
# Multilingual Hate Speech and Offensive Language Detection Using Natural Language Processing

Submitted September 2024, in partial fulfillment of
the conditions for the award of the degree **MSc Data Science.**

**Rohith Ganesan**
**20553375**

**Supervised by Dr. Milena Radenkovic**

School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature 

Date 20 / 09 / 2024

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the University of Nottingham's e-dissertation archive.

# Abstract

Detection of hate speech and offensive language is one of the major tasks in the sphere of Natural Language Processing (NLP), especially in terms of the multilingual usage of data on various platforms such as social networks. In this thesis, I focus on detecting hate speech and abusive language by analyzing datasets in Tamil, English, German, and Arabic, using various Machine Learning (ML) and Deep Learning (DL) models. The research applies different learners, including the traditional ML methods, such as Support Vector Machines, Logistic Regression, Random Forests, K-Nearest Neighbors, and Multilayer Perceptrons. As well, more sophisticated DL models are considered, including RNN and Bi-LSTM, as well as the cutting-edge Transfer Learning model BERT.

The outline initiated with considerable preparation such as text cleaning, and tokenization to advance model training stopword removal , stemming , lemmatization, and TF-IDF Vectorization. All the four datasets were used to train each model and performance was evaluated in terms of accuracy, precision, recall and F1 score. As it was speculated, their initial speculation was turned on its head and DL and TL models were often dethroned by traditional ML models, particularly with smaller datasets like Tamil and Arabic. This result indicates that, under constraints, such traditional models built upon hand coded features could yield less damaging if not satisfying results even with smaller data.

Besides, the deep and transfer learning models were resource-hungry and computationally intensive which required a longer training period. Also, data augmentation techniques were used to overcome the constraints of smaller datasets especially in the case of low-resource languages. This thesis emphasizes the contribution that conventional ML approaches can make in low-resource environments but also shows the directions needed in the future to make the deep learning models more efficient when large datasets and better computation facilities are available.

## Acknowledgements

I want to express my respect and gratitude to my supervisor, Dr. Milena Radenkovic, who was a great support, providing many helpful tips throughout the period of my research. Her knowledge and motivation played a critical role in the development and outcome of this project. Also, I would like to sincerely thank my parents for their support throughout my postgraduate studies. Also, my thanks goes to my brother Roshen Ganesan who provided useful comments on the project.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

With the rapid proliferation of digital communication and social media platforms, the ease of global connectivity has significantly transformed how individuals interact and share information. While this technological advancement has numerous benefits, it also presents challenges, notably the rampant spread of hate speech and offensive language. Hate speech, characterized by derogatory or discriminatory remarks targeting individuals or groups based on attributes such as race, religion, ethnicity, gender or sexual orientation poses severe threats to societal harmony and individual well-being. Social media platforms like Twitter, Facebook and Instagram have become breeding grounds for such toxic content, which can spread rapidly causing substantial harm. This necessitates the development of robust, automated detection systems capable of identifying and mitigating hate speech and offensive language to maintain a safer, more inclusive digital environment.

The history of hate speech detection dates back to the early 2000s when the rise of the internet and online forums highlighted the need for monitoring and controlling offensive content. Initial efforts relied heavily on manual moderation and rule-based approaches, which were limited in scope and scalability. As the volume of online content grew, so did the need for more efficient methods. The introduction of machine learning in the mid-2010s marked a significant advancement in hate speech detection, enabling the development of models that could learn from data and make predictions about new, unseen content. Early models primarily focused on English language content, but the increasing diversity of online users soon highlighted the need for multilingual capabilities.

Support Vector Machines (SVM) and Logistic Regression are examples of traditional machine learning models which use engineered features and work quite well to some extent. It is, however, common for these models to miss some contextual underspecified subtleties related to hate speech. Over the past couple of years, Deep Learning approaches, in particular transfer-learning based models such as BERT (Bidirectional Encoder Representations from Transformers) appear to be more sophisticated in capturing context changes by using the structure of the pre-trained model.

Within this research, machine learning (ML), deep learning (DL) and transfer learning techniques have been used to conduct a systematic review of hatred and offensive speech identification and performance evaluation in numerous languages in order to see which are the most successful in identifying hate and/ or offensive speech. The study seeks to emphasize the best ways for multilingual hate speech detection by exploring the possibilities of various approaches.

## 1.1 Motivation

The motivation behind this research stems from the urgent need to address the negative impact of hate speech and offensive language on social media. Hate speech can lead to real-world violence, discrimination and psychological harm to targeted individuals or groups. By developing effective detection mechanisms, these adverse effects can be reduced and a more respectful online discourse can be fostered. Moreover, governments and regulatory bodies worldwide are increasingly imposing stringent regulations on social media platforms to control the spread of hate speech. Automated detection systems can help these platforms comply with legal requirements and uphold ethical standards of communication. Manual moderation of social media content is labor-intensive, time-consuming, and often infeasible given the sheer volume of data generated daily. Automated systems provide a scalable solution to efficiently monitor and filter harmful content in real-time.

Advances in natural language processing (NLP), machine learning (ML) and deep learning (DL) offer powerful tools to tackle the complexities of hate speech detection. Exploring these technologies enables the development of more accurate and robust models that can handle diverse linguistic and contextual variations. Furthermore, hate speech is not confined to a single lan-

guage or region. Developing models that can effectively detect hate speech across multiple languages, such as English, Arabic, German and Tamil addresses the global nature of this problem and ensures inclusivity. By leveraging the latest advancements in NLP and ML, this research aims to create sophisticated detection systems that can identify and mitigate hate speech and offensive language, thereby contributing to a safer and more inclusive digital communication landscape.

## 1.2 Aims and Objectives

This research primarily focuses on exploring all available machine learning, deep learning as well as transfer learning methods and tools that would aid in the classification of hate and offensive speech in Tamil, English, German and Arabic languages. In light of this, the research aims to investigate if BERT categorizes hate speech better than other classical machine learning methods managed to achieve, since they aim to comprehend the fine details of the hate speech, eliminating the need for BERT's fixed contextual comprehension.
The objectives are:

- **Developing a base Architecture:** To design and carry out ML, DL and TL techniques for the classification of hate and offensive speeches.
- **Evaluate and Interpret Model Performance:** To evaluate these models in the various languages that the performance metrics have been presented.
- **Optimisation with data augmentation:** To investigate how data augmentation systems can be utilized to boost BERT in low resource settings.

## 1.3 Description of the work

This dissertation comprises 6 chapters in total. Chapter 2 describes the existing literature on hate and offensive speech identification, including previous works and focal points in Natural Language Processing (NLP), machine Learning and deep learning pertaining to euthanizing multilingual hate speech. Chapter 3 includes creative hypothesis formulation, data overview

and description and data cleaning relevant to all four languages used. It also details the machine learning, deep learning and transfer learning models used and the optimisation that has been carried out to improve them. Chapter 4 discusses the performance evaluation of the models, with attention to language and data reduction performance comparison. Chapter 5 presents results of different aspects of the work and their significance, as well as discusses how one can build on the current achievements of hate speech detection in the future. At last, chapter 6 draws to a close the project with a conclusion of comments made and restates the contribution made by this research and the impact it makes in practical usage.

# Chapter 2

# Background and Related Work

## 2.1 Background

**Overview**

Machine learning is a subset of artificial intelligence, where instead of writing explicit computer programs for a specific task, computer algorithms are developed to make decisions and predictions based on certain data. Learning enables the computerised systems to draw insights from the data, learn patterns and make decisions with little input from people. ML techniques are very important in the effort of detecting offensive language. ML solves this problem very well and makes it possible for content moderation to be done quickly and more efficiently through the effective enforcement of community standards.

Essentially, machine learning consists of developing AI models with the help of a model training dataset and based on patterns or correlations. After this procedure, the model can make predictions or assess new data. The volume and quality of training data are of great importance for any machine learning model's performance.

The primary machine learning techniques can be classified in three broad areas: supervised learning, unsupervised learning, and reinforcement learning.

- **Supervised learning** is whenever a model is trained on a supervised setting such as a labeled dataset, where the output for every input is known. This involves learning the mapping of the inputs to outputs and so it facilitates image classification or spam filtration.

- **Unsupervised learning** is when there is no labeled data available instead the algorithm is performing a specific operation of creating a result without knowing what the result should normally look like. This includes how and where data is arranged, commonly known as clustering, or how many features are included or reduced dimensionality of the data present.

- **Reinforcement learning** is when an agent interacts with an environment by taking actions and the agent is rewarded or penalized for the actions taken. This method is common in robotics, game playing, and other fields which involve a sequence of decisions to be made.

In recent years, machine learning has been gaining a lot of attention and popularity in providing insights, forecasting, automating and enhancing decision making processes in various sectors. Its uses range from targeted advertising and contents in video on demand platforms, self-driving cars, fraud detection among others. Yet there are some issues that still persist or needs to be addressed including privacy concerns, algorithmic biases and the explainability of machine learning or statistical models. Looking into the future, there will still be further advances of how technology and the society will be more integrated.

### 2.1.1 SVM

Support Vector Machines (SVM) are a well-defined group of supervised methods that focus on classification, regression and outlier detection. They are especially preferred for classification as they are quite effective and efficient for high dimensional spaces. In SVMs, the central concept is the search for that hyperplane, which separates the data into classes with the largest possible distance from the hyperplane. In a two-dimensional space, the hyperplane is a straight line that divides the plane in two such that either of the parts occupies data points of the same class. In the case of a higher dimensional space the hyperplane itself is said to be a subspace of one dimension less than the space itself. (for example, let's consider three dimensional space a plane becomes a hyperplane).
(**Image Source:** https://www.marktechpost.com/2021/03/25/introduction-to-support-vector-machines-svms/)

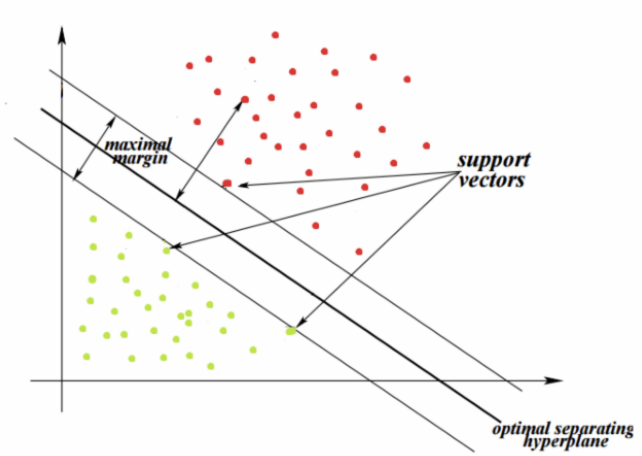



Figure 2.1: Support Vector Machine: Maximal Margin and Support Vectors

The best possible hyperplane is the one where the distance (also called margin) between the closest data points of two classes known as support vectors is maximized as seen in figure 2.1. These support vectors play a key role in positioning and constructing the hyperplane. For SVM, this margin is always enhanced since larger margins normally allow for better performance of the classifier with respect to unseen data.

For a given set of training data $\{(x_i, y_i)\}$ where $x_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$, the hyperplane can be defined by the equation:

$$\mathbf{w} \cdot \mathbf{x} - b = 0$$

Here:

- $\mathbf{w}$ is the weight vector (normal to the hyperplane),
- $b$ is the bias term,
- $\mathbf{x}$ represents the input feature vector.

The decision boundary is defined by this hyperplane, and the classification rule can be expressed as:

$$\text{Class}(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

**Maximizing the Margin**

The margin is the distance between the hyperplane and the closest data points from each class, known as support vectors. The margin $M$ is given by:

$$M = \frac{2}{\|\mathbf{w}\|}$$

To maximize the margin, $\|\mathbf{w}\|$ needs to be minimised subject to the constraint that all points are correctly classified:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1 \quad \text{for all } i$$

This leads to the optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1$$

### 2.1.2 Logistic Regression

Logistic Regression finds extensive use in the area of supervised learning, more commonly for binary classification problems. In contrast to linear regression that outputs numerical values, logistic regression estimates the likelihood of a binary class (i.e., a class can either be 0 or 1). Logistic Regression estimates the correlation of one or many explanation variables to a particular variable which takes either of two values, known as the binary variable using a form of the logistic function (also referred to as the sigmoid function). The sigmoid function, being a special case of the logistic function, transforms a real number between between 0 and 1, thus being useful for factor forecasting.

The logistic function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

where $z$ is the linear combination of the input features:

$$z = \mathbf{w} \cdot \mathbf{x} + b = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

Here:

- **w** is the vector of weights,
- **x** is the vector of input features,
- $b$ is the bias term.

The output of the logistic function $\sigma(z)$ represents the probability that the input belongs to class 1:

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}}$$

**Decision Boundary**

A final decision statement is then made by applying a threshold usually referred to as the cut-off probability, which is typically 0.5. If the threshold is reached or exceeded, class 1 is assigned. For all values below the threshold, the value is assigned to class 0.

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|\mathbf{x}) \geq 0.5 \\ 0 & \text{if } P(y = 1|\mathbf{x}) < 0.5 \end{cases}$$

### 2.1.3 Random Forest

Random Forest is an ensemble technique of machine learning and mostly applies to classification and regression problems. It constructs several decision trees during training and aggregates their results to enhance accuracy and prevent overfitting.

In a Random Forest, multiple trees are trained on a random sample of data with replacement described as bootstrap sampling. In addition, during the construction of trees, only a randomly selected subset of features is tried for splitting, which increases the diversity in trees. This randomness assures such trees are different enough and not related, therefore diminishing chances

of overfitting. In Random Forest, the final prediction is made based on all the tree predictions usually by voting for a classification problem or averaging for a regression problem.
One of the key benefits of random forest is its capability in dealing with larger and higher dimensional datasets, dealing with noise, and measuring feature significance. Certainly, however, it could be rather expensive in computation as it uses a lot of trees and it needs to be optimally configured on hyperparameters like the number of trees or even the maximum allowable depth of each tree.
Overall, random forest is an efficient and powerful tool that is commonly utilized in different areas, such as finance, healthcare, and image processing, owing to its accuracy and dependability.

### 2.1.4 KNN

K-Nearest Neighbors (KNN) is a rather straight forward and yet powerful supervised learning algorithm utilized for classification and regression problems. The basic idea is that closer the data in feature space, they are likely to belong to the same category or will have similar values. KNN can be characterized as a lazy learner as it does not explicitly construct a model during the training phase. Rather, it keeps all the training examples in memory and makes decisions in relation to where the data is located. If there is a need to classify a new data point or predict its value, KNN determines the distance between that point and all the training points. The k number of points which are closest to the new point are then located and classified into one of the classes that are most frequently present or the class values are averaged in case of a regression.
The value of k plays a significant role in KNN. Therefore, too small k makes the method oversensitive to the noise, and too large k creates the loss of details that could lead to the errors. The proper k value is often established by cross-validation.

#### Discrete metrics

The central aspect of KNN that drives its assumption is the notion of ”distance” between data points. Various distance metrics are possible and can be utilized depending on the type of data at hand:

1. **Euclidean Distance:** The most common distance metric, it is the straight-line distance between two points in Euclidean space. For two points $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$, the Euclidean distance is:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Euclidean distance works well in continuous, low-dimensional spaces but can suffer from the curse of dimensionality in high-dimensional spaces.

2. **Manhattan Distance:** Also known as the L1 norm or taxicab distance, it measures the distance between two points along the axes at right angles. It is calculated as:

$$d(p, q) = \sum_{i=1}^{n} |p_i - q_i|$$

Manhattan distance is effective when dealing with grid-like data, such as in image processing, where movements are restricted to horizontal and vertical directions.

3. **Minkowski Distance:** A generalization of both Euclidean and Manhattan distances, it introduces a parameter $p$, allowing for the tuning of the distance calculation:

$$d(p, q) = \left( \sum_{i=1}^{n} |p_i - q_i|^p \right)^{1/p}$$

When $p = 2$, Minkowski distance becomes the Euclidean distance, and when $p = 1$, it becomes the Manhattan distance. This flexibility makes Minkowski distance useful in various applications.

4. **Chebyshev Distance:** This metric considers only the largest absolute difference between coordinates of a pair of points:

$$d(p, q) = \max_i (|p_i - q_i|)$$

Chebyshev distance is useful in scenarios where the maximum distance in any one di-

mension is critical, such as in chess for calculating the number of moves a king would take.

5. **Hamming Distance:** Primarily used for categorical data or binary vectors, Hamming distance counts the number of positions where corresponding symbols differ:

$$d(p, q) = \sum_{i=1}^{n} [p_i \neq q_i]$$

Hamming distance is particularly useful in error detection and correction algorithms.

### 2.1.5 Multilayer Perceptron

A Multilayer Perceptron (MLP) is one variety of the class of feedforward artificial neural networks (ANN) and is considered the basic architecture in deep learning. It contains a number of node or neuron layers,an input layer, one or more hidden layers and an output layer as seen in figure 2.2.

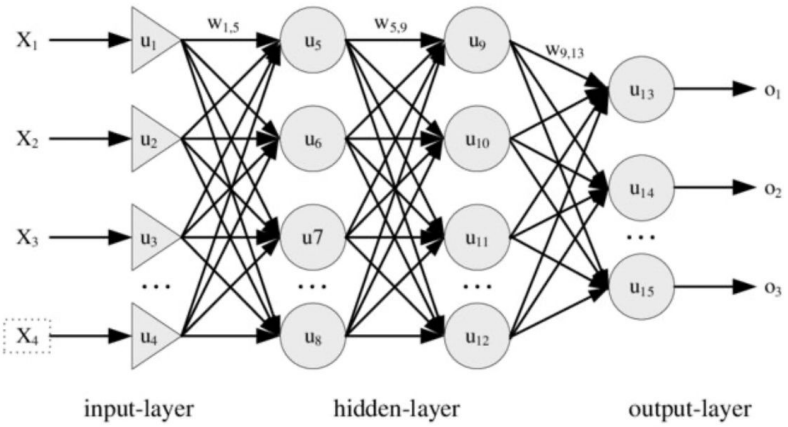



Figure 2.2: Architecture of Multilayer perceptron

(**Image Source:** https://www.researchgate.net/publication/265784353_Utilization_of_Neural_Network_for_Disease_Forecasting/figures?lo=1)

- **InputLayer:** This is the first level containing the raw data features. There is one neuron defined for each feature of the input data in this layer.

- **HiddenLayers:** There exists at least one hidden layer between input and output layers. Every neuron in one of the hidden layers takes in data from the neurons in the previous layer, processes the data through an activation function, and sends the output to another layer. Such layers concealed from end-users are designed to discover and underpin difficult and intricate relationships found in the information. The most commonly used functions are the ReLU (Rectified Linear Unit) and the sigmoid.

- **OutputLayer:** The output layer extends the classifiers by providing final prediction/classification. The number of neurons in this layer is equal to the number of target classes or output values.

MLPs can bring the answer using backpropagation along with some optimization technique like gradient descent. In the process of training, the network changes its weights with respect to the prediction error compared to the target values. This spiral develops further and further until a number of criteria of performance accuracy are satisfied by the model. MLPs are flexible and they are applicable for image, voice, and natural language processing. They can also be used in machine learning and AI to solve problems where some sort of non-linearity is exhibited. The MLPs are quite basic in themselves when compared to other complicated topologies of neural networks however, they are still seen as essential in the domain.

### 2.1.6 RNN

Recurrent Neural Networks (RNNs) are a category of neural network models that are structured to engage with sequential data in which the ordering of data points is significant. On the contrary, while most connective neural networks are feedforward, RNNs are characterized by the presence of directed cycle networks. These enable information to span through different time frames. They are hence very useful for time series, language and any data in sequential order tasks.
The fundamental concept of RNNs is that they include a hidden unit responsible for holding information about the preceding inputs in a sequence. This hidden state is refreshed at every step regardless of whether the inputs are sequential allowing the network to have some sort of 'memory' of the prior inputs.

Mathematically, at each time step $t$, the hidden state $\mathbf{h}_t$ is updated based on the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. The update can be represented as:

$$\mathbf{h}_t = \sigma(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)$$

where:

- $\mathbf{W}_{xh}$ is the weight matrix for the input,
- $\mathbf{W}_{hh}$ is the weight matrix for the hidden state,
- $\mathbf{b}_h$ is the bias vector,
- $\sigma$ is a nonlinear activation function, such as tanh or ReLU.

The output at each time step can be computed as:

$$\mathbf{y}_t = \mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y$$

where $\mathbf{W}_{hy}$ is the weight matrix mapping the hidden state to the output, and $\mathbf{b}_y$ is the bias for the output layer.

Although powerful, classic RNNs suffer from several limitations, the most common being the vanishing and exploding gradient problems. Nevertheless, these consistency intervals can also lead to inconsistency when the backpropagation is carried out across seams of nested timeframes. During BP, when gradients are extended outside beyond the cut-off time and they are recurrently passed back through time, these gradients can either decrease to zero (vanish) or increase (explode), which makes it impossible for the network to grasp long-term sequences. Of course, this is most troubling in cases when the network must learn to retain information for a large number of time steps which is often the case while dealing with long sequences of text or time-oriented data.

### 2.1.7 LSTM & Bi-LSTM

#### Long Short-Term Memory (LSTM)

Long-Short Term Memory, or LSTM, is a type of Recurrent Neural Network (RNN) that specifically addresses some of the problems related to the classical RNNs such as the shortness of the short-term memory. RNNs are effective for sequence data including time series or natural languages, since they make it possible to retain information. Still, this class of networks is challenged with the vanishing gradient problem, a phenomenon where the gradients backpropagated in time reduce with time, making it hard for the network to encode anything that goes beyond its short-term memory.

LSTM solves this problem by adding a memory cell that can keep its state for an extended period of time as well as three gates: input, forget and output for information flow control as seen in figure 2.3.

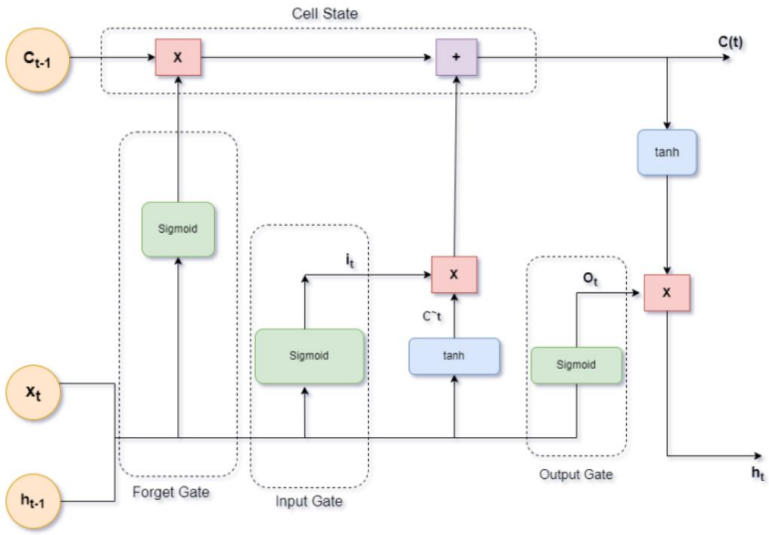



Figure 2.3: Architecture of LSTM

(**Image Source:** https://www.linkedin.com/pulse/understanding-lstm-in-depth-look-its-architecture-pros-babu-thomas/)

- **Memory Cell:** This factor is critical when there is a need to remember information for a long duration to predict events in further time in the correct context.
- **InputGate:** Informs how much new information should be ingested in memory cells. It

uses current input and the previous hidden state as inputs.

- **Forget Gate:** A component that determines the extent of the previous cell state that will be kept. It can 'forget' some old states that are not important.
- **OutputGate:** This gate helps decide what part of the memory cell will be taken as the hidden state for output.

The cell state $\mathbf{C}_t$ and hidden state $\mathbf{h}_t$ at time $t$ are updated through the following equations:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_C \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$

Here, $\sigma$ is the sigmoid function, $\odot$ denotes element-wise multiplication, and $\mathbf{W}$ and $\mathbf{b}$ are the weights and biases learned during training.

**Bidirectional LSTM (Bi-LSTM)**

Bidirectional LSTM (Bi-LSTM) adds the extra feature of sequential data analysis in two directions. It is a variation of LSTM in which, as opposed to standard LSTM, which travels from past to present or future, the flow of information includes both past as well as future context. Bi-LSTM, in contrast, utilizes two LSTMs: one that analyzes the sequence from beginning to end as forward LSTM, and the second one that analyzes the sequence from end to beginning as backward LSTM as seen in the figure 2.4.
(**Image Source:** https://www.researchgate.net/publication/344751031_Real-Time_Cuffless_Continuous_Blood_Pressure_Estimation_Using_Deep_Learning_Model/figures?lo=1)
The output of these two LSTMs is then combined, usually through simple concatenation, thus allowing the network to access each point in this sequence not only to the history but to the

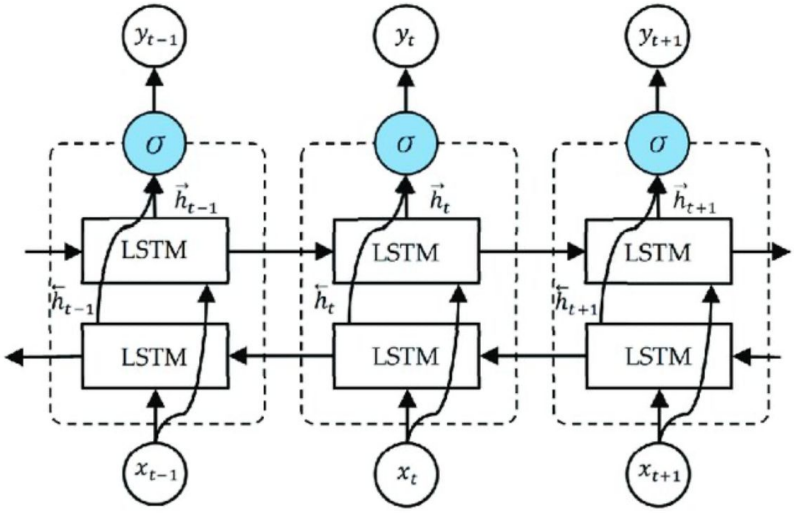



Figure 2.4: Architecture of Bi-LSTM

future perspective as well. This feature makes Bi-LSTM notably efficient in situations where there is a need to retrieve information that is both before and after a certain point in the sequence like in many NLP tasks such as named entity recognition or sentiment classification.

LSTM networks are popular for several applications such as speech recognition, language modeling and sequence prediction because they can remember long-term information. Bi-LSTM networks apply also to the Natural Language Processing area, where every word in the sentence has significance either before or after it while performing machine translation, text summarization or synthesizing speech.

LSTM networks are popular for several applications such as speech recognition, language modeling and sequence prediction because they can remember long-term information. Bi-LSTM networks apply also to the Natural Language Processing area, where every word in the sentence has significance either before or after it while performing machine translation, text summarization or synthesizing speech.

## 2.2 Related Work

The monitoring of hate speech and offensive language using NLP has gained a lot of attention in academic circles in recent years. People's interest is governed by the fact that there are plenty of social networking sites where this kind of information can make trouble quickly and there

is a need for well-developed and effective systems for addressing this. In overview of this eco systems of knowledge, several streaming formats arise and these are supervised, deep and hybrid models and localized applications of these strategies. Here an overview of the methods and techniques applied in these approaches is presented based on a study of relevant publications on the subject matter.

### 2.2.1 Supervised Learning Approaches

For hate speech detection, supervised learning has always been one of the basic approaches. Usually, this strategy uses classifiers to recognize examples of hate speech in labelled datasets. These models are said to be functioning when a large amount of the training data used takes from features and classifies them.

Fauzi et al (2018) [11] conducted an extensive study employing a set of supervised learning algorithms such as Naive Bayes, Random Forest, and Support Vector Machines (SVM) to identify hate speech on social media. They utilized Term Frequency-Inverse Document Frequency (TF-IDF) as the main feature extractor approach, which is well-known for its text classification efficiency. This ensemble method took advantage of the skills of individual models using multiple classifiers, which improved detection performance notably in addressing imbalanced nature hate speech datasets. The study showed that Naive Bayes captured probabilistic relationships between features effectively but Random Forest and SVM models had more robust decision boundaries to handle complex ones so the ensemble was really useful.

One of the important points that Fauzi et al. have highlighted in their work is the focus on imbalanced data sets, which is a common problem enclosing hate speech detection. In these situations, the imbalanced datasets exist particularly because of the lower instances of hate speech rather than non-hate speech acts. Therefore, models trained on such data become biased, which often leads to low sensitivity for that class that is of minority. This constraint was addressed by Fauzi et al. by trying out several data alteration methods such as oversampling the less represented group, which means increasing the number of instances of hate speech, or cost-sensitive learning, in which the model is trained to treat all speech acts alike but increases the punishment for wrongly labelling the hate speech. These methods worked well as they enhanced the models'

capability of identifying hate speech while maintaining the level of performance attained before the application of the techniques.

Likewise, Sari (2019) [15] engaged in logistic regression applied to detect hate speech on Twitter. They had an extremely thorough approach with feature engineering, extracting both lexical and semantic features from the text by including n-grams as well as word embeddings on top of TF-IDF. It was shown that a simple and well understood logistic regression model could achieve competitive accuracy when the dataset is balanced and carefully preprocessed. This result demonstrates the potential benefits of feature selection and preprocessing on traditional machine learning models.

One of the additional merits of Sari's work was their investigation of the different techniques of text preprocessing in order to enhance model performance. They researched how effective such techniques as stop words removal, stemming, and lemmatization on the accuracy of the logistic regression model. These results showed that stop words removal increased the models but not to a large extent whilst stemming and lemmatization were more useful in decreasing the dimensions of the features and enhancing the model generalization. This close survey of the data preprocessing in question suggests how useful it is to employ such methods depending on the nature of the data being studied.

### 2.2.2 Deep Learning Approaches

Deep learning brought a sea change in detection hate speech. Given their ability to automatically discover optimal features from raw data, deep learning models effectively model the intricacies of natural language-contextual dependencies as well as subtleties present in hate speech.

As noted by Rosa et al. (2018) [24], he was the first one to utilize Convolutional Neural Networks (CNNs) and Long/CD Short - Term Memory Neural Networks (LSTM) in the area of cyberbullying detection which has many of the properties of bias linguistics. The approach taken was to feed the words within a text to the CNNs to train them to identify smaller units within the text like phrases or sets of words and the LSTMs were used with the conversation to capture more distant context and temporal relations. The word embeddings as a Word2Vec enhanced the comprehension of these models such that they were able to know the semantics of

some words in context thus enabling them to detect hate speech in their complex forms even better. They also looked at several other architectures including the CNN and LSTM architectures, the last described architecture was the most successful in the model developments, showing the importance of the model architecture on the deep learning based text classification activities.

Pre-trained word embeddings were one of the strong points of Rosa et al.'s approach. Using word embeddings, the model can capture more semantic similarity between words based on how similar their context is in a large corpus other than what I get from simple bag-of-words representation. Words such as "hate" and "disgust," for example, may be semantically related, but there is a close association in the words that identify them, so word embeddings enable learners to learn these associations and increases its ability to capture hate speech without necessarily using similar vocabulary. Rosa et al.'s study also established that further tuning of the pre-trained embeddings over domain-specific, which enhances quality, makes them capture finer details in language on user-specified datasets.

Buoyed by the promising results, Tin Van Huynh (2019) [18] developed a hybrid deep learning model incorporating Bi-directional Gated Recurrent Units (Bi-GRU) and Convolutional Neural Networks (CNNs) as well as LSTM networks. Their model aimed at overcoming the challenges encountered with the use of a single deep learning model by employing the strengths of multiple models. The Bi-GRU component of the model was used to model the bi-directional dependencies in the written text, while the CNN layers were more concerned with the local feature extracts. The LSTM layers enhanced the context modeling of the model by remembering the previous state of all text sequences. This layered technique proved to be useful in the identification of context-specific hate speech that is often dependent on several sentences or the whole dialogue.

One interesting point about Tin Van Huynh's model was that they used Bi-GRUsin-steed of standard GRUs to make the input data processed in both forward and backward directions. This bidirectional processing is especially nice for NLP tasks, in which the meaning of a word or phrase might depend on both what came before it and what comes after. The improved contextual insight for each word in a sentence made it easier to predict accurately with the use of Bi-GRUs. The integration of CNNs and LSTM made the model reliable to pick up sophisticated textures

in data, which is definitely a breakthrough finding for hate speech detection.

### 2.2.3 Hybrid and Multimodal Approaches

Due to the weaknesses associated with mono-approaches to hate speech detection, several works have emerged which give rise to such terms as 'hybrid approaches' These approaches seek to exploit the merits of several models and lessen the shortcomings of each model.

Di Capua et al. [7] explored several concepts by proposing the use of unsupervised learning techniques in conjunction with supervised classifiers' approaches. Their methodology comprised one applying K-Means clustering to untapped datasets consisting of instances of hate speech labelled so as to be input into a supervised classifier. This method was especially good with the dynamic nature and the sheer volume of online hate that changes form from one site or context to another and renders a few forms all useless. Aided by unsupervised learning to spot patterns in the data prior to administering a model that relies on supervised learning, the model was better placed in terms of accuracy as well as generalization.

The method proposed by Di Capua et al. is particularly remarkable at the early stage of the procedure as it includes clustering methods. Clustering methods refer to the processes of detection of groupings or similarities of data attributes that are performed in most cases in the absence of targets in the data set. In case of detecting hate speech, clustering can assist in defining the types of hate speech, e.g. racial or sexist and others in such a manner that they are not abused. This helps not only to understand the various manifestations of hate speech but also offers a systematic approach to encode large volumes of data, which later on can be used to build supervised models. At this point using clustering alongside supervised approaches worked more ideal as models created could easily find their way to other data that had never been used.

Qian et al. [23] introduced another innovative approach by creating a phenomenological benchmark dataset for hate speech intervention in generative context. Their work was not limited to identifying hate speech as it included producing responses to defuse hateful interaction as well. The dataset introduced by them had the contextual dialog which was important in interpreting the appropriate hate speech. The generative model used in this study was built on the Sequential Model-to-Sequence or Seq2Seq architecture that was then improved. More sophisticated

techniques such as attention mechanisms and Reinforcement Learning (RL) were used to suggest techniques that could help improve the responses generated. This advancement marks an important move towards hate speech identification and control systems that are context aware, stressing the need for hate speech generation and action themes that allow for the engagement with the conversation as a whole rather than each individual post.

The research conducted by Qian et al. is of great interest due to its emphasis on both generative models. Which is a move away from the typical approaches which are mainly concerned with detection. Generative models, which generate new content upon feedback, are useful for detecting hate speech as well as stopping hateful conversations in their infancy. Adding other novelty to these studies is the application of recognizing deep learning techniques, that employ the Seq2Seq architecture common for all machine translation systems, which outputs relevant and appropriate to the context responses. The efficiency in the generated responses is further improved because of attention mechanisms that allow the model to attend to certain specified relevant sections of the input when generating the output. There was, however, the application of Reinforcement Learning, which was used to train the model to maximize a reward function based on how well the model's responses addressed the problem of hate speech in the proposed interventions.

### 2.2.4 Application to Specific Domains

The techniques for hate speech detection have also been adapted to a particular domain like sports or even gaming where hate speech can be common. Such implementations usually come with peculiar challenges, for example, understanding the language of the particular domain and factors or events, such as the performance of the team or the results of the game, which impact the hate speech spread level.

Groll et al. (2019) [16] tested the applicability of Random Forest models in predicting the outcomes of the FIFA World Cup of 2018 with the provision that team ability parameters were factored in. Although their main concern was predicting sports outcomes, the methods implemented in this research are applicable to the sphere of hate speech detection, especially with regards to the big data aspect and class imbalance problem. The study included the feature

selection stage focusing most on the variables that enhance the performance of the team, like player ratings and results of the previous matches played. Given that the Random Forest model is robust at high dimensional data accuracy levels, its use in predicting the outcome was successful. Thus, the practical applicability of these techniques to dealing with complex real-world datasets was shown.

The importance of the feature selection process in predictive modeling is properly demonstrated in Groll et al.'s study. In other words, once the problem of hate speech detection is framed, the right features, including the presence of specific keywords, the textual sentiment, or the use of hate words, have to be chosen; otherwise, the model performance will not be up to the mark. The other noteworthy feature of the study is the adoption of Random Forests, which is an ensemble modeling approach that relies on the predictions of many decision trees. Adoption of this method also insists upon the use of complex models that are able to address coupling between factors. Such concern is particularly useful in hate speech detection since the interpretation of a word or phrase dictates its meaning, thereby complicating simple heuristic approaches and such simple models are unlikely to unravel.

Most recently, Ulmer & Fernandez [30] carried out studies on predicting the winner of the soccer competitions of the English Premier League with the help of supervised algorithms. Bearing this in mind, their approach was particularly focused on the issue of feature selection on those variables that were believed to have the strongest relationships with the targets, such as team form, player injuries and previous match scores. The study included the application of one of the already mentioned models - logistic regression as well as Random Forests trained on multi-year databases. Their findings demonstrated the impossibility of forecasting the winner of a match in the English Premier League, a phenomenon characterized by high lability and competitiveness and where a lot of external factors come into play. These difficulties are quite similar to those that are presented in the analysis of hateful speech patterns, where the vocabulary is subject to tension and contextualization is of key importance.

Ulmer & Fernandez's work points include the importance of ongoing assessment of models as well as their modification. We've all been in dynamic circumstances in sports competitions or in an online discourse for that matter, where such environments can be subject to rapid changes,

thus, models should be continuously trained on fresh data if their up-to-date status is to be guaranteed. In this context, this means, the models used to combat hate speech should be redesigned to include newer forms of hate speech or motivated changes in the social media language. Turning to Ulmer & Fernandez, they have employed logistic regression, a relatively ”easy”, but as I know it’s a powerful model, which makes it possible to evaluate interpretable models concerning the causal factors presented as key issues for certain consequences. It is a frequently required degree of interpretability for hate speech detection because knowing the rationale for classifying the text into problematic evergreen helps improve the model for greater functionality as well.

### 2.2.5 Current Limitations

Although detection of hate speech has made great strides, there are still some challenges remaining. One of the most obvious challenges is the nature of the language which is dynamic and ever-changing, especially in the Internet settings where fresh slurs and offensive words are always coming up. Waseem & Hovy (2017) [33] emphasized and explained the obstacles to designing effective detection systems allowing reusability of present models. With respect to their study, they presented a concept of dynamic feature selection, which dealt with the model applying fresh features from revised data at all times. This approach dramatically enhanced the model’s success in the detection of new patterns of hate speech, especially those that are less obvious and come in coded language usage.

The dynamic feature selection method proposed by Waseem & Hovy is notable, especially in the scope of online communication where language is changing fast and neologisms or jargon can be created within a short duration. Waseem and Hovy’s extension is a very practical one that also modifies the features the model takes into account, ensuring that the model can adapt over time, even as the language employed by hate groups or susceptible users changes. This method also stresses the need for the regular extension and enhancement of existing systems for detecting hate speech, as systems that have been built but not improved upon will not be able to withstand the changing language.

In Waseem & Hovy (2017), the authors rely on the prevention of hate speech and what demo-

graphic information can capture audience information and how it can be best employed. Understanding the user's background information or demographic features such as gender, age and location helps construct a better picture of the situation in which hate speech takes place thereby enhancing the accuracy of detection. Also, this approach focused on the ethical aspects of using demographic information in machine learning including privacy issues and the risk of discrimination. As a result, the study did consider that when developing these types of systems in the future, such factors should be taken into account.

The issue of providing demographic data in hate speech detection systems as investigated by Waseem & Hovy is extremely troubling. There are ethical concerns that come up, such as the appropriate use of such information. While demographic attributes are useful for enriching the accuracy of detection techniques, they equally come with challenges with respect to privacy and unfairness. For example, using demographic information could lead to the model targeting members of an organization or a group based on candidate's gender or age, ethnicity, etc. In order to prevent such situations there is a need to put in place measures to avoid compromising the privacy of users and to ensure that the models are built and deployed in a manner that takes care of bias. This is possibly done through the application of differential privacy, whereby data on the individual level is obscured by the introduction of noise or determining how to analyze the model such that there is a fair performance of the model concerning all demographic classes. Going forward, there are certain areas that would most probably attract attention in future studies. Specifically, one area that particularly catches one's attention is analyzing scientific articles or videos and from those sources of data, building more complex detection models. The incorporation of multimodal data could improve the model's contextual understanding of hate speech, thus offering better detection. For example, hate speech that is conducted through images, such as memes or symbols, which are oftentimes lost by text-only analysis, can be targeted through images along with texts. The development of advanced generative models that can be used to intervene in hate speech and other aggressions is the theme of yet another contribution, which is consistent with the study of Qian et al. (2019) [23]. These models could come in handy where such interventions need to be carried out in real time within online discussions, some of which could assist in reducing tensions and enhancing healthy interaction. Most of such generative

models are trained on volumes of conversational data and thus will benefit from the introduction of reinforcement learning, where the model is trained such that it optimizes its interventions based on feedback from the users. This is likely to make the intervention much more efficient, as more user-centric and suitable responses will be given depending on the situation.

Furthermore, it is becoming evident that cross-platform and cross-lingual hate speech detection constitutes an important emerging field. It would almost certainly increase the scalability and applicability of these models were they capable of performing hate speech detection in various languages and platforms. Nonetheless, this poses several problems including the acquisition of diverse data sources and the creation of models with cross-language and culture generalizable capabilities. One way to solve these problems is to apply transfer learning, where one language or one platform model is adapted to another language or the same language but altered platform.

Finally, it is important to underline how complicated it is to stop the spread of hatred and xenophobia through these technologies. However, with the adoption of such technologies in society, the chances of their exploitation to violate the rights of free speech or to discriminate against particular communities also increase. Subsequent research should focus on risk reduction techniques including the building of customers that are responsibly managed, the application of privacy barriers and the engagement of various customers in the creation and distribution of such systems in the future. One possible approach to circumventing the issues associated with such systems is the adoption of explainable AI in the design of models. Users of the protections brace themselves so that all the actions of the intelligent systems such as which content is identified as hate speech and which isn't are clear to the target users.

# Chapter 3

# Methodology

This chapter discusses how an integrated strategy defined the research process. The methodology consists in establishing research objectives, gathering multilingual corpora, and implementing text processing techniques on the raw data so as to prepare it for the purposes of the intended analysis. Exploratory data analysis is performed to understand the characteristics and distribution of the available datasets, and a variety of models including machine learning models, deep learning models and transfer learning models are used for the performance evaluating on hate and offensive speech detection task. The methodology allows for structured data processes and models application in relation to the objectives of the research.

## 3.1 Research Questions

**Question-1: Will standard machine learning models for hate and offensive speech detection outperform transformer-based models such as BERT and deep learning models in the three languages: Tamil, English, German, and Arabic?** This may be due to its high capacity of capturing contextual information that may be important when using more sophisticated languages like hate speech. Most classical models on the contrary, are dependent on a handcrafted feature, while transformers might dynamically address the contextual relationships of words making them probably more suitable to understand the subtle form of hate speech spread out in different languages.

**Question-2: Are data augmentation techniques capable of enhancing the effectiveness of**

**transformer-based models and deep learning models at lower (33%, 50%, 75%) rather than at full (100%) amount of training data, especially in regards to the multilingual hate and offensive speech detection tasks ?** Transfer learning models may still able to excel when performing various tasks, although their effectiveness may decrease on account of a lack of data. The research aims to determine if data augmentation can ease this limitation, improving the models' performance in such low-resource tasks.

## 3.2 Data Acquisition

### 3.2.1 Tamil Dataset

The Tamil Hate Speech and Abusive Comment Detection Dataset [3] is a resource that delves into various aspects of the abusive languages only related to Tamil culture from the comments sections of YouTube video hosting activities. This dataset was created under the framework of a shared task in the DravidianLangTech@ACL 2022 workshop. The given task is directed towards recognizing the offensive content on social media (youtube) and requires the use of sociolinguistics in the wake of the expanding phenomenon of online social networking OSN. Such platforms have turned out to be fertile ground for a wide range of hate speech aimed at groups and individuals due to their race, gender, sexual orientation, or religion.
The dataset includes multiple thematic categories of hate speech and abusive language, covering a broad range of offensive behaviors such as:

- **Misogyny:** Hatred or prejudice against women.
- **Misandry:** Hatred or prejudice against men.
- **Homophobia**: Discrimination or dislike against individuals based on their sexual orientation.
- **Transphobia:** Discrimination against transgender individuals.
- **Xenophobia:** Fear or hatred of foreigners or people from different ethnicities.
- **Counter Speech:** Positive or neutral comments aimed at combating hate speech.

- **Hope Speech:** Content encouraging hope, positivity, or unity.

The dataset includes annotation at the comment/post level for each entry and an average length of one sentence is devoted towards each post/comment that describes the social media post. The posts have been categorized into multiple classes, including why the posts have not been classified as hate speech: none of the above, and in some cases more categories of hate speech and counter speech were provided. The final dataset contains comments exclusively in Tamil as well as bilingual comments (Tamil and English code-mixed). The Tamil dataset sample is given in table 3.1

This dataset will help in building machine learning models for hate speech and offensive comment detection and classification in the Tamil language. Since it addresses these problems in a more fine-grained manner, it also tries to solve the problems related to language-based abuse on social platforms. In this resource, all the aspects related to hate speech detection are elaborated so as to enhance the efficacy of detecting hate speech in languages with limited resources like Tamil. The corpus allows for a binary classification (offensive/non-offensive) but also extends to even more sophisticated tasks performing specific types of discrimination and different types of abusive acts targeted toward people.

**Sample Dataset of Tamil Hate Speech**

| S.no | Class | Tweets |
|---|---|---|
| 1 | Hope-Speech | உங்கள் பேச்சை எதிர்பர்த்தனே். நல்ல விளக்கம் அரமை. நியாயமன மறையில் பதிவுசெய்தள்ளீர்கள். நீண்ட கலம் வழ்வதற்கு வழ்த்தக்கள். |
| 2 | None-of-the-above | எச். ரசவல் இரமரக்கே இழிவு இவர் எல்லம் இரமர் பரம்பரையம். |
| 3 | None-of-the-above | கல்யணம்னுவரம்போதுதன் சதிமக்கியம் எங்களக்கு |
| 4 | None-of-the-above | Tentu KottaI இயன் நன் தமிழ் இந்து |
| 5 | Homophobia | Tamil Selvan அடே கோவிந்தனக்குபித்தவனே. |

Table 3.1: Sample Dataset of Tamil Tweets

### 3.2.2 Arabic Dataset

The Arabic Levantine Hate Speech and Abusive Language Dataset (L-HSAB) [17] is also the first-ever focused dataset for the detection of hate speech and abusive language in the Levantine dialects of Arabic. It originates in the countries of Syria and Lebanon. It was suggested at the ALW-2019, workshop co-organized with ACL-2019, held in Florence, Italy. This is because such countries which have the Levantine dialect as their mother tongue, make these countries experience a turmoil of political and social climate, making the people to engage in violent assaluts on the internet over every particular issue which no body wants to focus on, namely, abusive languages and hate speech.

The dataset consists of 5,846 tweets and offers two primary classification tasks:

1. **Binary Classification:** Categorizes tweets into 'Normal' (non-abusive) or 'Abusive' language.

2. **Multiclass Classification:** Further refines the classification by categorizing tweets as 'Normal', 'Abusive', or 'Hate' speech.

The dataset has been segmented into a training subset and a testing subset and manually annotated by three connotative arabic speakers from the levantine region. Such annotations will greatly aid in understanding the communication styles of abusive users in the Levantine Arabic language. The sample of Arabic dataset is given in the table 3.2. The three categories are defined as follows:

- **Normal:** Tweets that contain no offensive, aggressive, insulting, or profane language.

- **Abusive:** Tweets that contain offensive, aggressive, insulting, or profane content but are not specifically targeted as hate speech.

- **Hate:** Tweets that use abusive language targeted at a specific person or group, often dehumanizing or demeaning based on attributes such as race, gender, religion, or disability.

This dataset is very beneficial for the purpose of hate speech detection in a low-resource language such as Arabic since it presents politically sensitive raw data. Its dual binary and multiclass classification component is potent in developing and testing a variety of machine learning models

ranging from simple classifiers to sophisticated neural networks. This dataset is central to the exploration of the specific features of hate and offensive language in Arabic and contributes to the improvement of the universal sensitivity of natural language processing against the particular forms of hate speech in different languages.

**Sample Dataset of Arabic Hate Speech**

| S.no | Tweet | Class |
|---|---|---|
| 1 | وزير الخارجية اللبنانيي جبران باسيل قال في سلسلة تغريداته عقب اختتام القمة العربية التنموية الاقتصادية والاجتماعية لمسنا فت | normal |
| 2 | سورية بلد الحضارات نربطها بعجلة او بحيوان | normal |
| 3 | أخي الحاج إذا شعرت أنك محرجا من الانتقادات لتصريح جبران باسيل لا داعي لأن تهاجم المنتقدين | normal |
| 4 | ما فيك تعيش بلا ما تكذب فن ليل نهار وبكرة تلم الموضوع عند السيد | normal |
| 5 | هذا البطل الذي قاتل وواجهت بحياته لتحيا أنت يا واطي يا عيب الشوم | abusive |

Table 3.2: Sample Dataset of Arabic Tweets

### 3.2.3 German Dataset

The **German 2019-HASOC Dataset** [28] has been specifically created so that users can perform an analysis as well as detection of hate and offensive speech in German online texts. This dataset includes 4,669 texts, each of which is an individual sentence or post. The German dataset sample is given in table 3.3.

Each dataset contains the following four columns:

1. **Text ID**: This column is the primary key of the database which makes sure there is no duplication of the texts.

2. **Text**: The second column has the actual text that is made up of German social media content. The snippets are of different lengths averaging to have about **22.46 words** in the sentences.

3. **Task 1 (Binary Classification)**: This column allows for the classification of each post into two groups:

    - **NOT**: Not Offensive – Non-violent Linguistic Hate, the contents of the post are devoid of hate speech.
    - **HOF**: Hate Speech and Offensive Content - it implies that the contents of the post are offensive in some way or other.

4. **Task 2 (Multiclass Classification)**: In this column, further classification of offensive content will be provided for a better understanding. Depending on the post, this column classifies it into one of the following groups:

    - **NONE**: There is no offensive content.
    - **OFFN**: Offensive content, but does not contain hate speech.
    - **HATE**: Hate speech, a more serious form of offensive language.
    - **PRFN**: Profanity posts such as strong language or vulgar words but some may not be threatening or offensive.

This corpus primarily targets the German language and contains varying levels of offense language. Each post is marked with a certain level of severity of damage, which allows to differentiate the various types of harmful speech. This detailed classification of the dataset makes it especially valuable when assessing and teaching machine learning models for the tasks of binary or multiclass hate speech identification. The **German 2019-HASOC dataset** is important to train models in understanding and identifying hate speech in particular low-resource languages like German. Each annotation in the dataset contains information that is useful for performing binary or multiclass tasks across a range of NLP models, from classical to modern BERT and RoBERTa transformer-based models.

**Sample Dataset of German Hate Speech**

| S.no | ID | Tweets | Task-1 | Task-2 |
|---|---|---|---|---|
| 1 | hasoc_de_1 | Frank Rennicke – Ich bin stolz https://t.co/Cm6TD8w1k1 https://t.co/qynXso07Zn | NOT | NONE |
| 2 | hasoc_de_2 | ANSEHEN.....und danach bitte TEILEN...TEILEN...TEILEN.... https://t.co/z18lnbOWRQ | NOT | NONE |
| 3 | hasoc_de_3 | #Koeln Mohamed erkennt kein deutsches Recht sondern nur die #Scharia an. Das er den Kölner Dom kaputt machen wollte, war nur ein Scherz aber wenn er aus dem Knast rauskommt, hat er kein Mitleid mehr. https://t.co/xcSixH8JR2 | NOT | NONE |
| 4 | hasoc_de_4 | #SaudiArabien ist eine brutale islamische Diktatur und richtete kürzlich fünf Männer wegen ihrer Homosexualität hin. Welche Konsequenzen hat das für das Verhältnis zu Saudi-Arabien nach Meinung der Bundesregierung? Die Antwort im Video. https://t.co/ttMeReBXFE | NOT | NONE |
| 5 | hasoc_de_5 | BundespolizeI #München hat im 1. Quartal 2019 rund 3.380 illegale Einreisen registriert, die Migranten stammten v. a. aus Nigeria, Afghanistan, Serbien, Albanien und der TürkeI - je ein Drittel kam mit PKW bzw. per Bahn, ein Viertel mit dem Bus #Einreise https://t.co/vNzTXqLomN | NOT | NONE |

Table 3.3: Sample Dataset of German Tweets

### 3.2.4 English Dataset

The Hate Speech and Offensive English Language [6] on Twitter Dataset constitutes a study aimed at the cataloging of the English Twitter communications with the prospect of determining hate speech and offensive speech. This dataset provides an excellent source for the development and testing of machine learning systems aimed at abusive language detection from social media networks and in this case Twitter. The dataset includes a number of concrete methods which

were interrelated and resulted in a single space for use both for training and for testing.
The dataset consists of a particular tweet and a few operational columns to enhance understanding of its classification:

- **Count**: The number of annotations assigned to each tweet is mentioned in this column.
- **Hate Speech Count:** This column contains information on how many annotations pointed out the hate speech present in the tweet.
- **Offensive Language Count:** This column gives the number of annotations which represented the tweet in any offensive manner.
- **Neither Count:** This refers to the number of the annotations which did neither hate speech nor offensive language in a character.

The structure of the dataset permits the evaluators to measure the degree of harmfulness in every Twitter post by including several levels of annotating. This multi-class categorization helps create models in machine learning that are able to discriminate between hate speech, offensive language and neutral language, enhancing the understanding and detection of toxicity. The English dataset sample is given in the table 3.4.
This dataset is obtained from Kaggle, which is useful in the development of automated systems for hate speech. It provides a large, comprehensive and pre-structured dataset that is essential for building and optimizing hate speech and abusive language detection systems that function in real-time social media environments. Special attention was given to prescribing a thorough annotation system, which enables to delve into the grammatical constructions that accompany hate speech, allowing to target such intricacies that may be eluded by less sophisticated systems.

## 3.3 Preprocessing

In the area of natural language processing (NLP), an appropriate and careful preprocessing of unprocessed text data is critical in preparing it for model training. For this study, various preprocessing techniques were applied uniformly to the four languages: Tamil, English, German and Arabic. Translations were created for each language dataset to achieve uniformity of structure

| S.No | Count | Hate Speech | Offensive Language | Neither | Class | Tweets |
|---|---|---|---|---|---|---|
| 0 | 3 | 0 | 0 | 3 | 2 | !!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out... |
| 1 | 3 | 0 | 3 | 0 | 1 | !!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place!! |
| 2 | 3 | 0 | 3 | 0 | 1 | !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever fuck a bitch and she start to cry? You be confused as shit |
| 3 | 3 | 0 | 2 | 1 | 1 | !!!!!!!!! RT @C_G_Anderson: @viva_based she look like a tranny |
| 4 | 6 | 0 | 6 | 0 | 1 | !!!!!!!!!!! RT @ShenikaRoberts: The shit you hear about me might be true or it might be faker than the bitch who told it to ya #57361; |

Table 3.4: Sample Dataset of English Tweets

and sufficiency for the purposes of the model. The preprocessing steps for this study included, text cleaning, tokenization, stopword removal, stemming/lemmatization, label encoding and text representation with the help of TF-IDF vectorization.

### 3.3.1 Text Cleaning

The input text content often contains segments like punctuation, URLs, special characters, and emojis that are unnecessary and unhelpful when it comes to understanding the semantic aspect of the content. Hence, text pre-processing was done to improve the input quality. For English, a custom cleaning function was provided for removal of some stopwords, special symbols, and additional non-articulated characters like …..., ‘, and https:// links. The same was done for German and Tamil datasets, where all URLs, punctuation as well as non-numbers and english alphabets were purged. Arabic data on the other hand also involved lower casing as well as eliminating posting characters. These cleaning processes helped in ensuring that the input text in the corpus had no irrelevant details, which helped in the ease of tokenization and vectorization.

### 3.3.2 Tokenization

Tokenization involves converting a text into discrete units called tokens, which may be words or subword groups. For this investigation, it was noted that applying language tokenization over all four language datasets NLTK libraries for each language were used. For instance, in the case of the American English dataset, its documentation is just broken into words using the

"word_tokenize" function from the NLTK library. In the same way, in the German dataset, sentence tokenization using the "sent_tokenize" function of NLTK was done. In Arabic and Tamil, word-level tokenization was focused on, although it was noted that the tokenization process comprehensively captured the language (english-ml).

### 3.3.3 Stopword Removal

Stopwords are common terms that appear in sentences but have little or no meaning contribution (e.g. "is", "and", "the"). These were eliminated in order to lessen the mess in the text data. Some NLTK stopword lists were utilized on the English and German datasets respectively. As for Tamil and Arabic, specific language stopword lists were used employing extraction from appropriate NLP libraries which made sure such common words were systematically eliminated. This measure dramatically reduced the amount of words in the vocabulary which in turn enhanced the performance of further computation and training of the model.

### 3.3.4 Stemming/Lemmatization

Stemming and lemmatization are crucial for reducing words to their root or base forms. Stemming was used to reduce words to their root, typically resulting in more aggressive cuts to word forms. For example, the German language dataset applied the Cistem stemmer, which is optimised for German texts. The English dataset used Porter Stemmer for reducing word variants to their base forms. Arabic and Tamil, which involve morphologically rich languages, required specific stemming algorithms to handle variations in root words. Lemmatization was not heavily used due to its complexity across different languages, but where applicable, it ensured proper root-word representation.

### 3.3.5 Label Encoding

As for the target variable, or in other words, the class labels (hate, offensive, or normal), these variables were encoded using label encoding for all types of the datasets. This helped to facilitate the interpretation of the class labels as numbers, which is quite essential during the training of the machine learning models. The encoding process involved assigning integer values to the domain

labels ‘Hate’, ‘Offensive’, and ‘Neither’ e.g. 0 for Hate, 1 for Offensive, 2 for Neither(english-ml)(german-ml). This was consistently applied across all datasets.

### 3.3.6 TF-IDF Vectorization

Regarding the classic machine learning models, any text data was converted to its numerical form via Term Frequency-Inverse Document Frequency (TF-IDF) vectorisation. This approach makes it easy to evaluate the relevance of words in a collection of documents in a structural way by collecting both common and uncommon terminologies. In the English dataset, TF-IDF vectorization was applied taking into consideration a maximum feature of 5000. Content in the German dataset similarly employed TF-IDF vectorization with selective n-grams (unigrams, bigrams, and trigrams) to focus on short sequences of words. For Tamil and Arabic, the same TF-IDF mechanism was applied, taking into account the language scripts and tokenisation methods. The prepared vectorized text was then fed to classical machine learning techniques as well as deep learning techniques.

With the use of these preprocessing techniques, the datasets corresponding to all four languages were standardized to a format suitable for machine learning and deep learning operations while still maintaining the distinct features of each language and diminishing noise and useless data.

## 3.4 Data Exploration

### 3.4.1 Distribution of Target Classes

The analysis of the distribution of target classes in Figure 3.1 is very important because it assists in identifying whether there is any class imbalance that will affect how well the trained machine learning models are able to perform. In this study, the datasets together with Tamil, English, German, and Arabic, have each been examined to ascertain the composition of the class. A standard count plot using sns.countplot has been used to display this.

In the Tamil dataset, there are different hate speech categories: ”None-of-the-above”, ”Misogyny”, ”Misandry”, ”Homophobia” etc. Majority of classes actually relates to ‘none of the above’ while class distributions are highly unbalanced as other classes like ‘Homophobia’ and ‘Xeno-

phobia' are available in very few numbers.

In the English dataset, there are generally common three classes: "Hate Speech", "Offensive Speech", "Neither". There is also an overrepresentation of the class of 'Offensive Speech', which could bias the model due to the class distribution.

In the 'German dataset' classification involves classes such as '145 Misandry', '146 None-of-the-above' and '147 Counter-speech'. The distribution again indicates there is an imbalance; 'None-of-the-above' has the most occurrences and the rest are too few.

Lastly, in the dataset concerning Arabic, the classes have "Normal", "Abusive" and "Hate". Here, tweets in the class "Normal" are the most frequent in the other categories and are the most dominating class. Visualization of this imbalance helps to determine whether techniques like oversampling or under-sampling need to be employed so that the final model performs well across all the classes.

### 3.4.2 Distribution of Text Length

The distribution of text length in Figure 3.2 helps in understanding the diversity and intricacy of textual data. Such analysis utilizes histograms to visualize the text length distribution across the different datasets for each language.

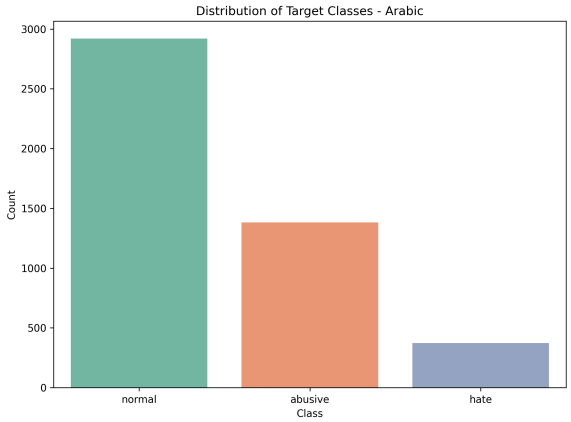
When it comes to the Tamil dataset, it is characterized by high text length variability with most text entries 10 to 50 words in length. Only a few texts are longer or shorter than this range which may reflect short social media postings alongside explanations.

One notable difference that can be appreciated visually is the distribution of the text lengths in the English dataset. As can be seen from the histogram plots, English tweets have an average length of around 9.6 words. Other texts feature an array of no shorter than a single word and up to about 50 word sentences, probably showing the varied nature of English Twitter data.

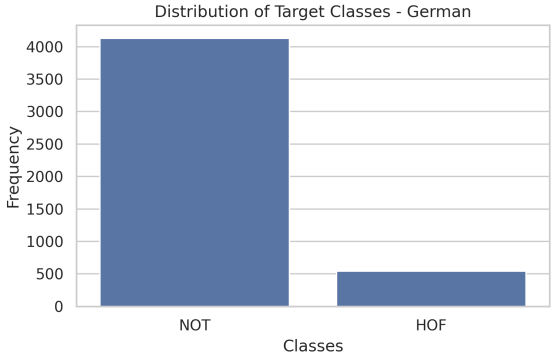
With regard to the size of the texts, the German dataset's distribution indicates that it usually falls between 20 and 60, with some outliers as well. This length is typical of replies on social networking sites, where users write lengthy comments in German, as German is a language that allows the formation of more complex structures.

The Arabic dataset is composed of many short and concise Tweets, the vast majority within

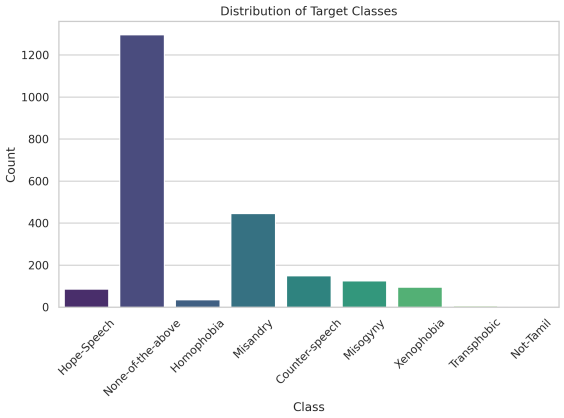

(a) English Dataset




(b) Arabic Dataset




(c) German Dataset




(d) Tamil Dataset

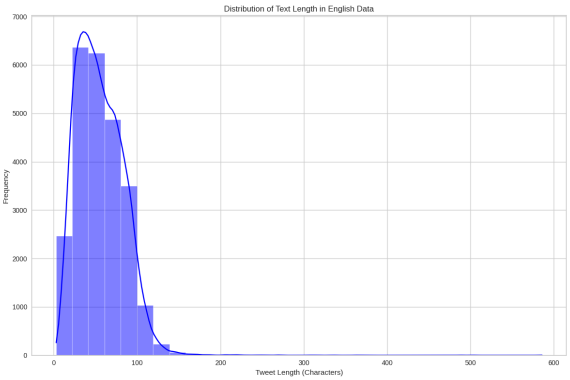
Figure 3.1: Class Distribution of samples on all four datasets

the 30-word limit, particularly fitting a social media environment. The statistical measures of text length are important for defining such parameters as tokenization and padding in neural networks.

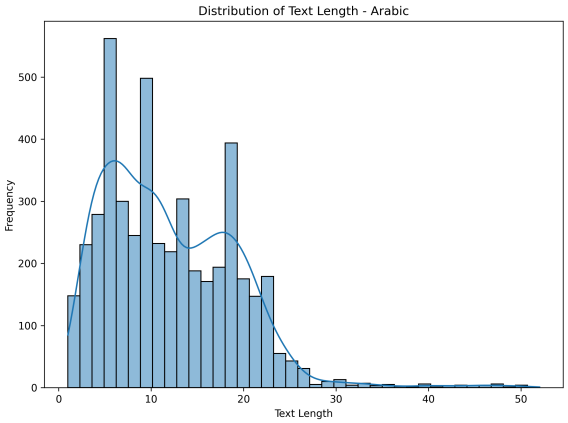
### 3.4.3 Distribution of Text Length by Class

Boxplots in Figure 3.3 show the distribution of text length over several target classes in text classification and shows how the length varies with the class label.

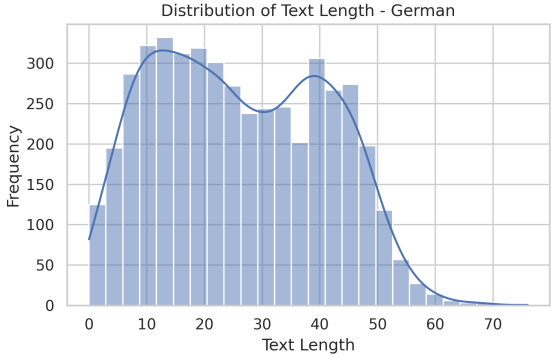
In the case of Tamil data, the text in the ”None-of-the-above” class can be generally longer than in such categories like ”Homophobia,” which contain shorter texts. This manifold is good for analyzing the relationship, if any correlation exists, between the comment’s length and the hate speech categories.



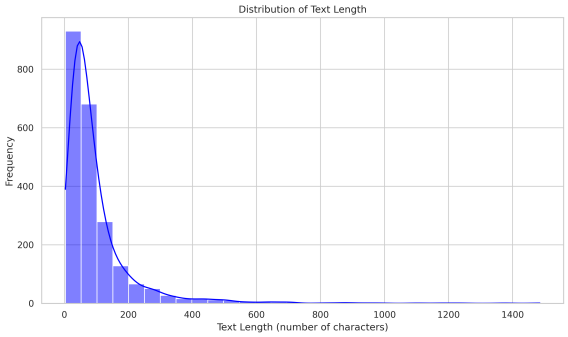

(a) English Dataset




(b) Arabic Dataset




(c) German Dataset



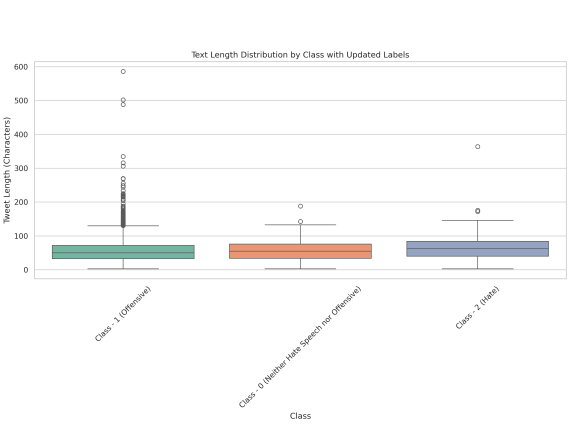

(d) Tamil Dataset

Figure 3.2: Text Length Distribution of samples on all four datasets

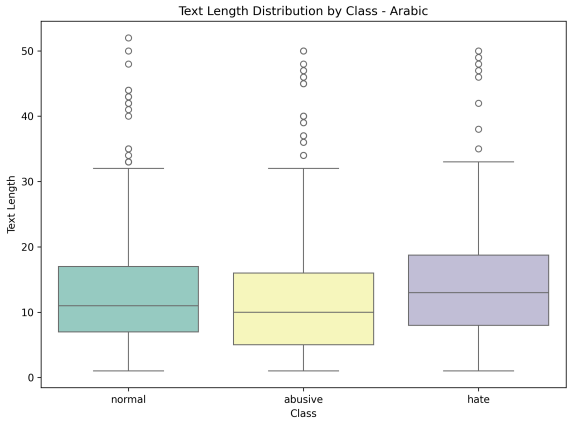
In regards to the English dataset, it appears that offensive speech has a slightly greater median paragraph length than both hate speeches and neutral comments. The boxplots pointers to this slight difference, showing the intricacies of offensive and hate speech in this case.

In the case of the German dataset, it has been noticed that comments are different in their length depending on the class. In this scenario, for instance, “None-of-the-above” entries typically take longer to comment on, while those of the class “Xenophobia” submit shorter texts. It is possible that these variations will affect the tokenisation and embedding of the text for machine learning models.

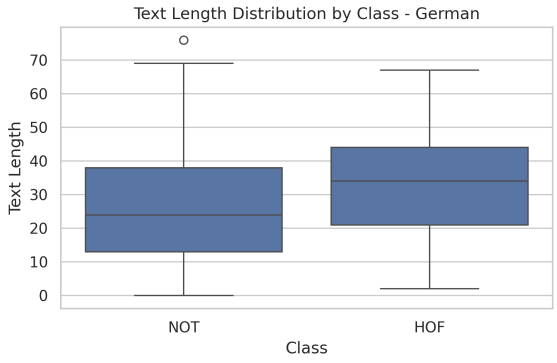
For the Arabic dataset, similar patterns emerge, where “Normal” class tweets are likely to have a higher number of words than “Abusive”, and “Hate” class tweets. This variation in text length may be attributed to the sociolinguistic aspects of hate talks in various languages.



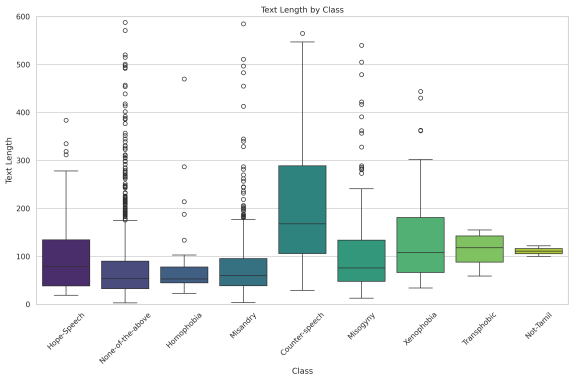

(a) English Dataset




(b) Arabic Dataset




(c) German Dataset




(d) Tamil Dataset

Figure 3.3: Class-based Text Length Distribution of samples on all four datasets

### 3.4.4 Word Cloud

In every class of the four datasets, word clouds have been produced in order to display the prominent words. Word clouds in Figure 3.4 give a general overview of the most important words among different documents or classes and help in determining words that are thematically related or frequently mentioned in a text corpus.

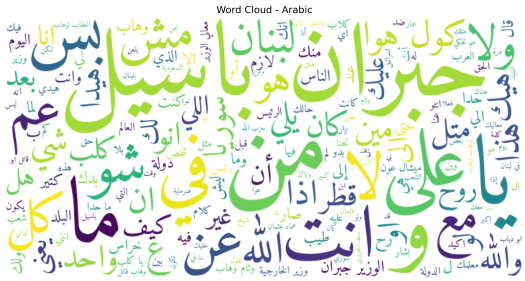
The Tamil dataset comprises a word cloud that thus shows that the social identity and gender-related terms were more common in classes “Misogyny” and “Misandry”. These visualisations are helpful in examining the frequent words used in various categories of hate.

The English dataset word cloud for ‘Offensive’ and ‘Hate Speech’ illustrates the vocabulary commonly used in hateful speech such as deriding or insultive words. The neutral class also contains those words as pronouns, connectors, and other basic vocabulary words.

The word clouds of the German dataset present common words in the 'None-of-the-above', and the hate speech categories which help in differentiating various speech types. There is also evidence in some categories of words that may evoke some counter speech or counter hate speech attempts.

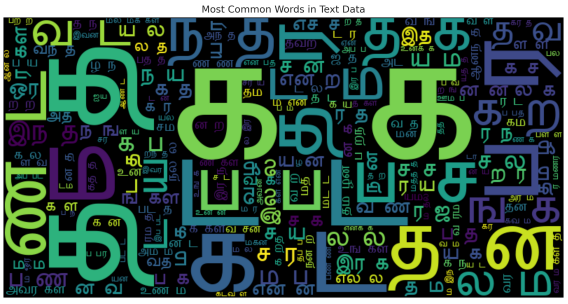
In the case of the Arabic dataset, word clouds are helpful in identifying the common words used in normal speech compared to hate speech and abusive speech with most words related to abusive language used in the abusing class. Taking into account the nature of the dataset, word clouds also allow making qualitative evaluations of the content, revealing the most common terms related to hate speech.


(a) English Dataset




(b) Arabic Dataset


(c) German Dataset




(d) Tamil Dataset

Figure 3.4: Generated Word Cloud on all four datasets

### 3.4.5 Bigrams and Trigrams

For examining word pairs and triples assists in the detection of frequent collocations present in the corpus, a necessary tool in comprehension and syntactical arrangement. In this methodology, CountVectorizer is used to obtain prevalent bigrams in Figure 3.5 and trigrams in Figure 3.6 from the given textual data.

For the Tamil dataset, usage of gender-related violence abuse words and derogatory bigram expression can be seen dominantly in the "Misogyny" and "Misandry" classes. From the trigrams angle, they indicate the combinations of abuse words that often appear in abusive remarks.

Hate and offensive bigrams and trigrams in the English dataset also include the common usage of targeted words or phrases that involve verbal violence. For example, certain identity-related derogatory terms used in combinations and hyphenated in some instances, are common in hate speeches.

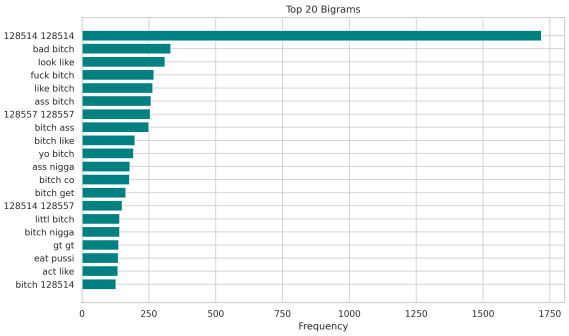
In the case of the German dataset, it was noticed that some bigrams and trigrams are often encountered in text in the xenophobic or homophobic contexts. This analysis aids in not merely spotting lone words but also entire strings of words that can be employed in such bad contexts.

In the case of the Arabic dataset, particular phrases consisting of bigrams and trigrams appear in a number of abuse and hateful tweets. Learning such phrases is useful in improving models of context when training the classifiers because in some cases, rather than looking at single words, combining them in particular ways are more predictive of hate speech.

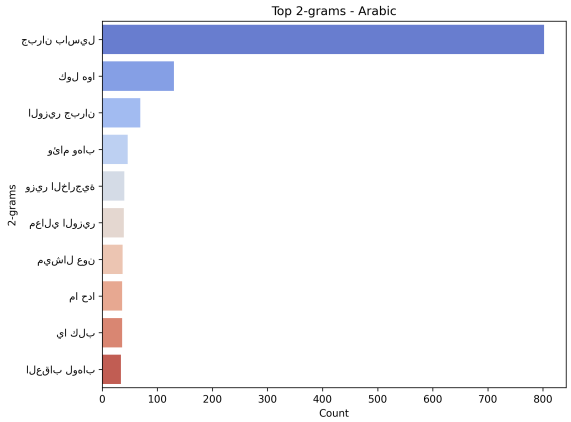
In a detailed analysis of the identified datasets in Tamil, English, German and Arabic, this study presents useful information about the layout and organization of the data. Such outcomes are essential for the next step model advancement and its evaluation.

## 3.5 Models

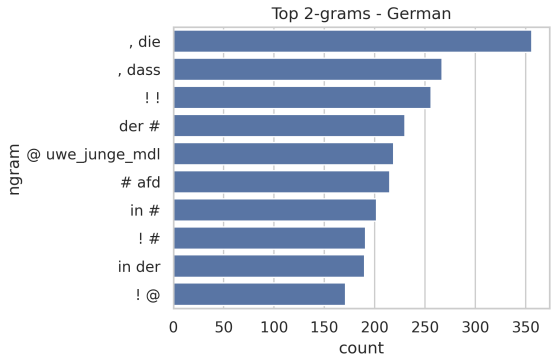
This section details the comparative analysis of several machine learning, deep learning and transfer learning models implemented in the recognition of hate and offensive speech in Tamil, Arabic, German and English. The choice of models is primarily influenced by their earlier performance in text classification tasks and their ability to manage the intricacies associated



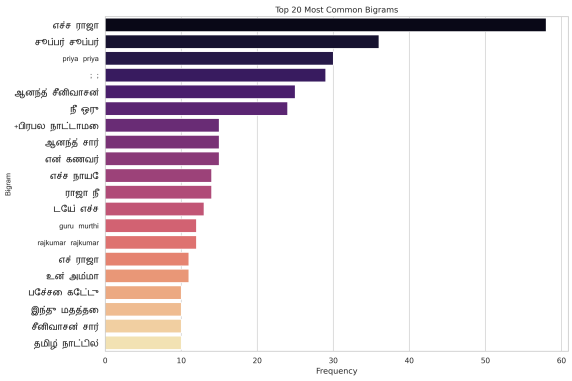

(a) English Dataset




(b) Arabic Dataset




(c) German Dataset




(d) Tamil Dataset

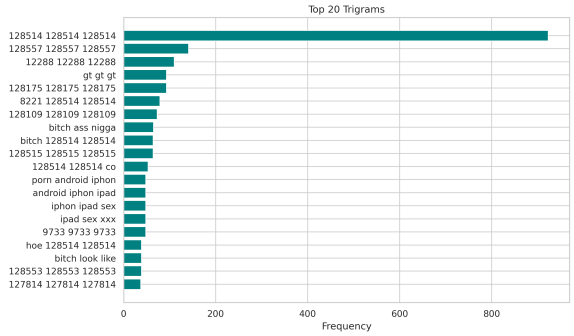
Figure 3.5: Top Frequency Bigrams on all four datasets

with hate speech detection in various languages.

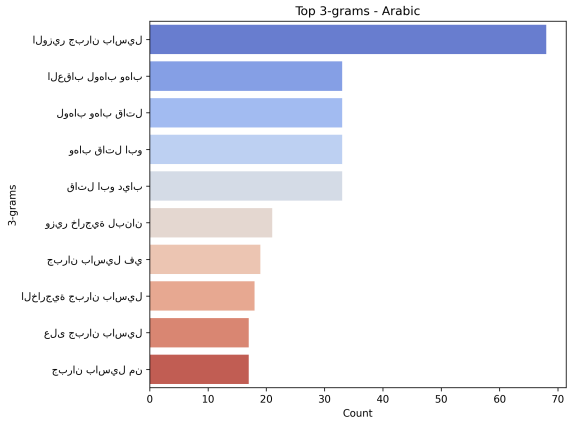
### 3.5.1 Machine Learning Models

#### Support Vector Machine (SVM)

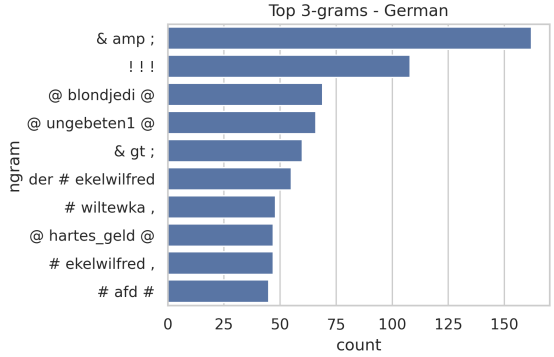
Support Vector Machines (SVMs) are well-suited for both binary and multiclass classification tasks, especially in high-dimensional feature spaces such as those created from text vectorization techniques like TF-IDF. In this project, LinearSVC was implemented to handle hate and offensive speech detection across Tamil, Arabic, German, and English datasets. SVM's capability to work efficiently with sparse matrices made it a strong choice for text classification problems, especially when using character and word n-gram representations.



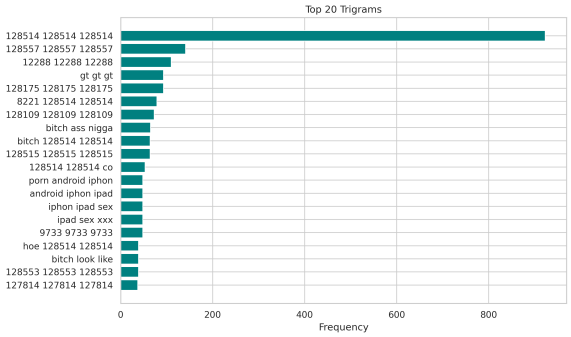

(a) English Dataset




(b) Arabic Dataset




(c) German Dataset




(d) Tamil Dataset

Figure 3.6: Top Frequency Trigrams on all four datasets

### Logistic Regression

Logistic Regression was employed as a baseline machine learning model across all datasets. Known for its effectiveness in multiclass classification problems with well-structured and high-dimensional data, it served as a fundamental model, particularly useful in handling the German and Tamil datasets. Its linear nature made it suitable for initial comparisons before deploying more complex models.

### Multilayer Perceptron (MLP)

The Multilayer Perceptron (MLP), a type of feedforward neural network, was used to capture non-linear relationships in the data. TF-IDF features were fed into the MLP for all four languages. The inclusion of hidden layers allowed the model to learn subtle distinctions in the text,

making it valuable for handling more nuanced classifications in languages like Tamil and Arabic. Cross-validation and hyperparameter tuning were applied to prevent overfitting, particularly in smaller datasets.

#### Random Forest

Random Forest, an ensemble learning model, was chosen for its ability to reduce variance and enhance generalization. Its use of multiple decision trees allowed for better handling of noisy data and unbalanced classes. This made it suitable for detecting hate and offensive speech in challenging text data, particularly in the Arabic and Tamil datasets, which contained significant variation.

#### K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a distance-based classification algorithm that assigns categories based on the proximity of data points to their nearest neighbors. Though computationally expensive for large datasets, KNN was included in the evaluation, particularly for the smaller Tamil and Arabic datasets, to explore its potential for detecting outliers in hate and offensive speech detection. However, its reliance on distance metrics in high-dimensional TF-IDF spaces posed challenges in these contexts.

### 3.5.2 Deep Learning Models

#### Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) were used for the English and Arabic datasets to model the temporal dependencies present in textual data. By accounting for word order and context, RNNs can capture sequential information that is essential for detecting hate speech embedded in longer text. Character-based embeddings were explored, though RNNs generally struggled with longer texts due to challenges such as the vanishing gradient problem.

**Bidirectional LSTM (Bi-LSTM)**

Bidirectional Long Short-Term Memory (Bi-LSTM) networks were employed to improve sequence comprehension by processing the text in both forward and reverse directions. This approach was particularly beneficial for the Arabic language, where key information may be distributed across the sentence. Bi-LSTM's ability to capture dependencies across the entire sequence allowed for a more comprehensive understanding of complex cases of hate speech.

**BERT (Bidirectional Encoder Representations from Transformers)**

BERT, a transformer-based language model, was used as the primary model for hate and offensive speech detection across all datasets. Its pre-training on large-scale text corpora, combined with the ability to understand bidirectional context, made it highly effective for capturing the nuances in hate speech. BERT was fine-tuned on specific datasets, leveraging its multilingual capabilities to handle the syntactic complexities of languages such as Arabic and German.

# Chapter 4

# Implementation

## 4.1 Environments

### 4.1.1 System Parameters

The models in this project turned out to be memory-intensive, especially during the phase of training RNN, Bi-LSTM and BERT deep learning models. For datasets in many languages such as Tamil, Arabic, German and English, at least 16 GB of RAM were crucial for proper training and preprocessing. With BERT models, it was 32 GB of RAM and more to enable use of all the specified memory and perform computations on longer text sequences efficiently. In order to speed up the training of the models, NVIDIA GPU types such as A100, T4, and L4 were used. The incorporation of these GPUs made parallel computations more efficient, especially for complex models such as BERT and Bi-LSTM, where training was completed in a shorter period when vigorously compared to for a CPU based computations. The utilization of GPUs helped in the efficient management of the large volumes of data and the computationally demanding NLP tasks. The project also employed powerful CPUs for tokenization, stopword deletion, and feature extraction through TF-IDF techniques. The minimum requirement of processors cannot be ignored, which are at least 8-core to process the data while using traditional models like Logistic Regression, SVM that are more compute-intensive.

### 4.1.2 Python Version

The project was carried out using Python 3.7 or higher versions in order to keep working with important libraries: Tensorflow, Pytorch, Transformers and Scikit-Learn. Python version 3.7+ also enabled modern optimization and performance needed to train and deploy both ML and DL models efficiently.

### 4.1.3 Libraries

There were many essential libraries used in the project which helped in data preparation, operation in particular model and assessment of the different languages and models used.

#### TensorFlow and PyTorch

Deep learning models have been designed using both TensorFlow and PyTorch (RNN, Bi-LSTM and BERT). These libraries also provide rich, high-level application program interfaces to create and train deep networks. The TensorFlow library was used as Keras was integrated into it for simplifying model development. On the other hand, PyTorch was used for quick model development because of its dynamic computation graph, which is a big plus while training the model as it is easier to debug.

#### Transformers

The Transformers library developed by Hugging Face played an important role in the BERT model implementation in the project. Implementation of hate speech detection system was facilitated by pre-trained transformer models provided by the library and tuning them on relevant tasks. The multilingualism of BERT made it very suitable to work with Tamil, English, German, and Arabic datasets.

#### Scikit-learn

Scikit-learn was an extensible library for Machine learning algorithms such as Logistic regression, SVMs, Random forest, and KNN. It would be utilized for things such as data preprocessing (for example, TF-IDF vectorization, Label Encoding) as well as model training. Its reasonably

mature algorithms as well as cross-validation tools such as metrics enabled dependable performance on classical ML models.

**NLTK (Natural Language Toolkit)**

The NLTK was largely employed for splitting text into smaller parts, taking out words that don't add much meaning, and preparing the text for further actions. The use of this library was demonstrated above while preparing text in English, but there were advanced options for enabling tokenization and preparing other languages as well.

**Matplotlib and Seaborn**

Visualizations were created using Matplotlib and Seaborn in order to perform data exploration using different graphs such as those showing class distribution and text length distribution as well as word clouds. These illustrations revealed certain patterns found in the datasets and they contributed to understanding how the models performed.

## 4.2 Model Parameters

The utilized parameters have been analyzed in the models developed for detecting hate and offensive speech across the different datasets in Tamil, Arabic, English and German languages.

### 4.2.1 Epochs

The number of epochs is relevant to the training of the model in that it indicates the amount of times the model will traverse through the whole training dataset during the training phase. When training RNN, Bi-LSTM or BERT deep learning models, the epoch value was between 5 and 30 depending on the data set and the model. In the case of the Arabic and Tamil datasets, the last epoch for Bi-LSTM and similar models was 5 in order to avoid overfitting while still keeping in mind a reasonable time spent training. On the other hand, comparatively easier machine learning methods like Logistic Regression, SVM, and Random Forest were mostly trained until convergence where any more epochs after a certain physiological time pose no

further performance increment. In the case of machine learning models, the number of epochs is not as concise as that, except for neural networks (MLPClassifier) the code specifies a limit of 200 iterations which is seemingly the number of epochs.

### 4.2.2 Batch Size

Batch size is the number of training samples a model takes before changing its internal parameters. When training deep learning models like RNN, Bi-LSTM and BERT, a batch size of 32 is a constant number across all the datasets. This is quite a reasonable size that encourages faster processing of the model without straining the memory capabilities. For example, in the Arabic and Tamil datasets, RNN, Bi-LSTM and BERT architectures were employed and had a batch size of 32 as seen in code. For other statistical learning such as SVM, Logistic Regression, Random Forest and KNN for example, such sizes do not apply the use of batch since they are not designed for such models. Rather, such models go through the entire data set or its part in one instance.

### 4.2.3 Learning Rate

The learning parameter manages the extent of modification on the model associated with the predicted loss in the process of training. In my construction of neural network models: RNN, Bi-LSTM and BERT, a variable learning rate was utilized when using the optimizer Adam. Adam optimizer defaults to a learning rate of $1 \times 10^{-5}$ usual in all my implementation aiming at letting the models converge fast without skipping towards the optimal loss for the models. Such approaches, as in the case of Logistic Regression or SVM, do not include the learning rate explicitly; rather it is somehow 'hidden', in a way that it is limited by the size of the solver action when minimizing the Logistic Regression for instance using a 'saga' solver. On these models, the code does not state any 'Custom Learning Rates' per se but simply takes advantage of the standard methods of learning rate in the saga solver and the likes.

### 4.2.4 Optimizers

It is essential to have optimizers as they allow modification of the model weights depending on the gradients. In the case of deep learning models, the Adam optimizer was implemented across the given datasets (Tamil, Arabic, German, and English) because of its adaptive learning rate and competence in large amounts of data. Since Adam integrates the strengths of RMSProp and Stochastic Gradient Descent (SGD), it is very popular with NLP tasks. It was noticed that for models such as BiLSTM and RNN that incorporated Adam, this led to faster convergence and minimal oscillations during the training process. Whereas for machine learning models such as Logistic Regression and SVM, such an optimization is built-in within the models. For instance, A solver called a 'saga' is used in the case of Logistic Regression because it is efficient when applied to big datasets that might also be sparse as in my representation of the text by TF-IDF. Random Forest and KNN will not be optimized in a classic way since they are based on trees splits and distance metrics, respectively.

### 4.2.5 Loss Functions

The loss function is an important part in training both machine learning and deep learning models. It measures how far off the prediction is from the true label and helps during the optimization process on how to modify the model weights to enhance prediction. Among hateful and offensive speech detection models, derivable electronic models primary loss functions applied are categorical cross-entropy for RNN and Bi-LSTM models and log-loss for Logistic Regression models.

#### Categorical Cross-Entropy Loss (For Deep Learning Models)

The Categorical Cross-Entropy is a loss function that is widely utilized in multiclass classification problems which concerns the classification of a single input in one out of multiple possible exclusive classes. In my case, the target classes might be "Hate Speech", "Offensive Speech" and "Neither" and the model output contains all three classes in the form of probability distribution across these classes.

**Mathematical Formula**: For a given training example with true class label $y$ and predicted

probabilities $p$, categorical cross-entropy is defined as:

$$L(y, p) = -\sum_{i=1}^{C} y_I \log(p_i)$$

**Where**:

- $C$ is the total number of classes.
- $y_I$ is the true label, which is 1 if the class is correct and 0 otherwise.
- $p_I$ is the predicted probability of the class $I$, output by the model.

In this formula, the negative logarithm of the probability $p_I$ estimated for the true class is taken and multiplied by the actual class label $y_I$, which is 1 for the actual class and 0 otherwise. It amounts to adding the loss over all the available classes and due to the negative sign the function penalises the model when its predicted probability for the true class is low.

**Interpretation:**

- In the event that the model estimates a high probability (closer to 1) towards the true class species then the log term approaches the figure of zero thereby minimizing the loss.
- In the unusual scenario where the model gives a low probability (closer to 0) towards the true class species, then this log term becomes quite large resulting in an excessive increase in the loss.

In real-time applications, RNN and its variants such as LSTM and Bi-LSTM use this loss function in model training wherein gradients are backpropagated and the weights of models are updated using optimizers such as Adam. A categorical cross-entropy loss is appropriate for hate and offensive speech detection tasks since the structure of such models requires predicting a probability over a number of classes.

**Logarithmic Loss (Log-Loss) for Logistic Regression**

In machine learning models such as Logistic Regression, the log-loss is used as a performance measure. The log-loss works well in binary or multiclass classification problems and guarantees

that models will foul up wrong predictions.

**Mathematical Formula:** In the case of a binary classification task (which can, however, be generalized into a multi class situation) the formula of log-loss is given as:

$$L(y, p) = -\left(y \log(p) + (1 - y) \log(1 - p)\right)$$

**Where**:

- $y$ is the true label (1 for the positive class, 0 for the negative class).
- $p$ is the predicted probability for the positive class.

In the multiclass case, the log-loss is a generalization of binary log-loss and closely resembles the categorical cross-entropy loss:

$$L(y, p) = -\sum_{i=1}^{C} y_I \log(p_i)$$

**Where**:

- $C$ is the number of classes, and $y_I$ is the true label (1 for the correct class, 0 for others).
- $p_I$ is the predicted probability for class $I$.

Log-loss heavily penalises models that are overconfident in their incorrect predictions (meaning if the model predicts the wrong category, for example, a high probability of 0.9, it would result in a high log loss value). Therefore, log-loss proves to be capable of assessing the model applied since it takes into account not only whether the prediction made is correct or not but also how confident that prediction is.

**Interpretation:**

- In the case of correct class predictions with high confidence, the log term become small and as a result, the global loss gets minimised.
- On the other hand, when a class is predicted wrongly or very low confidence is assigned to the correct class then log-loss goes very high.

**Hinge Loss for SVM (Support Vector Machine)**

In Support Vector Machines (SVM), the hinge loss is used for the purpose of training the model. Basically, it has been constructed for use with margin-based classifiers, which are designed to increase the distance separating the decision boundary from the closest point of the data (support vector).

**Mathematical Formula**: In the case of binary classification, hinge loss is mathematically expressed as.

$$L(y, f(x)) = \max(0, 1 - yf(x))$$

**Where**:

- $y$ is the true label (+1 for the positive class and -1 for the negative class).
- $f(x)$ is the decision function of the SVM model.

**Explanation**:

- If the predicted class $f(x)$ is correctly classified and lies beyond the margin (i.e., $yf(x) \geq 1$), the loss is 0.
- If the prediction is incorrect or lies within the margin (i.e., $yf(x) < 1$), the model incurs a positive loss.

The hinge loss seeks to classify the points correctly but makes sure that the distance from the decision boundary is adequately large to enhance generalisation.

# Chapter 5

# Evaluation

This chapter offers a systematic assessment of the models and techniques used in the project concerning their evaluations in cross-lingual and multi-experimental settings. The appraisal seeks to be able to answer the second research question satisfactorily by utilizing the various evaluation methods employed in the first question and the experimentation conditions such as data augmentation.

To begin with, an elaboration of the evaluation metrics has been adopted in the assessment of the model performance over four languages—Tamil, English, German, and Arabic, concentrating on critical metrics such as accuracy, precision, recall, and F1 scores. These parameters are crucial in making an assessment of comparatively advantageous areas of classical and deep learning models as well as transformer-type models.

Thereafter, model performance has been compared (Research Question 1) across the different datasets and the question of how the models deal with the problem of hate and offensive speech in different languages have been addressed. The extent of the data augmentation have been investigated (Research Question 2) by looking at how the amount of training data influences the results of the model for all languages.

## 5.1 Evaluation Metrics

Many evaluation metrics were used to measure the performance of machine learning, deep learning, and transformer models used in the project. These metrics provide an insightful overview

regarding the accuracy of the models on identifying hate and offensive speech in four languages: Tamil, English, German and Arabic. Consequently evaluation software was employed with the following parameters: Accuracy, Precision, Recall and F1-Score.

### 5.1.1 Accuracy

Among all measures, accuracy appears to be the most simple and straightforward to understand, especially considering its application in classification tasks. It is a ratio of the number of correctly predicted responses (both positive and negative) out of the total predicted by the model. Still, accuracy can be an inappropriate measure in the presence of class skew; this applies to hate speech detection where negative class, e.g. 'normal' often dominates.

**The formula for accuracy is:**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

**Where**:

- **TP** (True Positives): Correctly predicted positive samples
- **TN** (True Negatives): Correctly predicted negative samples
- **FP** (False Positives): Incorrectly predicted positive samples
- **FN** (False Negatives): Incorrectly predicted negative samples

### 5.1.2 Precision

Precision assesses the correctness of positive predictions made by the model. It is very important in hate speech detection, as false positives (normal content identified as hate) amplify the chances of overkill measures. Therefore, precision is prioritized when the repercussions of false positives are steep.

**The formula for precision is:**

$$\text{Precision} = \frac{TP}{TP + FP}$$

**Precision helps answer the question:** Of all the predictions that were classified as hate or offensive, how many were correct?

### 5.1.3 Recall

Recall, also known as sensitivity or true positive rate, determines how successful the model is in detecting positive samples. In the case of hate-speech detection, it refers to the amount of hate or offensive content that was detected as hate or offensive. High recall is important when the target is to reduce the number of false negatives since it guarantees that the majority of harmful content is captured.

**The formula for recall is:**

$$\text{Recall} = \frac{TP}{TP + FN}$$

**Recall answers the question:** Of all the actual hate or offensive content, how much did the model successfully identify?

### 5.1.4 F1-Score

The F1-Score aims to achieve a trade-off between precision and recall hence this is effective in situations with uneven class distribution. It is the arithmetic average of precision and recall thus all aspects are adequately evaluated finally. Unlike accuracy, F1-Score is a better metric when there are different weights associated with false positives and false negatives.

**The formula for the F1-Score is:**

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

This metric is particularly useful for hate speech detection because it balances the need to reduce both false positives (unnecessary censorship) and false negatives (missed harmful content).

## 5.2 Comparative Model Performance Across Languages (Research Question-1)

The first research question seeks to evaluate the performance of SVM, LR, RF, KNN, and MLP models along with three advanced models, which are RNN, Bi-LSTM and BERT. The assessment was carried out on four different datasets, namely, Tamil dataset, English dataset, German dataset and Arabic dataset. The results encompass various performance metrics in the form of accuracy, precision, recall and F1 scores.

### 5.2.1 Tamil Dataset

#### Machine Learning Models

Considering SVM for Tamil dataset from the Table 5.1, it was the best among other ML models with the accuracy of 70.53% and F1 score of 0.6761. The reason for SVM comparatively higher performance than other models can be attributed to its efficiency in separating hate speech, abusive and normalised comments in the given text. Since the Tamil language has many complex features, such as a variety of word endings and intricate grammar, most models like KNN and RF did not perform well in classification, but SVM has excelled because of its robust high-dimensional feature spaces. Logistic Regression on the other hand did not excel much, it was a little behind the best performer yet still earned a respectable accuracy of 66.61% and F1 score of 0.6587 by simply utilizing the fact that the data was separable to some extent. KNN performed poorly on every metric, obtaining the lowest accuracy of 55.17%, which indicates how poorly the proximity-based algorithms do within the Tamil dataset. RF had an adequate performance but except in terms of accuracy which was reduced because of overfitting when the features in the text were few which is commonly occurring in Tamil.

#### Deep Learning Models

In deep learning model performance from the Table 5.1, RNN reports slightly better accuracy, 68.39% compared to Bi-LSTM. The F1 score for RNN 0.6597 is higher than Bi-LSTM which suggests that RNNs though quite simplistic, are able to learn long-term dependencies

well enough in Tamil. However, BERT recorded the lowest performance suggesting that there are issues with transformer structure especially in reference to the Tamil language. This F1 score of 0.59 for BERT shows one of the reasons that low performance is due to the lack of adequate pre-trained resources for Tamil language. Another limitation with BERT is the fact that there is a high reliance on transfer learning which works for high-resource languages such as English. Some subtle aspects of Tamil seem to pose difficulty for BERT; one of the greatest weaknesses of the model is its rather superficial approach to multilingual pre-training.

| Model | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| **Machine Learning Models** | | | | |
| LR | 66.60% | 0.6597 | 0.6660 | 0.6587 |
| SVM | 70.53% | 0.6674 | 0.7053 | 0.6716 |
| RF | 68.03% | 0.6304 | 0.6803 | 0.6253 |
| KNN | 55.17% | 0.6196 | 0.5517 | 0.5679 |
| MLP | 66.96% | 0.6560 | 0.6696 | 0.6590 |
| **Deep Learning Models** | | | | |
| RNN | 68.39% | 0.6494 | 0.6839 | 0.6597 |
| Bi-LSTM | 67.85% | 0.6327 | 0.6785 | 0.6452 |
| BERT | 67% | 0.51209 | 0.65 | 0.59 |

Table 5.1: Performance Metrics for Machine Learning and Deep Learning Models on the Tamil Dataset

### 5.2.2 English Dataset

#### Machine Learning Models

Once again in the English dataset from the Table 5.2, SVM was the best classifier reaching 89.62% in accuracy and an F1 score of 0.8853. The explanation of SVMs success is with regard to their capability of processing high-dimensional feature vectors generated through TF-IDF vectorization which illustrates the difference between the offensive and non-offensive speeches in context of the English language. Also, Logistic Regression gave sensible results meaning that hate speech classification utilizing linear models is possible with relatively good text data preparation. Further analyzing the characters of model performance, random forest and MLP approaches were almost equal achieving nearly 0.9 F1 score and accuracy respectively. However, KNN was the least performer among ML models and still emphasizes that KNN-based distance

classification may not be effective in high-dimensional spaces where syntactic proximity loses its relevance compared to the semantic content of the words being classified.

#### Deep Learning Models

As English has many composed models available for language processing, it complimented BERT effectively. By leveraging the language understanding that BERT from the Table 5.2 had obtained beforehand, the model hit an accuracy of 77.67% with an F1 Measure of 0.7767, which illustrates the supremacy of transformers in high resource languages. In general, both RNN and Bi-LSTM did quite impressively, although they were not as effective as BERT. RNN and Bi-LSTM underperformed due to the lack of salient pre-training as possessed by BERT since these models cannot effectively handle long-range dependencies. The accuracy reports for Bi-LSTM were also good with Bi-LSTM achieving 77.44% while BERT managed to overtake this performance. This suggests that sequential models could build a structure out of the language data though transformer models reorients the word context and their inter relation better.

| Model | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| **Machine Learning Models** | | | | |
| LR | 89.15% | 0.8819 | 0.8915 | 0.8858 |
| SVM | 89.62% | 0.8852 | 0.8962 | 0.8885 |
| RF | 85.52% | 0.8405 | 0.8552 | 0.8327 |
| KNN | 80.76% | 0.8115 | 0.8076 | 0.8050 |
| MLP | 88.87% | 0.8769 | 0.8887 | 0.8814 |
| **Deep Learning Models** | | | | |
| RNN | 77.43% | 0.5999 | 0.7743 | 0.6759 |
| Bi-LSTM | 77.43% | 0.5996 | 0.7743 | 0.6759 |
| BERT | 77.66% | 0.6032 | 0.7766 | 0.6790 |

Table 5.2: Performance Metrics for Machine Learning and Deep Learning Models on the English Dataset

### 5.2.3 German Dataset

#### Machine Learning Models

In the case of the German dataset from the Table 5.3, most of the preprocessing was utilized by Random Forest which has produced optimal results (accuracy and F1 score both around 99.76%).

This is also due to the random forest's attribute of coping well with imbalanced classes and more so its nature of over fitting being less common due to the ensemble type of RF. The scores obtained by SVM were also reasonably good, but Random Forest's scope of generalization across varying features was the reason behind why RF did better than other models. In KNN, there has been improvement to its accuracy than what was noted in other datasets but none the less the figure stood at 86.24%. The efficiency of the ML models, especially that of Random Forest and SVM, is recommended especially in working with languages such as German which has well-structured and clear syntax and semantics.

### Deep Learning Models

It's impressive how Bi-LSTM yields excellent results in the German dataset from the Table 5.3. Its strength of capturing bidirectional dependencies proved quite effective in German leading to an F1 measure of 0.9926. Although Bi-LSTM outranked BERT, it is safe to say that Bi-LSTM is the best out of all in this case. One possible explanation for Bi-LSTM's high performance may be found in the structure of German which has certain complexities that can be easily captured by bidirectional models compared to transformer models. The ceiling effect for BERT at 84% indicates that the patterns learned through the pre-training phase enable the models to grasp contextual information; however, some of the more intricate structural aspects of the German language may require greater targeted approaches or more modes of understanding which Bi-LSTM simply offers.

| Model | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| **Machine Learning Models** | | | | |
| LR | 88.71% | 0.8285 | 0.8871 | 0.8494 |
| SVM | 89.22% | 0.7783 | 0.8822 | 0.9374 |
| RF | 99.76% | 0.9977 | 0.9976 | 0.9976 |
| KNN | 86.24% | 0.8564 | 0.8624 | 0.8138 |
| MLP | 99.76% | 0.9977 | 0.9976 | 0.9976 |
| **Deep Learning Models** | | | | |
| RNN | 81.38% | 0.7056 | 0.84 | 0.767 |
| Bi-LSTM | 99.29% | 0.993 | 0.9929 | 0.9926 |
| BERT | 84% | 0.7056 | 0.84 | 0.767 |

Table 5.3: Performance Metrics for Machine Learning and Deep Learning Models on the German Dataset

### 5.2.4 Arabic Dataset

#### Machine Learning Models

Once again, SVM has outperformed other ML models in this Arabic dataset from the Table 5.4 with an F1 score of 0.7489, where SVM is accepted as the best performing classifier. This is due to SVM's efficiency in such scenarios as small and skew datasets, where the feature space of hate vs non-hate speech may not be very well separated. Logistic Regression and MLP closely followed the pack, while Random Forest did worse than any other datasets compared possibly because of the aspects of Arabic syntax and semantics that are likely more feature dependent than what could be captured by decision trees. With an achievement of 66.49% in accuracy and F1 performance measuring 0.67, KNN showed the lowest performance.

#### Deep Learning Models

BERT had quite a hurdle in dissatisfaction in Arabic and particularly in smaller trained datasets or datasets that are morphologically rich, such as Arabic. From table 5.4 BERT score F1 equals 0.47 seems to be the evidence that this approach does not succeed in bridging the languages that are based on more resources. In line with that RNN and Bi-LSTM with F1 score of 0.47 were no better than BERT but performance results in general specify that more concrete or language-based enhancements may be required for Arabic to improve those figures.

| Model | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| **Machine Learning Models** | | | | |
| LR | 74.44% | 0.7396 | 0.7444 | 0.7417 |
| SVM | 76.83% | 0.7519 | 0.7683 | 0.7489 |
| RF | 72.64% | 0.7113 | 0.7264 | 0.6977 |
| KNN | 66.49% | 0.6792 | 0.6649 | 0.6701 |
| MLP | 73.41% | 0.7290 | 0.7341 | 0.7314 |
| **Deep Learning Models** | | | | |
| RNN | 62.39% | 0.3892 | 0.6239 | 0.4794 |
| Bi-LSTM | 62.39% | 0.3892 | 0.6239 | 0.4794 |
| BERT | 62.39% | 0.3893 | 0.6239 | 0.4794 |

Table 5.4: Performance Metrics for Machine Learning and Deep Learning Models on the Arabic Dataset

## 5.3 Impact of Data Augmentation on Model Performance (Research Question-2)

The second research question aims at investigating the effect of reducing the amount of the training data for the deep learning and transfer learning models (RNN, Bi-LSTM, and BERT) validated on the four datasets. The data augmentation was applied in 33%, 50%, and 75% splits to evaluate how well the model might withstand less data.

### 5.3.1 Tamil Dataset

#### RNN and Bi-LSTM

From the table 5.5 and 5.6, the performance of both RNN and Bi-LSTM was found to improve gradually and steadily with an increase in dataset size. Figure 5.1 shows a linear trend started on RNN with 66.96% accuracy at 33% data and culminated at 68.39% at 100% data. This rise continued on Bi-LSTM, where it began at 68.04% and then cut to 67.86%. The constant enhancement implies that the sequential two models are learning from a lot of training data and consequently are considered number-sensitive in Tamil. However, for BERT the situation was different as the model performed badly with small datasets starting with 48% accuracy at 33% of the data and climbing to 59% at 100% finally. It can be seen that BERT requires much larger data to perform effectively in Tamil likely due to its reliance on pre-trained embeddings that may not generalize well to Tamil.

| **1. Tamil Dataset - Accuracy** | | | | |
|---|---|---|---|---|
| **Model** | **33%** | **50%** | **75%** | **100%** |
| RNN | 66.96% | 66.79% | 67.32% | 68.39% |
| Bi-LSTM | 68.04% | 67.14% | 67.86% | 67.85% |
| BERT | 62% | 63% | 65% | 67% |

Table 5.5: Accuracy for different models on various splits of the Tamil dataset

| 1. Tamil Dataset - F1 Score | | | | |
|---|---|---|---|---|
| **Model** | **33%** | **50%** | **75%** | **100%** |
| RNN | 0.6223 | 0.6299 | 0.6315 | 0.6597 |
| Bi-LSTM | 0.6263 | 0.6317 | 0.636 | 0.6452 |
| BERT | 0.48 | 0.51 | 0.6 | 0.59 |

Table 5.6: F1 Scores for different models on various splits of the Tamil dataset

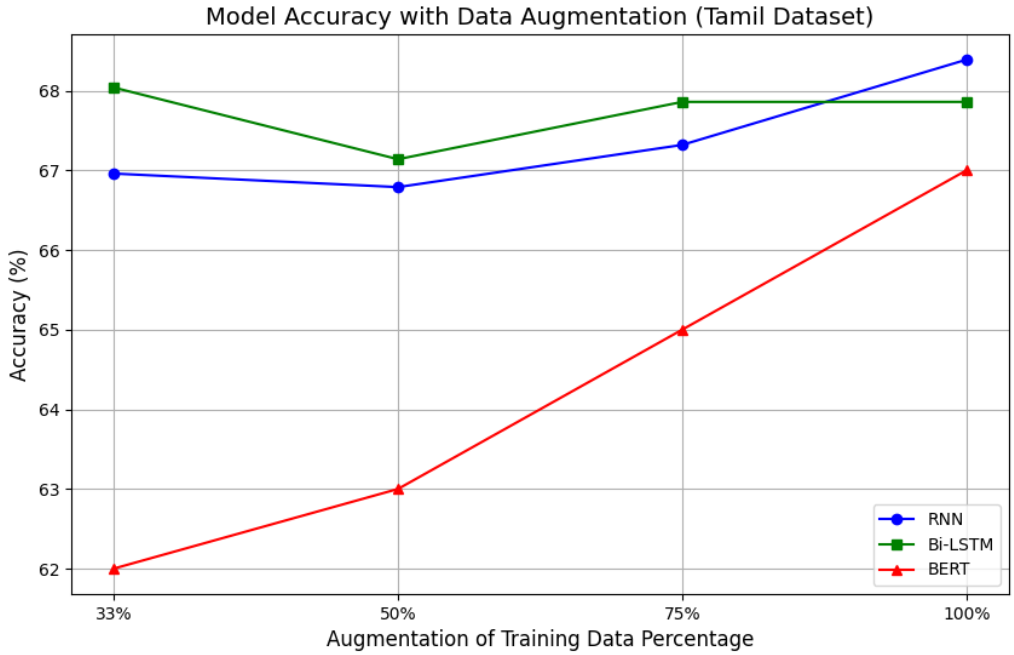



Figure 5.1: Model Accuracy with Data Augmentation (Tamil Dataset)

### 5.3.2 English Dataset

Among the models trained for English language from the Table 5.7 and 5.8, it can been see than in all data splits, the highest performance was recorded for BERT, as compared to RNN and Bi-LSTM. BERT achieved an accuracy of 77.57% with just 33% of the data and went on to reach 77.67% accuracy with 100% of the data. The fact that there is only a slight loss in the performance of models with a lesser amount of data shows that BERT is mainly designed for English language texts; this could be because BERT was first trained on many English texts before being fine-tuned. From the figure 5.2, the trends in accuracy were similar for Bi-LSTM and RNN, but with a smaller performance gap as shown by BERT. These trends suggest that Transformers perform better than sequential models even if they have been pre-trained in high-resource languages such as English with lesser training data.

| 2. English Dataset - Accuracy | | | | |
|---|---|---|---|---|
| **Model** | **33%** | **50%** | **75%** | **100%** |
| RNN | 76.58% | 77.40% | 77.34% | 77.44% |
| Bi-LSTM | 76.58% | 77.40% | 77.34% | 77.47% |
| BERT | 75.98% | 76.72% | 78.40% | 77.67% |

Table 5.7: Accuracy for different models on various splits of the English dataset

| 2. English Dataset - F1 Score | | | | |
|---|---|---|---|---|
| **Model** | **33%** | **50%** | **75%** | **100%** |
| RNN | 0.6642 | 0.6755 | 0.6746 | 0.6759 |
| Bi-LSTM | 0.6642 | 0.6755 | 0.6746 | 0.6759 |
| BERT | 0.6561 | 0.6662 | 0.6891 | 0.6790 |

Table 5.8: F1 Scores for different models on various splits of the English dataset

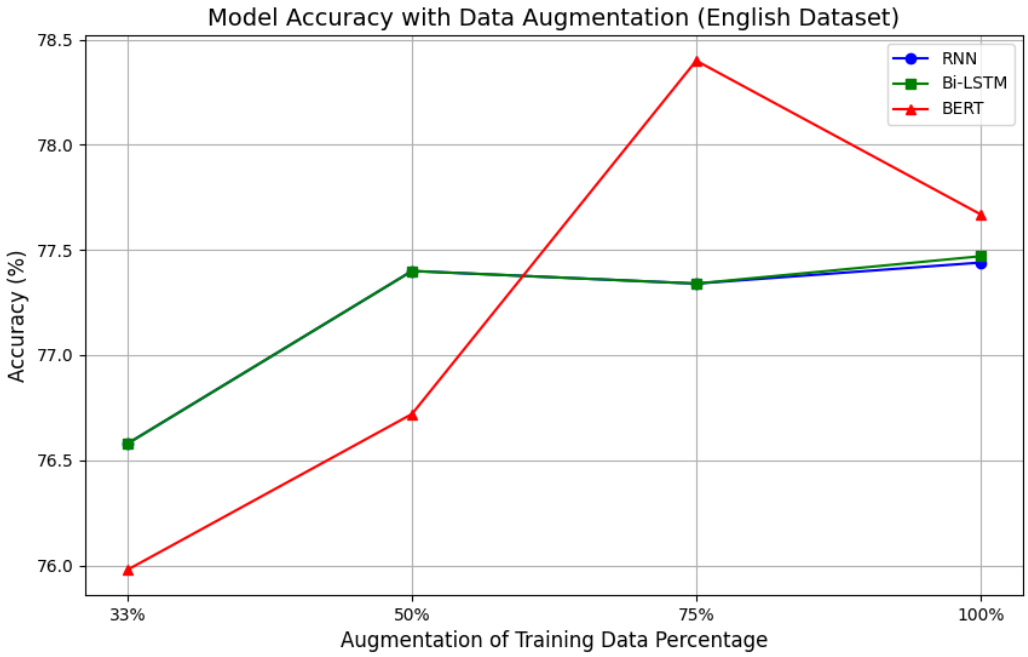



Figure 5.2: Model Accuracy with Data Augmentation (English Dataset)

### 5.3.3 German Dataset

In the German dataset from the Table 5.9 and 5.10, it has been noted that BERT as well as Bi-LSTM consistently performed better than RNN with increasing dataset size. Bi-LSTM achieved an accuracy rate of 99.26% with 100% of the data, whereas the accuracy rate of BERT did not go over 84% regardless of the size of the dataset. One possible reason why Bi-LSTM performed so well was because it was better at remembering long-term dependencies than BERT, which

appears to struggle due to the high syntactical nature of the German language. The observation from the figure 5.3 shows that Bi-LSTM performance improvement did not tapper off as the amount of data grew which amplifies the fact that the Bi-LSTM-based model is optimal for detecting German hate speech as it seems to effectively address the issue of sequence modeling.

| 3. German Dataset - Accuracy | | | | |
|---|---|---|---|---|
| **Model** | **33%** | **50%** | **75%** | **100%** |
| RNN | 84% | 84% | 84% | 81.38% |
| Bi-LSTM | 86.12% | 91.53% | 95.53% | 99.29% |
| BERT | 84% | 84% | 84% | 84% |

Table 5.9: Accuracy for different models on various splits of the German dataset

| 3. German Dataset - F1 Score | | | | |
|---|---|---|---|---|
| **Model** | **33%** | **50%** | **75%** | **100%** |
| RNN | 0.7670 | 0.7670 | 0.7670 | 0.7670 |
| Bi-LSTM | 0.8394 | 0.9035 | 0.9524 | 0.9926 |
| BERT | 0.7670 | 0.7670 | 0.7670 | 0.7670 |

Table 5.10: F1 Scores for different models on various splits of the German dataset

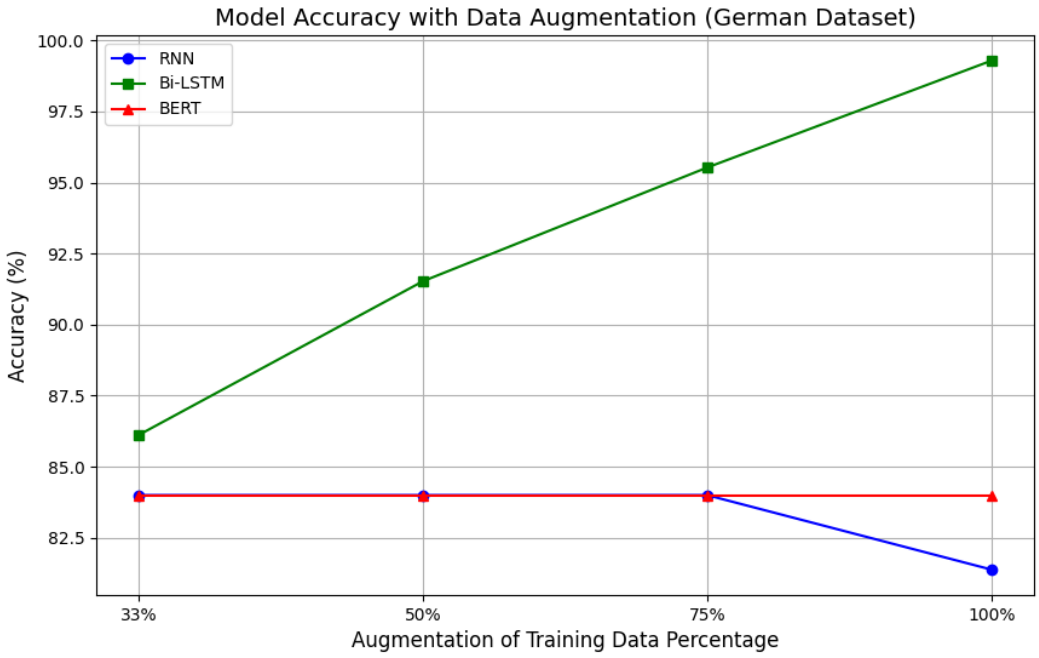



Figure 5.3: Model Accuracy with Data Augmentation (German Dataset)

### 5.3.4 Arabic Dataset

As was the case in Tamil dataset, RNN and Bi-LSTM models outperformed BERT model on Arabic dataset particularly when dataset was small as showcased in Table 5.11 and 5.12. In the scenario of 33% data size both RNN and Bi-LSTM reached the 62.39% accuracy while BERT achieved only 40.34% accuracy. For the complete dataset, BERT's performance turned out to be poor as well (accuracy of 62.39%). This means that the models with transfer learning such as BERT are not likely to work well for Arabic due to morphological richness without sufficient generic pre-training data. RNN and Bi-LSTM on the contrary proved to be more flexible toward the size of the datasets and steadily gained improvement as more data was added. From the Figure 5.4, there's a inconsistent performance of BERT on the Arabic dataset, where accuracy fluctuates as more data is introduced. The morphological complexity of Arabic may pose challenges for general pre-trained models like BERT, which may not be fine-tuned adequately for Arabic's rich inflectional structures.The significant drop in accuracy from 66.32% with 33% of the data to 40.34% with 50% of the data suggests overfitting. BERT may be overfitting to the small dataset and failing to generalize when the data increases, possibly due to learning spurious patterns from the smaller portion.

| **4. Arabic Dataset - Accuracy** | | | | |
|---|---|---|---|---|
| **Model** | **33%** | **50%** | **75%** | **100%** |
| RNN | 62.39% | 62.39% | 62.39% | 62.39% |
| Bi-LSTM | 62.39% | 62.39% | 62.39% | 62.39% |
| BERT | 66.32% | 40.34% | 64.53% | 62.39% |

Table 5.11: Accuracy for different models on various splits of the Arabic dataset

| **4. Arabic Dataset - F1 Score** | | | | |
|---|---|---|---|---|
| **Model** | **33%** | **50%** | **75%** | **100%** |
| RNN | 0.4794 | 0.4794 | 0.4794 | 0.4794 |
| Bi-LSTM | 0.4794 | 0.4794 | 0.4794 | 0.47944 |
| BERT | 0.5919 | 0.4380 | 0.569 | 0.4794 |

Table 5.12: F1 Scores for different models on various splits of the Arabic dataset

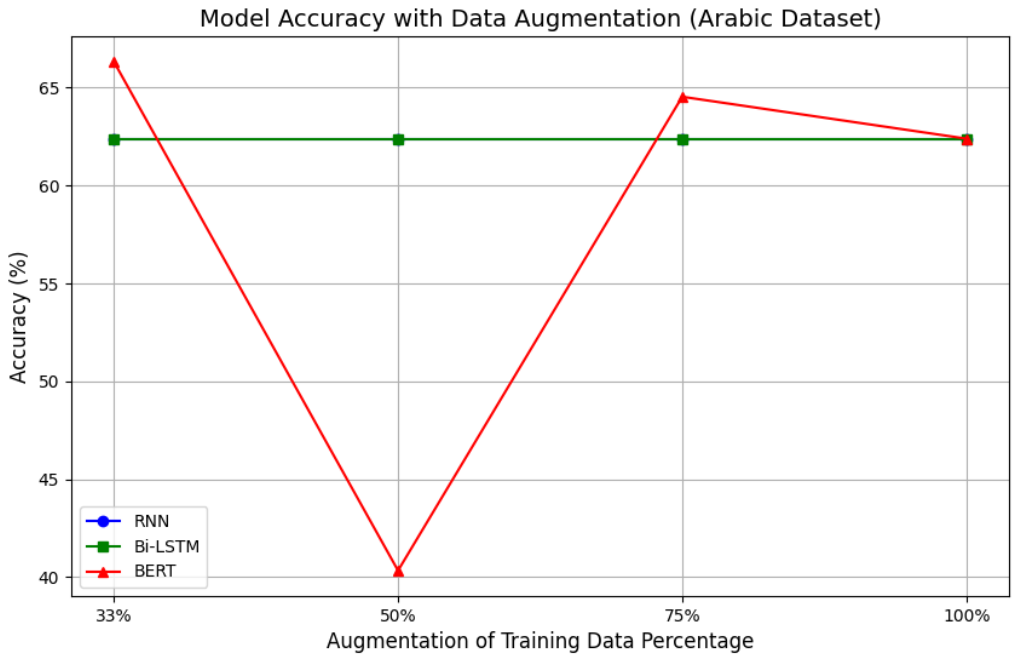



Figure 5.4: Model Accuracy with Data Augmentation (Arabic Dataset)

## 5.4 Discussion of Results

The trends that emerged from the comparison of traditional machine learning (ML) models with deep learning (DL) models and transfer learning models were surprising. Initially, it was presumed that the end results of deep learning models – RNN, Bi-LSTM and also those based on transfer learning such as BERT—would be better than the traditional machine learning models such as SVM, Logistic Regression or Random Forest due to the fact that neural networks work well with the matrices and span over a large context. But SVM and Random Forest which are considered as traditional ML models managed to outperform the majority of deep learning and transfer learning models in many datasets including Arabic dataset and German dataset where the deep models performed poorly.

A another major factor in this can be the characteristic of the text data itself. Practically conventional ML models with strong feature engineering techniques like TF-IDF vectorization perform reasonably well in text classification problems. These models are also capable of handling sparse and high-dimensional data efficiently with low computational complexity and do not need as much data as deep learning models do. Furthermore, conventional models, in this case being the SVM model, perform excellent in such situations in small datasets as they are not prone to

overfitting as much as neural networks do.

Models of deep learning have been more efficient in handling long-standing sequential dependencies and recognizing complex regularities, even though at the cost of higher data and computational requirements. This assumes the second most notable mechanism in shaping the outcome: Computational power. Although a powerful 40GB A100 GPU was available during the model training phase of deep and transfer learning, the time taken to train the models remained quite long. Under these limits, many precious optimizations on deep learning models could not be done, mainly because of the neural network layers and loss function tuning that could help improve the performance. Greater computational power would allow more room for adjustment for these learners, perhaps enhancing their performance.

Finally the conventional machine learning models were seen to outperform both deep learning and transfer learning models in this study, it is possible that with more appropriate computer resources and a more detailed and varied dataset, the deep learning models could be further tuned to give better performance. For example, increasing or decreasing the number of layers in the neural networks, using different optimizers, or applying additional regularization techniques are all ideas to improve model performance. For now, such computational restrictions and the small size of the datasets, in relative terms, were key factors limiting the deep and transfer learning models from achieving their full potential.

In this project, the only factors that hindered the performance of traditional machine learning models as compared to deep and transfer learning models were the size of the datasets, available computational power and plain text as input data for machine learning algorithms. While all of them performed quite well as ML models, resource optimization and collation of more compute power might change the scenario for neural network-based models, especially in the more complicated or larger set of text. The challenges regarding the performance of BERT on the Arabic dataset could be resolved by developing a BERT from scratch for Arabic. The general pre-trained BERT model is not qualified to deal with the issues of the isolated languages like Arabic, or German, and may not be able to grasp the intricacies of certain aspects of morphology of a language and syntax. This allows for a more suitable generalizations since it is an Arabic model pre trained on a wide linguistic data across various splits. More advanced data augmen-

tation techniques such as GANs or autoencoders may be employed in the effort of producing new and even more diverse sample training data. Such an action may help to diminish the data imbalance concerns and enhance the performance of the model. Lastly, there is a possibility of utilizing contingency techniques, integrating BERT with other types of models like RNNs or Bi-LSTMS, to harness the benefits of all features available. This would also make it possible to perform better on the complexity of Arabic and obtain less variance in performance on various data splits.

In conclusion, the analysis of research objectives 1 and 2 emphasizes that the strengths of machine learning, deep learning and transfer learning models differ across languages and size of data. In particular, traditional models such as SVM and Random Forest performed well with structured datasets such as German and English languages, while the deep learning models such as Bi-LSTM were very useful in languages with highly-inflected structures such as Tamil and Arabic. Although effective for English, BERT was less effective in low-resource languages with small size datasets. These works emphasize the need to use a suitable model considering the language, data resources and its complexity and therefore enhancing multilingual hate speech detection approaches.

# Chapter 6

# Conclusion and Future Work

## 6.1 Project Management

The project implemented appropriate project management processes in a logical sequential and timely manner in order to achieve the identified goals. The main phases of the project were subdivided into clear-cut phases with each of them dedicated to predominant activities such as data collection, data processing, modeling and assessment.

The initial phase began with planning and defining the project's goals: detecting hate and offensive speech across multiple languages using machine learning (ML), deep learning (DL), and transfer learning models. Given the diverse nature of these languages, the project required a tailored approach to data collection and model training. An agile approach was used to allow flexibility and continuous feedback, especially when moving between different datasets and modeling techniques.

For the data acquisition phase, text data from social networking sites such as Twitter and YouTube was obtained. The datasets varied in size where English comprised about 125,000 and where the Tamil, Arabic and German datasets had smaller datasets in the range of 5,000 to 6,000 rows. This difference in datasets was one of the most handled issue throughout the duration of the project.

Next, the preprocessing stage was very important to facilitate the model training on the multilingual data. Such processes as special character removal, lowercasing and various types of tokenization were implemented in a uniform manner on all the datasets. This phase also involved

language-specific preprocessing steps such as introduction of custom stemming/lemmatization methods in Tamil and Arabic that complicated the project. The cleaning of every dataset under consideration was performed thoroughly during pre-processing, since inconsistencies could impact model results.

In the model development stage, support vector machine, logistic regression, K-nearest neighbors, random forest, and multilayer perceptron neural networks were constructed and tested, along with recurrent neural network, bidirectional long short-term memory and BERT models on all datasets. The deep learning models especially BERT and BILSTM were computationally intensive with powerful GPUs being completely utilized.

During the evaluation stage, the performance of the models was assessed using metrics like accuracy, precision recall and F1 scores, aiming to obtain the best possible result. The project also carried out data augmentation techniques in order to determine how they influence model performance, especially with regard to small datasets. During this phase, a feedback loop was continuously in effect, allowing further iterations of the model design and evaluation processes to be pursued.

Finally, during the optimization and validation stage, the model performance was enhanced through hyperparameter tuning and grid search, although the latter was hampered by lack of computing power. In spite of the aforementioned challenges, the project delivered successful results, and set a basis for further work on the detection of hate speech in multilingual environments.

## 6.2 Challenges

There were some major problems that were faced during the implementation of the project. One of the main problems remained the high computational effort of the deep learning models such as BERT, RNN and Bi-LSTM. Even if there was a possibility to do training of the deep learning and transfer learning models on a very powerful 40GB A100 GPU, the duration taken in the training of these models was quite high. Training one dataset on average of 10 epochs which is the minimum takes over 50 hours, while testing per dataset takes about 3-7 hours. In view of these limitations, there were certain deep learning models that were not allowed to be fully tuned

especially with regards to the neural network layers and loss function tuning although they were potentially performance degrading. With more computational resources, it would be possible to further tune these models, beyond those already described to benefit the performance.

Another major consideration in the performance level of deep learning models and generalization to the problem is the size of the dataset. The English dataset contained about 25000 rows whereas the Arabic, Tamil and German datasets respectively ranged from about 5000 to 6000 rows each which is quite a small size for a text-based deep learning model. In such cases, data augmentation techniques tend to be less effective as these models need a lot of data to work effectively. It has been observed that in the case of the augmented datasets derived from it, there were 33%, 50%, and 75% of the data, the accuracy of the BERT model as well as models such as RNN and Bi-LSTM reached the maximum value or slightly increased. This implies that the size of the datasets restricted the models in learning the variations from the data which would have benefited them.

The optimization of the loss functions and the layers of the neural networks was performed under certain constraints of computational means. It is probable that the optimization of the models could be taken further and would bring about better performance if more extensive computational power and time were allocated.

## 6.3 Personal Reflections

On further retrospection on the project, one of my major reflections is that such big-scale companies like Amazon or Meta (YouTube, Twitter) would be able to take this project further as well. Given their enormous resources, the deep learning and transfer learning models may be further fine-tuned than what was achievable in this study. These companies, possessing large-scale infrastructure, would be up to the task of mounting the computational burden posed by models such as BERT and improving the project's potential in real-time detection of hate and offensive speech.

Moreover, working with multilingual datasets gave an interesting understanding of the language constructs, including the low-resource languages like Tamil and Arabic. Since there are no pre-trained models available for these languages, it became obvious that these areas will require

more resources and more time in order to obtain better results. This project has shown the advantages of the further investigations focusing on language-dependent models and the further progress which is essential in order to cope with bigger datasets and more sophisticated models. Nevertheless, this project has established a basis for some further research on hate speech detection, especially in multilingual and low-resource environments, and pointed out the possibilities for even more sophisticated applications.

## 6.4 Conclusion

The purpose of this project was to solve the problem of classifying hateful and offensive speech in multilingual documents through machine learning, deep learning and transfer learning approaches. The research employed these models on datasets in four different languages i.e. Tamil, English, German and Arabic. What differed from the initial expectation was that conventional machine learning models such as SVM and Random Forest did better than deep learning and transfer learning models in many cases. This surprising result points to the applicability of these models in performing text classification tasks with small amounts of data.

The findings of the project stressed the importance of the amount of computations and size of deep learning datasets. While it's possible that models such as RNN, Bi-LSTM and BERT can achieve better results than traditional machine learning approaches, they need a lot of computations and large datasets to do so. For instance, in the case of the Arabic and the Tamil datasets where data was very scarce, the models could not perform at their best.

In summary, this work proved the necessity of choosing models that fit the particularities of the data and the resolution of the problem. Although deep and transfer learning allow to achieve better results, classical approaches still possess reasonable efficiency especially in the presence of constraints on computation power. The results of the study add to the existing body of knowledge on multilingual natural language processing with an avenue for future works.

## 6.5 Future Work

These approaches should yield positive results even in the detection of hate speech and offensive language in multilingual systems. Some of the work in this area could be the enhancement of the performance of deep learning models using sophisticated data augmentation strategies. In addition to back-translation and other standard approaches, synthetic data could be produced using Generative Adversarial Networks (GANs) or different autoencoders, especially with regards to low-resourced languages such as those found in Tamil and Arabic. These types of techniques could address the problem of poor datasets and increase the performance of a model.

To explore this further, one possible direction is to use distributed and decentralized systems, specifically federated learning, for dealing with large datasets as well as aiding the training of deep learning architectures. This will lead to the improvement of models that can be trained on many autonomous devices or servers without uploading a large amount of information, ensuring that privacy is kept and the stress on central processing units is reduced. Another advantage of this strategy is that it would be particularly useful for training on the large-scale database obtained from social networks.

Moreover, it would also be interesting to investigate how neural networks can be optimized further by considering the architecture of the layers and the loss functions. Additional, hyper-parameter estimation techniques, like grid search or Bayesian approach, which might augment the neural model and protect it against overfitting will be efficient for models BERT, RNN and Bi-LSTM. In addition, consideration of the construction of language-specific pretrained models for low-resource languages could allow hate speech detection to be more precise and take into consideration the context.

Ultimately, these models could be of real use in social media sites such as Twitter and YouTube, as they would enable real-time monitoring and intervention. Future work could include field applications as well as model refinements in order to study evolving hate speech.

# Bibliography


[1] Badjatiya, P., Gupta, S., Gupta, M., and Varma, V. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion* (Perth, Australia, 2017), International World Wide Web Conference Committee (IW3C2), pp. 759–760.

[2] Badjatiya, P., Gupta, S., Gupta, M., and Varma, V. Deep learning for hate speech detection in tweets. In *Proceedings of the ACM Web Science Conference Companion (WWW'17)* (2017), pp. 759–760.

[3] BpHigh. Abusive comment detection dataset. `https://github.com/bp-high/BpHigh_at_Dravidian_Lang_Tech_ACL-2022/tree/main/Abusive_Comment_Detection/Data/Original%20Dataset`, 2022.

[4] Davidson, T., Warmsley, D., Macy, M., and Weber, I. Automated hate speech detection and the problem of offensive language. In *Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM)* (Montreal, Canada, 2017), Association for the Advancement of Artificial Intelligence (AAAI), pp. 512–515.

[5] Davidson, T., Warmsley, D., Macy, M., and Weber, I. Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media* (2017), vol. 11.

[6] Devastator, T. Hate speech and offensive language detection dataset. `https://www.kaggle.com/datasets/thedevastator/hate-speech-and-offensive-language-detection?resource=download`, 2023.

[7] Di Capua, M., Di Nardo, E., and Petrosino, A. Unsupervised cyber bullying detection in social networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)* (2016), IEEE, pp. 432–437.

[8] Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V., and Bhamidipati, N. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web (WWW 2015 Companion)* (Florence, Italy, 2015), ACM, pp. 29–30.

[9] Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V., and Bhamidipati, N. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web (WWW 2015 Companion)* (Florence, Italy, 2015), ACM, pp. 29–30.

[10] ElSherief, M., Kulkarni, V., Nguyen, D., Wang, W. Y., and Belding, E. Hate lingo: A target-based linguistic analysis of hate speech in social media. In *Proceedings of the Twelfth International AAAI Conference on Web and Social Media (ICWSM 2018)* (Stanford, California, USA, 2018), Association for the Advancement of Artificial Intelligence (AAAI).

[11] Fauzi, M. A., and Yuniarti, A. Ensemble method for indonesian twitter hate speech detection. *Indonesian Journal of Electrical Engineering and Computer Science 11*, 1 (2018), 294–299.

[12] Fortuna, P., and Nunes, S. A survey on automatic detection of hate speech in text. *ACM Comput. Surv. 51*, 4 (jul 2018).

[13] Fortuna, P., and Nunes, S. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR) 51*, 4 (2018), 1–30.

[14] Gao, L., Kuppersmith, A., and Huang, R. Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP)* (Taipei, Taiwan, 2017), Asian Federation of Natural Language Processing (AFNLP), pp. 774–782.

[15] Ginting, P. S. B., Irawan, B., and Setianingsih, C. Hate speech detection on twitter using multinomial logistic regression classification method. In *2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)* (2019), IEEE.

[16] Groll, A., Ley, C., Schauberger, G., and Van Eetvelde, H. Prediction of the fifa world cup 2018–a random forest approach with an emphasis on estimated team ability parameters. *arXiv preprint arXiv:1806.03208* (2018).

[17] Hermessi, H. Arabic levantine hate speech detection dataset. `https://www.kaggle.com/datasets/haithemhermessi/arabic-levantine-hate-speech-detection`, 2023.

[18] Huynh, T. V., Nguyen, D.-V., Nguyen, K. V., Nguyen, N. L.-T., and Nguyen, A. G.-T. Hate speech detection on vietnamese social media text using the bi-gru-lstm-cnn model. In *Proceedings of the VLSP Shared Task 2019: Hate Speech Detection on Social Networks* (2019), Vietnam National University Ho Chi Minh City.

[19] Kapil, P., Ekbal, A., and Das, D. Nlp at semeval-2019 task 6: Detecting offensive language using neural networks. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)* (Minneapolis, Minnesota, USA, 2019), Association for Computational Linguistics, pp. 587–592.

[20] Kwok, I., and Wang, Y. Locate the hate: Detecting tweets against blacks. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence* (Bellevue, Washington, USA, 2013), Association for the Advancement of Artificial Intelligence (AAAI), pp. 1621–1622.

[21] Pitsilis, G. K., Ramampiaro, H., and Langseth, H. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433* (2018).

[22] Pitsilis, G. K., Ramampiaro, H., and Langseth, H. Effective hate-speech detection in twitter data using recurrent neural networks. *Applied Intelligence 48* (2018), 4730–4742.

[23] Qian, J., Bethke, A., Liu, Y., Belding, E., and Wang, W. Y. A benchmark dataset for learning to intervene in online hate speech. In *Proceedings of the 2019 Conference on Em-*

*pirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing* (2019), Association for Computational Linguistics, pp. 4755–4764.

[24] Rosa, H., Ribeiro, R., Carvalho, J. P., Matos, D., and Coheur, L. A "deeper" look at detecting cyberbullying in social networks. In *Proceedings of the 2018 International Conference on Computational Linguistics (COLING)* (2018), Association for Computational Linguistics.

[25] Schmidt, A., and Wiegand, M. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media* (Valencia, Spain, 2017), Association for Computational Linguistics, pp. 1–10.

[26] Schmidt, A., and Wiegand, M. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media* (Valencia, Spain, 2017), Association for Computational Linguistics, pp. 1–10.

[27] Sohn, H., and Lee, H. Mc-bert4hate: Hate speech detection using multi-channel bert for different languages and translations. In *Proceedings of the 2019 International Conference on Data Mining Workshops (ICDMW)* (2019), IEEE, pp. 551–559.

[28] Speech, H. H., and Identification, O. C. Hasoc 2021 dataset. `https://hasocfire.github.io/hasoc/2021/dataset.html`, 2021.

[29] Sreelakshmi, K., Premjith, B., and Soman, K. P. Detection of hate speech text in hindi-english code-mixed data. *Procedia Computer Science 171* (2020), 737–744.

[30] Ulmer, B., and Fernandez, M. Predicting soccer match results in the english premier league. *CS229 Final Project, Stanford University* (2014). Available at Stanford University.

[31] van Aken, B., Risch, J., Krestel, R., and Löser, A. Challenges for toxic comment classification: An in-depth error analysis. In *Proceedings of the Second Workshop on Abusive*

*Language Online (ALW2)* (Brussels, Belgium, 2018), Association for Computational Linguistics, pp. 33–42.

[32] Warner, W., and Hirschberg, J. Detecting hate speech on the world wide web. In *Proceedings of the 2012 Workshop on Language in Social Media (LSM 2012)* (Montreal, Canada, 2012), Association for Computational Linguistics, pp. 19–26.

[33] Waseem, Z., and Hovy, D. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of NAACL-HLT 2016* (2016), Association for Computational Linguistics, pp. 88–93.

[34] Wiegand, M., Ruppenhofer, J., and Kleinbauer, T. Detection of abusive language: The problem of biased datasets. In *The 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Minneapolis, MN, USA, 2019), Association for Computational Linguistics, pp. 602–608.

[35] Wulczyn, E., Thain, N., and Dixon, L. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web (WWW)* (Perth, Australia, 2017), International World Wide Web Conference Committee (IW3C2), pp. 1391–1399.

[36] Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. Predicting the type and target of offensive posts in social media. *arXiv preprint arXiv:1902.09666* (2019).

[37] Zhang, Z., Robinson, D., and Tepper, J. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *Proceedings of the 2018 International Conference on Social Media, Wearable and Web Analytics* (New York, NY, USA, 2018), ACM.

[38] Zhang, Z., Robinson, D., and Tepper, J. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *ESWC 2018: The Semantic Web* (2018), N.-R. V. M. H. P. T. R. H. L. T. A. Gangemi, A. and M. Alam, Eds., vol. 10843 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 745–760.

# Appendix A

# User Manuals

This section provides a detailed guide for setting up, running, and utilizing the Python code. The code has been developed and tested using Python 3.6 and is compatible with several deep-learning frameworks and libraries, including TensorFlow and PyTorch. Below are the step-by-step instructions for configuring the environment, training the models, and using the various scripts.

### B.0.1 Setup Instructions

1. **Create a Conda Environment:**

   To ensure isolation and proper dependency management, it's recommended to create a conda environment:

   ```
   conda create --name myenv python=3.6
   ```

   Replace `myenv` with your preferred environment name.

2. **Activate the Conda Environment:**

   After creating the environment, activate it:

   ```
   conda activate myenv
   ```

3. **Install Required Libraries:**

   Ensure that the `requirements.txt` file is present in your working directory. It contains the list of libraries required for the project. To install the dependencies:

   ```
   pip install -r requirements.txt
   ```

   This command will install TensorFlow, PyTorch, Scikit-learn, NLTK, and other essential libraries.

### B.0.2 File Structure

The project directory includes several important folders that organize the scripts and datasets for training and evaluation. Here's a breakdown of the key directories:

- **data/**: Contains the datasets used for multilingual hate speech detection (English, Tamil, German, Arabic).
- **preprocessing/**: Scripts for text cleaning, tokenization, and preparation of data before training.
- **models/**: Includes implementation files for machine learning models (SVM, Logistic Regression) and deep learning models (RNN, Bi-LSTM, BERT).
- **training/**: Contains the main training scripts for different datasets and models. Each script can be configured with the dataset and model type via command-line arguments.
- **evaluation/**: Scripts for evaluating model performance using metrics like accuracy, precision, recall, and F1 score. It also includes visualization tools for creating confusion matrices and plotting results.

### B.0.3 Training and Testing Models

To train and test the models, navigate to the `training/` directory. Each model has its own configuration file and can be trained by executing the following command:

```
python train_model.py --model BERT --dataset Arabic
```

Arguments for Training and Testing:

- `--model`: Specify the model to train. Options include `SVM`, `RNN`, `Bi-LSTM`, `BERT`.
- `--dataset`: Select the dataset for training (English, Tamil, German, Arabic).
- `--epochs`: Set the number of training epochs. Default is 10.
- `--lr`: Specify the learning rate. Default is 0.001.

After training, the model performance metrics (accuracy, F1 score, etc.) will be saved in the `evaluation/` folder for further analysis.

### B.0.4 Running on GPU Server

This project has been designed to run on GPU-accelerated servers. To run the training scripts on a GPU server, ensure that the environment is activated and that the `requirements.txt` file has been installed. The project has been tested on the NVIDIA A100 GPU with 40GB memory. Here's an example command for running the training script on a GPU server:

```
python train_model.py --model BERT --dataset Tamil --gpu
```

Ensure that the environment is properly configured to support CUDA for GPU-based acceleration.