Hate Speech and Toxic Comment Detection

Anjali Haryani
haryani7.a@northeastern.edu
Seattle, Washington, USA

Abstract

In recent years, lower data costs and better internet access have led to a big increase in the number of people using the internet. High-speed connections have reached even remote areas, enabling more people to come online and express their ideas and opinions. However, this has also contributed to an increase in hate speech and online bullying, which discourages some individuals from sharing their thoughts and spreads negativity, depression, and racial or communal hostility. In extreme cases, it can even incite violence. As a result, it is crucial for social media platforms to ensure their environments are free from toxic content, fostering the free and open exchange of ideas.

ACM Reference Format:
Anjali Haryani. 2024. Hate Speech and Toxic Comment Detection. In . ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Social networking platforms have become a powerful medium for global communication, enabling users to share ideas and opinions freely. However, this freedom has also led to the spread of hate speech, toxic comments, and online harassment. These negative behaviors not only discourage open expression but also contribute to a harmful environment by promoting racial harassment, bullying, and mental health issues like depression.

Addressing the issue of hate speech and toxic content is a major challenge for social media platforms, as manual moderation is not scalable due to the sheer volume of user-generated content. This creates a need for automated systems that can efficiently detect and classify toxic comments in real time. The problem lies in accurately identifying harmful content, as it can take many forms and may involve complex language, slang, or context-specific meaning.

The aim is to develop machine learning models capable of detecting hate speech and toxic comments across various categories, helping social media platforms reduce the spread of harmful content and foster a safer, more inclusive online environment. This is critical to ensuring that individuals can express themselves freely without fear of abuse or harassment.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

1.1 Research Paper 1: Hate me, hate me not: Hate speech detection on Facebook

In the work by Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi (2017), the authors focused on stopping the spread of hate campaigns on Facebook. They created a system to detect hate speech in Italian comments using two machine learning methods: Support Vector Machines (SVM) and Long Short-Term Memory (LSTM) neural networks.

To solve the problem, they used language features like sentence structure, word meanings, and word relationships. The system could identify different levels of hate, such as no hate, weak hate, and strong hate. They collected data from public Facebook pages, and the comments were labeled by human reviewers.

In conclusion, the system worked well for identifying whether comments had hate speech or not, but it struggled to tell the difference between weak and strong hate due to disagreements in labeling and the small number of strong hate examples.

1.2 Research Paper 2: Exploring Hate Speech Detection in Multimodal Publications

In the work by Raul Gomez, Jaume Gibert, Lluis Gomez, and Dimosthenis Karatzas (2020), the authors aimed to detect hate speech in social media posts that include both text and images. They created a new dataset called MMHS150K, which has 150,000 tweets, each with both text and an image. The goal was to find out if using both text and images together (multimodal analysis) would make it easier to detect hate speech compared to just using text alone (unimodal analysis).

To solve the problem, they tested different models that combined the information from the text and images in the tweets. These models were compared to traditional models that only used text to detect hate speech. They used CNNs to analyze images and LSTMs to analyze text. They also used text recognition to pull out words from images, like memes or screenshots.

In conclusion, while images did add helpful context, the models that used both text and images did not perform better than the models that used only text. This was because of the difficulty in understanding how the text and images work together and the small amount of content that needed both for accurate detection.

1.3 Research Paper 3: A Lexicon-based Approach for Hate Speech Detection

The paper by Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long (2015), called "A Lexicon-Based Approach for Hate Speech Detection," is about creating a system to find hate speech in online posts, especially related to race, nationality, and religion. The authors wanted to build a tool that can recognize hateful messages by analyzing the tone and meaning of words in

forums, blogs, and comment sections. Their goal was to detect harmful speech aimed at certain groups.

To solve this problem, they used a rule-based approach. First, they found sentences that showed opinions or emotions, then they looked for specific words and phrases linked to hate speech. They built a list (lexicon) of hate-related words and patterns using WordNet and other methods. The system they built classified content into three categories: No hate, Weakly hate, or Strongly hate, and they tested it on 500 labeled paragraphs.

In conclusion, their approach worked well for identifying hate speech, especially when combining different types of features. However, they suggest that the system could be improved by using more data and advanced machine learning techniques to increase accuracy across different types of hate speech.

1.4 Research Paper 4: Automated Hate Speech Detection and the Problem of Offensive Language

The paper "Automated Hate Speech Detection and the Problem of Offensive Language" by Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber (2017) focuses on improving hate speech detection on social media, particularly Twitter. The authors address the challenge of separating hate speech from offensive language, as previous methods often incorrectly categorize offensive content as hate speech due to the presence of offensive terms.

To address this issue, they used a crowd-sourced hate speech lexicon to collect tweets containing potentially hateful keywords. A sample of these tweets was manually labeled into three categories: hate speech, offensive language, or neither. The team then trained a multi-class classifier using linguistic features like word n-grams, syntactic structure, and sentiment analysis to differentiate between the categories, with a focus on distinguishing subtle differences in language use.

The classifier performed well overall, achieving high precision and recall. However, it struggled with edge cases, where offensive language without explicit hate terms was often misclassified, and certain types of hate speech (e.g., less explicit or targeting less frequent groups) were harder to detect accurately. The authors suggest that future work should better account for context and less common forms of hate speech.

1.5 Research Paper 5: Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter

The paper "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter" by Zeerak Waseem and Dirk Hovy (2016) explores methods for identifying hate speech on social media, specifically Twitter. The authors aim to differentiate between hateful content and non-hateful content, focusing on sexist and racist language.

To address the problem, they created a dataset of over 16,000 annotated tweets, classifying them as sexist, racist, or neither. They used various features like character n-grams and extra-linguistic information (such as gender and location) to train a classifier for hate speech detection. They also developed a dictionary of common words associated with hate speech to improve the accuracy of their model.

Their results showed that character n-grams and gender information provided the best performance in detecting hate speech. However, they found that adding features like location and tweet length reduced the model's accuracy, highlighting the need to focus on the most relevant features for effective hate speech detection.

1.6 Research Paper 6: Detection of hate: speech tweets based convolutional neural network and machine learning algorithms

The paper, written by Hameda A. Sennary, Ghada Abozaid, Ashraf Hemeida, Alexey Mikhaylov, and Tamara Broderick, discusses using machine learning (ML) and deep learning (DL) techniques to detect hate speech in tweets. Specifically, the authors employed a TF-IDF-based feature engineering approach on eleven classifiers, including logistic regression, support vector machines, and convolutional neural networks, applied to three datasets.

They processed tweets using text preprocessing techniques like tokenization and stemming, then trained and evaluated classifiers with features extracted via TF-IDF. Among the datasets, their approach achieved over 99% accuracy on a merged dataset, showcasing the effectiveness of combining traditional ML and DL techniques.

While the proposed methods performed well on the datasets used, the study notes challenges in generalizing results to diverse hate speech forms, indicating a need for more robust cross-domain evaluations.

1.7 Research Paper 7: An approach to automatic classification of hate speech in sports domain on social media

In the work by Staša Vujičić Stanković and Miljana Mladenović, titled "An approach to automatic classification of hate speech in sports domain on social media" (Journal of Big Data, 2023), the authors address hate speech targeting athletes on social media, specifically in Serbian, a less-studied language for this problem.

To tackle this, they created domain-specific and domain-agnostic datasets and developed a hate speech lexicon in Serbian. They trained Bidirectional Long Short-Term Memory (Bi-LSTM) models using these datasets and evaluated their ability to classify hate speech in sports-related contexts.

The study demonstrated high precision but low recall in detecting hate speech, highlighting the need for further refinements in model training, dataset balance, and embedding techniques for better generalization across domains.

1.8 Research Paper 8: Is hate speech detection the solution the world wants?

In the paper titled "Is hate speech detection the solution the world wants?" by Sara Parker and Derek Ruths (PNAS, 2023), the authors critically examine the focus of the computer science (CS) community on automated hate speech detection and explore how

this emphasis aligns—or conflicts—with the perspectives of other stakeholders like governments, nonprofits, and platforms.

The authors analyzed discourse from these stakeholders, revealing a disconnect: while CS research predominantly emphasizes developing detection methods, other stakeholders prioritize holistic, collaborative solutions to mitigate hate speech's impact. The paper advocates for interdisciplinary collaboration to integrate technical tools with broader societal strategies effectively.

The study concludes that the CS community needs to shift from method-based thinking to solution-driven approaches, engaging other stakeholders to address online hate speech comprehensively. Without collaboration, efforts to mitigate hate speech risk remaining fragmented and ineffective.

1.9 Research Paper 9: A Comprehensive Review on Automatic Hate Speech Detection in the Age of the Transformer

In the paper titled "A Comprehensive Review on Automatic Hate Speech Detection in the Age of the Transformer" by Gil Ramos et al. (Social Network Analysis and Mining, 2024), the authors provide a detailed review of hate speech detection methodologies, highlighting the transition from traditional machine learning (ML) and deep learning (DL) approaches to the adoption of Transformer-based models.

They analyzed over 100 studies to evaluate the effectiveness of various techniques, such as Support Vector Machines (SVM), Long Short-Term Memory (LSTM) networks, and Transformer models like BERT. The review also covers datasets, languages, and challenges like algorithmic bias and resource limitations for low-resource languages. The authors emphasize the superior performance of Transformer models, while noting the computational costs and advocating for context-specific solutions.

The study concludes that Transformer-based models significantly advance hate speech detection but require balanced approaches integrating societal and technical solutions for broader applicability.

1.10 Research Paper 10: Exploring Automatic Hate Speech Detection on Social Media: A Focus on Content-Based Analysis

In the paper titled "Exploring Automatic Hate Speech Detection on Social Media: A Focus on Content-Based Analysis" by Francimaria R. S. Nascimento, George D. C. Cavalcanti, and Márjory Da Costa-Abreu (SAGE Open, 2023), the authors provide a comprehensive survey of hate speech detection methodologies, focusing on textual data in social media and exploring definitions, datasets, and algorithms.

The authors reviewed various approaches, including feature extraction methods such as bag-of-words, n-grams, text embeddings, and sentiment analysis, alongside machine learning models like SVM and neural networks. They identified challenges such as defining hate speech consistently, class imbalances in datasets, and the need for improved handling of low-resource languages.

The study concludes by highlighting the importance of integrating advanced detection techniques with interdisciplinary efforts to address the societal impact of hate speech effectively.

1.11 Research Paper 11: A Survey on Hate Speech Detection and Sentiment Analysis Using Machine Learning and Deep Learning Models

In the paper titled "A Survey on Hate Speech Detection and Sentiment Analysis Using Machine Learning and Deep Learning Models" by Malliga Subramanian et al. (Alexandria Engineering Journal, 2023), the authors review advancements in detecting hate speech and analyzing sentiment using machine learning (ML) and deep learning (DL) techniques.

They explore various datasets, traditional ML approaches like Support Vector Machines (SVM) and Decision Trees, and advanced DL models such as CNNs, RNNs, and Transformer architectures like BERT. The paper highlights challenges in data imbalance, language diversity, and context-dependent hate speech detection.

The study concludes by advocating for interdisciplinary research and improved methodologies to develop accurate and inclusive solutions for hate speech detection across languages and contexts.

1.12 Research Paper 12: Hate Speech Detection with ADHAR: A Multi-Dialectal Hate Speech Corpus in Arabic

In the paper titled "Hate Speech Detection with ADHAR: A Multi-Dialectal Hate Speech Corpus in Arabic" by Anis Charfi, Mabrouka Besghaier, Raghda Akasheh, Andria Atalla, and Wajdi Zaghouani (Frontiers in Artificial Intelligence, 2024), the authors introduce a novel hate speech dataset tailored for Arabic, addressing dialect diversity and imbalanced categories in existing resources.

They developed the ADHAR corpus, which includes 4,240 tweets covering Modern Standard Arabic (MSA) and four Arabic dialects (Egyptian, Levantine, Gulf, and Maghrebi). The dataset spans four hate speech categories—nationality, religion, ethnicity, and race—and employs balanced classes (hate and non-hate) to mitigate biases. They also demonstrated the dataset's utility in building effective hate speech classifiers, achieving F1-scores of up to 94 using the AraBERT model.

The study concludes that ADHAR significantly advances hate speech detection in Arabic by offering a balanced, multi-dialectal resource, while also highlighting the need for future work to expand dialectal coverage and adapt to evolving hate speech patterns.

1.13 Research Paper 13: Hate Speech Detection by Using Rationales for Judging Sarcasm

In the paper titled "Hate Speech Detection by Using Rationales for Judging Sarcasm" by Maliha Binte Mamun et al. (Applied Sciences, 2024), the authors address the challenge of detecting hate speech that is camouflaged as sarcasm, a nuanced issue in online communication.

The authors created a new dataset by annotating existing data with sarcasm labels and rationales, combining these with conventional hate/offensive rationales. They incorporated this into a BERT-based model with a novel preprocessing step that uses attention vectors. This approach improved classification accuracy and explainability metrics, such as plausibility and faithfulness, compared to traditional methods.

The study concludes that integrating sarcasm-based rationales into hate speech detection models enhances both performance and transparency, paving the way for more effective moderation of online content.

2 Overview

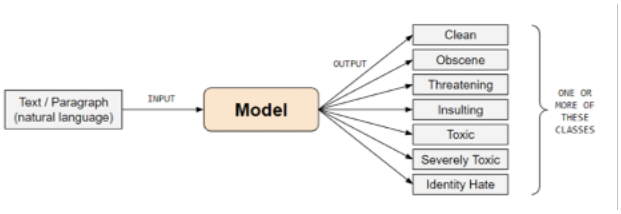

Figure 1: Overview

Social networking sites are now widely used for communication around the world. They allow people to freely share their opinions and ideas. However, some people use these platforms to spread harmful language, like hate speech and toxic comments. This has made it challenging for social networks to control such negative content. Detecting hate speech or toxic comments can be set up as a classification problem where a post could belong to one or more harmful categories. Many companies are now creating models to automatically flag this kind of content.

In this project, we analyze posts from different social media sites and build models to classify these posts into six categories: toxic, severe toxic, threat, identity hate, obscene, and insult. Using real social media comments is beneficial because they accurately represent the content often seen online. Since these comments contain noise, or unwanted data, we first cleaned the data by removing any noise and outliers. We will start by testing classical models, like support vector machines and logistic regression, on this task. Then, we will explore newer methods like tree-based ensemble models. Additionally, we will test more advanced models, such as recurrent neural networks. Finally, we used pretrained word embeddings—Word2Vec, FastText, and GloVe—to improve classification. We compared all models based on their accuracy, precision, recall, F1 scores which will help us evaluate their performance.

3 Problem Description

We have a dataset of comments from various social media platforms like Facebook, Twitter, and Instagram. The goal is to detect and flag comments that show hate or vulgarity. Specifically, the model categorizes each toxic comment into one of six labels: toxic, severely toxic, threatening, insulting, obscene, or identity hate. If a comment is not toxic, it is labeled as "clean," making it an extra category.

This problem involves natural language processing (NLP) since we are dealing with text comments. The data is often unstructured, noisy, and messy because it comes from the internet and social media. Therefore, extensive text cleaning and processing are necessary to make the data usable.

Figure 2: Multiclass and Multilable Classification

Our problem is both a multiclass and a multilabel classification:

- **Multiclass classification**: Each example is placed in one category out of several options.
- **Multilabel classification**: An example can belong to multiple categories at once.

In our case, a comment may fall into multiple categories, like being both toxic and obscene, making it a multilabel problem. Reducing the task to a binary classification (toxic or not toxic) would limit the model's application. For instance, some platforms may allow mild insults but not identity hate. Therefore, we explore other methods to handle these multiple labels in the next sections.

4 Dataset

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0

Figure 3: Dataset

The dataset used in this study was provided by Conversation AI (Jigsaw and Google) to examine negative online behaviors. It contains Wikipedia comments labeled across six types of toxicity. The dataset includes 143,346 non-toxic comments and 16,225 toxic comments, with a total of approximately 160,000 comments. A comment is considered toxic if it falls into at least one toxicity category.

Human raters on Kaggle manually assigned labels to these comments, with each comment labeled as either 0 or 1 for each category—0 meaning the comment does not belong to that toxicity class, and 1 meaning it does. This dataset was prepared using crowdsourced annotations, meaning that different users across the internet labeled the examples, adding an element of subjectivity. For instance, one person might find a comment offensive while another may not, which introduces judgment bias.

Further exploration of the dataset shows that other factors can affect perceptions of toxicity. For example, a comment's appearance

might change based on screen width, with some arrangements appearing as offensive symbols to certain annotators, while others see it as spam or clean. Additionally, some lengthy, detailed comments expressing opinions were sometimes marked as toxic without clear cause.

Due to these factors, achieving perfect classification is challenging, as some labels might be subjective or even incorrect.

5 Data Cleaning and Visualization

In this project, **data cleaning** is an important step to make sure our model works well. First, we checked the dataset for missing values and found that there were none.

```
4 missing_values =train.isnull().sum()
5 print('No. of null values =', missing_values)
```

```
No. of null values = id              0
comment_text     0
toxic            0
severe_toxic     0
obscene          0
threat           0
insult           0
identity_hate    0
dtype: int64
```

Figure 4: Missing Values

Then, we added a new column called 'clean' to mark comments that are non-toxic. This helps us easily separate comments without any toxicity, making classification more accurate.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate	clean
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	1
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	1
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	1
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	1
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	1

Figure 5: New Column

Good data cleaning is necessary to handle messy data, especially in online comments. By preparing the data carefully, we improve the quality and reliability of our model's results.

Data Visualization is an important part of data mining. It helps us see patterns and features in the dataset, which can guide us in choosing the right machine learning methods.

We started by creating a pie chart to show the different types of hate tags in the dataset. This chart revealed a big class imbalance. The "toxic" tag was the most common, while the "threat" tag had the fewest labels. Next, we added a "clean" tag to mark non-toxic comments and plotted a bar graph. This showed that a large portion of the comments were clean.

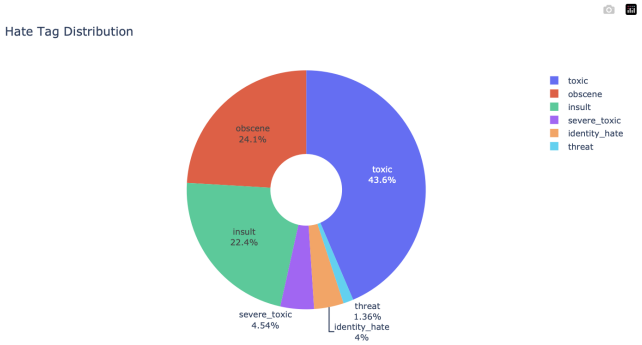

Figure 6: Pie Chart: Hate Tag Distribution

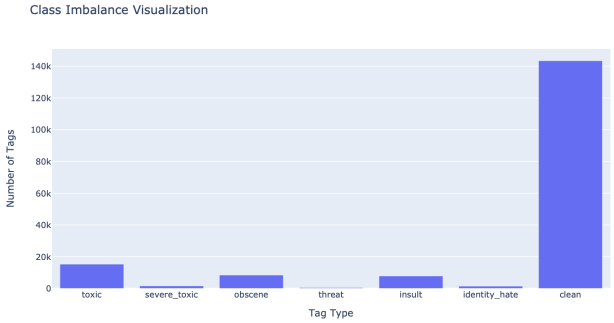

Figure 7: Bar Chart: Class Imbalance

Using these visualizations gives us a clear picture of the dataset and helps us understand its structure.

We visualized the length of comments in the dataset. From the visuals, we can see that a large number of comments have very few sentences, words, or letters. We plotted histograms to show the distribution of letter counts, sentence counts, word counts, and unique word counts in each comment.

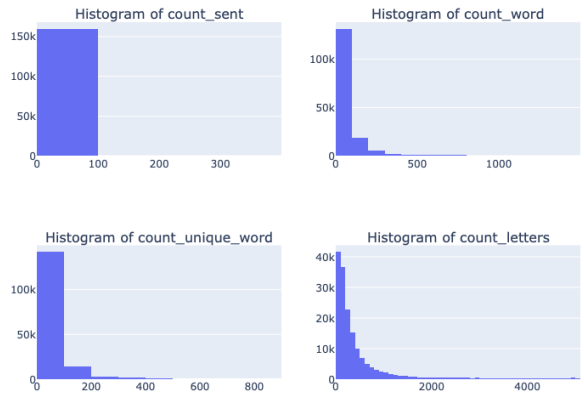

Figure 8: Length of Comments

H

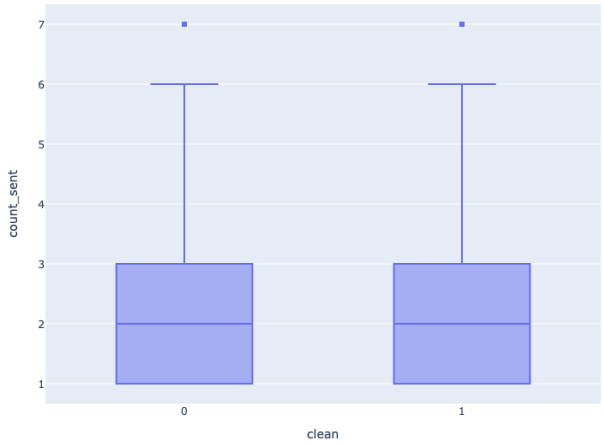

Figure 9: Box Plot: Sentence Count and Comment Toxicity

These visualizations help us understand the structure of the comments and highlight patterns in their length.

We used box plots to explore the correlation between comment length and toxicity. First, we plotted the relationship between sentence count and comment toxicity, finding that there is little correlation between the two.

Then, we examined the correlation between word count and toxicity. Here, we observed a slight trend: comments with fewer words tend to have higher toxicity.

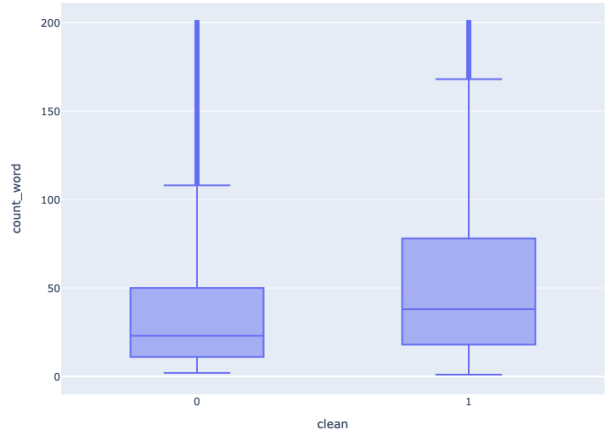

Figure 10: Box Plot: Word Count and Comment Toxicity

We also created a visualization for the multi-class classification. The bar plot reveals that many comments have only a single tag. However, there are some comments that contain five or six hate tags.

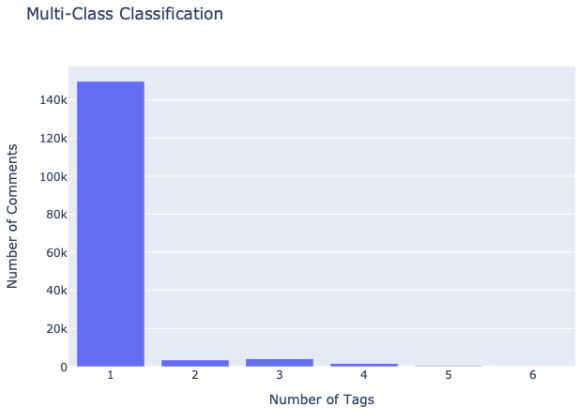

Figure 11: Multi-Class Bar Plot

We plotted a heatmap to show the correlation between the tags. From the heatmap, we observed a negative correlation between clean comments and toxic comments. Additionally, there is a strong correlation between certain tags, such as toxic-obscene and toxic-insult.

Figure 12: Correlation Between Tags

6 Data Preprocessing

In data preprocessing, there are two main steps: removing stop words and stemming.

Removing stop words involves eliminating common words that do not contribute much meaning to the text, such as "the," "is," and "and." This helps focus on the more significant words in the comments.

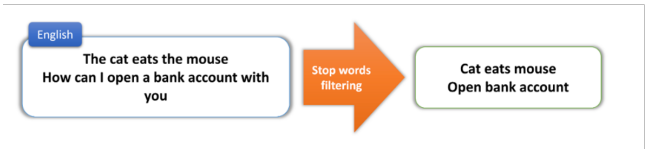

Figure 13: Stop Words

Stemming is the process of reducing words to their root form. For example, "running," "runner," and "ran" would all be reduced to the root word "run." This helps in treating different forms of a word as the same, making the analysis more effective.

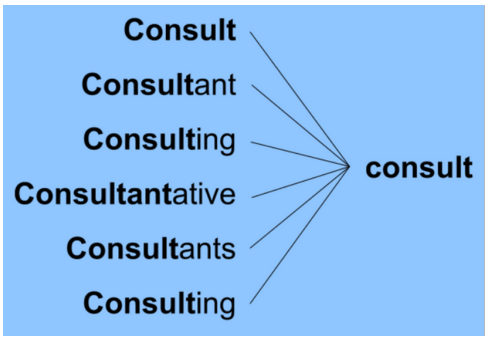

Figure 14: Stemming

Additionally, we converted all letters to lowercase and removed non-letter characters to further clean the data and ensure consistency in our preprocessing steps.

```
Cleaned Comment: ye mother child case michael jackson studi motiv reason judg upon cha
ract harshli wacko jacko tell ignor incrimin go continu refut bullshit jayjg keep thro
w jun utc

 Original Comment Text: Yes, because the mother of the child in the case against Micha
el Jackson was studied in here motives and reasonings and judged upon her character ju
st as harshly as Wacko Jacko himself.  Don't tell me to ignore it and incriminate myse
lf.  I am going to continue refuting the bullshit that Jayjg keeps throwing at me.   1
8:01, 16 Jun 2005 (UTC)
```

Figure 15: Cleaned Comment

7 Methodology: Initial Experimentation

The task at hand involves a multiclass classification of text data. The preprocessed text data remains in its natural language form, which is not directly interpretable by machine learning algorithms. Therefore, it is essential to convert this text into a machine-readable format through a process known as vectorization or feature extraction. This involves tokenizing the text into words, assigning numerical values to each token, and then transforming the entire text into a feature vector. The main techniques used for text feature extraction are as follows:

Feature Extraction

- **Tokenizing**: This involves breaking down the text into smaller units, typically words, which are then assigned an integer or floating-point value.
- **Counting**: The frequency of each token in a document is counted to determine the occurrence of words.
- **Normalizing**: This step penalizes the words that appear frequently across most documents, ensuring that they do not dominate the model.

Two commonly used vectorization methods are:

- **Count Vectorizer**: This method counts the frequency of each word across all documents, creating a feature matrix. However, it does not account for the importance of words across different documents, which may lead to less meaningful features.
- **TF-IDF Vectorizer**: This method not only counts word frequency but also normalizes it by considering the document frequency of each term. Words that appear frequently across many documents are penalized, making it more effective for capturing important terms in the text.

Word Embeddings

Word embeddings provide a more sophisticated way of representing text data by encoding words as dense vectors in a continuous vector space. Unlike traditional methods, embeddings capture semantic relationships between words, making them a powerful feature representation for natural language processing tasks.

- **Word2Vec**: A neural network-based approach that learns word representations by predicting the context words of a given word (or vice versa) using models like Skip-Gram or Continuous Bag of Words (CBOW).
- **GloVe (Global Vectors for Word Representation)**: GloVe is an unsupervised learning algorithm for obtaining vector representations of words by combining the strengths of both matrix factorization and local context-based approaches. It constructs a word-word co-occurrence matrix and derives embeddings such that their dot product approximates the logarithm of the probability of their co-occurrence. This method ensures that the embeddings capture both semantic and syntactic relationships.

In our experiments, GloVe embeddings pre-trained on large text corpora were used to initialize word vectors for the models. This initialization provided a meaningful representation of words, which improved the learning process and helped the models generalize better.

Deep Learning Models

For the classification task, we experimented with three different deep learning models, along with their variants incorporating GloVe embeddings, to handle the text data and perform multiclass classification:

- **LSTM Model**: A simple Long Short-Term Memory (LSTM) model was utilized to capture sequential dependencies in the text. LSTM networks are capable of learning long-term dependencies, making them effective for processing textual data.
- **LSTM with GloVe**: In this variant, GloVe embeddings were used to initialize the embedding layer. This allowed the LSTM to leverage pre-trained word representations, leading to faster convergence and improved performance.
- **CNN + LSTM Model**: A hybrid model combining Convolutional Neural Networks (CNN) and LSTM was applied. CNNs are particularly good at extracting local features from the text (such as phrases or patterns), while LSTMs help capture the temporal or sequential dependencies in the text. This combination allows the model to capture both local and sequential information, leading to better performance on complex tasks.

- **CNN + LSTM with GloVe**: Similar to the LSTM variant, the embedding layer was initialized with GloVe embeddings. This provided the CNN with richer input features, enhancing the feature extraction process.
- **Bi-LSTM Model**: A Bidirectional LSTM (Bi-LSTM) was used to process the text in both forward and backward directions. This enables the model to capture context from both past and future words in the sentence, providing a more comprehensive understanding of the text.
- **Bi-LSTM with GloVe**: GloVe embeddings were incorporated into this model to enhance the context representation. The pre-trained embeddings allowed the Bi-LSTM to start with a solid semantic foundation, improving its ability to generalize across diverse text data.

The models were trained using the binary cross-entropy loss function, which is suitable for multi-label classification tasks. We evaluated the performance of the models using several metrics, including accuracy, ROC-AUC score, and classification report. These metrics helped us understand how well the models were distinguishing between the different classes and identifying misclassifications. The results of these experiments were compared to determine the most effective approach for the given classification task.

Results

LSTM

The LSTM model achieved the following results during training and validation:

- **Precision:** 0.76 (average across all classes)
- **Recall:** 0.69 (average across all classes)
- **F1-Score:** 0.72 (average across all classes)
- **ROC-AUC (Validation):** 0.9807
- **Training Accuracy:** 0.9293
- **Validation Accuracy:** 0.9183

Classification Report (Validation Set):

	precision	recall	f1-score	support
toxic	0.85	0.74	0.80	1568
severe_toxic	0.63	0.25	0.35	155
obscene	0.85	0.80	0.83	872
threat	0.00	0.00	0.00	50
insult	0.70	0.70	0.70	804
identity_hate	0.00	0.00	0.00	146
micro avg	0.81	0.69	0.74	3595
macro avg	0.51	0.41	0.45	3595
weighted avg	0.76	0.69	0.72	3595
samples avg	0.07	0.06	0.06	3595

Figure 16: Classification Report LSTM

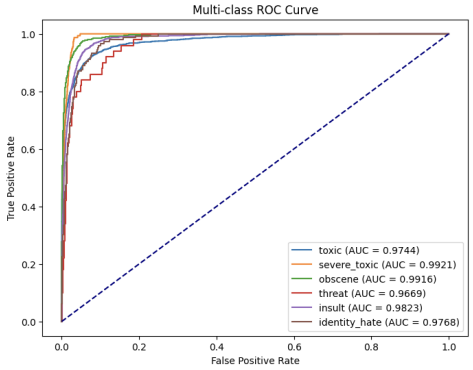

Figure 17: AUC-ROC Curve LSTM

The LSTM model showed strong overall performance with a high ROC-AUC score of 0.9807, indicating excellent ability to distinguish between classes. Its precision and recall are well-balanced, with slightly higher recall, demonstrating its ability to capture most positive cases effectively.

LSTM with GloVe Embedding

The LSTM + GloVe model achieved the following results:

- **Precision:** 0.81 (average across all classes)
- **Recall:** 0.61 (average across all classes)
- **F1-Score:** 0.69 (average across all classes)
- **ROC-AUC (Validation):** 0.9741
- **Training Accuracy:** 0.9201
- **Validation Accuracy:** 0.9162

Classification Report (Validation Set):

	precision	recall	f1-score	support
toxic	0.88	0.64	0.74	1568
severe_toxic	0.80	0.18	0.29	155
obscene	0.86	0.71	0.78	872
threat	0.00	0.00	0.00	50
insult	0.72	0.62	0.66	804
identity_hate	0.68	0.34	0.45	146
micro avg	0.82	0.61	0.70	3595
macro avg	0.66	0.41	0.49	3595
weighted avg	0.81	0.61	0.69	3595
samples avg	0.06	0.06	0.05	3595

Figure 18: Classification Report LSTM with Glove Embedding

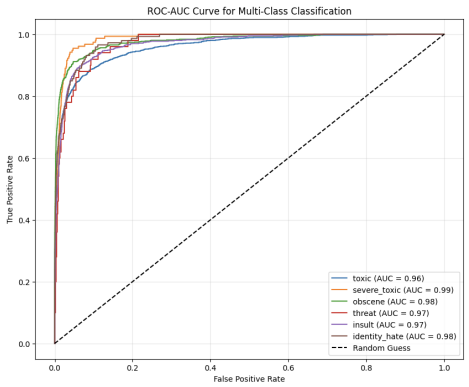

Figure 19: AUC-ROC Curve LSTM with Glove Embedding

This model benefits from pre-trained GloVe embeddings, which provide semantic context. However, its recall and F1-score are lower than those of the basic LSTM, suggesting that the embeddings may not generalize well for this dataset.

Bi-LSTM

The Bi-LSTM model achieved the following results:

- **Precision:** 0.78 (average across all classes)
- **Recall:** 0.67 (average across all classes)
- **F1-Score:** 0.71 (average across all classes)
- **ROC-AUC (Validation):** 0.9802
- **Training Accuracy:** 0.9294
- **Validation Accuracy:** 0.9174

Classification Report (Validation Set):

	precision	recall	f1-score	support
toxic	0.84	0.74	0.79	1568
severe_toxic	0.84	0.14	0.23	155
obscene	0.86	0.81	0.83	872
threat	0.00	0.00	0.00	50
insult	0.74	0.66	0.70	804
identity_hate	0.00	0.00	0.00	146
micro avg	0.82	0.67	0.74	3595
macro avg	0.55	0.39	0.43	3595
weighted avg	0.78	0.67	0.71	3595
samples avg	0.07	0.06	0.06	3595

Figure 20: Classification Report Bi-LSTM

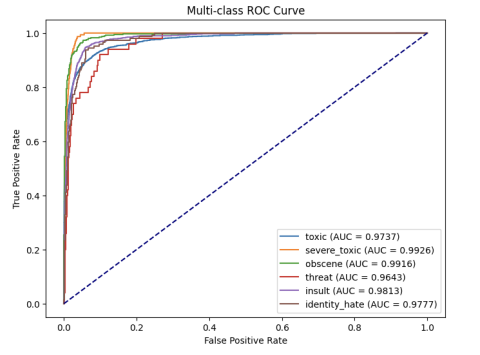

Figure 21: AUC-ROC Curve Bi-LSTM

Bi-LSTM leverages bidirectional processing for better context understanding. It performs comparably to the basic LSTM, though with slightly lower recall.

Bi-LSTM with GloVe Embedding

The Bi-LSTM + GloVe model achieved the following results:

- **Precision:** 0.75 (average across all classes)
- **Recall:** 0.69 (average across all classes)
- **F1-Score:** 0.71 (average across all classes)
- **ROC-AUC (Validation):** 0.9759
- **Training Accuracy:** 0.9194
- **Validation Accuracy:** 0.9130

Classification Report (Validation Set):

	precision	recall	f1-score	support
toxic	0.81	0.72	0.76	1568
severe_toxic	0.62	0.20	0.30	155
obscene	0.79	0.80	0.79	872
threat	0.00	0.00	0.00	50
insult	0.67	0.69	0.68	804
identity_hate	0.64	0.40	0.49	146
micro avg	0.76	0.69	0.72	3595
macro avg	0.59	0.47	0.51	3595
weighted avg	0.75	0.69	0.71	3595
samples avg	0.06	0.06	0.06	3595

Figure 22: Classification Report Bi-LSTM with Glove Embedding

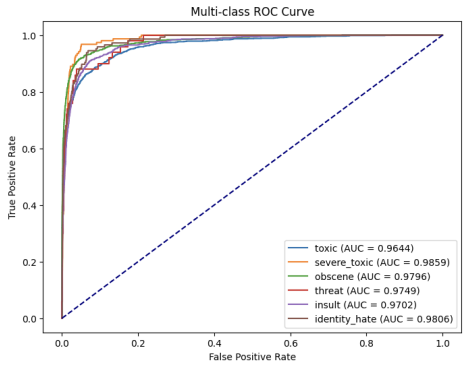

Figure 23: AUC-ROC Curve Bi-LSTM with Glove Embedding

While this model captures bidirectional dependencies and semantic context, its performance is slightly lower than the basic Bi-LSTM, likely due to poor embedding alignment.

CNN + LSTM

The CNN + LSTM model achieved the following results:

- **Precision:** 0.77 (average across all classes)
- **Recall:** 0.65 (average across all classes)
- **F1-Score:** 0.69 (average across all classes)
- **ROC-AUC (Validation):** 0.9774
- **Training Accuracy:** 0.9314
- **Validation Accuracy:** 0.9183

CNN + LSTM – Classification Report (Validation Set):

	precision	recall	f1-score	support
toxic	0.88	0.61	0.72	1568
severe_toxic	0.71	0.03	0.06	155
obscene	0.88	0.65	0.75	872
threat	0.00	0.00	0.00	50
insult	0.74	0.53	0.62	804
identity_hate	0.00	0.00	0.00	146
micro avg	0.84	0.54	0.66	3595
macro avg	0.53	0.30	0.36	3595
weighted avg	0.79	0.54	0.64	3595
samples avg	0.06	0.05	0.05	3595

Figure 26: Classification report CNN + LSTM with Glove Embedding

CNN + LSTM – Classification Report (Validation Set):

	precision	recall	f1-score	support
toxic	0.84	0.76	0.79	1568
severe_toxic	0.54	0.05	0.08	155
obscene	0.86	0.75	0.81	872
threat	0.00	0.00	0.00	50
insult	0.75	0.60	0.67	804
identity_hate	0.00	0.00	0.00	146
micro avg	0.82	0.65	0.73	3595
macro avg	0.50	0.36	0.39	3595
weighted avg	0.77	0.65	0.69	3595
samples avg	0.07	0.06	0.06	3595

Figure 24: Classification Report CNN + LSTM

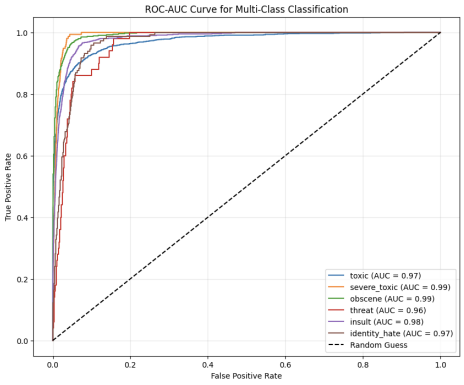

Figure 25: AUC-ROC Curve CNN + LSTM

CNN + LSTM combines spatial feature extraction and sequential modeling, yielding performance similar to the basic LSTM, though with slightly lower recall and F1-score.

CNN + LSTM with GloVe Embedding

The CNN + LSTM + GloVe model achieved the following results:

- **Precision:** 0.72 (average across all classes)
- **Recall:** 0.63 (average across all classes)
- **F1-Score:** 0.66 (average across all classes)
- **ROC-AUC (Validation):** 0.9759
- **Training Accuracy:** 0.9177
- **Validation Accuracy:** 0.9084

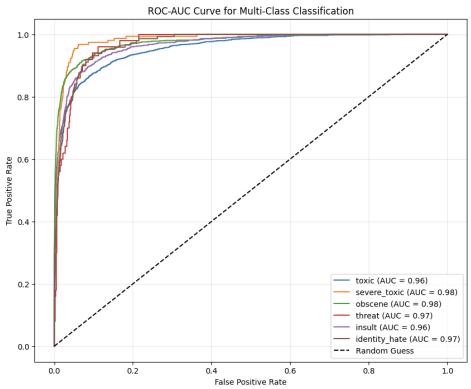

Figure 27: AUC-ROC Curve CNN + LSTM with Glove Embedding

This model incorporates semantic embeddings but shows the lowest overall performance, indicating suboptimal generalization.

Performance Comparison Table

Model	Precision (Val)	Recall (Val)	F1-Score (Val)
LSTM	0.76	0.69	0.72
LSTM + GloVe	0.81	0.61	0.69
Bi-LSTM	0.78	0.67	0.71
Bi-LSTM + GloVe	0.75	0.69	0.71
CNN + LSTM	0.77	0.65	0.69
CNN + LSTM + GloVe	0.72	0.63	0.66

Table 1: Model performance metrics: Precision, Recall, and F1-Score.

Model	ROC-AUC (Train)	ROC-AUC (Val)	Accuracy (Train)	Accuracy (Val)
LSTM	0.9883	0.9807	0.9293	0.9183
LSTM + GloVe	0.9771	0.9741	0.9201	0.9162
Bi-LSTM	0.9877	0.9802	0.9294	0.9174
Bi-LSTM + GloVe	0.9809	0.9759	0.9194	0.9130
CNN + LSTM	0.9851	0.9774	0.9314	0.9183
CNN + LSTM + GloVe	0.9809	0.9759	0.9177	0.9084

Table 2: Model performance metrics: ROC-AUC and Accuracy.

Best Model Suggestion

Based on the comparison, the **LSTM model** is the best choice due to its balanced precision and recall, slightly higher ROC-AUC (0.9807), and robust validation accuracy (0.9183). These metrics make it the most effective model for identifying toxic content across all classes.

Conclusion

The LSTM model emerged as the best choice for our task based on its superior performance across key metrics. Here are the top takeaways:

(1) **LSTM's Strength in Sequential Data:** LSTM demonstrated excellent performance in capturing long-range dependencies, making it well-suited for tasks like emotion recognition.
(2) **Pre-trained Embeddings:** While GloVe embeddings improved precision and recall, LSTM alone outperformed combined models, highlighting the effectiveness of LSTM in learning from raw data.
(3) **Comprehensive Evaluation Metrics:** Using multiple metrics, such as precision, recall, and ROC-AUC, ensures better model selection, especially in imbalanced datasets.
(4) **Simplicity Over Complexity:** A simpler LSTM model provided the best results, suggesting that adding complexity can sometimes lead to overfitting or inefficiencies.

Overall, this project reinforces the importance of model selection based on both the task at hand and the specific characteristics of the dataset. Understanding the strengths and limitations of different architectures and evaluation methods is key to making informed decisions in the field of machine learning and deep learning.

Future Work

The current model has shown promising results, but there are several key challenges that need to be addressed for further improvements:

1. **Data Labeling Issues**: The dataset was manually labeled by humans, which may introduce judgment bias. Different users may have varying opinions on the classification of comments, which can affect the consistency of the labels.

2. **Noisy Data**: Real-world datasets often contain noisy data, including slangs, contractions, non-English words, meaningless terms, and spam. Cleaning the dataset and handling such noise effectively can improve the model's performance.

3. **Unbalanced Dataset**: A large proportion of comments in the dataset are clean, while unclean comments are relatively rare. This imbalance, combined with the multi-class toxicity problem, can further aggravate model performance. Addressing this through techniques like data augmentation or class balancing would be beneficial.

4. **Computational Capability**: Natural Language Processing models, particularly Recurrent Neural Networks (RNNs), require significant processing power. Training these models efficiently demands the use of powerful GPUs. Future work should focus on optimizing the models for scalability and faster computation.

5. **Memory Issues**: The dataset includes over 100,000 training samples, which can be difficult to store and process in memory all at once. Implementing efficient data loading strategies, such as batching and caching, can help mitigate memory issues.

6. **Advanced Neural Networks**: Further improvements can be achieved by experimenting with more complex neural network architectures, such as Transformer-based models like BERT, GPT, or T5, for better context understanding and representation.

By addressing these challenges, the model's robustness, accuracy, and scalability can be enhanced, making it more suitable for real-world applications.

References

- **1. Research Paper** : "Hate me, hate me now: Hate speech detection on Facebook" by Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi
- **2. Research Paper:** "Exploring Hate Speech Detection in Multimodal Publications" by Raul Gomez, Jaume Gibert, Lluis Gomez, and Dimosthenis Karatzas
- **3. Research Paper:** "A Lexicon-based Approach for Hate Speech Detection" by Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long
- **4. Research Paper:** "Automated Hate Speech Detection and the Problem of Offensive Language" by Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber
- **5. Research Paper:** "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter" by Zeerak Waseem and Dirk Hovy
- **6. Research Paper:** "Detection of hate: speech tweets based convolutional neural network and machine learning algorithms" by Hameda A. Sennary, Ghada Abozaid, Ashraf Hemeida, Alexey Mikhaylov and Tamara Broderick
- **7. Research Paper:** "An approach to automatic classification of hate speech in sports domain on social media" by Staša Vujičić Stanković and Miljana Mladenović
- **8. Research Paper:** "Is hate speech detection the solution the world wants?" by Sara Parker and Derek Ruths
- **9. Research Paper:** "A Comprehensive Review on Automatic Hate Speech Detection in the Age of the Transformer" by Gil Ramos et al
- **10. Research Paper:**"Exploring Automatic Hate Speech Detection on Social Media: A Focus on Content-Based Analysis" by Francimaria R. S. Nascimento, George D. C. Cavalcanti, and Márjory Da Costa-Abreu
- **11. Research Paper:** "A Survey on Hate Speech Detection and Sentiment Analysis Using Machine Learning and Deep Learning Models" by Malliga Subramanian
- **12. Research Paper:** "Hate Speech Detection with ADHAR: A Multi-Dialectal Hate Speech Corpus in Arabic" by Anis Charfi, Mabrouka Besghaier, Raghda Akasheh, Andria Atalla, and Wajdi Zaghouani
- **13. Research Paper:** "Hate Speech Detection by Using Rationales for Judging Sarcasm" by Maliha Binte Mamun