|  |  |
| --- | --- |
| Hate speech Detection  Natural Language Processing | Yash Patkar  MAIB JAN 23 AJ23SYD006 |

Table of Contents

[Introduction 1](#_Toc152091133)

[Literature review 1](#_Toc152091134)

[Data Collection and Pre-processing 2](#_Toc152091135)

[Models 3](#_Toc152091136)

[Logistic Regression 3](#_Toc152091137)

[Decision Tree 4](#_Toc152091138)

[Random Forest 4](#_Toc152091139)

[AdaBoost (Adaptive Boosting) 5](#_Toc152091140)

[Support Vector Machine (SVM) 5](#_Toc152091141)

[Naive Bayes 6](#_Toc152091142)

[Neural Network (LSTM) 6](#_Toc152091143)

[Conclusion 7](#_Toc152091144)

# Introduction

Hate speech, defined as any communication that discriminates, stigmatizes, or incites violence against individuals or groups based on attributes such as race, religion, ethnicity, gender, or sexual orientation, poses a significant and pervasive challenge in today's digital age. With the rise of social media and online platforms, hate speech has found new avenues to spread and amplify, impacting individuals and communities globally. The consequences of hate speech can be severe, contributing to social division, discrimination, and even violence. Addressing this issue requires innovative solutions that harness technology to identify and mitigate the harmful effects of hate speech in online spaces.

# Literature review

The literature on hate speech detection commonly employs machine learning algorithms such as Support Vector Machines (SVM), Naive Bayes (NB), Logistic Regression (LR), Decision Trees (DTs), and K-Nearest Neighbour (KNN). LR consistently outperforms others, but performance varies based on the dataset. Most algorithms rely on n-grams, POS tags, and sentiment for hate speech detection, with some incorporating lexicons. However, contradictory conclusions arise from diverse datasets, and concerns exist about binary classification without considering word context.

Deep learning methods, particularly Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM), show promise. Studies report high accuracy with CNN and LSTM, often addressing binary hate detection. Challenges include limited datasets affecting pre-trained BERT models and difficulties in detecting hate speech. Hybrid approaches, combining traditional and deep learning, demonstrate improved performance, but generalization is hindered without large datasets.

Dataset challenges stem from varied features and imbalanced class distribution in hate speech. Traditional methods, e.g., SVM and NB, can outperform deep learning based on notability and features. Discrepancies exist, as seen in SVM outperforming Hierarchical Attention Network (HAN) in certain cases. Features related to user personalities and linguistic characteristics play roles in hate speech detection. Dataset issues include unbiased manual annotation and conflicts in keyword-based automated methods (Fatimah Alkomah, 2022).

The authors of the second paper propose a novel approach to hate speech classification by leveraging a multimodal deep learning architecture that combines semantic and emotion features extracted from speech. To address the absence of a dedicated hate speech video dataset, they manually curate the Hate Speech Detection Video Dataset (HSDVD). Due to the limited size of this dataset, transfer learning is employed to pre-train models responsible for capturing unimodal language and speech embedding.

Three distinct machine learning models are introduced in their methodology. The first model focuses on detecting hate speech in text and utilizes pre-trained transformer networks, such as BERT and ALBERT, trained on existing Twitter datasets. The second model, termed the Emotion Detection Multi-Task Learning (MTL) model, is trained on the IEMOCAP dataset to identify valence, arousal, and dominance levels in audio. Notably, a dimensional representation of emotions, defined by valence, arousal, and dominance attributes, is employed to address the challenge of discrete representation of complex emotions.

The third model, designed for Multimodal Learning (MML), employs a multilayer perceptron trained on HSDVD to specifically detect hate speech. This model selects the best-performing text and emotion models to generate embeddings as input. Additionally, the text-based models are fine-tuned and evaluated on HSDVD, creating a benchmark for comparison with the MML framework. Both models are rigorously tested on a holdout dataset from HSDVD, and two baselines are employed for experimental purposes.

Results indicate that the Multimodal Learning (MML) framework outperforms its respective baselines, demonstrating gains in precision and recall. This outcome substantiates the authors' hypothesis regarding the importance of incorporating multiple modalities for effective hate speech detection (Aneri Rana, 2022).

The third paper discusses the use of multiple languages for detecting hate speech. Urdu, spoken by over 300 million people globally, presents unique challenges in Natural Language Processing (NLP) due to its complex writing style, right-to-left orientation, context-sensitivity, diacritics, and absence of word capitalization. For hate speech detection in Urdu, challenges include the contextual usage of common words and the difficulty in tokenization, which is essential for accurate model understanding.

Contributions of this research include the creation of a hate base lexicon for Urdu, a carefully crafted dataset of 10,526 tweets for hate speech detection, and the exploration of various pre-trained embedding models. The study employs BERT in its vanilla form and introduces a BERT-CNN architecture, alongside other models like distil-BERT, FastText, FastText + BiGRU, and multi-lingual models such as XLM-Roberta. Results show that multi-lingual models outperform others.

The paper is organized with a literature review, dataset pipeline details, an extensive experimental setup, a proposed approach, and a formal solution definition. Results and discussions, an ablation study on different BERT forms, error analysis, and conclusions are presented in subsequent sections.

In summary, the research contributes to addressing the pressing issue of hate speech detection in low-resource languages like Urdu, shedding light on the importance of language diversity in technological advancements and emphasizing the need for automated mechanisms to curb harmful online behaviors (Raza Ali, 2022).

# Data Collection and Pre-processing

The data is collected from Kaggle (SAMOSHYN, 2020), this data has been labelled as offensive, hate and none of the above. The code snippet performs several preprocessing steps on a dataset, primarily focusing on cleaning and lemmatizing text data. It appears to be intended for text data related to tweets, as it includes steps such as removing URLs, HTML tags, special characters, punctuation, and stopwords. Additionally, it maps numerical class labels to more human-readable categories.

Firstly, the code adds a new column "labels" to the dataframe `data` based on the existing "class" column. The "class" column seems to represent numerical labels, where 0 corresponds to "Hate Speech," 1 corresponds to "Offensive Language," and 2 corresponds to "No Hate and Offensive." The mapping is done using the `map` function, creating a new column "labels" with human-readable class labels.

The `clean` function defined in the code is applied to the "tweet" column of the dataframe. This function is responsible for text cleaning, converting the text to lowercase, removing URLs, HTML tags, special characters, mentions (@), hashtags (#), non-alphanumeric characters, newlines, and digits. It also tokenizes the text into words using the `word\_tokenize` function from the Natural Language Toolkit (nltk) library. Furthermore, it filters out stopwords, which are common words that do not carry significant meaning in the context of natural language processing. The cleaned and tokenized tweets are then joined back into sentences and stored in the "tweet" column of the dataframe. The resulting dataframe is assigned to the variable `tweetData`.

Subsequently, the code defines a lemmatization function named `lemmatizing`. However, there appears to be an issue with the implementation of lemmatization. The lemmatization function currently does not modify the data and returns the input unchanged. The correct implementation should apply lemmatization to each word in the text data. Finally, the code attempts to lemmatize the "tweet" column in the `tweetData` dataframe using the defined lemmatization function. However, due to the issue in the lemmatization function, it does not effectively lemmatize the text.

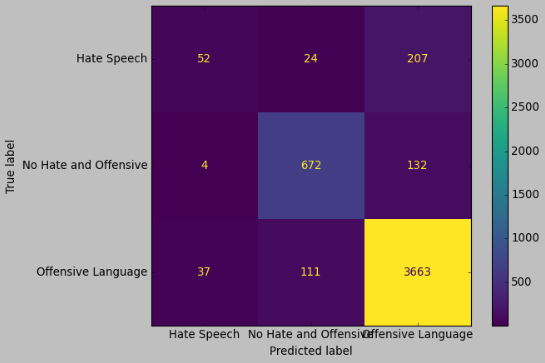
In summary, the code processes a dataset containing tweets by mapping numerical class labels to human-readable labels, cleaning the text data by removing various elements, and attempting to lemmatize the processed text, although there is a bug in the lemmatization function. The data is still skewed after cleaning and filtering so the code employees the SMOTE method to reduce the bias before putting it through the process of training.

# Models

Multiple models were used on the cleaned and processed data to get the best accuracy. The use of transfer learning did not take place given the compatibility of the data and the large computational time taken in doing so. The results of each model along with the advantage and disadvantage is given below.

## Logistic Regression

Logistic Regression is a linear classification algorithm that works well for binary and multiclass classification problems. In this code, it's used for binary classification. Logistic Regression models the probability that an instance belongs to a particular class. It's computationally efficient, interpretable, and does not require complex hyperparameter tuning. The regularization parameter ('C') and solver choice are optimized using grid search. Despite its simplicity, Logistic Regression can perform well when the relationship between features and target is approximately linear.



## Decision Tree

The Decision Tree classifier is a non-linear model that learns decision rules based on the features of the dataset. It is capable of handling both numerical and categorical data and can capture complex relationships. Decision Trees are interpretable, making it easy to understand the logic behind the predictions. However, they are prone to overfitting, and the performance can vary depending on the dataset's characteristics.

A screenshot of a computer screen

Description automatically generated

## Random Forest

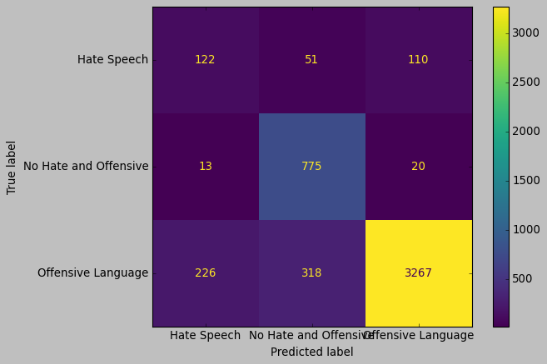
Random Forest is an ensemble method that builds multiple decision trees and merges their predictions to improve accuracy and reduce overfitting. It combines the simplicity of decision trees with the power of ensemble learning. Random Forest is robust, handles high-dimensional data well, and is less prone to overfitting compared to individual decision trees. It's effective for capturing complex relationships and achieving high predictive accuracy.

A screenshot of a computer screen

Description automatically generated

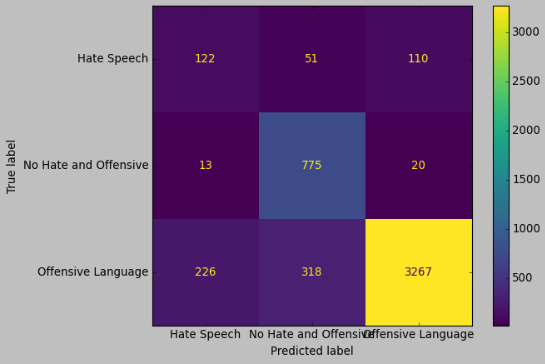
## AdaBoost (Adaptive Boosting)

AdaBoost is an ensemble method that combines multiple weak classifiers to create a strong classifier. It assigns weights to misclassified instances, allowing subsequent weak classifiers to focus more on them. AdaBoost adapts over iterations, giving more emphasis to difficult-to-classify instances. It performs well in practice, is less prone to overfitting, and is effective even with weak classifiers.



## Support Vector Machine (SVM)

SVM is a powerful linear and non-linear classifier that works well in high-dimensional spaces. It aims to find the hyperplane that best separates classes while maximizing the margin between them. In this code, a linear kernel is used. SVMs are effective in capturing complex decision boundaries and are less sensitive to outliers. However, they can be computationally expensive, especially with large datasets.



## Naive Bayes

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It assumes that features are conditionally independent given the class label, hence the term "naive." It's computationally efficient, easy to implement, and works well with high-dimensional data. In this code, Gaussian Naive Bayes is used, suitable for continuous features. It's particularly useful for text classification tasks, like spam detection.

A chart with numbers and a yellow square

Description automatically generated

## Neural Network (LSTM)

The LSTM (Long Short-Term Memory) neural network is a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data. LSTMs excel in processing and predicting sequences, making them suitable for tasks like natural language processing. The model defined in the code includes an embedding layer for word representations, LSTM layers for sequence processing, and a dense layer for classification. Neural networks, especially LSTMs, can capture intricate patterns in data but may require more computational resources and data compared to traditional models.

A graph with red and blue lines

Description automatically generatedA graph of training accuracy

Description automatically generated

Each model has its advantages, and the choice depends on the characteristics of the data and the specific requirements of the classification task. At the end we get the best accuracy from the Logistic Regression model based on the table below.

**Model Accuracy**

1 Logistic Regression 0.889841

2 Logistic Regression Tuned 0.894941

3 Decision Tree Classifier 0.866585

4 Random Forest Classifier 0.871073

5 AdaBoostClassifier 0.849449

6 SVC 0.849449

7 Naive Bayes 0.563446

8 LSTM 0.774337

# Conclusion

In conclusion, our exploration into hate speech classification has delved into the urgent and challenging realm of addressing harmful online behaviors. Hate speech, with its potential to contribute to social division and violence, requires innovative technological solutions. Through a comprehensive literature review, we gained insights into various machine learning and deep learning approaches employed in hate speech detection.

The literature revealed the prevalence of machine learning algorithms such as Logistic Regression, Decision Trees, Random Forest, AdaBoost, Support Vector Machines, Naive Bayes, and deep learning techniques like LSTM in hate speech classification. Notably, the Multimodal Learning (MML) framework showcased the significance of combining text and emotion features for enhanced hate speech detection. Additionally, our exploration extended to addressing hate speech in low-resource languages like Urdu, emphasizing the importance of language diversity in technological advancements.

Our data collection and pre-processing phase involved sourcing a labeled dataset from Kaggle and implementing meticulous text cleaning and lemmatization. Despite encountering challenges related to dataset biases and skewed class distributions, we mitigated these issues through the Synthetic Minority Over-sampling Technique (SMOTE) to improve model training.

For model development, we explored a diverse set of classifiers, from traditional models like Logistic Regression and Decision Trees to ensemble methods such as Random Forest and AdaBoost, as well as Support Vector Machines, Naive Bayes, and deep learning with LSTM. Each model brought its unique strengths and considerations to the table.

Ultimately, the Logistic Regression model emerged as the most effective, achieving the highest accuracy in hate speech classification. The comprehensive evaluation of each model's performance, including precision, recall, F1-score, and accuracy, provided a nuanced understanding of their capabilities and limitations.

This project not only contributes to the ongoing discourse on hate speech detection but also emphasizes the need for a nuanced and context-aware approach. Language, cultural diversity, and the evolving nature of online interactions necessitate ongoing research and adaptive methodologies to effectively curb hate speech. The findings of this project offer a stepping stone toward developing robust and reliable hate speech detection mechanisms, ultimately fostering a safer and more inclusive online environment.

# References

Aneri Rana, S. J. (2022). *Emotion Based Hate Speech Detection using Multimodel Learning .* Cornell University .

Fatimah Alkomah, X. M. (2022). *A Literature Review of Textual Hate Speech Detection Methods.* MDPI.

Raza Ali, U. F. (2022). *Hate speech detection on Twitter using transfer learning.* ScienceDirect.

SAMOSHYN, A. (2020). *Hate Speech and Offensive Language Dataset*. Retrieved from Kaggle: https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset