



MSU Graduate Theses

Fall 2023


Evaluation of Different Machine Learning, Deep Learning and Text Processing Techniques for Hate Speech Detection

Nabil Shawkat

Missouri State University, ns92s@MissouriState.edu

As with any intellectual project, the content and views expressed in this thesis may be considered objectionable by some readers. However, this student-scholar's work has been judged to have academic value by the student's thesis committee members trained in the discipline. The content and views expressed in this thesis are those of the student-scholar and are not endorsed by Missouri State University, its Graduate College, or its employees.

Follow this and additional works at: <https://bearworks.missouristate.edu/theses>

 Part of the [Computational Linguistics Commons](#), [Computer and Systems Architecture Commons](#), [Data Storage Systems Commons](#), and the [Science and Technology Studies Commons](#)

Recommended Citation

Shawkat, Nabil, "Evaluation of Different Machine Learning, Deep Learning and Text Processing Techniques for Hate Speech Detection" (2023). *MSU Graduate Theses*. 3913.

<https://bearworks.missouristate.edu/theses/3913>

This article or document was made available through BearWorks, the institutional repository of Missouri State University. The work contained in it may be protected by copyright and require permission of the copyright holder for reuse or redistribution.

For more information, please contact bearworks@missouristate.edu.

**EVALUATION OF DIFFERENT MACHINE LEARNING, DEEP LEARNING AND
TEXT PROCESSING TECHNIQUES FOR HATE SPEECH DETECTION**

A Master's Thesis

Presented to

The Graduate College of

Missouri State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science, Computer Science

By

Nabil Shawkat

December 2023

Copyright 2023 by Nabil Shawkat

EVALUATION OF DIFFERENT MACHINE LEARNING, DEEP LEARNING AND TEXT PROCESSING TECHNIQUES FOR HATE SPEECH DETECTION

Computer Science

Missouri State University, December 2023

Master of Science

Nabil Shawkat

ABSTRACT

Social media has become a domain that involves a lot of hate speech. Some users feel entitled to engage in abusive conversations by sending abusive messages, tweets, or photos to other users. It is critical to detect hate speech and prevent innocent users from becoming victims. In this study, I explore the effectiveness and performance of various machine learning methods employing text processing techniques to create a robust system for hate speech identification. I assess the performance of Naïve Bayes, Support Vector Machines, Decision Trees, Random Forests, Logistic Regression, and K Nearest Neighbors using three distinct datasets sourced from social media posts. To gauge the optimal approach, I employ Term Frequency-Inverse Document Frequency (TF-IDF), unigrams, bigrams, trigrams, a combination of unigrams and bigrams, and a combination of unigrams, bigrams, and trigrams for the machine learning models to analyze the text corpus. Given the imbalanced nature of the datasets, I implement both under-sampling and over-sampling techniques to investigate their impact on the results. I also investigated the performance of different deep learning algorithms on the three datasets. The results show that the Bidirectional Encoders Representations from Transformers (BERT) model gives the best performance among all the models on imbalanced datasets by achieving an F1-score of 90.6% on one of the datasets, and F1-scores of 89.7% and 88.2% on the other two datasets. Comparative analysis reveals that BERT and Robustly Optimized BERT Pretraining Approach (RoBERTa) outperform traditional Machine Learning (ML) algorithms, with F1-scores approximately 20% higher. The investigation indicates that RoBERTa, with its enhanced training strategies, comes remarkably close to the performance of BERT. The outcomes show the transformative impact of deep learning and pretrained models on hate speech detection, with larger, more diverse datasets further enhancing model performance.

KEYWORDS: hate speech, machine learning, social media, BERT, deep learning, RoBERTa, pretrained models, text classification

**EVALUATION OF DIFFERENT MACHINE LEARNING, DEEP LEARNING AND
TEXT PROCESSING TECHNIQUES FOR HATE SPEECH DETECTION**

By

Nabil Shawkat

A Master's Thesis
Submitted to the Graduate College
Of Missouri State University
In Partial Fulfillment of the Requirements
For the Degree of Master of Science, Computer Science

December 2023

Approved:

Jamil M. Saquer, Ph.D., Thesis Committee Chair

Mohammed Y. Belkhouche, Ph.D., Thesis Committee Member

Siming Liu, Ph.D., Thesis Committee Member

Julie Masterson, Ph.D., Dean of the Graduate College

In the interest of academic freedom and the principle of free speech, approval of this thesis indicates the format is acceptable and meets the academic criteria for the discipline as determined by the faculty that constitute the thesis committee. The content and views expressed in this thesis are those of the student-scholar and are not endorsed by Missouri State University, its Graduate College, or its employees.

ACKNOWLEDGEMENTS

I extend my heartfelt gratitude to the individuals who have been instrumental in the successful completion of my master's thesis. Foremost, I would like to thank my dedicated supervisor, Dr. Jamil Saquer. His guidance, profound knowledge, and mentorship have been invaluable in shaping the academic excellence of this thesis. Dr. Saquer's unwavering support from the first semester has played a pivotal role in my academic growth. I would like to thank the following people for their support during the course of my graduate studies.

I am equally grateful to the esteemed members of my thesis committee, Dr. Mohammed Y. Belkhouche and Dr. Siming Liu. Their expertise, insightful feedback, and commitment to the quality of this research have been crucial in shaping this research.

I also wish to express my deepest appreciation to my father and brother, who have been constant sources of support throughout my academic journey. Their belief in me and endless encouragement made this endeavor possible. Their support and love have been the driving force behind my determination to succeed.

Finally, to my friends and colleagues who provided their support and encouragement, your presence has been a source of strength. This thesis has come to fruition with the collective efforts of these individuals. I am profoundly grateful for each one of you.

TABLE OF CONTENTS

Introduction	Page 1
Literature Survey	Page 5
Background	Page 14
Hate Speech	Page 14
The Significance of Hate Speech Detection	Page 15
Traditional Approaches to Text Classification	Page 16
Data Preprocessing and Feature Engineering	Page 16
Machine Learning Approaches	Page 18
Machine Learning Models	Page 18
The Rise of Deep Learning in Text Classification	Page 20
Unraveling Deep Learning	Page 21
Deep Learning Approaches in Text Classification	Page 21
LSTM	Page 22
CNN	Page 24
BERT: Revolutionizing Natural Language Understanding	Page 26
Methodology	Page 30
Experimental Setup And Results	Page 40
Stage-1 of the proposed framework	Page 40
Stage-2 of the proposed framework	Page 41
Stage-3 of the proposed framework	Page 45
Stage-4 of the proposed framework	Page 47
Comparison Of The Models	Page 51

SVM Comparison	Page 51
RoBERTa Comparison	Page 53
Conclusion	Page 55
References	Page 57

LIST OF TABLES

Table 1. Best Results of ML Algorithms on the Original Datasets	Page 42
Table 2. Best Results of ML Algorithms on the Undersampled Datasets	Page 43
Table 3. Results of ML Algorithms on the Oversampled Datasets	Page 44
Table 4. Experiment Results of All Models	Page 46
Table 5. F1-score of BERT and RoBERTa Models using the Datasets	Page 54

LIST OF FIGURES

Figure 1. Overview of the System	Page 32
Figure 2. Architecture of LSTM	Page 34
Figure 3. Architecture of CNN	Page 36
Figure 4. Architecture of BERT	Page 39
Figure 5. Token IDs of a Sample	Page 48
Figure 6. Tokens and Attention Masks of a Sample	Page 49
Figure 7. Predicting Hate Speech from Dataset I	Page 50
Figure 8. Predicting Hate Speech from Dataset II	Page 50
Figure 9. Predicting Hate Speech from Dataset III	Page 50
Figure 10. Sample of Dataset I Used in BERT Model to Predict the Class which SVM couldn't Predict Correctly	Page 52
Figure 11. Sample of Dataset II Used in BERT Model to Predict the Class which SVM couldn't Predict Correctly	Page 52
Figure 12. Sample of Dataset III Used in BERT Model to Predict the Class which SVM couldn't Predict Correctly	Page 52

INTRODUCTION

Social media has revolutionized our ability to engage with a vast audience, eliminating barriers of location, gender, and race. While platforms like Facebook and Twitter offer numerous benefits, they are not without their drawbacks, with one significant concern being the proliferation of hate speech [9]. Studies indicate that a considerable portion, ranging from 10% to 40%, of social media and internet users face exposure to hate speech [15]. The consequences of such exposure are wide-ranging, leading to issues like momentary anxiety or, in extreme cases, even suicides among the victims [13]. The escalating crime of hate speech demands urgent attention and intervention. Several high-profile incidents have underscored the prevalence of hate speech on social media. This study zeroes in on the critical task of detecting hate speech online.

Hate speech can have devastating implications for victims and can result in social anxiety, despair, and suicidal thoughts [10, 11, 12]. Hate speech or using abusive words used to be limited to physical interactions between the harasser and the victim until the advent of social media. It is easier to identify and handle hate speech or attacking an individual situation when they occur in-person such as on a school premises with the existence of others who can intervene [17]. However, the emergence of the Internet and digital media has resulted in hate speech becoming a more pervasive problem, as it can happen at any time and any place. Additionally, with the online medium, the chances of such incidents going unnoticed and unaddressed are significantly higher. So, it is crucial to identify and recognize hate speech in combatting this significant issue. Nonetheless, detecting hate speech online is still a challenging task. The task of recognizing and pinpointing hate speech online is a complex challenge, characterized by its subjectivity and the

absence of a universally agreed-upon definition. The understanding of hate speech varies significantly among different groups, adding to the intricacy of its identification.

Abusive users target victims on a variety of topics, including race, religion, and gender, across multiple social media platforms. The lexicon and perceived meaning of terms change substantially across social media platforms depending on the issue of hate speech. Recognizing hate speech necessitates a comprehensive grasp of the contextual nuances and semantic intricacies embedded in the text. However, deploying straightforward algorithms reliant on lists of offensive words poses a formidable challenge, primarily attributable to the swift evolution of terms and hashtags across various social media platforms.

Advances in Machine Learning (ML), Deep Learning (DL), Natural Language Processing (NLP), and Graphics Processing Units (GPU) technology provide promising solutions to the challenges of automatic detection of hate speech. The aim of this study is to assess the efficacy of various machine learning and deep learning algorithms in identifying abusive text, employing diverse text processing methods such as TFIDF and n-grams. The initial phase of the investigation involves appraising the performance of traditional ML techniques to discern hate speech. I chose to test the effectiveness of the following six well-known ML algorithms: Naïve Bayes, Support Vector Machine (SVM), Decision Tree, Random Forests, Logistic Regression, and K Nearest Neighbors. The datasets I used in the research are imbalanced because there are significant differences in the distribution of the objects among the datasets and it indicates that the datasets are biased toward a particular class. An ML algorithm trained on such datasets will be biased toward the same class if the dataset is biased toward that class. In my case, most text samples belong to the non-abusive class and are biased toward that class. This is expected because in real-life situations with any corpus of text one would expect most text not to contain hate speech [14].

The traditional ML algorithms gave bad results when using imbalanced datasets. To handle the imbalanced class distribution, I either under-sample the majority class or over-sample the minority class so that all classes have the same number of objects. Over-sampling the minority class has the added advantage of increasing the size of a dataset, which helped achieve better results.

As we move further into the digital age, the sheer volume of data available for analysis continues to grow at an astonishing pace. The complex and high-dimensional nature of this data has led to the evolution of sophisticated machine learning algorithms. Traditional machine learning has served us well over the years, but we have begun to reach a point where these techniques alone are not enough to keep up with the complexity and volume of data we now face. This is where deep learning comes into play.

Deep learning, a subset of machine learning, leverages neural networks with many layers (hence "deep") to extract higher-level features from raw input data. Compared to traditional machine learning algorithms, deep learning algorithms can automatically learn feature representations from raw data, eliminating the need for manual feature extraction. This ability to handle raw, high-dimensional data and learn abstract features has made deep learning particularly effective in areas like image and speech recognition, natural language processing, and other data-intensive applications.

One such deep learning model that has been widely applied to image analysis is Convolutional Neural Networks (CNNs). CNNs apply a series of filters to the raw pixel data of an image to extract and learn higher-level features, which can then be used for classification. CNNs have been shown to be highly effective in tasks such as object detection and facial recognition.

Long Short-Term Memory (LSTM) units, on the other hand, have changed the game in processing sequential data for tasks such as language modeling and time series prediction. Unlike

traditional recurrent networks, LSTMs have an added ability to forget unnecessary information in a sequence, thus making them effective in capturing long-term dependencies.

In recent years, a new model architecture known as Transformers has emerged, designed specifically for tasks involving sequential data. Transformers address the problem of long sequences by using a mechanism called self-attention, allowing them to consider other words in the sentence simultaneously when processing each word. This has made Transformers exceptionally good at understanding context in natural language processing tasks.

A prominent example of a transformer-based model is BERT. BERT's bidirectional training approach allows it to understand the context of a word based on all of its surroundings (left and right of the word). This has led to groundbreaking results on a range of tasks in the natural language processing field, from sentiment analysis to question answering.

For the second part of the research, we used several deep learning techniques such as LSTM, CNN, and BERT over the datasets without over-sampling or under-sampling the datasets. BERT is designed to pre-train deep bidirectional representations from unlabeled text to help computers understand the meaning of ambiguous language in text by using surrounding text to establish context. Consequently, by fine-tuning the pre-existing BERT model with an appended output layer, it becomes possible to develop cutting-edge models for various tasks, including but not limited to question answering and language inference. This achievement is attained without requiring significant task-specific architectural alterations [25]. In my experiments, BERT shows its capability in classifying text and understanding context, resulting in F1-scores of 89.7%, 88.2%, and 90.6% on three different datasets.

LITERATURE SURVEY

Hate speech is an act of crime that unfortunately has been occurring more often nowadays online. Use of social networks, social media and online gaming has led to people communicating all over the world but with the price of being targeted or bullied. Detecting hate speech to avoid hate crimes from happening online is a crucial matter but the task is extremely challenging. In this section, we briefly review existing literature and work on detecting hate speech.

The study of online hate speech detection is comparatively recent. In [1], researchers collected data from the field of automatic hate speech detection, which led them to categorize the field as “Law and Social Sciences” and “Computer Science.” Most of the papers are from “Law and Social Sciences” but there is still significant amount of work done in the “Computer Science” field that helps one understand the topic and its challenges. “Cyberbullying” and “Online Harassment” are other terms that are sometimes used to mean hate speech [2].

Social media platforms have become breeding grounds for hate speech, making the detection of such content essential for maintaining a safe online environment. Detecting hate speech in social media networks presents unique challenges due to the informal nature of the language used and the prevalence of user-generated content. Researchers have developed specific techniques tailored to social media platforms by leveraging platform-specific features. User interactions, including replies, likes, and retweets, provide valuable contextual information that can aid in hate speech detection. Hashtags and mentions are often indicative of hate speech content and can be used as additional signals. Moreover, understanding the distinct characteristics and dynamics of social media data is crucial for effective hate speech detection in these environments [42, 43]. Detecting hate speech across different domains is essential for comprehensive monitoring

and mitigation efforts. Hate speech can manifest differently depending on the context, such as news articles, online forums, or comment sections. Researchers have focused on developing techniques that can generalize across different domains, allowing hate speech detection models to adapt and perform well in various settings. Transfer learning approaches have been explored to leverage knowledge learned from one domain and apply it to another. Domain adaptation methods aim to minimize the domain shift between labeled training data and the target domain where hate speech detection is needed. Cross-domain hate speech detection enables the identification of hateful content in diverse sources and promotes a more comprehensive understanding of the prevalence and nature of hate speech [44, 45].

In automatic hate speech detection, it is important to understand the definition of hate speech, specifically online hate speech. To identify hate speech, organizations and companies have made definitions. It helps the users to understand that some words and sentences are prohibited if they match the policy and definition of hate speech. Encyclopedia of the American Constitution, European Union Commission, international minorities associations, and the policies of Facebook, YouTube, and Twitter have their own definitions of hate speech [6]. All of these definitions have different goals in mind when defining hate speech; some discussed how an attack on a particular group (such as people with specific race, ethnicity, religion, gender, or disability) qualifies as hate speech, while others discussed how an attack on an individual (based on any characteristic) qualifies as hate speech. In [5], online harassment is defined restrictively as a kind of action in which a user intentionally annoys one or more other users in a web community. However, a definition that gives a broader domain to understand hate speech is given by the authors in [1], "Hate speech is language that attacks or diminishes, that incites violence or hate against groups, based on specific characteristics such as physical appearance, religion, descent, national or ethnic

origin, sexual orientation, gender identity or other, and it can occur with different linguistic styles, even in subtle forms or when humor is used.” Most definitions lead to the idea that hate speech is defined by characteristics that incite violence or hate, attacks, or diminishes, has specific targets, and whether humor is hurtful or not.

NLP techniques and models are used in existing research to identify and categorize abusive language that targets a person based on their race, religion, ethnicity, national origin, disability, sex, sexual orientation, or gender identity [22, 23, 24]. Supervised learning models like Logistic Regression, SVM, Random Forests, and DL, as well as semi-supervised techniques like bootstrapping, are primarily used in research on the classification of hate speech [19, 24].

As hate speech detection models are increasingly deployed in real-world settings, there is a growing demand for transparency and interpretability. Users and stakeholders want to understand the reasoning behind the model's decisions and identify the specific features or words that contribute to the classification of hate speech. Researchers have explored various methods to provide explanations for hate speech detection models [40, 41]. Attention mechanisms have been employed to highlight the most influential words or phrases in the input text.

While the development of hate speech detection systems is crucial for addressing online abuse, it is essential to address ethical considerations and potential biases associated with these systems. Bias can arise from the training data used to develop hate speech detection models, leading to disproportionate false positives or false negatives for specific demographic groups. For example, if the training data is imbalanced and predominantly represents one particular group, the model may exhibit biased behavior in its predictions. To mitigate these biases, researchers have emphasized the importance of developing fair and unbiased hate speech detection models.

To achieve fairness and mitigate biases, several strategies have been proposed. Careful selection of training data is crucial to ensure that the dataset is representative of different demographic groups and social contexts. It is essential to include diverse voices and perspectives in the training data to avoid perpetuating biases. Moreover, considering the impact on marginalized communities is crucial throughout the development process. Regular evaluation and auditing of hate speech detection models can help identify and address biases as they arise. Additionally, active engagement with stakeholders, including community representatives and advocacy groups, can provide valuable insights and perspectives in addressing biases effectively [38, 39]. By addressing ethical considerations and biases, researchers can develop hate speech detection models that are fair, robust, and sensitive to the needs and experiences of diverse user groups. These efforts contribute to the development of more inclusive and equitable online environments.

With the global nature of online communication, the detection of hate speech in multilingual settings has become a pressing challenge. Hate speech is not limited to a specific language, and it is essential to develop detection models that can handle diverse linguistic contexts. However, obtaining labeled data for every language is often expensive and time-consuming. To address this issue, researchers have explored transfer learning techniques for hate speech detection in multiple languages. Cross-lingual embeddings, which map words from different languages into a shared vector space, enable the transfer of knowledge from a resource-rich language to low-resource languages. Multi-task learning, on the other hand, leverages shared representations and jointly trains models on multiple related tasks to improve performance in low-resource languages. These approaches facilitate the development of more inclusive hate speech detection systems that can effectively handle diverse linguistic contexts and mitigate language barriers [30, 31]. However, in the thesis, I am focusing on using datasets that are based on English language.

In addition to the detection of hate speech, research efforts have also focused on developing effective mitigation strategies [7, 33, 34]. Hate speech can have significant negative impacts on individuals and communities, and it is crucial to address and counteract its influence. Counter-speech has emerged as a potential strategy to reduce the impact of hate speech by promoting positive and informative messages. Counter-speech involves responding to hate speech with non-hostile and constructive dialogue, aiming to challenge and educate individuals who engage in or are exposed to hate speech. User reputation systems, which assign reputation scores to users based on their past behavior, can help identify and mitigate the dissemination of hate speech by giving more weight to trusted and reputable users. Community moderation, where users report and flag offensive content, combined with algorithmic interventions, can further aid in detecting and reducing the visibility of hate speech content. These strategies collectively contribute to fostering a safer and more inclusive online environment.

Automating detecting hate speech is complicated when dealing with the data itself and gathering it. Social networks are the common grounds for the datasets that have been used in related works. Most related works are done using Twitter datasets [1]. However, some works are done using more than one dataset from different social media like Q&A forums, Wikipedia talk pages and Twitter datasets [4]. The datasets mainly focus on two types of communities: discussion-style and chat-style [5]. Discussion-style environments consist of long posts on one predefined topic. On the contrary, chat-style environments consist of short sentences or a bunch of words. Twitter datasets are more like chat-style environments where the conversations are mostly short. An example of such a dataset is the Twitter dataset created by Waseem and Hovy [7], which I use as one of the datasets in my research.

To assess the performance of hate speech detection models, appropriate evaluation metrics and benchmark datasets are essential. Various evaluation metrics have been employed to measure the effectiveness of hate speech detection algorithms. Precision represents the proportion of correctly classified hate speech instances out of all instances predicted as hate speech. Recall measures the proportion of correctly classified hate speech instances out of all actual hate speech instances. F1-score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance. These evaluation metrics enable researchers to quantitatively compare different models and algorithms based on their performance on hate speech detection tasks [35].

In addition to evaluation metrics, the availability of benchmark datasets plays a crucial role in advancing hate speech detection research. Researchers have developed benchmark datasets that contain labeled examples of hate speech and non-hate speech instances [36, 37]. These datasets serve as standardized resources for evaluating hate speech detection models and facilitate comparative studies. Examples of such benchmark datasets include the HateEval dataset, which focuses on hate speech against immigrants and women in Twitter, and the OffensEval dataset, which addresses offensive language identification in social media. By using these benchmark datasets, researchers can compare the performance of different models, identify areas for improvement, and encourage the development of more robust hate speech detection systems.

In [21], the authors analyze directed hate speech and generalized hate speech on Twitter using linguistic and psycholinguistic methods. To extract features from the tweets, they employ a variety of NLP techniques, including part-of-speech tagging, and entity recognition. Additionally, for psycholinguistic methods, they use the Linguistic Inquiry and Word Count (LIWC) software, which is a text analysis tool that identifies psychological and emotional correlates of language use [26, 32]. The authors in [20, 23] investigate Islamophobia on Twitter. In [20], the authors carry

out a longitudinal analysis to investigate the connection between the trend on Twitter and the offline occurrences that took place during the COVID-19 pandemic months. This type of analysis involves monitoring changes or patterns in variables of interest over time and understanding the relationships among them as they evolve. The study emphasizes the importance of social media platforms taking action to prevent the spread of hate speech. In [23], they develop ML classifiers to identify Islamophobic tweets. In [19], Chandra et al. present a multimodal DL system that uses text and images from Twitter and Gab posts to identify antisemitic content and the specific antisemitism category it belongs to.

Automatic approaches and implementations to detect hate speech are done by machine learning algorithms and different feature selection methods. Machine learning algorithms that are usually used in different papers are Logistic Regression, Random Forests, SVM, DT, and Naïve Bayes. To use a dataset, the text must be weighted or processed properly to be used in the machine learning algorithm or it will not give good results. This leads to two major problems: language issue and annotation issue [7]. Language issue means not being able to understand the context of a sentence. Annotation is always a huge issue while extracting the text and labeling it. Feature selection methods help with the annotation problem when you know the context and semantics [5]. In a paper about cyberbullying [3], the authors proposed a method called the participant-vocabulary consistency (PVC) model which looks for parameter setting for all the users and key phrases that characterize the tendency of hate speech or not. In my research, I use datasets that have already been labeled, where the class for each sentence is known, whether it is hateful speech or not.

Deep learning approaches have gained significant attention in the field of hate speech detection due to their ability to capture complex linguistic patterns and semantic relationships in

text data. Hate speech is often characterized by subtle nuances and contextual cues that require sophisticated modeling techniques. Deep learning models, such as CNNs and recurrent neural networks (RNN), have shown promise in capturing the sequential and hierarchical nature of text. CNNs leverage filters to extract local features from different parts of the input text, while RNNs, particularly LSTM networks, can effectively model the temporal dependencies in a sequence of words. These models have demonstrated improved accuracy and generalization capabilities compared to traditional machine learning algorithms in hate speech detection tasks [46, 27].

The incorporation of contextual embeddings has emerged as a powerful technique in hate speech detection. Contextual embeddings refer to word representations that capture the meaning and semantic context of a word based on its surrounding words in a sentence. Pre-trained transformer models, such as BERT, have shown remarkable performance in a wide range of natural language processing tasks. BERT generates contextualized word embeddings by considering the entire sentence and capturing the relationships between words. These embeddings provide a rich representation of text that considers the context and context-dependent meanings of words [28, 29]. By leveraging contextual embeddings, hate speech detection models can better capture the nuances and subtleties of hate speech, resulting in improved accuracy and robustness.

Prepossessing the text is an important task to detect hate speech with the help of ML. In previous works [1, 5, 8, 18, 20], researchers used different techniques for preprocessing, but all were infused with TF-IDF and n-grams. In most cases, TF-IDF helped get better results than n-grams. The work in [5] used both TF-IDF and n-grams. For my experiment, I used both TF-IDF and n-grams, which gave better performance than previous works [1, 5, 8]. Another problem with detecting hate speech is the imbalanced datasets. Inspired by [14], we over-sampled and under-sampled the datasets to get balanced datasets for the machine learning algorithm. ML algorithms

and various feature selection techniques are used to detect hate speech. The ML algorithms Logistic Regression, Random Forests, SVM, DT, and Naive Bayes are frequently used in various papers. Data preprocessing is a critical step in preparing a dataset for machine learning algorithms. Without proper weighting and processing of text data, the algorithm's performance may suffer. Researchers used a variety of preprocessing methods in earlier works [1, 6, 7, 16, 18] but each one included TF-IDF and n-grams. TF-IDF typically assisted in obtaining better results than n-grams. Both TF-IDF and n-grams were used in the research in [4]. Both TF-IDF and n-grams were used in my experiment, and both performed adequately but the transformers-based DL model gave the best result.

BACKGROUND

In this section, I briefly describe key concepts, hate speech, machine learning algorithms, deep learning algorithms, BERT, and class balancing utilized in this work to provide some context.

Hate Speech

In the age of the Internet, social media, and online communities, the power of communication has transcended physical boundaries, providing a global platform for discourse, expression, and connectivity. However, this newfound freedom of expression has not come without its challenges. One such challenge is the proliferation of hate speech, a phenomenon that threatens the very essence of open dialogue and inclusivity in the digital realm. Hate speech, characterized by its harmful and discriminatory language targeted at individuals or groups based on their race, ethnicity, religion, gender, or other protected attributes, poses serious societal and ethical concerns.

The alarming rise in hate speech across online platforms has sparked considerable debate and led to calls for its swift identification and mitigation. Hate speech, when left unchecked, can perpetuate stereotypes, fuel hostility, and incite violence against marginalized communities. It erodes the principles of diversity, equality, and respect that are essential for the healthy functioning of democratic societies. Therefore, addressing hate speech in the digital landscape has become an urgent imperative for individuals, online platforms, and governments alike.

In this context, the field of hate speech detection has emerged as a pivotal area of research and technological development. Hate speech detection seeks to harness the power of computational

linguistics and machine learning to automatically identify and categorize hate speech content within vast volumes of online text data. By doing so, it not only assists in enforcing community guidelines on online platforms but also plays a crucial role in promoting positive online interactions, fostering tolerance, and protecting the vulnerable from harm.

The Significance of Hate Speech Detection

Hate speech detection systems are not merely tools of convenience; they are a defense against the potential harm inflicted by online hate speech. These systems serve several significant purposes:

Early Intervention: By swiftly identifying hate speech, these systems enable timely intervention and moderation, preventing the spread of toxic content before it causes significant harm.

Enhanced User Experience: Hate speech detection helps create safer online spaces by allowing platforms to enforce community guidelines, ensuring that users can engage in meaningful and respectful discourse.

Fostering Inclusivity: By mitigating hate speech, these systems foster inclusivity and make online spaces more welcoming for individuals from diverse backgrounds.

Legal Compliance: In many regions, hate speech is subject to legal consequences. Detection systems assist platforms in adhering to local and international laws by identifying and reporting illegal content.

Research and Analysis: Hate speech detection aids researchers, policymakers, and analysts in studying and understanding the prevalence and dynamics of hate speech, helping to shape effective policies and countermeasures.

Social Responsibility: Platforms that actively combat hate speech demonstrate a commitment to their users' well-being and contribute to building a more inclusive and empathetic digital society.

As the demand for more sophisticated hate speech detection systems grows, so does the need for advanced methods and technologies. This thesis explores the state-of-the-art techniques employed in hate speech detection, focusing on machine learning approaches, and contextual embeddings. By delving into these areas, we aim to contribute to the ongoing efforts to create a safer and more inclusive digital environment for all.

Traditional Approaches to Text Classification

Text classification, a fundamental task in natural language processing (NLP), involves categorizing text documents into predefined classes or categories based on their content. Over the years, several traditional approaches to text classification have been devised that made it possible to apply machine learning models to text datasets. In this section, we explore some of the classical methods employed for text classification.

Data Preprocessing and Feature Engineering

One of the initial steps in hate speech detection involves the preprocessing of textual data in a format suitable for machine learning models. Before subjecting textual data to any hate speech detection model, it is imperative to preprocess the data comprehensively. Data preprocessing encompasses a series of steps aimed at cleaning, organizing, and enhancing the quality of the text. This typically involves tasks such as text normalization, which includes removing punctuation, and stemming or lemmatizing words to their base forms. Furthermore, stop words, which are common words with little semantic value, are often eliminated to reduce noise in the data. Data

preprocessing also involves handling missing values and dealing with imbalanced datasets. The success of any hate speech detection model hinges on the effectiveness of these preprocessing steps, as they lay the foundation for subsequent analysis and model training. Traditional techniques like Term Frequency (TF), and Term Frequency-Inverse Document Frequency (TF-IDF) are commonly used to convert text documents into numerical vectors. TF measures how frequently a term (word) appears in a document. It is a simple count of how many times a word occurs. TF is used to identify the importance of a word within a document, assuming that words occurring more frequently are more significant. Then TF-IDF is a numerical statistic used to evaluate the importance of a word in a document relative to a collection of documents (corpus). It combines TF (Term Frequency) and IDF (Inverse Document Frequency), where TF measures a word's importance in a single document, and IDF measures how unique or rare a word is across the entire corpus. TF-IDF is widely used in information retrieval and text mining. These methods allow machine learning models to work with structured input data, where each word or term is assigned a unique dimension in the feature space.

Feature engineering techniques, such as n-grams and word embeddings like Word2Vec or Global Vectors for Word Representation (GloVe), further enhance the representation of text by capturing contextual and semantic information. N-grams are contiguous sequences of N items or words in a text. They are used in natural language processing to capture contextual information and relationships between words. For example, in the sentence "I love deep learning," the 2-grams (bigrams) would be "I love", "love deep", and "deep learning", while the 3-grams (trigrams) would be "I love deep" and "love deep learning". Word2Vec is a neural network-based model for word embeddings. It maps words to high-dimensional vector spaces where words with similar meanings are located closer together. Word2Vec captures semantic relationships between words and is useful

for various NLP tasks like word similarity, document classification, and sentiment analysis. GloVe is another word embedding model that, like Word2Vec, generates vector representations for words. However, GloVe focuses on the global co-occurrence statistics of words within a corpus. It creates word vectors by considering how frequently words appear together across the entire dataset. GloVe is known for its ability to capture semantic relationships and analogies between words. These engineered features serve as the foundation for training machine learning models.

Machine Learning Approaches

Machine learning forms the core of hate speech detection, where models are trained on labeled datasets comprising both hateful and non-hateful content. Various algorithms, ranging from traditional classifiers like Support Vector Machines (SVM) and Decision Trees to more advanced ensemble methods like Random Forests, are employed in this context. These algorithms learn to distinguish hate speech from benign content by identifying patterns, relationships, and discriminative features within the data.

Machine Learning Models

The emergence of hate speech as a pervasive issue in the digital landscape has led to the development of various machine learning techniques aimed at its detection and mitigation. Hate speech, characterized by discriminatory, offensive, or harmful language targeted at individuals or groups based on attributes such as race, religion, gender, or ethnicity, poses a significant challenge for online platforms, social media networks, and society at large. Addressing this issue requires the utilization of cutting-edge machine learning models and techniques. In this section, we explore models of machine learning that are used in this thesis.

Naïve Bayes Classification: Naïve Bayes is based on Bayes' theorem, which calculates the probability of a document belonging to a particular class based on the probabilities of words occurring in that class. The "naïve" assumption is that words are conditionally independent of each other within a class, which simplifies calculations. Despite this simplification, Naïve Bayes often works well for text classification. It is commonly used in spam email detection and sentiment analysis due to its simplicity and good performance [48].

Support Vector Machines (SVM): SVM aims to find a hyperplane that best separates data points from different classes. In text classification, documents are transformed into high-dimensional feature vectors, with each dimension corresponding to the presence or absence of a word. SVM finds the hyperplane that maximizes the margin between classes. It is effective for text classification tasks due to its ability to handle high-dimensional data and its robustness to overfitting [47].

Decision Trees: Decision trees are hierarchical structures that recursively split data into subsets based on attribute values (words in text classification). At each node, the tree selects the attribute that best separates the data. The result is a tree structure that can be used for classification. Decision trees are interpretable and allow understanding of feature importance in text classification. They work well for tasks where interpretability is crucial.

Random Forests: Random Forests are an ensemble learning method that combines multiple decision trees to improve classification accuracy. Each tree is built using a subset of the data and a random subset of features. The final prediction is based on a majority vote from individual trees. Random Forests enhance the performance of decision trees by reducing overfitting and increasing accuracy.

Logistic Regression: Logistic Regression is a linear model used for binary or multiclass classification. In text classification, it assigns weights to each word, and the weighted sum of word occurrences is transformed using the logistic function to produce class probabilities. Logistic Regression is a simple yet powerful model for text classification. It's especially useful when the relationship between words and classes is approximately linear [49].

K-Nearest Neighbors (KNN): KNN is an instance-based classification method where each document is represented as a point in a high-dimensional space. Classification is determined by the majority class among the k-nearest neighboring data points. KNN is simple and intuitive but requires selecting an appropriate distance metric and the number of neighbors (K). It can be useful in text classification and detecting hate speech.

The utilization of machine learning in hate speech detection marks a significant step toward creating safer and more inclusive digital spaces. As this field continues to evolve, it is imperative to remain at the forefront of cutting-edge research, continuously adapting and developing models and techniques that combat the ever-evolving landscape of hate speech on the Internet.

The Rise of Deep Learning in Text Classification

In recent years, deep learning has emerged as a transformative force in the field of text classification, revolutionizing the way to analyze and understand textual data. In this section, I explore the essence of deep learning and shed light on why it has become the cornerstone of modern text classification.

Unraveling Deep Learning

Deep learning, a subset of machine learning, is a neural network-based approach that has proven to be exceptionally adept at modeling and solving complex textual tasks [50]. At its core, a neural network consists of artificial neurons organized into layers, mimicking the structure of the human brain. Deep neural networks, also known as deep learning models, differentiate themselves by featuring multiple hidden layers, providing the depth required to capture intricate textual patterns. Deep learning models have an unparalleled ability to comprehend the contextual meaning of words and phrases within text. They can capture nuances, idiomatic expressions, and sentiment, making them invaluable for tasks like sentiment analysis, and hate speech detection. Deep learning models are remarkably versatile, proving effective across a multitude of text classification domains. Whether it is classifying texts or understanding sentiments, deep learning models adapt well to diverse textual challenges. This paradigm shift has not only enhanced the accuracy of text classification but has also expanded the possibilities for understanding and utilizing textual data in the digital environment.

Deep Learning Approaches in Text Classification

Deep learning approaches have garnered considerable attention in the field of text classification, primarily owing to their exceptional capacity to extract intricate patterns and semantic relationships within textual data. In this section, we explore the deep learning techniques that I used to detect hate speech.

LSTM

In recent years, deep learning has revolutionized the field of artificial intelligence, and one of its most prominent achievements is in natural language processing (NLP). LSTM is a specialized type of recurrent neural network (RNN) that has garnered significant attention and success in sequence modeling tasks. In this section, I will delve into the foundational concepts of LSTM, its architecture, working mechanisms, and its diverse applications across various domains. By the end of this section, you will have a proper understanding of LSTM's capabilities and why it has emerged as a powerful tool.

The Challenge of Sequential Data

Many real-world datasets, such as time series data, speech signals, and natural language text, exhibit a sequential nature where the order of elements matters. Traditional neural networks, designed for independent and identically distributed data (i.i.d), are ill-equipped to handle sequential information. This limitation inspired the development of RNNs, which can process sequences by maintaining a hidden state that evolves with each new element in the sequence.

Introducing LSTM

To address the shortcomings of standard RNNs, Hochreiter and Schmidhuber introduced LSTM in 1997 [51]. LSTM is designed to capture long-range dependencies effectively, making it well-suited for tasks involving sequential data. The key innovation in LSTM lies in its memory cell, which allows it to store and retrieve information over extended periods. This capability enables LSTM to maintain essential information even after processing a vast amount of sequential data.

Parameters and Hyperparameters in LSTM

LSTM models have several key parameters and hyperparameters that impact their performance and generalization capability.

1. **Number of LSTM Units:** The number of LSTM units or cells determines the capacity of the LSTM layer to capture information. A higher number of units may allow the model to capture more complex patterns but may also increase computational complexity.
2. **Sequence Length:** The sequence length is an important consideration in LSTM. Longer sequences require more memory and may increase training time. It's essential to strike a balance between the sequence length and the LSTM's memory capacity.
3. **Activation Functions:** Activation functions introduce non-linearity to the model, enabling it to capture complex relationships. Common activation functions used in LSTM include the sigmoid function for gating mechanisms and the hyperbolic tangent (tanh) function for the output.
4. **Dropout:** Dropout is a regularization technique commonly applied in LSTM models to prevent overfitting. It randomly sets a fraction of the LSTM units to zero during training, forcing the network to learn more robust representations.
5. **Learning Rate:** The learning rate controls the step size during gradient descent optimization. It is a crucial hyperparameter that affects the training process, and an appropriate learning rate can significantly impact the convergence and performance of the LSTM.

LSTM has demonstrated exceptional performance across various applications, making it a versatile and powerful tool for sequence modeling tasks. LSTM is widely used in NLP for tasks such as sentiment analysis, name entity recognition, machine translation, and text generation [52].

Its ability to understand the sequential nature of language and capture contextual dependencies makes it well-suited for these tasks.

LSTM has emerged as a powerful architecture for sequence modeling tasks, particularly in the domain of natural language processing. Its ability to capture long-range dependencies, handle variable-length sequences, and discern contextual nuances has made it a prominent choice for a wide range of NLP applications.

CNN

CNN has been a revolutionary development in the field of deep learning and has shown remarkable success in computer vision tasks. However, their applications are not limited to visual data alone; they have also proven to be highly effective in text classification tasks. In this section, we will delve into the fundamental concepts of CNN, its architectural components, and how it is adapted and utilized for text classification, with a particular focus on its relevance in detecting hate speech.

Understanding CNN

At its core, a CNN is designed to learn spatial hierarchies of features automatically and adaptively from input data. It was originally inspired by the visual processing system of living organisms, particularly the hierarchical organization of visual cortex in the brain [53]. CNNs are well-known for their ability to detect patterns in images, such as edges and textures, through the application of convolutional filters.

The basic building blocks of a CNN include convolutional layers, activation functions, pooling layers, and fully connected layers. The convolutional layers are the heart of the network

and consist of filters, also known as kernels, that slide over the input data to perform element-wise multiplications and obtain feature maps. These feature maps capture local patterns and represent them as spatially organized representations.

Layers in CNN for Text Classification

Adapting CNN for text classification tasks requires addressing the one-dimensional nature of textual data. In traditional CNNs used for images, the convolutional filters operate on two-dimensional grids. However, text data is one-dimensional, typically represented as sequences of words or characters. To accommodate this, a technique called word embedding is employed to transform each word into a dense vector representation, creating an embedding matrix as the input to the CNN.

CNN for Text Classification

CNNs are particularly advantageous in text classification due to their ability to identify local features and capture hierarchical relationships in sequences. In text classification, CNN is trained to learn meaningful patterns from word embeddings, such as specific phrases or linguistic structures that indicate the class of the input text.

Moreover, CNN's robustness to text variability is highly advantageous in hate speech detection. Online hate speech frequently includes variations in language use, including slang, abbreviations, and creative spellings. CNN's ability to detect local features rather than relying on exact matches makes it well-equipped to handle these variations, ensuring accurate identification of hate speech across diverse textual inputs.

BERT: Revolutionizing Natural Language Understanding

BERT (Bidirectional Encoder Representations from Transformers) is a groundbreaking NLP model that has reshaped the landscape of deep learning-based language understanding. Introduced by Google AI researchers in 2018, BERT brought unprecedented advancements in contextual word representation, enabling the model to capture bidirectional context and outperform previous NLP models on various tasks. In this section, we will delve into the key components, architecture, training process, and applications of BERT, highlighting why it has become a pivotal milestone in NLP [25, 28].

The Challenge of Contextual Word Representation

Language understanding heavily relies on capturing the context in which words appear. Traditional NLP models, such as word embeddings and context-free models like Word2Vec and GloVe, treat each word as an isolated entity, neglecting the contextual information. This limitation led to suboptimal performance in downstream NLP tasks that require a deeper understanding of the context.

Introducing BERT: A Bidirectional Approach

BERT addresses the context problem by introducing a novel bidirectional approach. Unlike previous models that processed text in a unidirectional manner (either left-to-right or right-to-left), BERT utilizes a transformer-based architecture to analyze words in both directions simultaneously. This bidirectional approach allows BERT to understand the context and dependencies of each word in a sentence more comprehensively.

The Transformer Architecture

The transformer architecture is the backbone of BERT and is based on the self-attention mechanism. Self-attention enables the model to weigh the importance of each word in relation to all other words in the sentence, enabling more informed word representations. The transformer consists of an encoder and a decoder, but in BERT, only the encoder is used for unsupervised pre-training.

BERT's Pre-training Process

The pre-training process is a crucial phase of BERT, where it learns unsupervised from vast amounts of unannotated text data. BERT is pre-trained on a large corpus by predicting missing words from the context in a masked language modeling (MLM) task. During MLM, a certain percentage of words are masked, and the model is tasked with predicting the masked words based on the surrounding context.

Additionally, BERT utilizes a next sentence prediction (NSP) task during pre-training to foster a deeper understanding of sentence relationships. In NSP, two sentences are paired, and the model must determine whether the second sentence follows the first in the original text. This task encourages BERT to grasp contextual relationships between sentences.

BERT's Architecture: From Layers to Hidden States

BERT consists of several layers of encoders, each containing a fixed number of self-attention heads and feed-forward neural networks. The self-attention heads enable BERT to capture dependencies between words, while the feed-forward networks process each word's representation. By stacking multiple layers, BERT gains a deeper understanding of the context and can model complex linguistic patterns.

The output of each BERT layer is the hidden states of all words in the input sequence. These hidden states contain rich contextual information for each word and can be extracted for downstream tasks or fine-tuning.

Fine-Tuning BERT for Specific Tasks

While BERT's pre-training enables it to capture general language representations, its true power comes from fine-tuning. After pre-training on a massive corpus, BERT is fine-tuned on specific tasks using labeled data. Fine-tuning involves adjusting the model's parameters to the task at hand, allowing it to adapt its representations to the specifics of the downstream task.

Applications of BERT

BERT's ability to generate powerful contextual word representations has led to its adoption across various NLP tasks [25, 54]. Some of the notable applications of BERT include:

1. **Text Classification:** BERT is widely used for text classification tasks such as sentiment analysis, topic classification, and spam detection. Its contextual understanding of text allows it to achieve state-of-the-art performance in these tasks.
2. **Named Entity Recognition (NER):** BERT excels in NER, where it can identify and classify entities such as names, locations, and organizations within a text.
3. **Question Answering:** BERT is employed in question-answering systems, enabling them to comprehend complex queries and provide accurate responses.
4. **Machine Translation:** BERT's contextual embeddings have been leveraged to improve machine translation systems by enhancing the quality of the encoded source and target language sentences.

5. Information Retrieval: BERT's contextual representations have been integrated into search engines to improve the relevance and accuracy of search results.

BERT Variant

Since its inception, BERT has inspired numerous extensions and variants to cater to different use cases and improve its performance further. One of the notable variants is RoBERTa, short for "A Robustly Optimized BERT Pretraining Approach," is an extension of BERT that optimizes the pre-training process by increasing the training data, using larger batch sizes, and training for more iterations [55]. This results in improved performance and better generalization capabilities. RoBERTa represents a pivotal milestone in the evolution of transformer-based language models. Developed by Facebook AI, RoBERTa builds upon the success of BERT by refining its pretraining process. Like BERT, RoBERTa uses a masked language modeling objective during pretraining. It learns to predict masked words within a sentence, which encourages the model to understand the contextual relationships between words. It introduces dynamic masking, which means that during training, it randomly masks out and replaces different spans of text in each batch. This technique encourages the model to focus on different parts of the text in each training iteration, improving its robustness and generalization. On the other hand, RoBERTa doesn't use the next sentence pretraining objective. The choice between RoBERTa and BERT often depends on the specific requirements and constraints of the NLP task.

METHODOLOGY

The Hate Speech detection framework is divided into four stages: Stage-1 preprocesses the datasets that was used to detect hate speech. In Stage-2, I implement traditional machine learning algorithms on the datasets and evaluate the results which helped detect hate speech. In Stage-3, I implement CNN and LSTM on the datasets and compare the results with the traditional ML algorithms. In Stage-4, I apply BERT to detect hate speech as it is a new approach. In this section, I discuss the four stages of my research.

Stage-1: In the first stage the datasets were preprocessed to clean them and prepare the data to be able to use the machine learning algorithms on them. For the preparation, TF-IDF, unigram, bigram, trigram, combination of unigram and bigram, and combination of unigram, bigram, and trigram were used for the text corpus to be read by the machine learning techniques to see which approach gives better results.

This stage consists of preprocessing the datasets which are required for the ML algorithms. In the experiment, I use three datasets which are already labeled. Dataset I is created by Z. Waseem and D. Hovy [7]. It consists of tweets that were manually labeled for hate speech by human annotators. The authors collected the dataset by searching Twitter for specific keywords related to hate speech, such as racial slurs and derogatory terms. The dataset contains a total of 16,907 tweets with 1,976 data labeled as racism, 3,430 labeled as sexism and, 11,501 labeled as neither racist nor sexist content. The dataset is not necessarily representative of all instances of hate speech on Twitter but is instead intended to evaluate ML models to detect hate speech. Dataset II is released by Impermium on Kaggle [14]. It contains social media comments that are labeled as either

insulting or not insulting. The aim of this dataset is to develop ML models for automatically detecting insults in online comments. The dataset includes 6,594 comments in English, collected from various social media platforms such as Facebook, Twitter, and Reddit. Each comment was also labeled by human annotators as either insulting or not insulting. The dataset also includes additional information such as the date and time of the comment, however, I only focus on the comments and labels from the dataset. The dataset contains 3,947 training data samples with 1,049 samples labeled as insulting and 2,898 labeled as non-insulting. It also contains 2,647 test samples with 6,93 labeled as insulting and 1,954 labeled as non-insulting. Dataset III is created by T. Davidson, D. Warmesley, M. Macy, and I. Weber [15]. It is made up of tweets that are manually classified as hate speech. Like the previous datasets, this dataset covers tweets that are in English. It contains 24,783 tweets with 1,430 labeled as hate speech, 19,190 labeled as offensive language, and 4,163 labeled as neither.

In the experiments, I consider samples labeled “offensive language” as “hate speech”. My next step after preprocessing includes under-sampling or over-sampling the datasets to make them balanced, where the resulting datasets have the same number of objects in each class. An overview of the steps I followed is shown in Figure 1. Then I apply the ML algorithms and DL models on the datasets and compare the results. Additionally, for the experiment, I consider label 0 as not hate speech and labels 1 and 2 (for example, in case of a class label such as “offensive language”) as hate speech in all three datasets.

Stage-2: The second stage is to evaluate the machine learning algorithms. Under-sampling and over-sampling were used to balance the datasets. Then compare the results with original unbalanced datasets.

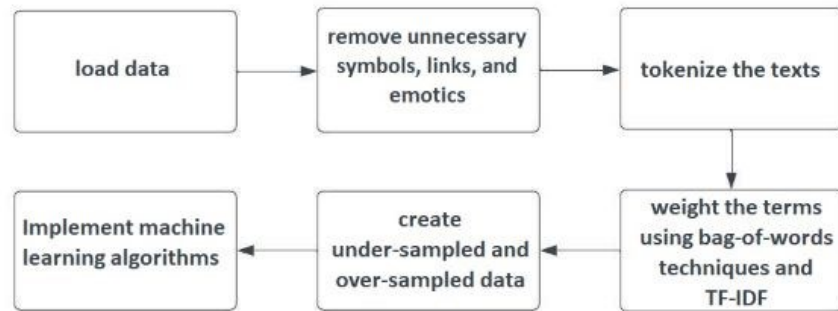


Figure 1: Overview of the System

First, I evaluated the conventional ML algorithms using just the preprocessed datasets. The ML methods that were used are: Naïve Bayes, SVM, Decision Trees, Random Forests, Logistic Regression, and K-Nearest Neighbors.

Stage-2 consists of under-sampling the datasets. I used the lowest labeled category as the focal interest of the sampling and used the same number to cut down the other labeled categories. Then used the previous ML algorithms on the under-sampled datasets and compare with previous results.

Finally in Stage-2, I added more data to the category of the datasets with fewer objects to make the number of objects equal to the highest labeled category of the datasets. After over-sampling, I used the ML algorithms and compared the results.

Stage-3: The third stage is to use the popular deep learning algorithms LSTM and CNN to detect hate speech. I used GloVe word embeddings with LSTM and CNN since GloVe uses a global

matrix factorization approach to directly optimize the word vectors based on their co-occurrence counts, in contrast to other word embedding techniques like Word2Vec that depend on predicting the context of a word.

LSTM for Hate Speech Detection

The detection of hate speech in online communication is a pressing challenge, as it involves analyzing complex language patterns and identifying subtle nuances that convey harmful intentions. LSTM's ability to capture long-range dependencies is especially critical in detecting hate speech, where context plays a vital role in understanding the true meaning of the text. For instance, LSTM can recognize that a word or phrase used early in the text may significantly impact the classification decision later on.

Hate speech often involves veiled threats, sarcasm, and other subtle expressions that may be challenging to detect without an understanding of the text's broader context. LSTM's memory cell allows it to retain such information, making it adept at discerning hate speech from seemingly innocuous statements. Moreover, online communication platforms often involve user-generated content with variable lengths, including short tweets, comments, or longer posts. LSTM's dynamic memory allocation enables it to process texts of different lengths efficiently, ensuring consistent and accurate hate speech detection across diverse inputs.

The LSTM Architecture

At the heart of LSTM are the memory cell and three fundamental gates: the input gate, the forget gate, and the output gate. Figure 2 shows the architecture of LSTM model [50].

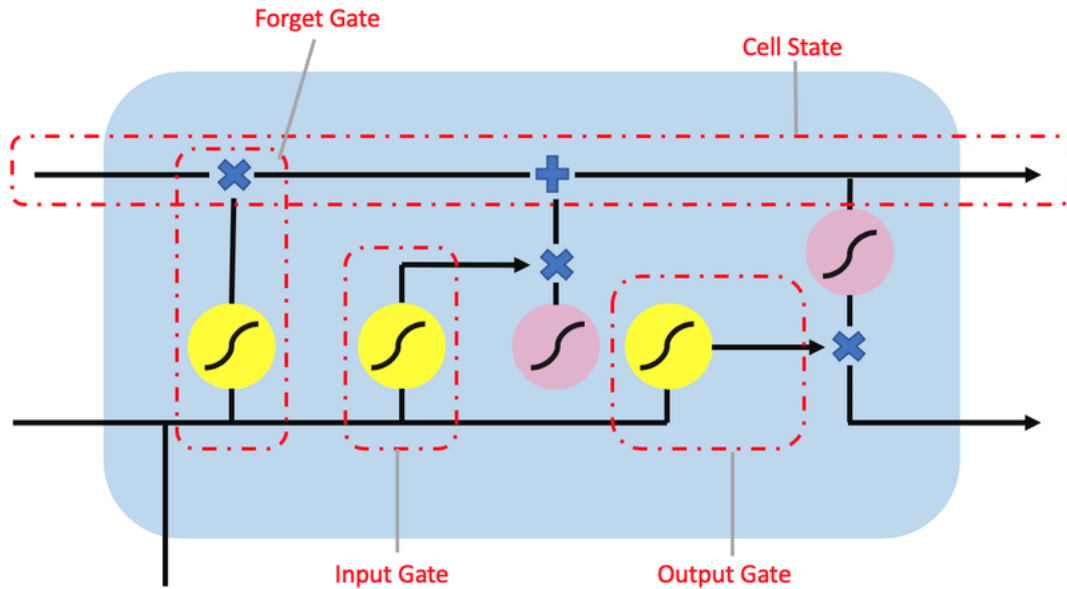


Figure 2: Architecture of LSTM

1. Memory Cell: The memory cell is the crux of LSTM and functions as a conveyor belt, carrying information over time steps. At each time step, the memory cell decides what information to keep and what to discard. This dynamic memory allocation is the essence of LSTM's ability to capture long-range dependencies.
2. Input Gate: The input gate controls the flow of new information into the memory cell. It takes input from the current time step and the previous hidden state, and through a sigmoid activation function, it decides which information should be added to the memory cell.
3. Forget Gate: The forget gate regulates what information to discard from the memory cell. It takes input from the current time step and the previous hidden state and, through a sigmoid activation function, decides which information to forget from the memory cell.

4. Output Gate: The output gate determines the output of the current time step. It takes input from the current time step and the previous hidden state, along with information from the memory cell, and through a sigmoid activation function, it decides what information to output as the hidden state for the current time step.

The LSTM Working Mechanism

During the forward pass of LSTM, the input sequence is fed into the network one element at a time. At each time step, the LSTM performs computations using the input gate, forget gate, and output gate, updating the memory cell and hidden state accordingly. The hidden state carries information that has been distilled from the entire input sequence, while the memory cell retains important information for long periods.

CNN for Hate Speech Detection

Hate speech detection is a challenging task in natural language processing, primarily because of the complex and context-dependent nature of hate speech. CNN's ability to capture local contextual information is crucial in identifying the discriminatory language and harmful intentions present in hate speech. The use of convolutional filters allows CNN to detect specific n-grams that are indicative of hate speech, even if they are embedded within larger text segments. This makes CNN well-suited for identifying hate speech in online communication platforms like social media, where content is often brief and contextual cues are essential.

Another crucial aspect is the scalability and efficiency of CNN. Hate speech is pervasive on social media platforms with millions of posts and comments generated every second. CNN's parallel processing capabilities allow it to analyze large-scale datasets in real-time, making it an

ideal candidate for hate speech detection on such platforms. Figure 3 shows the architecture of CNN.

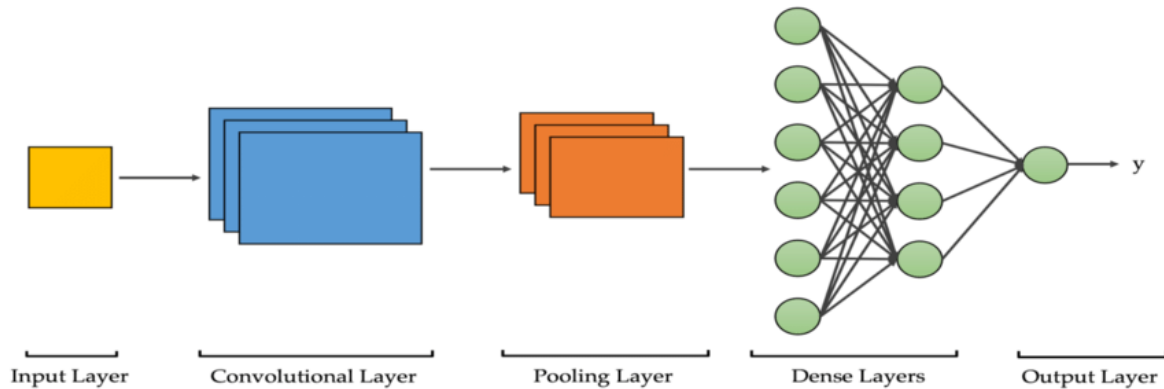


Figure 3: Architecture of CNN

Parameters and Architectural Considerations

Word Embeddings: Word embeddings are the foundational input for CNN in text classification. These dense vector representations are learned from large corpora using techniques like Word2Vec or GloVe, and they encode semantic relationships between words. The embedding layer provides a continuous representation of each word, allowing CNN to process textual information effectively.

Convolutional Filters: In text-based CNN, the convolutional filters slide over the sequence of word embeddings. These filters capture n-gram features, where n refers to the number of words considered together. For example, a 1-gram filter represents single words, a 2-gram filter considers pairs of adjacent words, and so on. By using different filter sizes, the network can learn features at varying contextual levels.

Pooling Layers: After the convolutional layer, pooling layers are used to reduce the dimensionality of the feature maps. Max pooling is commonly employed, where the maximum value within a specific window is taken, effectively capturing the most salient features. This down-sampling reduces computational complexity and retains essential information.

Activation Functions: Activation functions introduce non-linearity to the model, allowing it to capture complex relationships between features. In text classification, common activation functions include Rectified Linear Unit (ReLU) and Hyperbolic Tangent (tanh).

Fully Connected Layers: Following the convolutional and pooling layers, the fully connected layers aggregate the information and make the final predictions. These layers are typically combined with softmax activation in multi-class text classification to produce probability distributions over possible classes.

Stage-4: The fourth stage of the proposed system framework uses BERT on the cleaned datasets as well as under- and oversampling them before using the same BERT model again. I started the research with the datasets, where I employed machine learning techniques on the actual data, i.e., without using over- or under-sampling. However, unlike standard ML algorithms, the model produced better results because it is intended to address NLP-related tasks. Words are converted into fixed-dimension vector representations via the token embeddings layer. Then compare the BERT model by using original cleaned datasets with the best result using traditional machine language. I also compare them with LSTM and CNN.

BERT for Hate Speech Detection

BERT's contextual understanding of language and ability to capture complex linguistic patterns make it an excellent choice for hate speech detection. Hate speech often involves subtle expressions and context-specific language that require a deep understanding of the context to be effectively detected.

Traditional NLP models struggle with hate speech detection due to their inability to capture such contextual nuances. BERT's ability to model bidirectional context allows it to grasp the subtleties of language used in hate speech and distinguish between harmful and non-harmful content. Moreover, hate speech often evolves over time and across different online communities. BERT's fine-tuning capability enables it to adapt and learn from specific labeled datasets, making it a powerful tool for handling the dynamic nature of hate speech in online environments.

BERT has revolutionized natural language understanding and significantly advanced the field of NLP. Its ability to capture contextual word representations bidirectionally has led to state-of-the-art performance on various NLP tasks. The success of BERT in hate speech detection highlights its effectiveness in handling complex language data and addressing the challenges of identifying harmful content in online platforms. With ongoing research and the development of new BERT-based variants, the future holds exciting possibilities for further advancements in NLP.

I will use BERT without preprocessing the dataset. BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

Figure 4 shows the architecture of BERT.

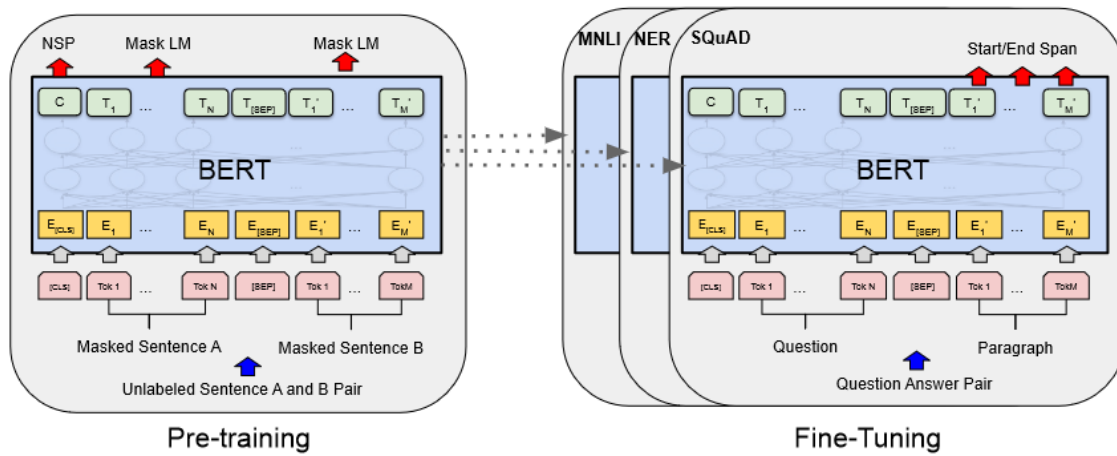


Figure 4. Architecture of BERT

EXPERIMENTAL SETUP AND RESULTS

In this section, I cover the research progress I have made to develop the proposed framework and present the results I obtained to conclude the research. I use the scikit-learn Python library for the experiments. The datasets are transformed into vectors by implementing bag-of-words and tokenizing the text. Tokenized terms are weighted by different data processing techniques (bag-of-words/unigram, bigram, trigram, combination of unigram and bigram, and combination of unigram, bigram, and trigram) and applying TF-IDF. Each data processing technique was tested separately to find out the technique that gives the best performance results. I use the TensorFlow library for the deep-learning models.

Stage-1 of the proposed framework

Stage-1 of the proposed hate speech detection framework consists of preprocessing the datasets which are required for the ML algorithms. The preprocessing phase involves preparing three labeled datasets for ML algorithms. Dataset I, curated by Z. Waseem and D. Hovy, consists of 16,907 tweets manually labeled for hate speech, with 1,976 racism, 3,430 sexism, and 11,501 neither racist or sexist. While not fully representative, it serves to evaluate ML models for hate speech detection. Dataset II, from Impermium on Kaggle, includes 6,594 English comments labeled as insulting or not insulting, with 1,049 insulting and 2,898 non-insulting training samples, and 693 insulting and 1,954 non-insulting test samples. Dataset III, created by T. Davidson et al., contains 24,783 English tweets manually classified as hate speech, with 1,430 hate speech, 19,190 offensive language, and 4,163 neither. Offensive language samples are considered as hate speech in the experiments.

All the datasets were full of a lot of Internet jargon and gibberish terms. So, I make sure that the datasets are clean, and training and testing ready for the machine learning algorithms. The first step of my preprocessing is to remove the unnecessary symbols (such as "~", "\\\\", "-_-", etc), links and emoticons. I also eliminate irrelevant data (such as "your WS/na fluke", "@...PetrMailbox @SidneyCatsby BOOM" etc) that would have a negative impact on the accuracy of the model in order to maintain consistency and the highest accuracy possible for ML models. I eliminate those posts because they lack context. Since I use scikit-learn python library for my experiment, the datasets are transformed into vectors by implementing bag-of-words and tokenizing the text. Tokenized terms are weighted by different data processing techniques (bag-of-words/unigram, bigram, trigram, combination of unigram and bigram, and combination of unigram, bigram, and trigram) and applying TF-IDF. Each data processing technique was tested separately to find out the technique that would give the best performance results.

I experimented with different training and testing percentage ranges for dividing the data into training and testing sets. Following a great deal of testing, I discovered that allocating 90% of the data to the training set and 10% to the testing set yields the highest accuracy on average. I used this allocation for the results. In my experiments, I used 10-fold cross validation. I used accuracy and F1-score to evaluate the models because the datasets are imbalanced.

Stage-2 of the proposed framework

Stage-2 of the framework consists of applying ML algorithms on the preprocessed datasets and also under-sampling and over-sampling the datasets and again applying ML algorithms on them. I started the experiments with Dataset I, where I used the machine learning algorithms on the original data, i.e., without applying over-sampling or under-sampling.

Since the dataset is not balanced with more text samples labeled as neither sexist nor racist than samples labeled sexist or racist, the results on the original data show poor results. I concentrate on F1 measure as it is the best performance measure to use for imbalanced datasets. The F1-score when using TF-IDF on Naïve Bayes is 0.743. However, KNN has given the lowest F1-score while using Unigrams, which is 0.456. I even used Grid Search to find the best parameters in the algorithms but did not get a better result. The lowest F1-score on the original data is found using bigrams on KNN, which is 0.344. However, The F1-score, using TF-IDF on SVM, is 0.756 which is the highest. It also gives the highest Accuracy, 0.827. On Dataset II, the F1-score of Naïve Bayes is 0.622 and is the highest with the help of TF-IDF. I also got the lowest F1-score on KNN using trigrams. On Dataset III, the highest F1-score I obtained was from SVM as well, also using TF-IDF. The lowest F1-score also comes by using KNN with trigrams as well. Table 1 shows the best results I obtained from all the ML algorithms with the word embeddings techniques.

Table 1: Best Results of ML Algorithms on the Original Datasets

Model	Dataset I		Dataset II		Dataset III	
	ACC(%)	F1(%)	ACC(%)	F1(%)	ACC(%)	F1(%)
Naïve Bayes-TFIDF	81.4	74.3	83.2	62.2	86.2	56.7
SVM- TFIDF	82.7	75.6	82.5	60.0	91.3	70.8
Decision Tree- TFIDF	79.9	71.4	77.8	59.3	88.2	67.9
Random Forests- Unigrams	81.8	72.1	82.4	54.8	86.4	65.8
Logistic Regression-TFIDF	82.1	75.1	80.8	57.7	90.1	68.3
KNN-Unigrams	71.9	45.6	74.8	51.4	81.4	54.5

To make the datasets more balanced on labels, I tested by under-sampling the data. This didn't give the best results but gave a better F1-score overall. But under-sampling didn't improve

the result of Dataset I, as it resulted in F1-score of 0.745 for Naïve Bayes using TF-IDF. The F1-score using trigram is the lowest on Naïve Bayes, 0.517. Trigrams gave the lowest F1-score using any machine learning algorithms.

I attribute this to the lower frequency of trigram phrases in the dataset, which made it harder for the ML algorithms to classify based on using three-word phrases as the data features. F1-score using TF-IDF on Logistic Regression is 0.786. It is the highest I get from the under-sampled data. I explain the low results with under-sampling to the fact that under-sampling reduces the size of the dataset used for training and testing, which could make the performance of a ML algorithm not as good as compared to when having more data. For Dataset II and Dataset III, the results were mediocre, just about the same as Dataset I but did give a better F1-score than the original dataset overall. Table 2 shows the best results over all the datasets that I obtained using all the ML algorithms with the word embeddings techniques.

Table 2: Best Results of ML Algorithms on the Undersampled Datasets

Model	Dataset I		Dataset II		Dataset III	
	ACC(%)	F1(%)	ACC(%)	F1(%)	ACC(%)	F1(%)
Naïve Bayes-Unigrams	75.5	74.5	79.3	81.3	74.9	75.0
SVM- TF	76.8	76.9	80.1	79.9	80.5	80.2
Decision Tree- Uni & Bigrams	70.2	70.4	74.2	75.1	76.3	75.9
Random Forests- Unigrams	77.0	76.9	79.4	79.3	78.4	78.0
Logistic Regression- TFIDF	78.4	78.6	74.7	73.2	79.4	78.9
KNN-Unigrams	50.3	48.4	68.7	70.8	56.5	56.0

To get more data and balanced class distributions, I over-sampled the data. Table 3 shows the best results of the ML algorithms by over-sampling the datasets. Over-sampling resulted in the best performance results for all algorithms. The F1-score on Naïve Bayes using the combination of unigram, bigram, and trigram is 0.882. The best F1-score I found was using the combination of unigram, bigram, and trigram on Random Forests, which is 0.970. My explanation for the best performance with oversampling is that over-sampling the data and using the combination of n-grams resulted in having more and balanced data with better features for the ML algorithms, which in turn helped getting the best accuracy, and F1-score using most ML algorithms.

Table 3: Results of ML Algorithms on the Oversampled Datasets

Model	Dataset I		Dataset II		Dataset III	
	ACC(%)	F1(%)	ACC(%)	F1(%)	ACC(%)	F1(%)
Naïve Bayes-Ngrams Combined	88.8	88.2	87.7	88.9	96.0	96.0
SVM -Ngrams Combined	95.9	95.9	94.0	94.2	97.4	97.4
Decision Tree- TF	92.9	92.8	88.7	89.3	96.6	96.5
Random Forests-Ngrams						
Combined	97.0	97.0	96.0	96.0	98.3	98.2
Logistic Regression-Ngrams						
Combined	95.7	95.7	93.6	93.9	97.3	97.2
KNN- Unigrams	87.8	87.7	84.3	84.2	92.5	92.4

Dataset II also gave the best results when I over-sampled the dataset. However, this dataset came with separate training data and testing data. Over-sampling both data separately and then applying the machine learning algorithms did not give promising results as the training data was not of sufficient amount to help weight the terms more. So, I combined the training and testing

data and then over-sampled it and then used 20% of the data for testing. This helped get really good results similar to what I obtained with Dataset I. Again, the best F1-score I get is using the combination of unigram, bigram, and trigram on Random Forest, which is 0.960. I also experimented with a third dataset that I refer to as Dataset III in the same way, and I got the best results when the dataset is over-sampled. The best F1-score I get again is using the combination of unigram, bigram, and trigram on Random Forest, which is 0.982.

The experimental results in this table show that over-sampling an unbalanced dataset and using n-grams as word embeddings for the tested ML algorithm gives excellent results. Even for KNN algorithms, better results can be obtained when the datasets are balanced with oversampling. After Random Forest, SVM and Logistic Regression give very good results. The F1-score using SVM on Dataset I is 0.959, Dataset II is 0.942, and Dataset III is 0.974. Then, the F1-score using Logistic Regression on Dataset I is 0.957, Dataset II is 0.939, and Dataset III is 0.972.

Stage-3 of the proposed framework

The Stage-3 of the framework mostly focuses on the original imbalanced dataset since in the real world most of the datasets are imbalanced and one needs to build a model based on that. So, I use the original datasets and apply LSTM and CNN deep learning algorithms. I loaded GloVe-100D embedding file for the experiment, which is a pre-trained word embedding model that was developed by the Stanford NLP Group [56].

GloVe stands for Global Vectors, and the “100D” refers to the dimensionality of the word vectors, and then loaded embedding vectors of words which comes in GloVe file and others (words which are not found in the GloVe file) were equated to 0. It is because, with a pre-trained word

embedding model like GloVe, not all words in a given dataset or vocabulary may be present in the pre-trained model. In this case, the embedding vectors for these words can be set to zero. This approach ensures that all words in the dataset are represented by a vector, even if their vector values are zero. Also, the embedding matrix was created that was used in the embedding of the models for the dimension and weights.

The LSTM model reports the lowest F1-score for each dataset among the DL models, having the highest F1-score of 82.9% on Dataset III as shown in Table-4. Nonetheless, the ML models performed worse than the DL models, with SVM model having the best F1-score among the ML models (75.6%). The CNN model performs adequately with the best F1-scores as shown in Table 4 among all other algorithms and models that were used till now to detect hate speech.

Table 4: Experiment Results of All Models

Model	Dataset I		Dataset II		Dataset III	
	ACC(%)	F1 (%)	ACC(%)	F1 (%)	ACC(%)	F1 (%)
BERT	93.4	89.7	89.8	88.2	89.6	90.6
CNN-GloVe	87.7	82.8	85.6	82.5	87.3	83.2
LSTM-GloVe	86.8	81.1	84.2	80.4	85.7	82.9
Naïve Bayes-TFIDF	81.4	74.3	83.2	62.2	86.2	56.7
SVM-TFIDF	82.7	75.6	82.5	60.0	91.3	70.8
Decision Tree-TFIDF	79.9	71.4	77.8	59.3	88.2	67.9
Random Forests-Uniwords	81.8	72.1	82.4	54.8	86.4	65.8
Logistic Regression-TFIDF	82.1	75.1	80.8	57.7	90.1	68.3
KNN-Uniwords	71.9	45.6	74.8	51.4	81.4	54.5

Stage-4 of the proposed framework

Stage-4 of the proposed framework consists of applying BERT on the cleaned datasets and also under-sampling and over-sampling the datasets and again applying the same model of BERT on them. I started my experiments with Dataset I, where I used the machine learning algorithms on the original data, i.e., without applying over-sampling or under-sampling but unlike the conventional ML algorithms, the model gave better results as it is designed for solving NLP-related tasks. The Token Embeddings layer transforms words into vector representations of fixed dimension.

In the case of BERT, each word is represented as a 768-dimensional vector and each token is given an ID and the same words are given the same ID. A sample of the token IDs is shown in Figure 5. The BERT model also needs the attention mask to understand the importance of the words. In the experiment, I used the attention mask tensors, tokens and IDs to make BERT understand the words. One sample is shown in Figure 6. Then I used a base BERT model that consists of 12 encoders, whereas BERT large model has 24 encoders. However, the results were simply close to the result on BERT large.

Tokens	Token IDs
bitch	7743
##es	2229
be	2022
like	2066
ni	9152
##gga	23033
##s	2015
ain	7110
##t	2102
shit	4485
.	1012

Figure 5: Token IDs of a Sample

In my experiment, BERT model performs the best overall on the original datasets, with F1-score of 89.7% on Dataset I, 88.2% on Dataset II, and 90.6% on Dataset III. It makes CNN the second-best model after BERT. Then LSTM did almost as good as CNN and then the ML algorithms were the ones to fall behind in detecting hate speech. Table 4 shows the evaluation results from the nine models on the original Dataset I, Dataset II, and Dataset III.

After the results, I gathered six posts from the Internet where three of them were hateful comments ('You deserve to die.', 'You're a girl, you should know better.', 'Let's Kill Jews, and let's kill them for fun. #killjews') and three of them were non-hateful comments ('It was a fun day.', 'I think you have a lot of potential, and if you work on your public speaking skills, you could really excel in your career.', 'How was your weekend? Did you do anything fun?'). Then gave the BERT model to detect hate speech after training and testing from the three datasets.

Tokens	Token IDs	Attention Mask
[CLS]	101	1
so	2061	1
where	2073	1
is	2003	1
your	2115	1
evidence	3350	1
of	1997	1
the	1996	1
intentional	21249	1
destruction	6215	1
of	1997	1
water	2300	1

Figure 6: Tokens and Attention Masks of a Sample

Figure 7 shows that after training on the Dataset I, even though BERT has gotten a good result, it was not able to detect one of the comments as hateful. Figure 8 shows that after using the Dataset II, only one hateful sentence was not detected correctly. Figure 9 shows Dataset III which also has given the best result to detect the correct label using BERT. It was able to detect all the sample comments correctly. BERT may not be able to give 100% accuracy and F1-score, but it shows the capability of BERT after using the six random samples. For all the datasets, I know that I had more data on the non-hateful posts and that helped the model learn non-hateful posts with the knowledge of context.

Input Sentence: You deserve to die.
Predicted Class: Non-Hate
Input Sentence: You're a girl, you should know better.
Predicted Class: Hateful
Input Sentence: Let's Kill Jews, and let's kill them for fun. #killjews
Predicted Class: Hateful
Input Sentence: It was a fun day.
Predicted Class: Non-Hate
Input Sentence: I think you have a lot of potential, and if you work on your public speaking skills, you could really excel in your career.
Predicted Class: Non-Hate
Input Sentence: How was your weekend? Did you do anything fun?
Predicted Class: Non-Hate

Figure 7: Predicting Hate Speech from Dataset I

Input Sentence: You deserve to die.
Predicted Class: Hateful
Input Sentence: You're a girl, you should know better.
Predicted Class: Hateful
Input Sentence: Let's Kill Jews, and let's kill them for fun. #killjews
Predicted Class: Non-Hate
Input Sentence: It was a fun day.
Predicted Class: Non-Hate
Input Sentence: I think you have a lot of potential, and if you work on your public speaking skills, you could really excel in your career.
Predicted Class: Non-Hate
Input Sentence: How was your weekend? Did you do anything fun?
Predicted Class: Non-Hate

Figure 8: Predicting Hate Speech from Dataset II

Input Sentence: You deserve to die.
Predicted Class: Hateful
Input Sentence: You're a girl, you should know better.
Predicted Class: Hateful
Input Sentence: Let's Kill Jews, and let's kill them for fun. #killjews
Predicted Class: Hateful
Input Sentence: It was a fun day.
Predicted Class: Non-Hate
Input Sentence: I think you have a lot of potential, and if you work on your public speaking skills, you could really excel in your career.
Predicted Class: Non-Hate
Input Sentence: How was your weekend? Did you do anything fun?
Predicted Class: Non-Hate

Figure 9: Predicting Hate Speech from Dataset III

COMPARISON OF THE MODELS

In the previous section, BERT demonstrated promising results in detecting hate speech within the online texts. It is essential to compare BERT with traditional ML algorithms, specifically SVM, as SVM provided the best results among traditional ML methods. This comparison aims to assess whether BERT excels at understanding context, as it yielded approximately 20% higher F1 results than SVM. Furthermore, in this section, I also conducted a comparison with another pretrained model, RoBERTa. The goal was to determine if BERT's performance in detecting hate speech is unique or if other models, such as RoBERTa, can achieve similar results.

SVM Comparison

After testing the datasets with an SVM model, I collected the tweets that the model predicted incorrectly. I then applied these data to the BERT model. Interestingly, I found that even the tweets incorrectly predicted by the SVM model had context, and BERT was able to predict them correctly as shown in Figure 10, Figure 11, and Figure 12. It's worth noting that not all the tweets predicted wrongly by SVM had context. Some of the tweets were lengthy in Dataset II, but BERT still predicted them accurately. It appears that traditional ML algorithms like SVM tend to perform well in predicting the majority class but struggle with the minority class.

Input Sentence: Maybe the girls should have less tickets on themselves and worry about the cooking. #MKR.
Actual Class: Hateful | Predicted Class: Hateful"

Input Sentence: but a girl doesnt run as fast as a boy #LikeAGirl so focus on what she can do. like get pregnant and glorify motherhood.
Actual Class: Hateful | Predicted Class: Hateful

Input Sentence: An idiot converts to Islam. What is your point
Actual Class: Hateful | Predicted Class: Hateful

Input Sentence: You are wrong about that. By definition, if they are really Muslims, they are terrorist.
Actual Class: Hateful | Predicted Class: Hateful

Input Sentence: Less than 1% of Muslims are killed by Israelis., More than 90% are killed by other Muslims. So what does this bigot want
Actual Class: Hateful | Predicted Class: Hateful

Input Sentence: Every time some deluded idiot converts to Islam the muslims make propaganda out of it.
Actual Class: Hateful | Predicted Class: Hateful

Figure 10: Sample of Dataset I Used in BERT Model to Predict the Class which SVM couldn't Predict Correctly

Input Sentence: Can we have your address so that when you die we can all come and pay our respects, and then urinate into your urn before they flush it?
Actual Class: Hateful | Predicted Class: Hateful

Input Sentence: LeBron, no matter who says what, you are an incredible player and teammate. You have played your heart out and we can all see that. I know that you really dont care what he said, but just letting you know, you and the rest of the team make me proud.
Actual Class: Non-Hateful | Predicted Class: Non-Hateful

Input Sentence: Hey nutbag..youre a traitor...Obama has more intelligence in his little finger than your peanut brain.....Go to pakistan and spout your venom...youlll be murdered onn the spot...You psychos love to talk about no freedom-but you are liars....You have the freedom to speak like an idiot....arent you happy now goofball, LMAO
Actual Class: Hateful | Predicted Class: Hateful

Input Sentence: Nnamdi chuwkwu you are nothing but dick sucker, i wish im BOKO.. i will bomb your generation.. .you need to relocate to ZOO
Actual Class: Hateful | Predicted Class: Hateful

Input Sentence: K_K_K or Nazi, I cant tell which disease you suffer from, Could you please blog some more
Actual Class: Hateful | Predicted Class: Hateful

Input Sentence: You are not one of us. Fuck off fag
Actual Class: Hateful | Predicted Class: Hateful

Figure 11: Sample of Dataset II Used in BERT Model to Predict the Class which SVM couldn't Predict Correctly

Input Sentence: They say our state bird is rare...I literally see one at least once a day
Actual Class: Non-Hateful | Predicted Class: Non-Hateful

Input Sentence: How is that not a yellow on Ronaldo for full on body bumping the ref?
Actual Class: Non-Hateful | Predicted Class: Non-Hateful

Input Sentence: My hair is trash. I need to put a masque on it.. But that takes so much energy that I don't have.
Actual Class: Non-Hateful | Predicted Class: Non-Hateful

Input Sentence: He no doubt copied it from someone else. Alert: The mock turtleneck is back.
Actual Class: Non-Hateful | Predicted Class: Non-Hateful

Input Sentence: Biden was a winner last night, Yea right like Charlie Sheen Really was a a winner too.... #winning lol
Actual Class: Non-Hateful | Predicted Class: Non-Hateful

Input Sentence: There's gotta be like 20 white trash motherfuckers living in that house and I don't think any of them have a job
Actual Class: Non-Hateful | Predicted Class: Hateful

Figure 12: Sample of Dataset III Used in BERT Model to Predict the Class which SVM couldn't Predict Correctly

As we can see in the examples that I used in the written thesis, both Dataset I and Dataset II had more incorrect predictions in the minority class, which was the Hateful class as shown in Figure 10, 11. However, BERT performed significantly better. In Dataset III, the minority class was the Non-Hateful class, and SVM often made incorrect predictions for this class as shown in Figure 12. Yet again, BERT outperformed SVM in these cases.

RoBERTa Comparison

I used RoBERTa, another pretrained model, for comparison with BERT. Additionally, RoBERTa is pretrained on a larger and more diverse corpus, including 160GB of text from the Internet, encompassing sources like BookCorpus, English Wikipedia, Common Crawl, and other web data. In contrast, BERT's training data includes a mix of BooksCorpus (800 million words) and English Wikipedia (2.5 billion words).

While BERT was trained for 1 million steps with a batch size of 256, RoBERTa was trained for 500,000 steps with a substantially larger batch size of 8,000. Notably, RoBERTa diverges from BERT by solely employing the MLM objective, omitting the NSP objective used in BERT.

In my thesis, I employed the RoBERTa and the results were consistently close to those obtained with BERT, as detailed in Table 5. Notably, Dataset I produced relatively worse results when using RoBERTa, but Dataset II showed an improvement compared to BERT. These variations can be attributed to several factors, including the quality and representativeness of the training data. For Dataset I, it's possible that the NSP training objective was better suited, leading

to improved results with BERT. Nevertheless, the overarching observation is that pretrained models excel in understanding context and predicting hate speech.

Table 5: F1-score of BERT and RoBERTa Models using the Datasets

	BERT	RoBERTa
DATASET I	89.7%	86.4%
DATASET II	88.2%	88.6%
DATASET III	90.6%	89.1%

CONCLUSION

I developed and evaluated multiple ML and deep learning models for hate speech identification on social media. My approach involved utilizing text from three distinct datasets and deploying a range of models, including BERT, CNN with GloVe, LSTM with GloVe, and six traditional ML algorithms with TF-IDF and n-grams. Additionally, for the traditional ML algorithms, I employed both oversampling and undersampling techniques on the datasets.

The traditional ML algorithms exhibited satisfactory performance, achieving F1-scores of approximately 70%. Notably, Dataset I yielded the best results among them. SVM stood out as the top-performing algorithm, with an F1-score of 75.6%.

Upon applying undersampling to the datasets, I observed improvements in F1-scores across all datasets. Naïve Bayes excelled with an impressive F1-score of 81.3% on Dataset II. The most outstanding results were obtained by oversampling the datasets, with F1-scores surpassing 90% for almost all traditional ML algorithms. Among them, Random Forests achieved the highest F1-score of 98.2% on Dataset III. Notably, the combination of n-grams consistently provided superior results, especially after oversampling, where TF-IDF proved most effective. As a result, I conclude that imbalanced datasets significantly impact the performance of traditional ML algorithms.

In real-world scenarios, where datasets are often imbalanced, deep learning techniques become essential to achieve optimal results. All my deep learning models achieved F1-scores exceeding 80%. Notably, CNN and LSTM outperformed traditional ML algorithms, but BERT consistently delivered the highest F1-scores across all datasets, with a remarkable F1-score of 90.6% on Dataset III and close to 90% on the other datasets.

In the comparative analysis, I observed that BERT and RoBERTa consistently outperformed traditional machine learning algorithms. While SVM emerged as the top-performing traditional model, BERT and RoBERTa exhibited F1 scores that were approximately 20% higher. Moreover, it was intriguing to note that RoBERTa, with its enhanced training strategies, came remarkably close to the performance of BERT, indicating that substantial modifications can refine the robustness of pretrained models.

The results also highlighted the sensitivity of the outcomes to the dataset used. Different datasets presented unique challenges, emphasizing the importance of dataset selection and diversity. The role of NSP training objectives, as seen in BERT, contributed to differences in performance across datasets.

In conclusion, this research underscores the transformative impact of deep learning and pretrained models on hate speech detection. The advancements in deep learning models, coupled with the richness of pre-training data, have elevated the field, enabling better understanding of context and more effective hate speech prediction. The results also indicate that larger datasets with more hate speech samples improve the performance of the deep learning models. While the performance of these models can vary depending on the dataset characteristics, the overall trend indicates a promising future for leveraging deep learning in addressing the critical issue of online hate speech.

REFERENCES

- [1] Fortuna, P., & Nunes, S. (2019). A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys*, 51(4), Article 85, 30 pages.
- [2] Wulczyn, E., Thain, N., & Dixon, L. (2017). Ex Machina: Personal Attacks Seen at Scale. doi:10.1145/3038912.3052591.
- [3] Agrawal, S., & Awekar, A. (2018). Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms. In G. Pasi, B. Piwowarski, L. Azzopardi, & A. Hanbury (Eds.), *Advances in Information Retrieval. ECIR 2018* (Vol. 10772, Lecture Notes in Computer Science, pp. 139-153). Springer. https://doi.org/10.1007/978-3-319-76941-7_11.
- [4] Yin, D., Xue, Z., Hong, L., Kontostathis, A., Davison, B. D., & Edwards, L. (2009). Detection of Harassment on Web 2.0.
- [5] MacAvaney, S., Yao, H.-R., Yang, E., Russell, K., Goharian, N., & Frieder, O. (2019). Hate speech detection: Challenges and solutions. *PLoS ONE*, 14(8), 1–16. <https://doi.org/10.1371/journal.pone.0221152>.
- [6] Kumar, R., Ojha, A. K., Malmasi, S., & Zampieri, M. (2018). Benchmarking Aggression Identification in Social Media. *TRAC@COLING 2018*, 1–11.
- [7] Waseem, Z., & Hovy, D. (2016). Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter, 88-93. <https://doi.org/10.18653/v1/N16-2013>.
- [8] Rezayi, S., Balakrishnan, V., Arabnia, S., & Arabnia, H. R. (2018). Fake News and Cyberbullying in the Modern Era. *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, 7–12. <https://doi.org/10.1109/CSCI46756.2018.00010>.
- [9] Cook, S. (2022, January 29). Cyberbullying facts and statistics for 2018-2022. <https://www.comparitech.com/Internet-providers/cyberbullying-statistics>.
- [10] Korte, L. (2017, May 30). Youth suicide rates are rising. School and the Internet may be to blame. <https://www.usatoday.com/story/news/nationnow/2017/05/30/youth-suicide-rates-rising-school-and-Internet-mayblame/356539001>.
- [11] Rosenblatt, K. (2017, August 1). Cyberbullying tragedy: New Jersey family to sue after 12-year-old daughter's suicide. <https://www.nbcnews.com/news/us-news/new-jersey-family-sue-school-district-after-12-year-old-n788506>.
- [12] Hinduja, S., & Patchin, J. W. (2010). Bullying, cyberbullying, and suicide. *Archives of Suicide Research*, 14(3), 206–221. <https://doi.org/10.1080/13811118.2010.494133>.
- [13] Giumetti, G. W., & Kowalski, R. M. (2022). Cyberbullying via social media and well-being. *Current Opinion in Psychology*, 45. <https://doi.org/10.1016/j.copsyc.2022.101314>.
- [14] Imperium. (2012). Detecting insults in social commentary dataset. <https://www.kaggle.com/c/detecting-insults-in-social-commentary>.

- [15] Davidson, T., Warmesley, D., Macy, M. W., & Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. ICWSM.
- [16] Hang, O. C., & Dahlan, H. M. (2019). Cyberbullying Lexicon for Social Media. 2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS), 1–6. <https://doi.org/10.1109/ICRIIS48246.2019.9073679>.
- [17] Simpson, J., & Saquer, J. (n.d.). Case Study Analyzing the Effectiveness of Machine Learning and Genetic Programming for Detecting Cyberbullying.
- [18] Solitana, N. T., & Cheng, C. K. (2021). Analyses of Hate and Non-Hate Expressions during Election using NLP. 2021 International Conference on Asian Language Processing (IALP), 385–390. <https://doi.org/10.1109/IALP54817.2021.9675186>.
- [19] Chandra, M., Pailla, D., Bhatia, H., Sanchawala, A., Gupta, M., Shrivastava, M., & Kumaraguru, P. (2021). Subverting the Jewtocracy: Online Antisemitism Detection Using Multimodal Deep Learning. In 13th ACM Web Science Conference 2021 (Virtual Event, United Kingdom) (pp. 148–157). <https://doi.org/10.1145/3447535.3462502>.
- [20] Chandra, M., Reddy, M., Sehgal, S., Gupta, S., Buduru, A. B., & Kumaraguru, P. (2021). "A Virus Has No Religion": Analyzing Islamophobia on Twitter During the COVID-19 Outbreak. In Proceedings of the 32nd ACM Conference on Hypertext and Social Media (pp. 67–77).
- [21] ElSherief, M., Kulkarni, V., Nguyen, D., Wang, W. Y., & Belding, E. (2018). Hate lingo: A target-based linguistic analysis of hate speech in social media. In Proceedings of the International AAAI Conference on Web and Social Media, Vol. 12. California, USA.
- [22] Cambria, E., & White, B. (2014). Jumping NLP curves: A review of natural language processing research. IEEE Computational intelligence magazine, 9(2), 48-57.
- [23] Khan, H., & Phillips, J. L. (2021). Language agnostic model: Detecting islamophobic content on social media. In Proceedings of the 2021 ACM Southeast Conference (pp. 229–233). Virtual.
- [24] Schmidt, A., & Wiegand, M. (2017). A survey on hate speech detection using natural language processing. In Proceedings of the fifth international workshop on natural language processing for social media (pp. 1–10). Valencia, Spain.
- [25] Acheampong, F. A., Nunoo-Mensah, H., & Chen, W. (2021). Transformer models for text-based emotion detection: A review of BERT-based approaches. Artificial Intelligence Review, 54, 5789–5829.
- [26] How It Works. Accessed on: February 17, 2023. [Online]. Available: <https://www.liwc.app/help/howitworks>.
- [27] De la Pena Sarracén, G. L., Pons, R. G., Cuza, C. E. M., & Rosso, P. (2018). Hate speech detection using attention-based lstm. EVALITA evaluation of NLP and speech tools for Italian, 12, 235.
- [28] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), 4171–4186.

- [29] Mutanga, R. T., Naicker, N., & Olugbara, O. O. (2020). Hate speech detection in twitter using transformer methods. *International Journal of Advanced Computer Science and Applications*, 11(9).
- [30] i Orts, Ò. G. (2019, June). Multilingual detection of hate speech against immigrants and women in Twitter at SemEval-2019 task 5: Frequency analysis interpolation for hate in speech detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (pp. 460-463).
- [31] Jacobs, C., Rakotonirina, N. C., Chimoto, E. A., Bassett, B. A., & Kamper, H. (2023). Towards hate speech detection in low-resource languages: Comparing ASR to acoustic word embeddings on Wolof and Swahili. *arXiv preprint arXiv:2306.00410*.
- [32] Pennebaker, J. W., Booth, R. J., & Francis, M. E. (2007). *LIWC: Linguistic Inquiry and Word Count*. Erlbaum.
- [33] Sap, M., Card, D., Gabriel, S., Choi, Y., & Smith, N. A. (2019, July). The risk of racial bias in hate speech detection. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 1668-1678).
- [34] Davidson, T., Warmusley, D., Macy, M., & Weber, I. (2017). Automated hate speech detection and the problem of offensive language. *Proceedings of the 11th International AAAI Conference on Web and Social Media (ICWSM)*, 512–515.
- [35] Van Rossum, G., & Drake Jr, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.
- [36] Wiegand, M., Siegel, M., Ruppenhofer, J., & Kuhn, J. (2018). Overview of the GermEval 2018 shared task on the identification of offensive language. *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval)*, 1–17.
- [37] Zampieri, M., Nakov, P., Rosenthal, S., Atanasova, P., Karadzhov, G., Mubarak, H., ... & Çöltekin, Ç. (2020). SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020). *arXiv preprint arXiv:2006.07235*.
- [38] Dixon, L., Li, J., Sorensen, J., Thain, N., & Vasserman, L. (2018, December). Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society* (pp. 67-73).
- [39] Sap, M., Gabriel, S., Qin, L., Jurafsky, D., Smith, N. A., & Choi, Y. (2019). Social bias frames: Reasoning about social and power implications of language. *arXiv preprint arXiv:1911.03891*.
- [40] Jain, S., & Wallace, B. C. (2019). Attention is not explanation. *arXiv preprint arXiv:1902.10186*.
- [41] Yadav, S., Kaushik, A., & McDaid, K. (2023, July). Understanding Interpretability: Explainable AI Approaches for Hate Speech Classifiers. In *World Conference on Explainable Artificial Intelligence* (pp. 47-70). Cham: Springer Nature Switzerland.

- [42] Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V., & Bhamidipati, N. (2015, May). Hate speech detection with comment embeddings. In Proceedings of the 24th international conference on world wide web (pp. 29-30).
- [43] Perifanos, K., & Goutsos, D. (2021). Multimodal hate speech detection in greek social media. *Multimodal Technologies and Interaction*, 5(7), 34.
- [44] Peng, M., Zhang, Q., Jiang, Y. G., & Huang, X. J. (2018, July). Cross-domain sentiment classification with target domain specific information. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 2505-2513).
- [45] Xue, J., Li, Y., Li, Z., Cui, Y., Zhang, S., & Wang, S. (2023). A Cross-Domain Generative Data Augmentation Framework for Aspect-Based Sentiment Analysis. *Electronics*, 12(13), 2949.
- [46] Kim, Y. (2014). Convolutional neural networks for sentence classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1746–1751.
- [47] Zhang, W., Yoshida, T., & Tang, X. (2008). Text classification based on multi-word with support vector machine. *Knowledge-Based Systems*, 21(8), 879-886.
- [48] Raschka, S. (2014). Naive bayes and text classification i-introduction and theory. arXiv preprint arXiv:1410.5329.
- [49] Aseervatham, Sujeevan & Gaussier, Eric & Antoniadis, Anestis & Burlet, Michel & Denneulin, Yves. (2012). Logistic Regression and Text Classification. 10.1002/9781118562796.ch3.
- [50] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. <http://www.deeplearningbook.org>.
- [51] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10), 2451-2471.
- [52] Wang, S., & Jiang, J. (2015). Learning natural language inference with LSTM. *arXiv preprint arXiv:1512.08849*.
- [53] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. O'Reilly Media.
- [54] Hugging Face. (2023). Transformers Documentation. URL: https://huggingface.co/docs/transformers/model_doc/bert.
- [55] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.
- [56] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. URL: <https://nlp.stanford.edu/projects/glove/>.