Enhanced Sarcasm Detection in Code-Mixed Tamil-English Text Using GRU and LSTM with SMOTE and Padding Techniques

Kogilavani Shanmugavadivel[1], Navbila K[1,*] and Sridhar S[1]

[1]*Department of AI, Kongu Engineering College, Perundurai, Erode.*

Abstract

The intricacy of language and contextual subtleties in code-mixed languages, like Tamil-English, create special difficulties when attempting to identify sarcasm. In order to identify sarcasm, this study compares the effectiveness of machine learning models such as XGBoost, LightGBM, and CatBoost with deep learning models such as LSTM and GRU. Machine learning models and GRU were subjected to the Synthetic Minority Over-sampling Technique (SMOTE) in order to rectify the class imbalance, and sequence pre-padding was employed by LSTM. The findings show that SMOTE enhances macro-average F1 scores and accuracy for the majority of models. Notably, with a macro F1 score of 0.69 and an accuracy of 0.73, LSTM with Pre-padding performed the best. A lower macro-average F1 score of 0.34 was obtained by the GRU model, in contrast, highlighting the challenge of sarcasm recognition in code-mixed languages. For this task, LSTM with padding turned out to be the most efficient model overall.

Keywords

Sarcasm Detection, Code-Mixed Text, SMOTE, LSTM, GRU, Padding techniques.

1. Introduction

The interaction between several languages and their grammatical structures makes it difficult to detect sarcasm [1]in code-mixed languages like Tamil-English [2]. Since models must comprehend subtle differences between the two languages within a single sentence, traditional natural language processing (NLP) techniques frequently falter in these situations [3]. To overcome these difficulties and raise the accuracy of detection, more sophisticated deep learning and machine learning methods are required. This paper investigates a hybrid method that combines deep learning models like LSTM and GRU with machine learning models like XGBoost, LightGBM, and CatBoost [4, 5]. One major issue is class disparity, where examples of sarcasm are less common. We generate synthetic samples to improve detection performance [6] by using the Synthetic Minority Over-sampling Technique (SMOTE) to machine learning models and GRU [7]. In order to guarantee uniform input lengths for LSTM, which is essential to its performance, we employ sequence pre-padding [8]. Our results demonstrate that, in terms of accuracy and macro-average F1 score, LSTM with sequence Pre-padding performs better than the other models [9]. The LSTM performs better when processing code-mixed language data due to its ability to handle sequence lengths, whereas SMOTE improves the performance of GRU and machine learning models [10, 11]. This emphasizes how crucial it is to control input length and data imbalance in order to accurately detect sarcasm in code-mixed languages [12].

2. Related Works

In the paper Sarcasm detection framework employing context, emotion and sentiment features [13], pre-trained transformers and CNN models are used to extract context[14], emotion and sentiment information for sarcasm identification from diverse datasets. Our work employs SMOTE to enhance sarcasm recognition in code-mixed Tamil-English text, addressing class imbalance and Pre-padding

Forum for Information Retrieval Evaluation, December 12-15, 2024, India. https://ai4bharat.iitm.ac.in

*Corresponding author.

kogilavani.sv@gmail.com (K. Shanmugavadivel); navbilak.22aid@kongu.edu (N. K); sridhars.22aid@kongu.edu (S. S)

difficulties in sequence-based models [15]. With an LSTM model, which is designed to tackle the difficulties of mixed-language environments, we obtain improved results, raising the accuracy to 0.73 and the macro-average F1 score to 0.69.

The study, Sarcasm identification using news headlines dataset [16], presents a sizable dataset of news headlines from HuffPost and The Onion and focuses on sarcasm recognition using a hybrid neural network. It does, however, primarily address sarcasm in formal, single-language contexts, leaving difficulties in mixed-language and informal settings. Our approach outperforms theirs because we concentrate on the more difficult issue of sarcasm recognition in code-mixed Tamil-English text, which is a naturally multilingual and informal situation. We employ SMOTE to address the issue of class imbalance. We obtain better accuracy of 0.73 and a F1 score of 0.69 by employing an LSTM model without Pre-padding, indicating increased reliability for practical application [17].

For sarcasm detection, the work "Sarcasm Identification in Text with Deep Learning Models and GloVe Word Embedding" [18] employs deep learning models like CNN, Bi-LSTM, and LSTM, finding that Bi-LSTM outperforms traditional methods on the SarcasmV2 corpus. Our research builds on this by addressing the more complex challenge of detecting sarcasm in code-mixed Tamil and English text. We implement padding techniques for LSTM and use SMOTE to mitigate class imbalance, achieving an accuracy of 0.73 and a macro F1 score of 0.69. This demonstrates enhanced flexibility and performance in managing multilingual data compared to their single-language approach [19].

The study Multimodal Sarcasm identification (MSD) in Videos using Deep Learning Models focuses on verbal and non-verbal cues for sarcasm identification in videos, and on the MUSTARD dataset, it performs better with multimodal data. Our study uses SMOTE and padding to solve the linguistic problems in code-mixed Tamil-English text in order to handle sarcasm. Our model obtains higher accuracy (0.73) and F1 score (0.69) than their video-based approach, which makes it more appropriate for processing huge amounts of text input in contemporary digital interactions.

3. Dataset Description and SMOTE Class Distribution

The dataset used in this study comprises utterances with mixed Tamil and English codes classified as sarcastic or non-sarcastic. Example sentences are shown in Table 1. Our dataset is available at: https://codalab.lisn.upsaclay.fr/competitions/19310#participate. The sarcasm identification process is made harder by the code-mixing of the data because Tamil and English have different syntax and grammar.

Table 1
Example Sentences for Sarcastic and Non-Sarcastic Instances

Text	Label
I think Rajnikant sir look like Iron man	Non-Sarcastic
Bigilku ahppu ready saugada Vijay fans	Sarcastic

3.1. Potential Biases in the Dataset

The distribution of sarcastic and non-sarcastic labels in the dataset shows a notable imbalance. About 73.5 % of the data is classified as non-sarcastic, whereas only 26.5 % is classified as sarcastic. This disparity may cause the model to favor the non-sarcastic class, which would impair its ability to detect sarcasm. As shown in Table 2, the distribution of sarcastic and non-sarcastic labels before SMOTE reveals a significant class imbalance.

The Synthetic Minority Over-sampling Technique (SMOTE) was used to create synthetic samples for the sarcastic class in order to rectify this imbalance. Predictions made by the training set may be skewed toward the majority class (non-sarcastic) as a result of this imbalance until addressed properly. To mitigate the class imbalance, we applied the SMOTE to the training set. SMOTE works by

Table 2
Distribution of Class Labels Before SMOTE

Label	No. of Instances
Non-Sarcastic	21,740
Sarcastic	7,830

Table 3
Distribution of Class Labels After SMOTE

Label	No. of Instances
Non-Sarcastic	21,740
Sarcastic	21,740

generating synthetic samples for the minority class (sarcastic sentences) based on existing samples, thereby balancing the class distribution. As shown in Table 3, the distribution of class labels after applying SMOTE is balanced, with equal instances for both sarcastic and non-sarcastic classes.

3.2. Synthetic Sample Generation for Minority Class

To develop synthetic samples for the minority class, SMOTE takes random data points and creates new samples between these points and their closest neighbors.In this study, 13,910 synthetic samples were generated for the sarcastic class, increasing its representation in the dataset.By growing the minority class's size while preserving its original distribution, this method improves the dataset's balance. Consequently, there is less bias towards the majority class during training as the minority class—sarcastic sentences—is better represented.

4. Methodology

In order to detect sarcasm in code-mixed Tamil-English text, our work employs a methodology that integrates deep learning models and machine learning approaches. As shown in Figure 1, the methodology consists of the following steps:

4.1. Data Preparation

The first step involves **Dataset Collection**, where text samples labeled as either sarcastic or non-sarcastic are gathered. This is crucial for creating a robust training dataset. Following this, **Preprocessing** is conducted, which includes tokenization to split the text into individual words or tokens. This step is essential for converting the raw text into a format suitable for model training. Furthermore, label encoding is applied to convert textual labels into numerical format, facilitating the training of the machine learning models.

4.2. Handling Class Imbalance

To address the issue of class imbalance, we implement the **Synthetic Minority Over-sampling Technique (SMOTE)**. SMOTE generates synthetic examples of the minority class (sarcastic text) by interpolating between existing data points. This technique helps to balance the dataset, thereby improving model performance, especially for machine learning algorithms, which tend to be biased toward the majority class when imbalances exist.

4.3. Model Development

For the **LSTM Model**, we apply **Pre-padding** to standardize input sequences to 100 tokens, padding shorter ones and truncating longer ones. The architecture includes an embedding layer, two LSTM layers with dropout, and a dense output layer with a sigmoid activation for binary classification.

The **GRU Model**, using SMOTE for class imbalance, does not require pre-padding or truncation. This allows the GRU to learn effectively from variable-length sequences, making it more flexible while utilizing the balanced dataset.

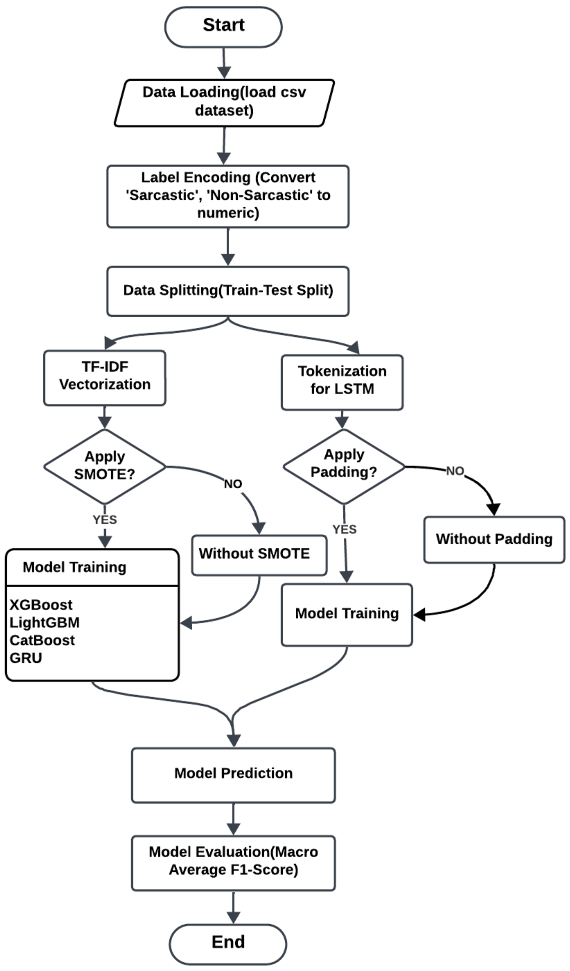

Figure 1: Work Flow

4.4. Model Evaluation

For **Performance Metrics**, we evaluate model performance based on accuracy and the macro F1 score. Accuracy provides an overall measure of model performance, while the macro F1 score is crucial in the context of an imbalanced dataset, as it considers both precision and recall. This ensures a balanced evaluation of model performance across both classes.

4.5. Results Analysis

In the **Outcome Summary**, we observed that SMOTE significantly improved the performance of most models, particularly the deep learning models. The LSTM model, with its pre-padding approach, achieved the best results, recording a macro F1 score of 0.69 and an accuracy of 0.73. The GRU model, which also employed SMOTE but without pre-padding, performed well as well but slightly lagged behind the LSTM in effectively managing long-range dependencies in the data.
This structured approach highlights the significance of addressing sequence length consistency and class imbalance in sarcasm detection. It demonstrates how both GRU and LSTM models contribute to the effectiveness of the detection procedures.

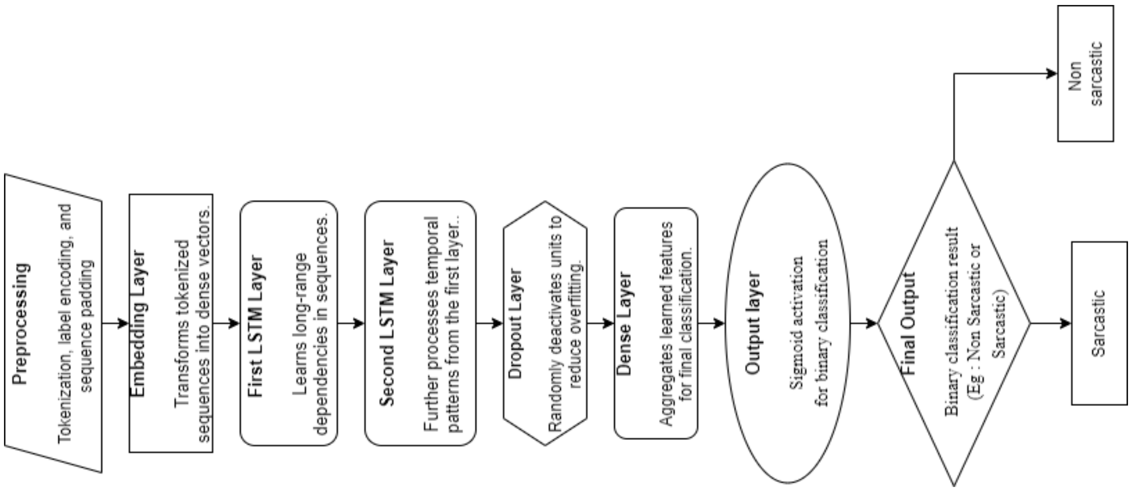
Preprocessing
Tokenization, label encoding, and sequence padding
Embedding Layer
Transforms tokenized sequences into dense vectors.
First LSTM Layer
Learns long-range dependencies in sequences.
Second LSTM Layer
Further processes temporal patterns from the first layer..
Dropout Layer
Randomly deactivates units to reduce overfitting.
Dense Layer
Aggregates learned features for final classification.
Output layer
Sigmoid activation for binary classification
Final Output
Binary classification result (Eg : Non Sarcastic or Sarcastic)
Sarcastic
Non sarcastic

Figure 2: Architectural Overview of LSTM for Sarcasm Detection (Rotated)

5. Architectural Overview of Deep Learning Models: LSTM and GRU for Sarcasm Detection

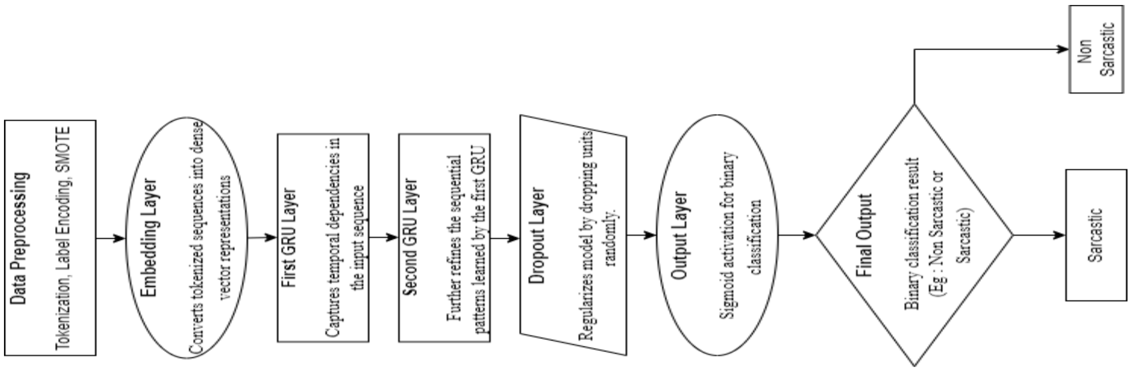
Data Preprocessing
Tokenization, Label Encoding, SMOTE
Embedding Layer
Converts tokenized sequences into dense vector representations
First GRU Layer
Captures temporal dependencies in the input sequence
Second GRU Layer
Further refines the sequential patterns learned by the first GRU
Dropout Layer
Regularizes model by dropping units randomly.
Output Layer
Sigmoid activation for binary classification
Final Output
Binary classification result (Eg : Non Sarcastic or Sarcastic)
Sarcastic
Non Sarcastic

Figure 3: Architectural Overview of GRU for Sarcasm Detection (Rotated)

The architecture of the LSTM model, as shown in Figure 2 comprises several critical components that work together to enable effective sarcasm detection. The process begins with **Preprocessing**, where the data is prepared by tokenizing the text, encoding the labels, and padding the sequences to ensure uniformity in input length. Following this step, an **Embedding Layer** is utilized to convert the words into dense numerical vectors, effectively capturing the semantic meaning of the words in a continuous space.

The first LSTM layer is designed to learn long-term dependencies within the sequences, recognizing patterns and relationships over extended timeframes. This is followed by a second LSTM layer, which refines the patterns identified by the previous layer, enhancing the model's ability to grasp more complex relationships within the data. To mitigate the risk of overfitting, a **Dropout Layer** is incorporated, which randomly turns off a subset of units during training. This helps improve the model's generalization capabilities.

After the LSTM layers, a **Dense Layer** combines the learned features, facilitating the decision-making process. The final output is produced by an **Output Layer**, which employs a Sigmoid activation function to enable binary classification. Ultimately, the model predicts whether the input text is "Sarcastic" or "Non-sarcastic."

The architecture of the GRU model, as depicted in Figure 3 shares similarities with the LSTM model but

also features distinct characteristics. The process begins with **Data Preprocessing**, which includes tokenizing the input text, encoding the labels, and applying SMOTE to address class imbalance. Like the LSTM model, the GRU model also utilizes an **Embedding Layer** to transform tokenized sequences into dense vector representations.
The first GRU layer captures the temporal dependencies in the input data, learning to recognize relevant patterns over time. This is followed by a second GRU layer, which further refines the patterns learned by the first layer. To prevent overfitting, a **Dropout Layer** is integrated into the architecture, similar to the LSTM model.
Finally, the **Output Layer** applies a Sigmoid activation function for binary classification, culminating in the model's prediction of either "Sarcastic" or "Non-sarcastic."

5.1. Differences Between LSTM and GRU Other than Architectures

While the primary distinction between the architectures shown in Figures 2 and 3 lies in the choice of the recurrent unit (LSTM vs. GRU), several other differences contribute to their performance and efficiency:
1. Memory Management: LSTM architectures include cell states, allowing them to maintain long-term memory across time steps more effectively. In contrast, GRUs combine the cell and hidden states, simplifying the memory management process.
2. Gating Mechanisms: LSTMs have three gates: input gate, forget gate, and output gate, which control the flow of information. GRUs, on the other hand, have only two gates: the update gate and the reset gate, leading to a more streamlined process.
3. Complexity: Due to the additional gating mechanisms and cell states, LSTMs are typically more complex than GRUs. This complexity can make LSTMs slower to train, while GRUs can often achieve comparable performance with fewer parameters.
4. Performance: In some cases, GRUs may outperform LSTMs on specific tasks or datasets due to their simpler architecture and reduced training time, while in other scenarios, LSTMs may be more effective at capturing long-range dependencies.
5. Use Cases: The choice between LSTM and GRU may also depend on the specific application. LSTMs are often preferred for tasks requiring the retention of longer sequences, while GRUs can be effective for shorter sequences or when computational efficiency is prioritized.
This detailed comparison highlights not only the differences in the recurrent units themselves but also the implications these differences have on the overall architecture and performance of the models.

5.2. Class Imbalance and Sequence Handling in GRU and LSTM

In the GRU model, SMOTE is used to address class imbalance by generating synthetic samples for the minority class, ensuring a more balanced training dataset. This helps the model generalize across both classes. GRU's architecture efficiently handles variable-length sequences without requiring Pre-padding, making it robust for tasks involving shorter-term dependencies. In contrast, the LSTM model, which is inherently better at capturing long-term dependencies, does not rely on SMOTE. LSTM generalizes well despite data imbalance and requires pre-padding to ensure consistent sequence lengths, allowing it to effectively learn temporal patterns without being affected by variations in input length.

6. Hyperparameter Tuning Analysis

Although we tested a number of algorithms, the LSTM model that employed the pre-padding technique consistently outperformed the others in sarcasm detection.The number of LSTM layers, batch size, and learning rate were systematically tested. Batch sizes of 32 and 64 were evaluated for training stability and speed. The Adam optimizer's default learning rate was used for optimization. The model architecture included 128 units in the first LSTM layer and 64 in the second. We used random and grid search to find the best configuration. The goal was to maximize accuracy and macro F1-score. The model was trained for 10 epochs with a 0.2 validation split. This process helped track overfitting. It significantly improved model performance on both training and validation datasets.

7. Results and Discussion

7.1. Evaluation metrics

Our main evaluation statistic is the macro F1-score. The F1-score for each class (sarcastic and non-sarcastic) is individually calculated by the macro F1-score, which then averages these values. Since this approach gives equal weight to each class, it is a suitable metric for datasets with uneven class distributions, like ours. Accuracy is also used to evaluate the model's performance by measuring the percentage of correct predictions. Additionally, precision and recall are also important performance metrics that provide insights into the model's effectiveness in identifying each class accurately.

7.2. Results

Significantly surpassing other models, the LSTM model with Pre-padding obtained the highest macro F1-score of 0.69. GRU struggled the most to recognize sarcasm, receiving a SMOTE score of just 0.34. As shown in Table 4, the LSTM model's superior performance highlights the effectiveness of incorporating Pre-padding in handling sequence data for sarcasm detection.

Table 4
Macro Average F1-Score Comparison of Models

Algorithm	**Macro avg F1-Score**	
	with SMOTE	**without SMOTE**
LSTM (with Padding)	0.69	0.32
LightGBM	0.53	0.50
CatBoost	0.53	0.50
XGBoost	0.51	0.49
GRU	0.34	0.32

The LSTM model with Pre-padding achieved the highest accuracy of 0.73, demonstrating its effectiveness for sarcasm detection. With SMOTE applied, the GRU model produced the second-highest accuracy at 0.64. Other machine learning models, including XGBoost, LightGBM, and CatBoost, achieved similar performance, with an accuracy of 0.59 for each. As shown in Table 5, all models showed reduced accuracy without SMOTE, highlighting the importance of SMOTE in mitigating class imbalance and improving model performance.

Table 5
Accuracy Comparison of Models

Algorithm	**Accuracy**	
	with SMOTE	**without SMOTE**
LSTM (with Padding)	0.73	0.70
GRU	0.64	0.55
XGBoost	0.59	0.58
LightGBM	0.59	0.58
CatBoost	0.59	0.58

The LSTM model with pre-padding excelled in sarcasm detection, achieving the highest precision (0.75) and recall (0.65). In contrast, the GRU model had lower precision (0.58) and recall (0.45), indicating it missed many sarcastic instances. XGBoost and LightGBM displayed moderate precision (0.61, 0.60) and recall (0.50, 0.52), reflecting a balanced but limited capacity for detection. As shown in Tables 6 and 7 compare the precision and recall of different models. , all models demonstrated improved precision and recall with SMOTE applied, compared to performance without SMOTE.

Table 6
Precision Comparison of Models

Algorithm	with SMOTE	without SMOTE
LSTM (with Padding)	0.75	0.70
GRU	0.58	0.55
XGBoost	0.61	0.57
LightGBM	0.60	0.56
CatBoost	0.62	0.58

Table 7
Recall Comparison of Models

Algorithm	with SMOTE	without SMOTE
LSTM (with Padding)	0.65	0.60
GRU	0.45	0.40
XGBoost	0.50	0.45
LightGBM	0.52	0.48
CatBoost	0.48	0.46

7.3. Impact of Pre-padding in LSTM

7.3.1. Pre-Padding Operation

For the LSTM model to standardize input sequences, the pre-padding operation is essential. Shorter sequences are padded with zeros at the start until they reach the designated maximum length (max_sequence_length) in order to guarantee that all sequences are of the same length. To fit this maximum length, longer sequences are, on the other hand, truncated from the beginning. For instance, setting the maximum length to 5 would truncate a sequence of length 6 to fit the limit, while padding a sequence of length 3, like [1, 2, 3], to [0, 0, 1, 2, 3]. In addition to facilitating effective batch processing, this constant sequence length improves the model's capacity for learning and generalization during training.

7.3.2. Accuracy Comparison (With vs Without Padding)

When Pre-padding is used, the accuracy of the LSTM model improves. As shown in Figure 4, by guaranteeing that input sequences have a consistent length, padding improves performance by enabling the LSTM to handle sequences of different lengths more skillfully.

7.3.3. Macro Average F1-Score Comparison (With vs Without Padding)

The macro F1 score is much greater with Pre-padding, much like accuracy is. This suggests that when padding is included, the model achieves a better balance between precision and recall across classes. As shown in Figure 5, by stabilizing model predictions over a range of sequence lengths, Pre-padding enhances classification performance overall.

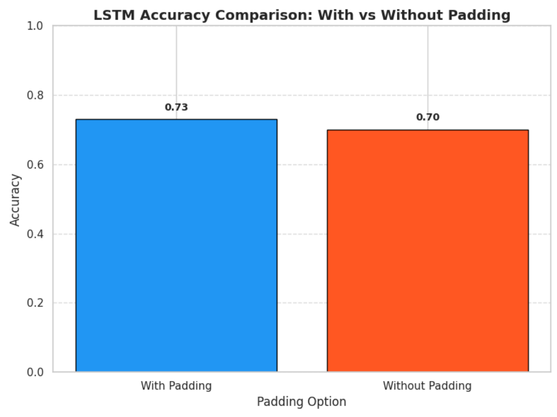

Figure 4: LSTM Accuracy Comparison

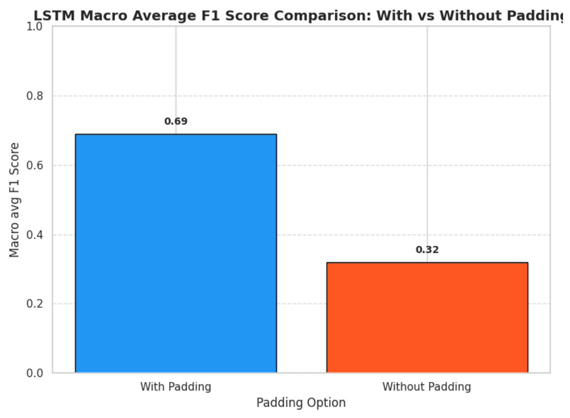

Figure 5: LSTM Macro Average F1-Score Comparison

7.3.4. Confusion Matrix Insights

The LSTM model's classification performance is displayed in the confusion matrix heatmap. As shown in Figure 6, twenty-843 non-sarcastic statements and 6,853 sarcastic statements were accurately detected by the model; 977 sarcastic statements were mistakenly classed as non-sarcastic, and 897 non-sarcastic statements were mistakenly classified as sarcastic. Due to class imbalance, the model performs well, as evidenced by the strong diagonal, which skews significantly toward non-sarcastic predictions.

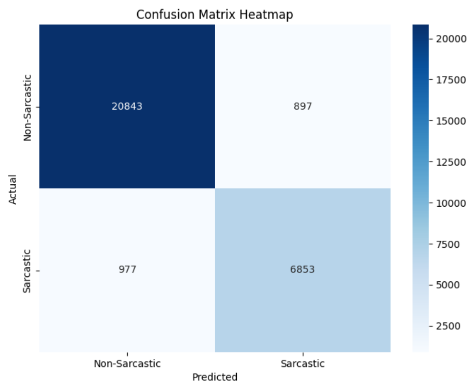

Figure 6: Confusion Matrix for LSTM Model with Padding

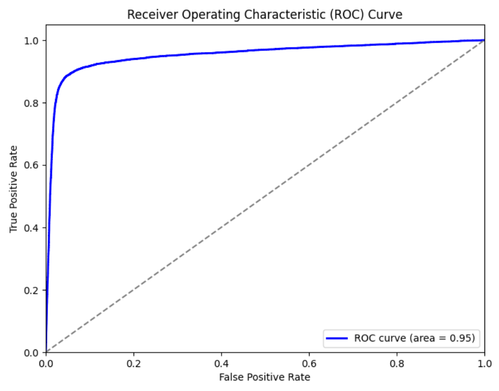

Figure 7: ROC Curve for LSTM Model with Padding

7.3.5. ROC Curve Insights

The Receiver Operating Characteristic (ROC) curve shows a high area under the curve (AUC) of 0.95 for the LSTM model with pre-padding, indicating good model performance in distinguishing between sardonic and non-sarcastic phrases. As shown in Figure 7, the closer the curve is to the upper left corner, the better the true positive rate of the model is compared to the false positive rate.

8. Conclusion

The difficult task of sarcasm identification in code-mixed Tamil-English text was the focus of this work. This task is complicated by linguistic nuances and class imbalance. Our research showed that deep learning models—specifically, LSTM with sequence Pre-padding—performed better than conventional machine learning techniques, with a macro-average F1 score of 0.69 and an accuracy of 0.73. The class imbalance problem was successfully reduced by using the Synthetic Minority Over-sampling Technique (SMOTE), which improved the performance of machine learning and GRU models. In the end, the LSTM model's architecture proved to be more successful in keeping the input sequences for sarcasm detection in this situation consistent.

9. Future Research Scope

Our future research will focus on improving sarcasm detection in code-mixed text by utilizing sophisticated models that are more sensitive to contextual cues, like transformer architectures and attention-based mechanisms. To increase the robustness of the model, we also want to add more examples and dialects to our dataset. Investigating semi-supervised learning strategies will also aid in addressing class imbalance and improve model performance. Our ultimate objective is to create an advanced model that can identify sarcasm in real time. This model may find useful in sentiment analysis and social media monitoring.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT in order to: drafting content, grammar and spelling check, etc. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] B. R. Chakravarthi, N. Sripriya, B. Bharathi, K. Nandhini, T. Durairaj, R. Ponnusamy, P. K. Kumaresan, K. K. Ponnusamy, C. Rajkumar, Overview of sarcasm identification of dravidian languages in dravidiancodemix@fire-2024, in: Forum of Information Retrieval and Evaluation FIRE - 2024, DAIICT, Gandhinagar, 2024.

[2] S. Chanda, A. Mishra, S. Pal, Sarcasm detection in tamil and malayalam dravidian code-mixed text, in: FIRE (Working Notes), 2023, pp. 336–343.

[3] B. R. Chakravarthi, N. Sripriya, B. Bharathi, K. Nandhini, S. C. Navaneethakrishnan, T. Durairaj, R. Ponnusamy, P. K. Kumaresan, K. K. Ponnusamy, C. Rajkumar, Overview of the shared task on sarcasm identification of dravidian languages (malayalam and tamil) in dravidiancodemix, in: Forum of Information Retrieval and Evaluation FIRE-2023, 2023.

[4] J. Aboobaker, E. Ilavarasan, Performance analysis of various sarcasm detection algorithms based on feature extraction methods, in: International conference on humans and technology: A holistic and symbiotic approach to sustainable development: ICHT 2022, 2022.

[5] M. Pathak, A. Jain, μboost: An effective method for solving indic multilingual text classification problem, in: 2022 IEEE Eighth International Conference on Multimedia Big Data (BigMM), 2022.

[6] N. Sripriya, T. Durairaj, K. Nandhini, B. Bharathi, K. K. Ponnusamy, C. Rajkumar, P. K. Kumaresan, R. Ponnusamy, C. N. Subalalitha, B. R. Chakravarthi, Findings of shared task on sarcasm identification in code-mixed dravidian languages, FIRE 2023 16 (2023) 22.

[7] S. Chatterjee, S. Bhattacharjee, K. Ghosh, A. K. Das, S. Banerjee, Class-biased sarcasm detection using bilstm variational autoencoder-based synthetic oversampling, Soft Computing (2023).

[8] A. Kumar, V. T. Narapareddy, V. A. Srikanth, A. Malapati, L. B. Murthy, Sarcasm detection using multi-head attention based bidirectional lstm, in: IEEE Access (Volume: 8), 2020.

[9] R. Akula, I. Garibay, Explainable detection of sarcasm in social media, in: Proceedings of the Eleventh Workshop on Computational Approaches to ..., 2021, 2021.

[10] B. R. Chakravarthi, Hope speech detection in youtube comments, Social Network Analysis and Mining 12 (2022) 75.

[11] B. R. Chakravarthi, A. Hande, R. Ponnusamy, P. K. Kumaresan, R. Priyadharshini, How can we detect homophobia and transphobia experiments in a multilingual code-mixed setting for social

media governance, International Journal of Information Management Data Insights 2 (2022) 100119.

[12] B. R. Chakravarthi, N. Sripriya, B. Bharathi, K. Nandhini, N. Chinnaudayar, C. N. Subalalitha, T. Durairaj, R. Ponnusamy, P. K. Kumaresan, K. K. Ponnusamy, C. Rajkumar, Overview of the shared task on sarcasm identification of dravidian languages (malayalam and tamil) in dravidiancodemix, in: Forum of Information Retrieval and Evaluation FIRE - 2023, 2023.

[13] O. Vitman, Y. Kostiuk, G. Sidorov, A. Gelbukh, Sarcasm detection framework using context, emotion and sentiment features, Expert Systems with Applications (2023).

[14] S. Khan, M. Fazil, V. K. Sejwal, M. A. Alshara, R. M. Alotaibi, A. Kamal, A. R. Baig, Bichat: Bilstm with deep cnn and hierarchical attention for hate speech detection, Journal of King Saud University - Computer and Information Sciences (2022).

[15] D. Jain, A. Kumar, G. Garg, Sarcasm detection in mash-up language using soft-attention based bi-directional lstm and feature-rich cnn, Applied Soft Computing (2020).

[16] R. Misra, P. Arora, Sarcasm detection using news headlines dataset, AI Open (2023).

[17] D. S. Chauhan, G. V. Singh, A. Arora, A. Ekbal, P. Bhattacharyya, An emoji-aware multitask framework for multimodal sarcasm detection, Knowledge-Based Systems (2022).

[18] G. Chandrasekaran, D. J. Hemanth, M. Saravanan, Sarcasm identification in text with deep learning models and glove word embedding, in: 2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2022.

[19] N. A. Helal, A. Hassan, N. L. Badr, Y. M. Afify, A contextual-based approach for sarcasm detection, Open access (2024).