and decreasing the hidden layer size, but the results remained unchanged. This suggests that the problem is not due to insufficient capacity but rather a fundamental issue with learning dynamics.

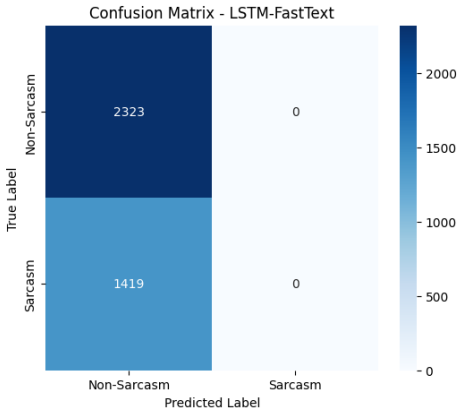

Figure 10: Confusion Matrix of FastText & LSTM

- **Gradient clipping and activation functions**: Initially, we used the sigmoid activation function, followed by attempts with `BCEWithLogitsLoss`, which is often employed to prevent gradient saturation in the 0-1 range. We also applied gradient clipping, but none of these adjustments improved performance. This indicates that while the vanishing gradient problem may exist, the root cause is likely more complex.

8.4 Proposed Improvements for LSTM

To address these issues, the following modifications should be considered:

- **Exploring different architectures**: Since sarcasm detection may require deeper contextual understanding, attention mechanisms or Transformer-based models could be more effective.
- **Using bidirectional LSTM**: Since sarcasm often relies on context from both earlier and later words, a bidirectional LSTM could enhance feature extraction.
- **Experimenting with alternative loss functions**: While we tested `BCEWithLogitsLoss`, other loss functions such as focal loss could potentially improve the model's ability to differentiate between sarcastic and non-sarcastic tweets.

8.5 Comparison of Models

Figure 11 compares the four key performance metrics across all models. Our findings reinforce that deep learning models (CNN, LSTM) generally outperform traditional machine learning

methods (SVM). Among them, CNN with SkipGram achieves the best performance, confirming that **local text patterns are more useful than sequential dependencies for sarcasm detection**.

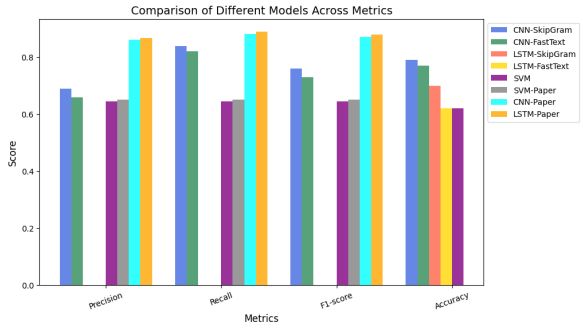

Figure 11: Comparison of the 4 scores for the different models

However, our results are not as strong as those in the referenced works, potentially due to:

- **Computational limitations**: We used a CPU instead of a GPU, restricting our dataset size and training efficiency.
- **Differences in data preprocessing**: Unlike previous studies, we removed usernames, which might have contained contextual clues for sarcasm detection.
- **Lack of additional training techniques**: The reference works might have included fine-tuning strategies that we did not implement.

Despite these challenges, our experiments provide valuable insights into sarcasm classification and highlight the strengths and weaknesses of different modeling approaches.

9 Conclusion and Future Work

In this project, we tackled sarcasm detection using SVM, CNN, and LSTM models, along with word embeddings such as SkipGram and FastText. Our experiments demonstrated that CNN outperformed the other models in capturing the essence of sarcasm, while SkipGram proved more effective in handling rare words. Although our models performed reasonably well, they still struggled with more complex sarcastic contexts.

Future research could focus on exploring advanced models such as Transformer-based architectures and BERT, or employing ensemble techniques to further enhance performance. Additionally, due to computational constraints, we were unable to experiment with different word embedding dimensions. Future work could investigate how varying the embedding size impacts model performance, as larger dimensions may provide richer semantic representations at the cost of increased computational complexity.

Another important direction is addressing the poor classification performance of LSTM models. Our findings suggest that LSTMs may suffer from vanishing gradient issues, leading to suboptimal learning. Future improvements could include using bidirectional LSTMs, incorporating attention mechanisms, or experimenting with alternative loss functions such as focal loss to improve classification accuracy.

Extending sarcasm detection to other languages, integrating additional contextual features such as sentiment analysis or pragmatic cues, and utilizing more diverse datasets would also contribute to enhancing model robustness. Moreover, incorporating multimodal data, such as images or audio, could provide a more comprehensive understanding of sarcasm, enabling models to capture nuances beyond textual information.

10 Contributions

Task	Responsible Member
Data Preprocessing	Leqi Liu
Feature Extraction	Leqi Liu & Jiren Ren
Model Construction	Leqi Liu & Jiren Ren
Model Training & Testing	Leqi Liu & Jiren Ren
Data Visualization	Jiren Ren
Final Report & Slide	Leqi Liu & Jiren Ren

Table 2: Project Contributions

References

Ghosh, A., & Veale, D. T. 2016. Fracking Sarcasm using Neural Network. *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis,* pages 161–169, San Diego, California. Association for Computational Linguistics.

Hochreiter, S., & Schmidhuber, J. 1997. Long short-term memory *Neural computation* 9(8):1735-1780.

Kreuz, R., & Caucci, G. 2007. Lexical Influences on the Perception of Sarcasm. *In Proceedings of the*

Workshop on Computational Approaches to Figurative Language, pages 1–4, Rochester, New York. Association for Computational Linguistics.

Burfoot, C., & Baldwin, T. 2009. Automatic Satire Detection: Are You Having a Laugh?. *In Proceedings of the ACL-IJCNLP 2009 Conference Short Papers,* pages 161–164, Suntec, Singapore. Association for Computational Linguistics.

Tungthamthiti, P., Shirai, K., & Mohd, M. 2014. Recognition of Sarcasms in Tweets Based on Concept Level Sentiment Analysis and Supervised Learning Approaches. *Pacific Asia Conference on Language, Information and Computation,*