Aamina Atlaswala, 2024, 12:5
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Emoji-Enhanced Sarcasm Detection: A Comparitive Study of Logistic Regression and LSTM Models

Aamina Atlaswala, Dr. Rakhi Gupta, Nashrah Gowalker
Department of Information Technology.
KC College Mumbai, India

Abstract- Sarcasm is a form of language defined using words or phrases that express the opposite of their actual meaning, often with a purpose of mocking or criticizing something or someone. Sarcasm detection is crucial for mining, sentiment analysis, detecting cyberbullies, online trolls, and other similar activities. Detecting Sarcasm is a part of Sentimental Analysis. This research focuses on detecting sarcasm in text with emojis using the Logistic Regression Model, a traditional machine learning approach, and the Long Short- Term Memory (LSTM) Model a deep learning technique and provides a comparative study of models from the two branches of Artificial Intelligence. The emojis in the dataset were converted to their text descriptions to infer their actual usage in the text. And then the models were trained on the preprocessed data.

Keywords- Sarcasm, Emoji Processing, Long Short-Term Memory (LSTM) Model, Logistic Regression Model, Machine Learning, Deep Learning, Sentiment Analysis, Comparative Study

I. INTRODUCTION

The magnitude of data generated through social media today is colossal. They are good for data analysis since they are very personal [1]. Multiple social media platforms use this data to provide personalized feed and reviews. This personalize feed and reviews might not be 100% correct. This is because the sarcasm is not filtered. Sarcasm in text is an obstruction for less accuracy.

To solve it, in this research Logistic Regression Model and Long-Short-Term Memory Models are utilized to detect sarcasm along with emoji. Emojis play a nuanced role in digital communication and have a potential to convey sarcastic intent as they often offer non-explicit and sometimes ambiguous cues. This ambiguity has a potential to fuel hate-speech, trolling, or cyber-bullying under the guise of sarcasm[2]. The emojis are preprocessed before training the models and making the models do detection i.e. emoji are converted in to text.

Machine learning (ML) is a branch of artificial intelligence (AI) and computer science that focuses on the using data and algorithms to enable AI to imitate the way that humans learn, gradually improving its accuracy.

Logistic regression is a supervised machine learning algorithm used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not. Logistic regression is a statistical algorithm which analyze the relationship between two data factors. Long Short-Term Memory is an improved version of recurrent neural network designed by Hochreiter & Schmid Huber. LSTM is well-suited for sequence prediction tasks and excels in capturing long-term dependencies. Deep learning is a subset of machine learning that uses multi-layered neural networks, called deep neural networks, to simulate the complex decision-making power of the human brain. Therefore, in this paper, we study the novel problem of exploiting emojis for sarcasm detection on social media[5].

© 2024 Aamina Atlaswala. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly credited.

II. METHODOLOGY

In this section, we outline our proposed methodology, which focuses on utilizing emoji data to address the challenge of multimodal sarcasm detection within a multitask framework[3].
The methodology followed in doing this research has the following steps.

1. Data Collection
First step was to collect data. For which a dataset of tweets from GitHub is used as a secondary data. This was a labelled dataset. That was downloaded from GitHub in csv file format.

2. Data Pre-Processing
Second step is data pre-processing in which data is cleaned and prepared for the models to be trained on. Data preprocessing is a foundational step in readying datasets for various data-driven tasks, including both machine learning and deep learning. This crucial phase involves a series of operations designed to clean, structure, and refine raw data, ensuring its suitability for subsequent modelling stages. Whether applying traditional machine learning algorithms or deep neural networks, preprocessing tasks remain fundamental. Common operations include handling missing values, removing duplicates, scaling numerical features, and encoding categorical variables. For text data, tasks like tokenization, lowercasing, and removing stop words are commonplace. The effectiveness of machine learning and deep learning models is intricately tied to the quality of the pre-processed data, emphasizing the critical role of preprocessing in extracting meaningful patterns and insights from diverse datasets.

Following are the preprocessing steps applied to refine and prepare the dataset for subsequent modelling stages:

Removing words "sarcasm", "sarcastic", "sarcastically": Utilizes basic string manipulation in Python without the need for external libraries. This step ensures the explicit removal of terms that might bias sentiment analysis.

Removing '#' tags only keeping the word ahead of it: Achieved through Python's string manipulation. The '#' symbol and subsequent characters are discarded, retaining only the relevant word. No external libraries are required.

Removing '@' tags with the name ahead of '@': Employing Python string operations, this step eliminates Twitter handles (@username) from the tweet, maintaining the remaining text. External libraries are unnecessary for this task.

Replacing emojis with their descriptions: Python's string handling, possibly aided by the 'emoji' library, transforms emojis into text descriptions for standardization. Ensures consistent representation of emoticons.

Handling numbers: Standard Python string manipulation is applied, removing or normalizing numerical characters in the tweet. No external libraries are needed for this numerical processing.

Handling non-English alphabets: Leverages Python string methods to either remove or replace non-English characters. External libraries are not employed for this task.

Handling symbols: Python's string manipulation addresses symbols, including punctuation and special characters, ensuring a clean text for analysis. No external libraries are required.

Handling unnecessary spaces: Basic Python string operations are used to clean up unnecessary whitespace, enhancing tweet readability and consistency without relying on external libraries.

Counting repeated letters in each word to create a 'repetition' column: Python's string manipulation counts repeated letters within words. This step generates a 'repetition' feature. No external libraries are necessary.

Counting capital letters in each word to create a 'capital' column: Utilizes Python's string methods to count capital letters within words, creating a 'capital' feature without reliance on external libraries.

Calculating sentiment scores for each word using SentiWordNet: Involves leveraging the 'nltk' library and the SentiWordNet lexical resource in Python. Assigns sentiment scores to words for a nuanced understanding of sentiment. Loading and Parsing: The initial step involves loading the SentiWordNet file into the NLP environment. Subsequently, the content is parsed meticulously to extract pertinent information crucial for sentiment analysis. Mapping to Text

Data: Every entry in the SentiWordNet file corresponds to a synset (POS,ID) and is accompanied by associated positivity (PosScore), negativity (NegScore), and objectivity (ObjScore) scores. Simultaneously, the text data, which may consist of reviews or tweets, undergoes tokenization into words or phrases.

Mapping Words to Synsets: During the preprocessing phase, words or phrases extracted from the text data are meticulously mapped to their corresponding synsets in the SentiWordNet file. This mapping entails matching words to their WordNet IDs and associating them with the appropriate part of speech (POS).

Calculating Sentiment Scores: Utilizing the positivity, negativity, and objectivity scores from SentiWordNet, sentiment scores are systematically computed for each word or phrase in the text. These sentiment scores play a pivotal role in comprehending the overall sentiment of the text, where positive and negative scores signify the intensity of sentiment.

Aggregation and Normalization: Individual sentiment scores attributed to words or phrases are often aggregated to derive an encompassing sentiment score for a sentence or document. Moreover, these scores may undergo normalization to ensure consistency and comparability across diverse texts.

Threshold Determination: A critical aspect involves setting a threshold based on the sentiment analysis results. This threshold aids in classifying the overall sentiment of the text, categorizing it as positive, negative, or neutral.

Integration into NLP Models: The processed sentiment scores seamlessly integrate into broader NLP models. This integration proves beneficial for various tasks, including sentiment classification, opinion mining, and emotion analysis.

SentiWordNet: SentiWordNet employs a comprehensive scoring system to assign sentiment scores to words based on their positivity, negativity, and neutrality. These scores span a range from -1 (most negative) to 1 (most positive), with 0 indicating neutrality. The breakdown of scores is as follows:

Positivity Score (PosScore)

- Represents the positive sentiment intensity of a word.
- Range: 0 to 1 (higher values indicate greater positivity).

Negativity Score (NegScore)

- Represents the negative sentiment intensity of a word.
- Range: 0 to 1 (higher values indicate greater negativity).

Objectivity Score (ObjScore)

- Represents the neutrality or objectivity of a word.
- Range: 0 to 1 (higher values indicate greater neutrality).

These nuanced sentiment scores facilitate a detailed understanding of a word's sentiment, contributing to more precise sentiment analysis in natural language processing applications.

Identifying positive and negative words based on sentiment scores: Building upon the 'nltk' library, this step determines the sentiment of words in the tweet, categorizing them as positive or negative based on their sentiment scores.

Tokenizing the tweet: Utilizes the 'nltk' library in Python to tokenize the tweet, breaking it into individual words or tokens for subsequent analysis.

3. Model Training

There is significant future scope for this research. If the limitations outlined in this paper, particularly time and economic constraints, can be addressed, utilizing the paid Twitter API to access real-time tweet data would be a highly beneficial next step.

Real-time data would allow for more accurate and up-to-date analysis of sarcasm in tweets, enhancing the relevance and applicability of the model. This would improve the system's responsiveness to current trends in language, emoji usage, and the evolving nature of sarcasm on social media.

Another promising direction for future research is the incorporation of context-based datasets. Sarcasm often relies heavily on the context in which it is expressed, and building models that are aware of the broader conversational or situational context could significantly improve sarcasm detection.

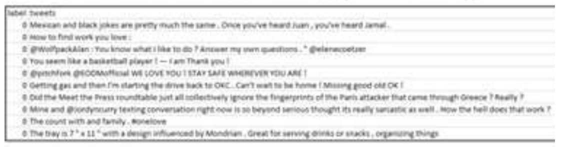
Developing datasets that provide contextual information, rather than analyzing isolated tweets or short text segments, would help models to better interpret sarcasm.

Furthermore, expanding the scope to include visual memes and videos offers another potential area for future exploration. Sarcasm is not limited to textual expression; it often appears in multimedia formats.

By utilizing datasets made up of images, videos, and memes, and integrating context-aware models, researchers could create more comprehensive sarcasm detection algorithms. These algorithms would be better equipped to detect sarcasm in multimedia formats where meaning is conveyed through a combination of visual and textual elements.

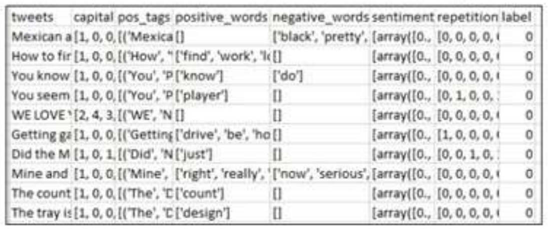
4. Data Visuals

GitHub Dataset Before Preprocessing (trainemo.csv):



label	tweets
0	Mexican and black jokes are pretty much the same . Once you've heard Juan , you've heard Jamal .
0	How to find work you love :
0	@WolfpackAlan : You know what I like to do ? Answer my own questions . " @elenecoetzer
0	You seem like a basketball player ! — I am Thank you !
0	@pitchfork @6ODMofficial WE LOVE YOU ! STAY SAFE WHEREVER YOU ARE !
0	Getting gas and then I'm starting the drive back to OKC . Can't wait to be home ! Missing good old OK !
0	Did the Meet the Press roundtable just all collectively ignore the fingerprints of the Paris attacker that came through Greece ? Really ?
0	Mine and @jordyncurry texting conversation right now is so beyond serious thought its really sarcastic as well . How the hell does that work ?
0	The count with and family . #onelove
0	The tray is 7 " x 11 " with a design influenced by Mondrian . Great for serving drinks or snacks , organizing things

GitHub Dataset After Preprocessing (trainemo.csv):



tweets	capital	pos_tags	positive_words	negative_words	sentiment	repetition	label
Mexican a	[1, 0, 0,	[('Mexica	[]	['black', 'pretty',	[array([0.,	[0, 0, 0, 0,	0
How to fir	[1, 0, 0,	[('How', '	['find', 'work', 'l	[]	[array([0.,	[0, 0, 0, 0,	0
You know	[1, 0, 0,	[('You', 'P	['know']	['do']	[array([0.,	[0, 0, 0, 0,	0
You seem	[1, 0, 0,	[('You', 'P	['player']	[]	[array([0.,	[0, 1, 0, 0,	0
WE LOVE	[2, 4, 3,	[('WE', 'N	[]	[]	[array([0.,	[0, 0, 0, 0,	0
Getting ga	[1, 0, 0,	[('Getting	['drive', 'be', 'ho	[]	[array([0.,	[1, 0, 0, 0,	0
Did the M	[1, 0, 1,	[('Did', 'N	['just']	[]	[array([0.,	[0, 0, 1, 0,	0
Mine and	[1, 0, 0,	[('Mine',	['right', 'really', '	['now', 'serious',	[array([0.,	[0, 0, 0, 0,	0
The count	[1, 0, 0,	[('The', 'D	['count']	[]	[array([0.,	[0, 0, 0, 0,	0
The tray is	[1, 0, 0,	[('The', 'D	['design']	[]	[array([0.,	[0, 0, 0, 0,	0

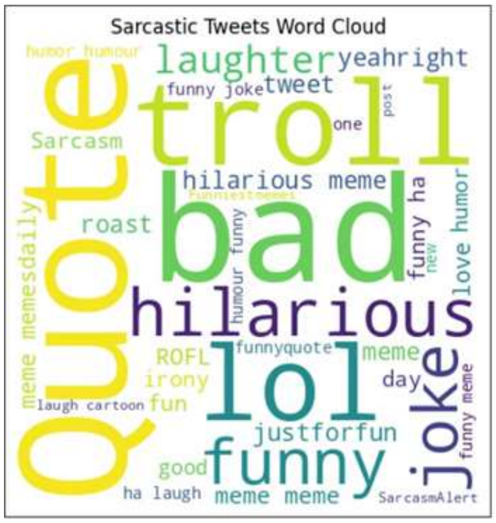


Fig 1: Non-Sarcastic Tweets Word Cloud from GitHub Dataset

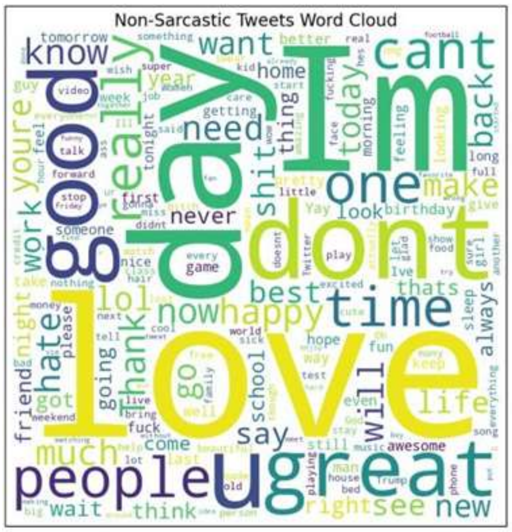


Fig: Non-Sarcastic Tweets Word Cloud from GitHub Dataset

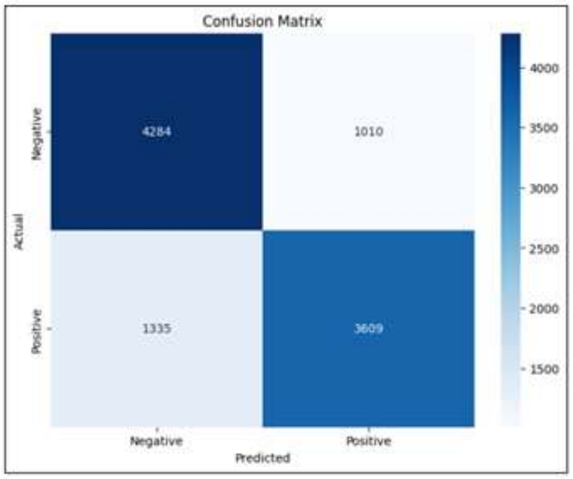


Fig: **Confusion Matrix of LSTM Model Trained with GitHub Dataset**

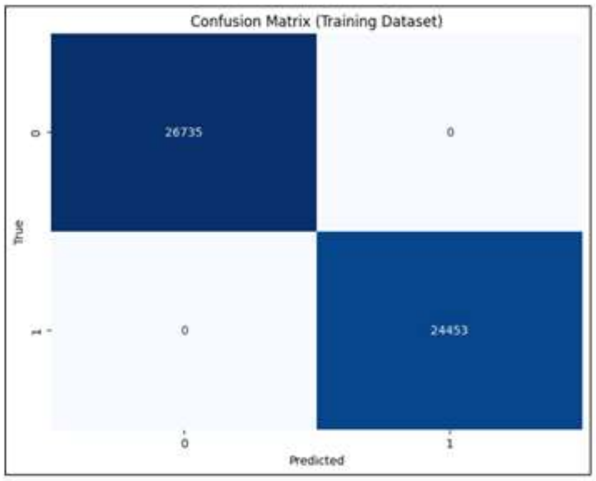


Fig: **Confusion Matrix of Logistic Regression Model Trained with GitHub Dataset (Training csv file)**

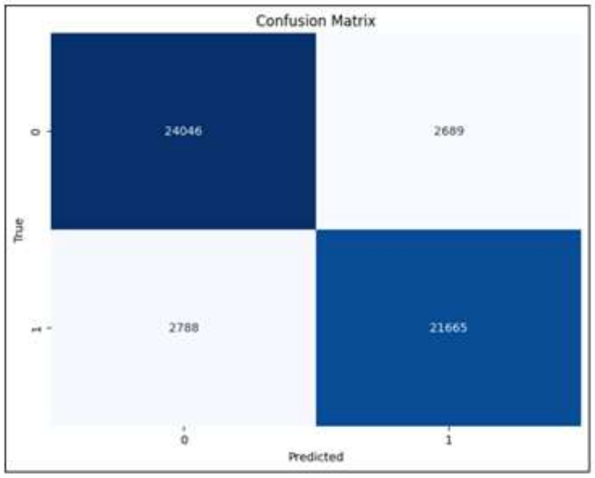
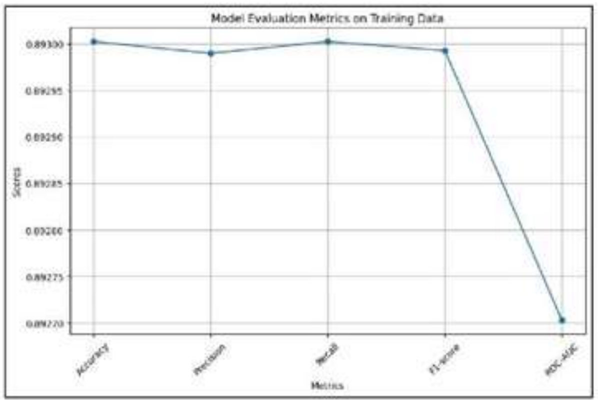


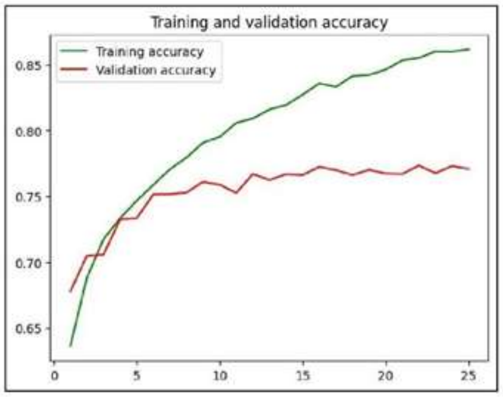
Fig: **Confusion Matrix of Logistic Regression Model Trained with GitHub Dataset (Testing csv file)**

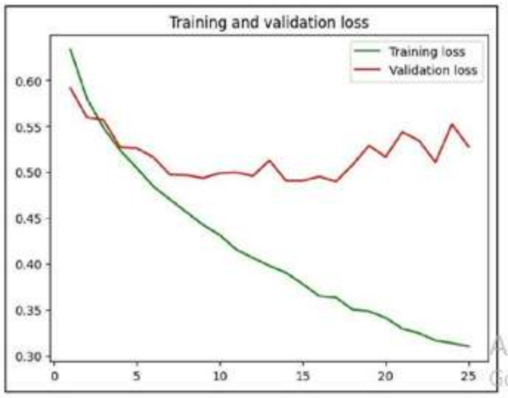
Logistic Regression Model Evaluation Metrics:



Branch of AI	Models	Accuracy
Machine Learning	Logistic Regression	0.754784063578379
Deep Learning	LSTM	0.8930022661561303

Training and Validation Accuracy and Loss Curve for LST Model on GitHub Dataset:

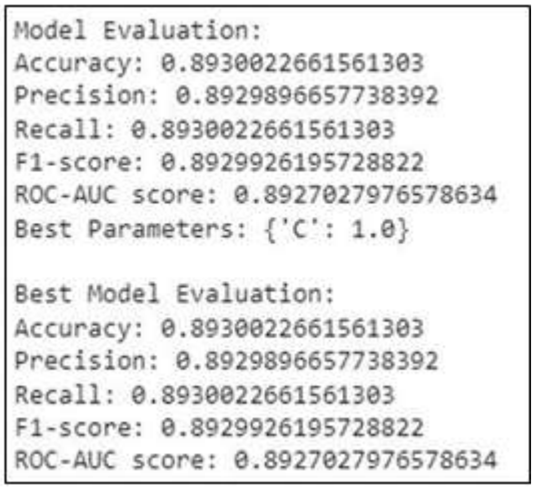




III. RESULTS

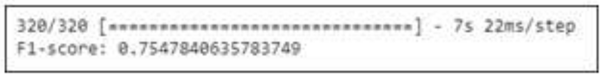
The accuracy of Logistic Regression Model is 0.754784063578379 and of Long-Short-Term Memory is 0.8930022661561303. This result shows that the Machine Learning Model is more accurate and optimal than deep Learning Model

This has proved the hypothesis of this research false, as it was assumed that Deep Leaning Model will be more optimal.



```
Model Evaluation:
Accuracy: 0.8930022661561303
Precision: 0.8929896657738392
Recall: 0.8930022661561303
F1-score: 0.8929926195728822
ROC-AUC score: 0.8927027976578634
Best Parameters: {'C': 1.0}

Best Model Evaluation:
Accuracy: 0.8930022661561303
Precision: 0.8929896657738392
Recall: 0.8930022661561303
F1-score: 0.8929926195728822
ROC-AUC score: 0.8927027976578634
```



```
320/320 [==============================] - 7s 22ms/step
F1-score: 0.7547840635783749
```

IV. CONCLUSION

Emojis provide a new dimension to social media communication. We study the role of emojis for sarcasm detection on social media. We propose a new deep learning model by introducing an attention layer which helps to model the text and emojis simultaneously for sarcasm detection. The empirical results on real-world datasets demonstrate the effectiveness of the proposed framework[5].

This comparative study highlights the significance of selecting appropriate Machine Learning models for the task, with the Logistic Regression Model proving to be the most optimal for detecting sarcasm in text that includes emojis. The model's performance was notably enhanced by effective preprocessing techniques, which played a crucial role in improving the accuracy of sarcasm detection.

In particular, the preprocessing of both text and emojis was critical to achieving these optimal results. Handling emojis correctly, alongside refining the textual data, allowed the model to better capture nuanced expressions of sarcasm, which often rely on the combined meaning of text and visual symbols. This careful preprocessing contributed to the overall success of the Logistic Regression Model in this context.

Limitations
In this research, we encountered several limitations that impacted our methodology and the overall scope of the study. One of the primary constraints was time. The limited timeframe restricted our ability to conduct more extensive data collection and in-depth analysis. Certain aspects of the research that required a longer duration, such as longitudinal studies or repeated measures, had to be scaled back or adjusted. This may have reduced the breadth of insights we could have gained, particularly when trying to capture evolving trends or behaviors over time.

Additionally, economic limitations posed a significant challenge. Accessing real-time tweet data through the Twitter API, which was integral to our research, became prohibitively expensive due to high pricing tiers. As a result, we were unable to collect live data directly from the platform, forcing us to rely on alternative methods such as historical datasets or third-party sources. This limitation affected the freshness and relevance of the data, potentially influencing the accuracy of our findings and limiting our ability to make real-time predictions or analyses.

Future Scope
There are a lot of future scope on the research. If the limitations mentioned in this paper are not a hurdle, then utilizing paid Tweeter API and getting real-time tweets data will be optimal decision.

One more future scope on this research is to use context- based dataset in which that models are made aware of the context for sarcasm.

Another future scope on this research topic is to use visual meme and videos and then utilizing a dataset made up of videos and picture and forming a context-based data set would result in a more purposeful sarcasm detection algorithm.

REFERENCES

1. A. Ghosh and T. Veale, "Magnets for sarcasm: Making sarcasm detection timely contextual and very personal", Proc. Conf. Empirical Methods Natural Lang. Process., pp. 482-491, 2017. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp =&arnumber=9420094
2. An attention approach to emoji focused sarcasm detection Vandita Grovera and Hema Banatib https://pubmed.ncbi.nlm.nih.gov/39286068/
3. An emoji-aware multitask framework for multimodal sarcasm detection https://www.sciencedirect.com/science/article/abs/pii/S095070512201 0176
4. Exploiting Emojis for Sarcasm Detection Jayashree Subramanian* , Varun Sridharan* , Kai Shu and Huan Liu Computer Science and Engineering, Arizona State University, Tempe, AZ, USA https://www.cs.emory.edu/~kshu5/files/sbp_2019_usarcasm.pdf
5. Exploiting Emojis for Sarcasm Detection DOI:10.1007/978-3-030-21741-9_8 https://www.researchgate.net/publication/333831290_Exploiting_Em ojis_for_Sarcasm_Detection
6. GitHub Testing Dataset https://raw.githubusercontent.com/muhammadadyl/SarcasmDetection/master/data/train.tsv
7. GitHub Training Dataset https://github.com/muhammadadyl/SarcasmDetection/blob/master/dat a/test.txt