”Cross-Lingual Sarcasm Detection in Social Media: Deep Learning and Machine Learning Approaches”

Sykam Sumanjali, Suda Hari Priya, Likhitha Shree S, Souvik Gorain
Department of AI,Amrita Vishwa Vidyapeetam, Coimbatore, India

***Abstract*—This study explores sarcasm detection in social media text for both English and Hindi. We employ a Long Short-Term Memory (LSTM) network for English sarcasm detection on a Reddit dataset. LSTMs excel at capturing sequential information, crucial for understanding context in sarcasm identification.**

For Hindi sarcasm detection on Twitter data, we leverage Doc2Vec embeddings combined with a Random Forest Classifier. Doc2Vec is a technique for generating document embeddings, allowing us to capture the semantic meaning of Hindi text. By using Doc2Vec in conjunction with a Random Forest Classifier, we aim to detect sarcasm effectively in Hindi social media text.

By comparing the performance of LSTMs on English Reddit data and the Doc2Vec with Random Forest Classifier approach on Hindi Twitter data, we investigate the effectiveness of different techniques for sarcasm detection across languages and social media platforms. This research contributes to the development of robust sarcasm detection systems for diverse languages and online communication environments.

I. Introduction

Sarcasm is a common linguistic phenomenon in social media, where users often convey their messages with irony and humor. Detecting sarcasm automatically presents a significant challenge for natural language processing (NLP) systems due to its subtle and context-dependent nature. Effective sarcasm detection is essential for applications such as sentiment analysis, opinion mining, and improving the accuracy of recommendation systems.

This study focuses on sarcasm detection in social media text for two distinct languages: English and Hindi. By leveraging advanced machine learning models tailored to each language, we aim to enhance the accuracy and robustness of sarcasm detection systems across diverse linguistic and social media environments.

For English sarcasm detection, we employ a comprehensive approach that includes tokenization, word embeddings, and a Long Short-Term Memory (LSTM) network. Tokenization is the process of breaking down text into individual tokens or words, which serves as the first step in text preprocessing. Following tokenization, word embeddings are generated to convert words into dense vector representations that capture semantic relationships. LSTMs, known for their ability to capture sequential dependencies and contextual information, are then used to model the text and identify sarcastic content. This pipeline is trained and evaluated on a Reddit dataset, leveraging the platform's rich and varied user interactions.

For Hindi sarcasm detection, we undertake data cleaning and preprocessing as the initial step to ensure the quality of the input text. This includes removing noise, handling misspellings, and normalizing text. Subsequently, we use the Gensim library to build word vectors and employ the Doc2Vec model to create document embeddings that capture the context of the entire text.

This research provides a comparative analysis of the effectiveness of DL and ML models in sarcasm detection for English and Hindi languages. By evaluating these models on different social media platforms—Reddit for English and Twitter for Hindi—we gain insights into the adaptability and performance of these techniques in various linguistic and contextual settings.

Our findings contribute to the development of more sophisticated and accurate sarcasm detection systems, facilitating better handling of sarcasm in multilingual social media environments. This work not only enhances the understanding of sarcasm in NLP but also supports the creation of more inclusive and comprehensive sentiment analysis tools.

II. LITERATURE REVIEW

Sarcasm detection in social media text is a well-studied yet challenging task in Natural Language Processing (NLP). Here, we explore relevant research in this area, focusing on the papers provided.

Key Points and Gaps in the Literature

Sarcasm Detection in Natural Language Processing This paper provides a general overview of sarcasm detection techniques in NLP, discussing various approaches like lexicon-based methods, rule-based systems, and machine learning algorithms. It emphasizes the importance of understanding context and sentiment for accurate sarcasm detection. However, it lacks a focus on advanced deep learning techniques and does not explore multilingual sarcasm detection extensively.

Sentiment Analysis of Social Network Data for Early Crisis Informatics While not solely focused on sarcasm detection, this paper highlights the challenges of sentiment analysis in social media due to informal language and irony. This is particularly relevant to sarcasm detection as it underscores the need for robust NLP techniques. However, it provides limited insights on specific sarcasm detection models. [5]

A Hybrid Approach for Sarcasm Detection in Twitter Data Using Sentiment Analysis and Machine Learning Techniques This paper proposes a hybrid approach combining sentiment analysis with machine learning techniques. It demonstrates that the hybrid approach outperforms sentiment analysis alone, yet it is focused only on English language Twitter data and

provides limited evaluation on other social media platforms. [1]

Irony Detection in Social Media Text Using Deep Learning Looking towards more recent advancements, this paper explores irony detection in social media text using deep learning techniques. It highlights the effectiveness of deep learning models in capturing complex linguistic features associated with irony, including sarcasm. However, it primarily focuses on English text and lacks a comprehensive comparison with non-deep learning methods. [3]

A Survey on Sarcasm Detection Using Deep Learning This survey provides a comprehensive overview of deep learning techniques for sarcasm detection and discusses various deep learning architectures employed in this domain. However, it lacks detailed discussion on cross-lingual sarcasm detection and minimal focus on dataset diversity and platform-specific challenges. [2]

Sarcasm Detection in Hindi Sentences using Support Vector Machine This paper investigates sarcasm detection in Hindi using Support Vector Machines (SVM) and demonstrates the applicability of machine learning for non-English languages. However, it is limited to SVM, lacking exploration of other advanced techniques, and focused on a single language without cross-lingual insights.

Findings of the WNUT-1.2 Workshop on Sarcasm Detection Finally, the findings of this workshop provide insights into the performance of various sarcasm detection methods on different datasets. It is a benchmark study evaluating the effectiveness of existing techniques, yet it is focused on benchmarking rather than developing new techniques and limited to datasets used in the workshop, potentially lacking broader applicability. [6]

Abdullah Yahya Abdullah Amer and Tamanna Siddiqu have proposed a novel algorithm for sarcasm detection using a supervised machine learning approach. This study builds on the foundational work in the field, incorporating advanced techniques to improve the accuracy of sarcasm identification.

Prior research by Joshi et al. (2017) highlighted the complexity of sarcasm detection, emphasizing the need for contextual understanding and the limitations of traditional machine learning methods [8]. In contrast, Amer and Siddiqu's approach leverages supervised learning, which involves training a model on labeled data to recognize patterns indicative of sarcasm.

Another significant contribution to the field is the work by Poria et al. (2016), who explored the use of deep learning techniques for sentiment analysis, showing that these methods can effectively capture the subtleties of human emotions [9]. Amer and Siddiqu extend this by integrating supervised learning with a focus on sarcasm, demonstrating improved performance over conventional models.

Additionally, the study by Ghosh and Veale (2016) provided insights into feature extraction for sarcasm detection, suggesting that a combination of lexical, syntactic, and semantic features can enhance model accuracy [10]. Amer and Siddiqu's algorithm incorporates these insights, utilizing a comprehensive feature set to train their model.

Despite these advancements, challenges remain, particularly in terms of dataset variability and the inherent ambiguity of sarcastic language. The novel algorithm by Amer and Siddiqu addresses some of these issues by employing a robust training process and validating the model on diverse datasets, showing promise for real-world applications.

In conclusion, Amer and Siddiqu's novel algorithm represents a significant step forward in sarcasm detection using supervised machine learning. By building on previous research and addressing existing challenges, their approach offers improved accuracy and robustness, contributing valuable advancements to the field of NLP.

In conclusion, these papers showcase the ongoing development of various techniques for sarcasm detection in social media text. Machine learning, deep learning, and hybrid approaches all show promise in this domain. Additionally, research is beginning to explore sarcasm detection in languages beyond English, emphasizing the need for cross-lingual approaches for effective social media analysis. As research progresses, sarcasm detection systems will become more robust and versatile, enabling a deeper understanding of sentiment and communication patterns within online communities.

III. DATA BASE

For English sarcasm detection we are using two datasets as mentioned in the following sections

FASTTEXT CRAWL 300D 2M DATASET

This dataset consists of 300-dimensional word embeddings trained using FastText on a vast web corpus containing approximately 2 million words. FastText is an open-source library developed by Facebook AI Research that provides efficient tools for text representation and classification.

Context

The word embeddings are trained using FastText's continuous bag-of-words (CBOW) or skip-gram model on a large collection of text data crawled from the web. The model learns to represent words as dense vectors in a continuous vector space based on their context and semantic meaning. Each word in the vocabulary is represented by a 300-dimensional vector. These vectors capture semantic relationships between words, allowing for tasks such as word similarity, analogy completion, and text classification.

The dataset is widely used in natural language processing tasks such as sentiment analysis, text classification, named entity recognition, and machine translation. Researchers and practitioners leverage these pre-trained word embeddings as a foundational component in building various NLP models and applications.

SARCASM ON REDDIT

[7] This dataset contains 1.3 million sarcastic comments from the Internet commentary website Reddit. The dataset was generated by scraping comments from Reddit containing the \s (sarcasm) tag. This tag is often used by Redditors

to indicate that their comment is in jest and not meant to be taken seriously, and is generally a reliable indicator of sarcastic comment content. The dataset includes both balanced and imbalanced versions, with the true distribution ratio being approximately 1:100. It comprises 1.3 million sarcastic statements, along with the comments they responded to, as well as many non-sarcastic comments from the same source.

TWEETS DATASET IN HINDI LANGUAGE

This dataset consists of over 16,000 tweets (including both sarcastic and non-sarcastic) for researchers interested in working on Sarcasm Detection in Hindi.

Number of Sarcastic tweets: 6051

Number of Non-Sarcastic Tweets: 10128

These tweets were extracted and prepared using tweet scraping code from the Github repository of Mr. Griffin Leow. The code was modified to extract tweets in native Hindi and with specific hashtags. The dataset covers tweets from January 1, 2012, to June 23, 2020.

IV. METHODOLOGY

Data Loading and Configuration: Essential libraries for data handling, model creation, and text processing are imported to ensure the necessary tools are available for each step of the analysis. Pandas settings are adjusted to display all columns and the entire text content, facilitating better inspection and debugging of the data.

Text Preprocessing: A preprocessing function is developed to clean and standardize the text data. This involves converting text to lowercase, removing HTML tags, URLs, special characters, and extra spaces, handling contractions, and ensuring punctuation is appropriately spaced. The preprocessing function is then applied to the text data to prepare it for further analysis and model training.

Tokenization and Sequencing: The Keras Tokenizer is used to convert the cleaned text data into sequences of tokens (words), transforming the text into a numerical format suitable for model input. The token sequences are padded to ensure uniform length across all inputs, which is necessary for efficient batch processing in neural networks.

Embedding Layer Preparation: Pre-trained FastText embeddings are loaded to represent words in a dense vector format, capturing semantic relationships between words. An embedding matrix is created to map the vocabulary words to their corresponding embedding vectors. This matrix is used to initialize the embedding layer in the model.

Model Architecture: The input layer is defined, specifying the input shape. An embedding layer is initialized with the pre-trained embedding matrix to convert input tokens into dense vectors. A bidirectional LSTM layer is added to capture sequential dependencies and contextual information in the text data. A global max pooling layer is implemented to reduce the dimensionality of the output from the LSTM layer and retain the most significant features. Dense layers with ReLU activation are added for non-linear transformation, and dropout layers are included to prevent overfitting by randomly setting a fraction of input units to zero during training. The output layer is defined with a sigmoid activation function for binary classification, indicating whether a comment is sarcastic or not.

Model Compilation and Summary: The model is compiled using binary cross-entropy loss and the Adam optimizer, with accuracy specified as the evaluation metric. The model summary is displayed to review the architecture and parameter details.

Model Training: The batch size and number of epochs for training are set. The model is trained on the training data, with a validation split to monitor the model's performance on unseen data during training. This helps in early stopping and model tuning.

Model Evaluation: The model's training and validation performance are tracked to ensure it is learning effectively and not overfitting. The model is tested on sample inputs to verify its sarcasm detection capability and interpret the results.

By following these steps, the methodology leverages pre-trained embeddings, advanced neural network architectures, and robust preprocessing techniques to build an effective sarcasm detection model. This structured approach ensures that the model can learn the nuances of sarcastic language and generalize well to new, unseen data.

V. TRADITIONAL ML METHODS

Before preprocessing, we explored the dataset to understand its structure. The labels and comments were checked for their data types and shapes. Additionally, sample entries from the dataset were examined to get an idea of the content and format of the comments.

Text Preprocessing Text preprocessing is a crucial step in preparing the data for machine learning models. In this project, we used the following preprocessing techniques:

Fig. 1. Pre Processing

- **Tokenization:** The comments were tokenized using a regular expression tokenizer to split the text into individual words.
- **Stop Words Removal:** Common stop words were removed to reduce noise in the data.
- **Stemming:** Words were stemmed using the Snowball Stemmer to reduce them to their root forms.

A custom tokenizer function was implemented to perform these steps.

TF-IDF Vectorization To convert the text data into numerical features, we used TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This technique transforms

the text into a matrix of TF-IDF features, which reflects the importance of each term in the document relative to the entire dataset.

Train-Test Split The dataset was split into training and testing sets using an 80-20 split. This ensures that the models can be trained on a portion of the data and evaluated on unseen data to test their performance.

Machine Learning Models Two machine learning models were trained and evaluated for sarcasm detection:

- **Logistic Regression:** A logistic regression model with class balancing was trained on the TF-IDF transformed training data. This model is suitable for binary classification tasks.
- **Naive Bayes:** A Multinomial Naive Bayes classifier was also trained on the same data. This model is often used for text classification tasks due to its simplicity and effectiveness.

The performance of the models was evaluated using various metrics, including accuracy, confusion matrix, and classification report. The results were compared to determine which model performed better on the sarcasm detection task.

VI. Performing ML on Hindi

Data preprocessing involves several steps to clean and prepare the text data for analysis:

- **Handling Missing Data:** Empty entries are removed to ensure data quality.
- **Text Cleaning:** URLs, mentions, hashtags, and punctuation are removed. Emojis are also stripped out using regular expressions.
- **Tokenization:** The text is tokenized using NLTK's word tokenizer.
- **Stop Words Removal and Stemming:** Common stop words are removed, and words are stemmed to their root forms to reduce noise and dimensionality.

A custom text processing function is applied to clean and standardize the tweets.

TF-IDF Vectorization To convert the textual data into numerical format, TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is employed. This technique transforms the text into a matrix of TF-IDF features, capturing the importance of each term in the document relative to the entire dataset.

Feature Engineering with Doc2Vec We use Gensim's Doc2Vec model for generating document embeddings. Each tweet is tokenized, and a TaggedDocument is created for each entry, associating the text with its label. The Doc2Vec model is trained on these tagged documents to create feature vectors representing the semantic content of each tweet.

Model Training and Evaluation The feature vectors and corresponding labels are split into training and validation sets using a 70-30 split. A RandomForestClassifier is trained on the training set to classify tweets as sarcastic or non-sarcastic. The model's performance is evaluated using several metrics:

- **Accuracy:** The percentage of correct predictions.
- **ROC-AUC Score:** Measures the model's ability to distinguish between classes.
- **Confusion Matrix:** Provides a detailed breakdown of correct and incorrect classifications.

VII. Results

Performance of LSTM Model on English Reddit Dataset: The LSTM model trained on the English Reddit dataset achieved a validation accuracy of 72.76%. This indicates that the model performs reasonably well in distinguishing between sarcastic and non-sarcastic comments in the Reddit dataset.

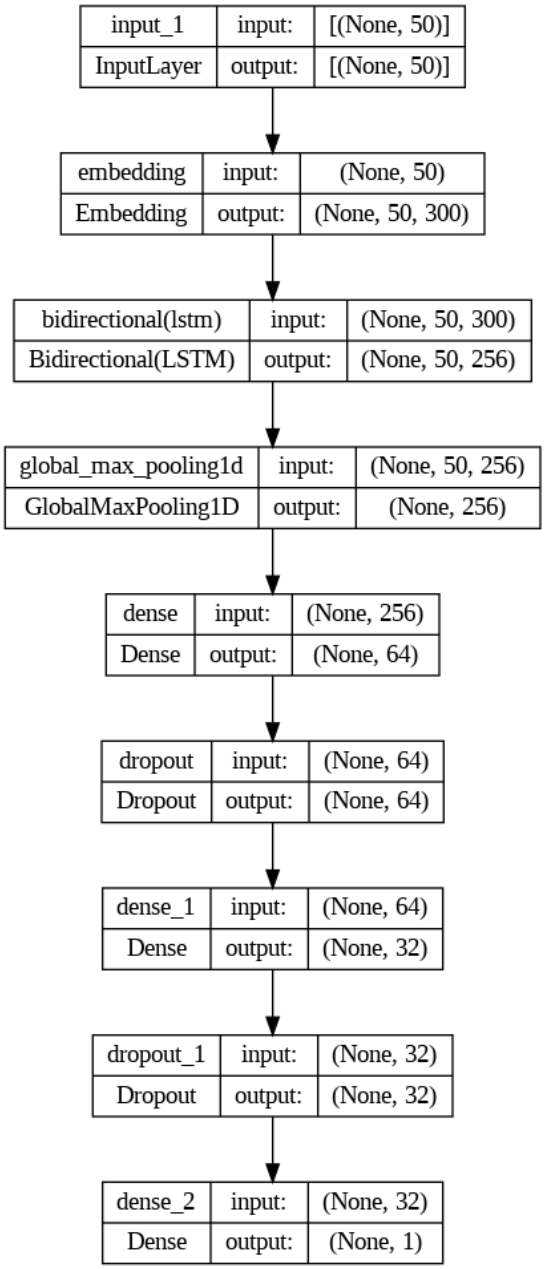

Fig. 2. Model

A. LSTM Model Predictions

After training the LSTM model, we utilized it to predict the probabilities of sarcasm for given sentences. Below are examples of sentences along with their predicted probabilities:

Sentence	Pred Prob of Sarcasm
Sun rises from the east	0.08
Oh, because that's obviously what I meant.	0.85
Sure, because that makes total sense.	0.99

TABLE I
EXAMPLES OF SENTENCES WITH THEIR PREDICTED PROBABILITIES OF SARCASM

From the probability analysis, we observe that the model tends to assign higher probabilities of sarcasm to sentences containing certain linguistic cues or patterns commonly associated with sarcasm. However, there are cases where the model exhibits uncertainty, as reflected in lower predicted probabilities.

B. Using ML

The table shows performance metrics for sarcasm detection on a Reddit English dataset using two machine learning models: Logistic Regression and Naive Bayes. Logistic Regression demonstrates a slightly higher recall of 72.22%, indicating that it is marginally better at correctly identifying sarcastic comments. On the other hand, Naive Bayes achieves a higher precision of 71.49%, which means it is more effective at correctly classifying comments as sarcastic when they are indeed sarcastic.

Model	Accuracy	Precision	Recall	F1
Logistic Regression	72.22	72.23	72.22	72.23
Naive Bayes	71.40	71.49	71.40	71.42

TABLE II
PERFORMANCE COMPARISON OF LOGISTIC REGRESSION AND NAIVE BAYES CLASSIFIERS

C. Sarcasm Detection Performance for Hindi tweets

This section presents the performance of the doc2vec and random forest classifier combination for sarcasm detection on Hindi tweets. We evaluated the model using two key metrics: Accuracy and Receiver Operating Characteristic (ROC) Area Under the Curve (AUC) score.

Metric	Score
Accuracy	0.9705
RoC AUC	0.9609

TABLE III
PERFORMANCE METRICS FOR SARCASM DETECTION

The model achieved a high accuracy score of 97.05%, indicating that it correctly classified a significant portion of the Hindi tweets as sarcastic or non-sarcastic. Additionally, the RoC AUC score of 0.9609 suggests a strong ability of the model to distinguish between sarcastic and non-sarcastic tweets. These results demonstrate the effectiveness of the doc2vec and random forest approach for sarcasm detection in Hindi text.

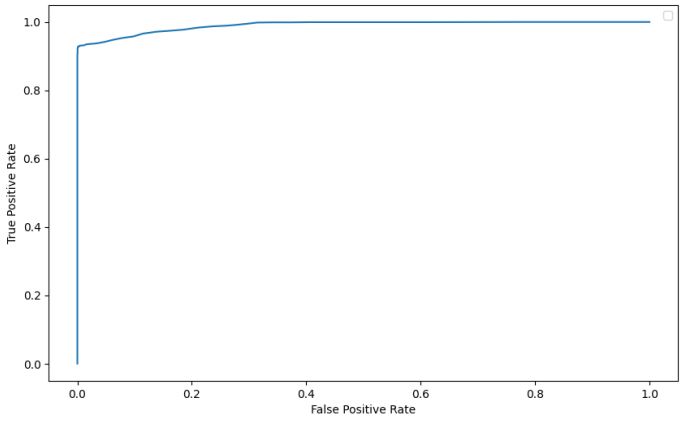

Fig. 3. Plot

VIII. CONCLUSION

In conclusion, this study delved into sarcasm detection across social media platforms and languages, employing distinct methodologies for English and Hindi text. For English, we harnessed the power of Long Short-Term Memory (LSTM) networks on Reddit data, capitalizing on their ability to capture sequential information crucial for understanding contextual nuances in sarcasm identification. On the other hand, for Hindi text sourced from Twitter, we utilized Doc2Vec embeddings in tandem with a Random Forest Classifier. This approach allowed us to capture the semantic meaning of Hindi text and effectively detect sarcasm in the social media discourse.

Through comparative analysis of the performance of LSTMs on English Reddit data and the Doc2Vec with Random Forest Classifier approach on Hindi Twitter data, we gained insights into the efficacy of different techniques for sarcasm detection across languages and social media platforms. Our findings contribute significantly to the advancement of robust sarcasm detection systems, offering valuable implications for researchers and practitioners interested in natural language understanding and sentiment analysis in multilingual and diverse social media contexts.

REFERENCES

[1] [Bharti et al.(2018)]bharti2018hybrid Bharti, S., Sharma, S., & Jain, A. K. (2018). A hybrid approach for sarcasm detection in twitter data using sentiment analysis and machine learning techniques. Procedia Computer Science, 132, 178-186.

[2] [Poria et al.(2019)]poria2019survey Poria, S., Gelbukh, A., Lopez, O., & Moreno, A. (2019). A survey on sarcasm detection using deep learning. Neural Networks, 113, 60-77.

[3] [Singh et al.(2023)]singh2023irony Singh, A., Singh, A., Sharma, L., Bali, A. (2023). Irony detection in social media text using deep learning. International Journal of Digital Development and Sustainability, 12(1), 14-26.

[4] [Singh and Shukla(2020)]singh2020sarcasm Singh, S., & Shukla, P. K. (2020). Sarcasm detection in hindi sentences using support vector machine. In 2020 International Conference on Innovative Trends in Information Technology (ICITI) (pp. 1-5). IEEE.

[5] [Santir et al.(2014)]santir2014sentiment Santir, S., Luo, J., Moore, C., Xiang, Y. (2014). Sentiment analysis of social network data for early crisis informatics. In 2014 IEEE International Conference on Big Data (Big Data) (pp. 93-100). IEEE.

[6] [Wullich and Qazmar(2020)]wullich2020findings Wullich, M., & Qazmar, Z. (2020). Findings of the WNUT-1.2 Workshop on Sarcasm Detection. In Proceedings of the Fourth Workshop on NLP for Conversational AI (WNUT-1.2) (pp. 1-6). Association for Computational Linguistics.

[7] @unpublishedSARC, authors=Mikhail Khodak and Nikunj Saunshi and Kiran Vodrahalli, title=A Large Self-Annotated Corpus for Sarcasm, url=https://arxiv.org/abs/1704.05579, year=2017

[8] Joshi, A., Sharma, V., Bhattacharyya, P. (2017). Harnessing context incongruity for sarcasm detection. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers) (pp. 757-762).

[9] Poria, S., Cambria, E., Gelbukh, A., Hussain, A. (2016). Sentiment analysis in Twitter using combined feature sets. In Proceedings of the 13th International Conference on Natural Language Processing (pp. 305-313).

[10] Ghosh, A., Veale, T. (2016). Fracking sarcasm using neural network. In Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis (pp. 161-169).

[11] Alqahtani, A., Alhenaki, L., Alsheddi, A. (2023). Text-based Sarcasm Detection on Social Networks: A Systematic Review.

[12] Misra, R., Arora, P. Sarcasm detection using news headlines dataset.