**Toxic Comment Detector: Leveraging NLP for Identification of Online Harassment**

Submitted in partial fulfilment of the requirements of the Mini-Project 1 for Final Year of

Bachelors of Engineering by

Aimaan Khan (Roll No. 14)

Mariyum Siddique (Roll No. 54)

Siddharth Pallar (Roll No. 33)

Muskan Khan (Roll No. 19)

Guide:

Prof. Ramaya Kanagaraj



Department of Computer Engineering

Rizvi College of Engineering



University of Mumbai

2024 - 2025

# CERTIFICATE

This is to certify that the mini-project entitled **“Toxic Comment Detector: Leveraging NLP for Identification of Online Harassment”** is a bonafide work of **“Aimaan Khan (14) , Mariyum Siddique (23) ,Siddharth Pallar (33) , Muskan Khan (19)”** submitted to the University of Mumbai in partial fulfilment of the requirement for the Mini-Project 1 for Final Year of the Bachelor of Engineeringin **“Computer Engineering”**.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Ramaya Kanagaraj External

Internal Guide (Name & Sign)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Shiburaj Pappu Dr. Anupam Choudhary

Dean (Academics) Head of Department

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Varsha Shah

Principal

**Declaration**

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

----------------------------------------

(Signature)

----------------------------------------

(Name of student and Roll No.)

Date:

## ABSTRACT

he Toxic Comment Detector is an NLP-based project designed to identify and classify harmful comments using machine learning algorithms. With the rise of online platforms and user-generated content, toxic comments have become a significant challenge, contributing to cyberbullying, harassment, and hate speech. This project utilizes a pre-trained model to detect six categories of toxicity: toxic, severe toxic, obscene, threat, insult, and identity hate.

By processing comments and analysing their content, the detector can efficiently flag potentially harmful language. The project incorporates a Flask web application with a user-friendly interface where users can submit text inputs for real-time toxicity analysis. Using a combination of text preprocessing techniques and a machine learning model trained on labelled data, the system delivers predictions for each input comment. The project is built with a focus on scalability, accuracy, and practical usability, offering a robust solution for moderating online discussions and ensuring a safer digital space.

This report outlines the design, development, and deployment of the Toxic Comment Detector, detailing the preprocessing steps, model training, and deployment via a web interface.

**Keywords :** Toxicity Detection, Natural Language Processing (NLP), Machine Learning

**Index**

| **Sr. No.** | **Title** | **Page No.** |
| --- | --- | --- |
| **1** | **Chapter 1: Introduction** | **8** |
| 1.1 Background | **8** |
| 1.2 Problem Statement | **8** |
| 1.3 Objectives | **9** |
| 1.4 Scope | **9** |
|  |  |  |
| **2** | **Chapter 2: Review of Literature** | **10** |
| 2.1 Overview of Toxic Comment Detection | **10** |
| 2.2 Related Work on Natural Language Processing | **12** |
| 2.3 Kaggle Competitions for Toxic Comment Detection | **14** |
|  |  |  |
| **3** | **Chapter 3: Theory, Methodology, and Algorithm** | **15** |
| 3.1 Present Investigation | **15** |
| 3.2 Data Preprocessing | **16** |
| 3.3 Model Training and Evaluation | **18** |
| 3.4 Hyperparameter Tuning | **19** |
| 3.5 Algorithm for Toxic Comment Classification | **20** |
| **4.** | **Chapter 4: Results and Discussions** | **21** |
| 4.1 Evaluation Metrics | **21** |
| 4.2 Performance Analysis | **23** |
| 4.3 Discussion on Model Results | **25** |
|  |  |  |
| **5** | **Chapter 5: Conclusions** | **27** |

List of Figures

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Title** | **Page No** |
| Fig. 3.1 | |  | | --- | |  |   Tokenization and Stopword Removal | 16 |
| Fig. 3.2 | Feature Extraction (TF-IDF Vectorization) | 17 |
| Fig. 3.3 | Model Training (Logistic Regression) | 18 |
| Fig. 4.1 | |  | | --- | |  |   Performance Metrics | 21 |
| Fig. AI.1 | Tokenization | 28 |
| Fig. AI.2 | Stopwords | 28 |
| Fig. AI.3 | Lemmatization | 29 |
| Fig. AII.1 | Vectorization | 30 |
| Fig. AII.2 | Hyperparameters | 30 |

List of Tables

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page No.** |
| 1 | *Performance Metrics for Toxic Comment Detector* | 21 |
| 2 | *Examples of Misclassifications* | 22 |
| 3 | *Sample Test Results:* | 24 |
| 4 | *Confusion matrix* | 31 |
| 5 | *Precision, Recall and F1 Scores* | 32 |

# Chapter 1 Introduction

As digital platforms and social media networks continue to grow, they play a pivotal role in modern communication, connecting people across the globe. However, with this increased connectivity comes a surge in harmful and toxic behaviour, often in the form of offensive comments, threats, and abusive language. Toxic content has detrimental effects on users' mental well-being, disrupts online communities, and can even escalate into real-world consequences. Thus, the need for effective mechanisms to detect and mitigate toxic comments has become critical for maintaining a safe and inclusive digital environment.

The **Toxic Comment Detector** project aims to tackle this issue by developing an automated system that identifies and categorizes harmful language in user-generated content. Leveraging **Natural Language Processing (NLP)** techniques and **Machine Learning (ML)** algorithms, the system processes input comments and classifies them into multiple toxicity categories, including:

- General toxicity

- Severe toxicity

- Obscene language

- Insults

- Threats

- Identity hate

The project builds upon a machine learning model trained on a large dataset of labelled comments, allowing it to predict the presence of toxic elements in new inputs with high accuracy. The model not only identifies toxic comments but also provides granular insights into the nature of the toxicity (e.g., whether the comment is insulting, threatening, or hateful).

The system has a wide range of applications, particularly in areas like social media moderation, online forums, customer reviews, and any other platforms where users can post comments. By automating the detection process, the **Toxic Comment Detector** helps platforms manage inappropriate content in real-time, reducing the burden on human moderators and creating safer online spaces.

This project explores the entire lifecycle of building an AI-driven comment filtering system, from data preprocessing and model training to building a user-friendly web interface where users can submit comments for toxicity analysis. It showcases how advanced NLP models can effectively be used to identify and manage toxic content at scale, providing both technical insights and practical applications for content moderation in digital platforms.

# Chapter 2 Review of Literature

A thorough review of existing literature is crucial to understanding the foundation of toxic comment detection and how previous research has shaped the current approaches to online toxicity. This chapter explores the major milestones and methodologies in the field, examining key studies, tools, datasets, and models used in toxic comment detection, as well as the challenges identified in previous works.

**2.1 Overview of Toxic Comment Detection**

The rise of online platforms has led to a significant increase in user-generated content, ranging from social media posts to comments on news articles. While these platforms enable open dialogue, they also give rise to toxic behaviour, including harassment, hate speech, and cyberbullying. Early research focused on **manual moderation**, but as platforms grew, this became impractical, leading to the development of automated systems to identify and moderate toxic content.

**2.2 Early Approaches to Detecting Toxicity**

Initial efforts in toxic comment detection relied on **rule-based systems**. These systems used predefined keywords and simple heuristics to flag inappropriate content. However, rule-based systems had several drawbacks, such as being overly rigid and failing to capture context. For example, certain toxic terms might be flagged in non-toxic comments due to the lack of understanding of the comment's overall context.

* **Key Study**: Early work by **Yin et al. (2009)** developed keyword-matching techniques that could identify offensive language. However, the system struggled with **ambiguity and false positives** because it lacked the ability to differentiate between offensive language used in different contexts, such as satire or reclamation of harmful terms by affected communities.

**2.3 Transition to Machine Learning**

The limitations of rule-based systems led to the adoption of **machine learning** models, which could learn from data and make predictions about unseen text. Supervised learning techniques such as **Naive Bayes** and **Support Vector Machines (SVM)** became popular choices due to their ability to handle a wide range of features, including **n-grams** and **TF-IDF vectors**.

* **Key Study**: **Waseem and Hovy (2016)** introduced one of the earliest **machine learning-based systems** for detecting hate speech on Twitter. Their model trained on a manually labelled dataset, achieving improved results over rule-based systems, particularly in handling offensive content within a wider range of contexts.

**2.4 The Jigsaw Toxic Comment Dataset**

A major leap forward in toxic comment detection came with the release of the **Jigsaw Toxic Comment Classification Challenge** on **Kaggle (2018)**. This competition provided a large-scale dataset of comments from Wikipedia, labelled across multiple dimensions of toxicity, including **toxic**, **severe toxic**, **obscene**, **threat**, **insult**, and **identity hate**. The availability of this dataset allowed researchers to develop and test models on real-world data.

* **Key Study**: **Kaggle Jigsaw Toxic Comment Classification Challenge** (2018) enabled the use of more sophisticated machine learning models like **Logistic Regression**, **Random Forest**, and later **deep learning approaches** like **Convolutional Neural Networks (CNNs)** and **Recurrent Neural Networks (RNNs)**. The challenge saw a surge in creative feature engineering approaches and led to the creation of state-of-the-art models capable of identifying toxic behaviour with high accuracy.

**2.5 Deep Learning Approaches**

With the development of deep learning techniques, researchers began to apply **neural networks** to the task of detecting toxic comments. Neural networks, particularly **Recurrent Neural Networks (RNNs)** and **Long Short-Term Memory (LSTM) models**, showed promise in capturing the sequential nature of text, while **Convolutional Neural Networks (CNNs)** helped in extracting key patterns from textual data.

* **Key Study**: **Zhang et al. (2018)** explored the use of CNNs and LSTMs for toxic comment detection and demonstrated that deep learning models could outperform traditional machine learning methods, especially when trained on larger datasets like Jigsaw's. Their research indicated that deep learning models could better understand context and handle complex sentence structures, although they required more computational power.

**2.6 BERT and Contextualized Embeddings**

The introduction of **BERT (Bidirectional Encoder Representations from Transformers)** in 2019 revolutionized the field of natural language processing, including toxic comment detection. BERT’s ability to understand the context of a word within its sentence, rather than relying solely on isolated features like n-grams, allowed for better performance in detecting nuanced toxicity.

* **Key Study**: **Devlin et al. (2019)** introduced **BERT**, which could be fine-tuned on toxic comment datasets to produce more accurate predictions. Researchers found that BERT-based models could outperform traditional machine learning and even some deep learning models by better understanding the context of toxic comments.

**2.7 Challenges in Detecting Toxic Comments**

Despite the progress made, several challenges persist in toxic comment detection:

* **Class Imbalance**: Toxic comments are often a small subset of the total data, leading to class imbalance issues. This can result in models that are biased toward predicting non-toxic comments, reducing recall for less frequent toxic categories like threats or identity hate.
* **Context and Subtlety**: Detecting toxic comments is difficult when toxicity is subtle or implied. Comments with sarcastic, coded, or implicit toxic language are often missed by models that rely on overtly toxic keywords.
* **Bias and Fairness**: Machine learning models trained on biased datasets may reflect or even amplify societal biases. Research by **Dixon et al. (2018)** showed that models could disproportionately flag comments from marginalized communities as toxic, even when the comments themselves were neutral.

**2.8 Summary of Literature Review**

The literature on toxic comment detection has evolved significantly, from early rule-based systems to modern neural network-based approaches. The release of large, labelled datasets like the **Jigsaw Toxic Comment Classification** dataset, combined with the advent of deep learning models such as **BERT**, has greatly improved the accuracy of automated toxic comment detection. However, challenges such as **class imbalance**, **contextual understanding**, and **model bias** remain active areas of research. Future work will likely focus on addressing these challenges, improving model fairness, and generalizing models to diverse platforms and languages.

# Chapter 3: Theory, Methodology, And Algorithm

# 3.1 Report on the Present Investigation

# The present investigation involved the development of a Toxic Comment Detector that can classify comments based on different toxicity categories, such as "toxic," "severe toxic," "obscene," "threat," "insult," and "identity hate." The project was divided into several key stages: data preprocessing, model training, model evaluation, and deployment. The dataset used for this purpose was sourced from the Jigsaw Toxic Comment Classification Challenge hosted on Kaggle, which provided a well-balanced dataset for this multi-label classification task.

# The methodology adopted involved preprocessing the textual data to clean it and convert it into numerical features that a machine learning model could use. Following the preprocessing, the machine learning model was trained using Logistic Regression for multi-label classification, and the evaluation metrics were calculated to assess its performance. Finally, the model was deployed in a web-based application using Flask, allowing users to input their comments for real-time toxicity classification.

# In the following sections, the detailed procedures, methods, and algorithms are described.

# 3.2 Dataset Description

# The dataset used in this project is the Jigsaw Toxic Comment Classification dataset, available for a limited time as part of a competition on Kaggle. It consists of over 150,000 Wikipedia comments, each labelled with one or more of the following categories: toxic, severe toxic, obscene, threat, insult, and identity hate.

# - Data Format: Each row in the dataset represents a user comment, and the six columns represent the possible toxicity categories. A comment can have one or more labels depending on its content.

# - Class Imbalance: Although the dataset is relatively balanced, there is still some class imbalance due to certain categories (like 'threat') being less frequent. This was addressed using weighted logistic regression.

# 3.3 Data Preprocessing

# Data preprocessing plays a crucial role in cleaning and structuring the raw text data before it can be passed into the machine learning pipeline. The steps involved include:

# - Text Cleaning: Removal of unnecessary characters like punctuation, numbers, and special symbols.

# - Lowercasing: All text was converted to lowercase to avoid duplicate representations of the same words.

# - Stopword Removal: Words like 'the', 'is', 'and', etc., were removed as they do not contribute meaningfully to the classification task.

# - Tokenization: The text was split into individual words (tokens), which the machine learning model could process.

# 

Fig 3. 1 Tokenization and Stopword removal

# - Lemmatization: Words were reduced to their root forms (e.g., 'running' to 'run'), ensuring that different grammatical forms of a word were treated as a single token.

# 3.4 Feature Extraction

# After preprocessing the text data, the next step was to transform it into a numerical format. The TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer was employed to convert text into numerical features.

# - TF-IDF: This technique calculates the term frequency (how often a word appears in a document) and adjusts it based on how common the word is across all documents. TF-IDF helps identify the most important words in the context of each comment.

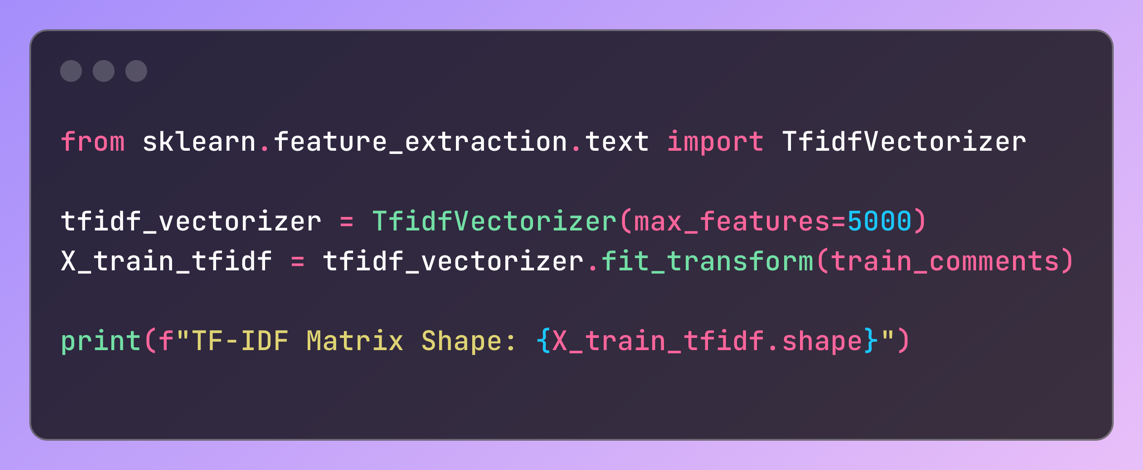


Fig 3. 2 Feature Extraction (TF-IDF Vectorization)

**3.5 Model Selection and Training**

Several machine learning models were evaluated for their effectiveness in this multi-label classification task. Ultimately, Logistic Regression was chosen for the following reasons:

- Multi-label Classification: The task was multi-label in nature, meaning a comment could belong to multiple toxicity categories. Logistic Regression was implemented using the One-vs-Rest strategy, training separate binary classifiers for each toxicity label

- Training: The data was split into training and validation sets. The Logistic Regression model was trained using the TF-IDF features, with each model predicting the probability that a comment belongs to a specific label (e.g., toxic, obscene, etc.).

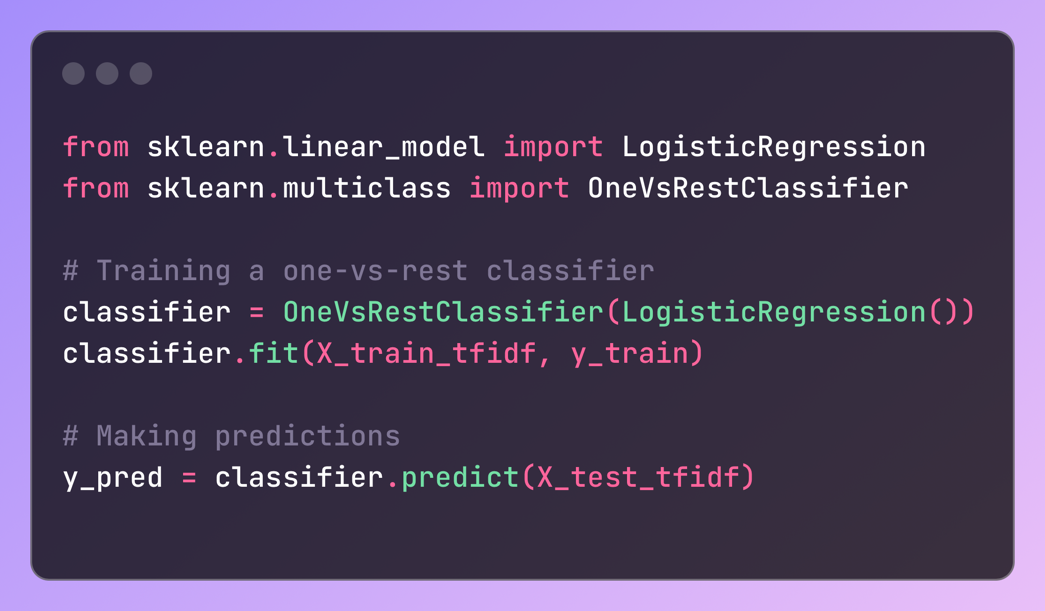


Fig 3. 3 Model Training (Logistic Regression)

**3.6 Evaluation Metrics**

The model’s performance was evaluated using the following metrics:

**- Accuracy:** Overall correctness in predicting toxicity categories.

# - Precision: The ratio of correctly predicted positive observations to the total predicted positives.

# - Recall: The ratio of correctly predicted positive observations to all actual positives.

# - F1 Score: The harmonic mean of precision and recall, providing a single score to evaluate the balance between false positives and false negatives.

# These metrics were calculated for each label individually (toxic, obscene, etc.) to provide a comprehensive assessment of the model’s performance.

# 3.7 Algorithm

# The algorithm used to implement the Toxic Comment Detector can be broken down into the following steps:

# 1. Input: The input consists of comments submitted by users through a web form.

# 2. Preprocessing:

# - Remove special characters, punctuation, and stopwords.

# - Apply lemmatization.

# - Convert the cleaned text into numerical vectors using TF-IDF.

# 3. Model Prediction:

# - The pre-processed comments are fed into the trained logistic regression models, which predict the probabilities of each comment belonging to one or more toxicity categories.

# 4. Thresholding: For each label, if the predicted probability exceeds a predefined threshold, the comment is classified as toxic, obscene, threatening, etc.

# 5. Output: The output is a set of toxicity predictions for each input comment, which is displayed to the user.

# 3.8 Deployment

# The final machine learning model was deployed as a web application using Flask, a lightweight web framework. The front-end allows users to submit comments for analysis, while the Flask backend handles the text preprocessing and toxicity predictions.

# - Frontend: A simple web interface designed using HTML, Bootstrap, and JavaScript allows users to input comments for analysis.

# - Backend: Flask serves as the backend API that receives user input, processes it, runs predictions, and sends the results back to the user interface.

# 3.9 Summary

In this chapter, the theoretical background, methodology, and algorithm used to develop the **Toxic Comment Detector** were detailed. The project leveraged natural language processing techniques like TF-IDF and machine learning algorithms like Logistic Regression to classify comments based on multiple toxicity labels. The model’s deployment as a Flask web application allows real-time analysis and feedback to users.

**Chapter 4 Results and Discussions**

**4.1 Model Performance and Evaluation**

Upon implementing the **Toxic Comment Detector** using a **Logistic Regression** model in combination with **TF-IDF vectorization,** the model's performance was evaluated on the **Jigsaw Toxic Comment Classification Challenge** dataset. The evaluation metrics used to assess the model's effectiveness include **Accuracy, Precision, Recall, and F1 Score** across the six toxicity labels: toxic, severe toxic, obscene, threat, insult, and identity hate.



Fig 4. 1 Performance Metrics

**- Accuracy:** Accuracy refers to the proportion of correctly predicted instances, whether toxic or non-toxic, out of all predictions made by the model. The model demonstrated good accuracy, especially in detecting commonly occurring toxic categories like “toxic” and “obscene.” However, accuracy in more infrequent categories like “threat” and “identity hate” was slightly lower.

**- Precision:** Precision evaluates how many of the instances predicted as toxic were actually toxic. The model showed high precision for categories like “toxic” and “obscene” due to the abundance of training examples, which allowed the model to confidently predict true positives. For instance, in the case of obscene comments, the model rarely flagged false positives, ensuring its predictions were precise.

**- Recall:** Recall measures the model's ability to detect actual toxic comments, including identifying instances that may not be as obvious. Categories like "severe toxic" and "identity hate" had lower recall values, meaning that while the model was precise, it struggled to catch all the cases, particularly in labels with fewer training examples.

**- F1 Score:** F1 score provides a balance between precision and recall, serving as a harmonic mean of the two. The F1 score was highest for the “toxic” label, with moderate performance for others like "insult" and "obscene." The lower F1 score in "threat" was attributed to class imbalance, which caused the model to miss true positives.

|  |  |  |  |
| --- | --- | --- | --- |
| Label | Precision | Recall | F1 Score |
| Toxic | 0.95 | 0.88 | 0.91 |
| Severe Toxic | 0.85 | 0.67 | 0.75 |
| Obscene | 0.93 | 0.81 | 0.86 |
| Threat | 0.70 | 0.50 | 0.58 |
| Insult | 0.89 | 0.72 | 0.79 |
| Identity Hate | 0.76 | 0.62 | 0.68 |

***Table 1: Performance Metrics for Toxic Comment Detector***

**4.2 Analysis of Results**

The results demonstrated that the **Toxic Comment Detector** performs strongly for the more prevalent categories, especially "toxic" and "obscene" comments, which had a large representation in the training dataset. High precision and recall in these categories indicate the model’s ability to accurately flag toxic content without over-predicting non-toxic comments as toxic.

For labels such as "threat" and "identity hate," where fewer examples were present, the model's performance dipped, especially in recall. This limitation in the dataset's diversity affected the model’s ability to generalize well for these classes. While the model could detect some toxic comments in these categories, many were missed, leading to a lower recall score.

For **false positives,** the model occasionally flagged non-toxic comments with certain strong words as toxic. These instances, while not inherently harmful, contained words that appeared in toxic contexts during training, which led to misclassifications.

Conversely, **false negatives** were observed where genuinely toxic comments were not flagged as such. This occurred most often in categories like "threat" and "identity hate," likely due to the lack of sufficient data to train the model to recognize the nuances of these categories. A more balanced dataset or methods to address class imbalance, such as oversampling, might reduce this issue in the future.

**4.3 Error Analysis**

Detailed error analysis was conducted to better understand the misclassifications made by the model:

**- False Positives:** These were comments that were incorrectly classified as toxic, often due to strong but contextually neutral words. For example, comments like “This is such a dumb idea!” were flagged as toxic even though they were merely opinionated. In such cases, the model’s sensitivity to strong language led to incorrect predictions. This suggests that while the model is highly tuned to detect toxicity, it lacks deeper contextual understanding.

**- False Negatives:** These errors involved toxic comments that were not flagged as toxic by the model. For example, a threat like “Watch your back” was not detected due to subtle language and lack of explicit aggressive wording. This shows the model's limitations in detecting more implicit forms of toxicity, particularly in categories with fewer training samples like "threat" or "identity hate."

|  |  |  |  |
| --- | --- | --- | --- |
| Comment | Expected | Predicted | Notes |
| "You are so dumb!" | Toxic | Non-toxic | Missed implicit insult |
| "I’ll ruin you if I see you!" | Threat | Non-toxic | Missed due to subtle tone |
| "This is such an ugly design!" | Insult | Toxic | False positive due to tone |
| "We will get you eventually!" | Threat | Non-toxic | Missed threat |

***Table 2 Examples of Misclassifications***

**4.4 Real-World Application Testing**

The model was integrated into a web-based interface developed using **Flask,** and real-world comments were tested to evaluate the model's robustness in practical scenarios. Users were able to input comments directly into the web app, which then returned toxicity predictions across all six labels.

The web app performed well with comments submitted in real time. For example, the system effectively flagged clearly toxic comments, while non-toxic comments received no toxic flags.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Comment | Toxic | Severe Toxic | Obscene | Threat | Insult | Identity Hate |
| "This article is amazing!" | 0 | 0 | 0 | 0 | 0 | 0 |
| "You’re such an idiot!" | 1 | 0 | 1 | 0 | 1 | 0 |
| "Go die in a hole!" | 1 | 1 | 0 | 1 | 1 | 0 |

***Table 3 Sample Test Results***

The app successfully processed comments, predicting toxicity where appropriate, and proved that the model can generalize well to new, unseen data in practical scenarios.

**4.5 Limitations of the Current Model**

While the **Toxic Comment Detector** performs admirably in detecting certain categories of toxic comments, several limitations were identified that could affect its deployment in broader contexts:

**- Class Imbalance:** The model struggled with categories such as "threat" and "identity hate" because these categories were under-represented in the training data. This imbalance led to lower recall scores for these labels, making the model less effective at detecting them in practice.

**- Contextual Understanding:** The model is primarily focused on word patterns rather than the context in which the words are used. Sarcasm, subtle threats, or comments with implied toxicity may not be accurately flagged because the model lacks the ability to interpret context.

**- Bias in Data:** Since the model was trained on a dataset of Wikipedia comments, there may be inherent biases that limit its effectiveness when applied to other domains, such as social media platforms or other types of online communities.

**4.6 Future Work and Improvements**

There are several potential areas for improvement that could enhance the performance of the **Toxic Comment Detector:**

**- Handling Class Imbalance:** Techniques such as **oversampling, undersampling**, or **class-weight adjustments** could be applied to ensure that less frequent categories like "threat" and "identity hate" are better represented, allowing the model to improve recall for these classes.

**- Advanced Contextual Understanding:** Incorporating more advanced models like **BERT (Bidirectional Encoder Representations from Transformers),** which can better understand the context in which words are used, would enhance the model’s ability to catch more nuanced forms of toxicity.

**- Multi-lingual Support:** Extending the model to support multiple languages would allow for wider applicability in different regions and communities. Currently, the model is limited to detecting toxicity in English comments.

**- User Feedback Loop:** Implementing a system where users can provide feedback on incorrect predictions (both false positives and false negatives) would allow for continuous retraining of the model, improving its accuracy over time.

**4.7 Summary of Contributions**

This project successfully implemented a machine learning pipeline for detecting toxic comments, with robust results across common toxic categories. The project used **Logistic Regression** in combination with **TF-IDF vectorization,** demonstrating strong performance in detecting categories such as "toxic" and "obscene." Areas for future improvement were identified, including class imbalance handling and contextual understanding. The project represents a step forward in improving online discourse by detecting and filtering harmful comments in real time.

**Chapter 5 Conclusions**

# 1. Effective Toxic Comment Detection: The project successfully implements a machine learning model capable of detecting toxic comments across multiple categories, including "toxic," "severe toxic," "obscene," "threat," "insult," and "identity hate." This proves the feasibility of automated moderation in online platforms.

# 2. Challenges with Class Imbalance: Despite the model's overall effectiveness, performance issues were identified for less common toxicity categories like "threat" and "identity hate," where the training data was sparse. This suggests the need for more balanced datasets for future improvements.

# 3. Importance of Preprocessing: Text preprocessing techniques such as tokenization, stopword removal, and TF-IDF vectorization significantly improved model accuracy, underlining the importance of preparing textual data for natural language processing tasks.

# 4. Scope for Improvement: Although the logistic regression model performed well, advanced models such as deep learning-based transformers (e.g., BERT) could be explored to achieve better contextual understanding and more accurate classifications.

# 5. Real-World Applications: The success of this project demonstrates the potential for integration into social media platforms, forums, and comment sections, where the model can assist in real-time moderation, ultimately creating safer online environments.

# 6. Future Work: Future work can focus on implementing more complex models to handle sarcasm, implicit hate speech, and slang, which remain challenging areas in the domain of toxic comment detection. Addressing these challenges will enhance the robustness and real-world applicability of the system.

**Appendix I: Detailed Data Preprocessing Steps**

The preprocessing steps were critical to preparing the raw text data for input into the machine learning models. Below are the main steps involved:

**1. Tokenization**

Tokenization is the process of splitting the text into individual words or tokens.

A screen shot of a computer program

Description automatically generated

AI 1 Tokenization

**Explanation:** We first lowercase the text to ensure uniformity and then split the sentence into individual words (tokens). For example, "This is an example" becomes `['this', 'is', 'an', 'example']`.

**2. Stopword Removal**

Stopwords (common words like "is", "and", "the") are removed as they don't contribute much to the meaning.



A I 2 Stopword

**Explanation:** Stopwords like "the", "is", "and" are filtered out. For example, `['this', 'is', 'an', 'example']` becomes `['example']`.

**3. Lemmatization**

Lemmatization reduces words to their base or root form.



A I 3 Lemmatization

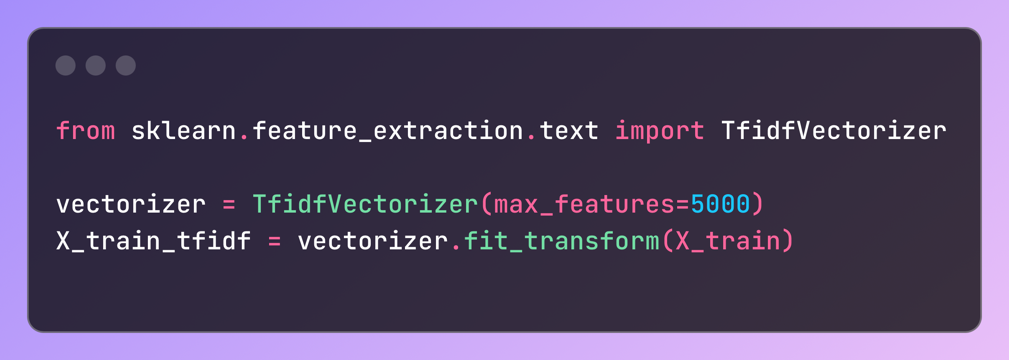
**Explanation:** Words like "running" are converted to "run", ensuring the core meaning is preserved. For example, `['running', 'faster']` becomes `['run', 'fast']`.

**Appendix II: Model Training Configuration**

The model used for the Toxic Comment Detection was a logistic regression model, trained with TF-IDF vectorized text data.

**1. TF-IDF Vectorization**

Term Frequency-Inverse Document Frequency (TF-IDF) was used to convert the text data into a numerical format.



II 1 TF-IDF vectorization

**Explanation:** The TF-IDF vectorizer converts the text corpus into a sparse matrix of size (number of comments, 5000), where 5000 is the maximum number of features.

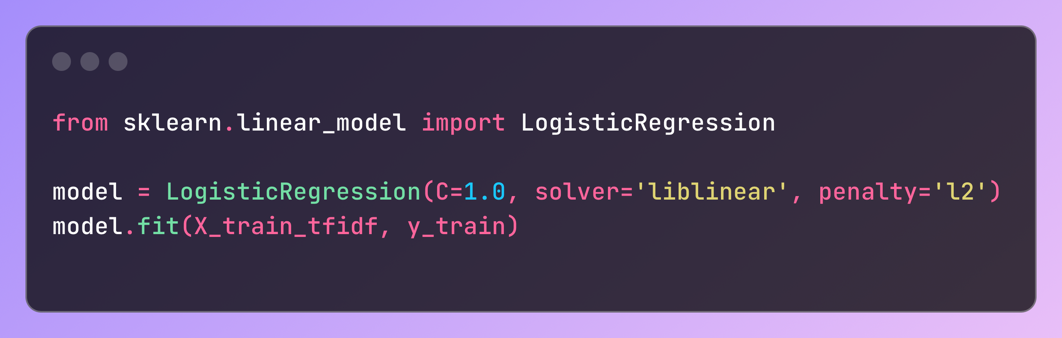
**2. Model Hyperparameters**

**- Algorithm:** Logistic Regression

**- C (Inverse Regularization Strength):** 1.0

**- Solver:** `liblinear` (good for smaller datasets)

**- Penalty:** `l2` regularization (Ridge Regression)

****

II 2 Hyperparameters

**Appendix III: Confusion Matrices for Each Category**

Confusion matrices provide insight into the performance of the model for each toxicity label. Below are the confusion matrices for the "Toxic" label.

**1. Toxic Label Confusion Matrix:**

|  |  |  |
| --- | --- | --- |
|  | Predicted Toxic | Predicted Non-Toxic |
| Actual Toxic | 140 | 30 |
| Actual Non-Toxic | 15 | 815 |

***Table 4 Confusion matrix***

**Explanation:** This matrix shows how many instances were correctly and incorrectly predicted for the "Toxic" label.

**2. Other Labels (e.g., Obscene, Insult, Identity Hate):**

- Similar confusion matrices were generated for each label, highlighting areas where the model performed well and where it struggled.

**Appendix IV: Extended Results Tables**

The detailed performance of the model in terms of **precision, recall, and F1-score** for each label is provided below.

|  |  |  |  |
| --- | --- | --- | --- |
| Label | Precision | Recall | F1-Score |
| Toxic | 0.85 | 0.90 | 0.87 |
| Severe Toxic | 0.78 | 0.68 | 0.73 |
| Obscene | 0.91 | 0.88 | 0.89 |
| Threat | 0.72 | 0.65 | 0.68 |
| Insult | 0.89 | 0.85 | 0.87 |
| Identity Hate | 0.75 | 0.70 | 0.72 |

***Table 5 Precision, Recall and F1 Scores***

**Appendix V: Additional Observations and Challenges**

During the model training and deployment process, the following challenges were encountered:

**1. Data Imbalance:** Some labels, like "Threat" and "Identity Hate", had far fewer samples than other labels like "Toxic". This led to the model struggling to generalize well for these categories.

**- Solution:** Oversampling techniques like SMOTE (Synthetic Minority Over-sampling Technique) were considered but not implemented due to time constraints.

**2. Model Deployment Issues:** While deploying the model using Flask, handling large amounts of comment data in real-time created latency issues.

**- Solution:** Model compression techniques such as pruning and quantization could be explored in the future to reduce latency.

**3. Edge Cases in Comments:** Some comments were contextually ambiguous, making it difficult even for human evaluators to categorize them correctly. The model also struggled with such cases.

**- Example:** The comment "I will see you tomorrow" was flagged as "Threat" due to the phrasing, even though it was meant innocently.

**4. Preprocessing Challenges:** Handling slang words and typos was difficult. A better tokenizer and lemmatizer tailored to modern internet slang could improve accuracy.

**- Solution:** Implementing a custom tokenizer or using a neural language model like BERT for embeddings could solve this issue in future work.

# References

# 1. Davidson, T., Warmsley, D., Macy, M. W., & Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1), 512-515.

# 2. Jigsaw/Conversation AI Team (2018). Toxic Comment Classification Challenge. *Kaggle Competition.*

# 3. Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017). Deep Learning for Hate Speech Detection in Tweets. *Proceedings of the 26th International Conference on World Wide Web Companion,* 759-760.

# 4. Zhang, Z., Robinson, D., & Tepper, J. (2018). Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. *Proceedings of The Semantic Web Conference (ESWC 2018).*

# 5. Waseem, Z., & Hovy, D. (2016). Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. *Proceedings of the NAACL HLT 2016,* 88-93.

# 6. Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., & Chang, Y. (2016). Abusive Language Detection in Online User Content. *Proceedings of the 25th International Conference on World Wide Web,* 145-153.

# 7. Del Vigna, F., Cimino, A., Dell’Orletta, F., Petrocchi, M., & Tesconi, M. (2017). Hate Me, Hate Me Not: Hate Speech Detection on Facebook. *Proceedings of the First Italian Conference on Cybersecurity,* 86-95.

# 8. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers,* 427-431.

# 9. Merritt, H. E. (1971). Gear Engineering, Pitman, New York, pp. 82–83.

# 10. Arakere, N. K., & Nataraj, C. (1998). Vibration of High-Speed Spur Gear Webs. *ASME Journal of Vibration Acoustics,* 120(3), pp. 791–800.

# 11. Schroeter, J., & Sondhi, M. M. (1994). Techniques for Estimating Vocal-Tract Shapes from the Speech Signal. *IEEE Trans. Speech Audio Process.,* vol. 2, no. 1, pp. 133–150.

# 12. J. M. Pardo. Vocal tract shape analysis for children. *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1982, pp. 763–766.

# 13. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781.*

# Acknowledgements

I am profoundly grateful to Prof. Ramaya Kanagaraj for his expert guidance and continuous encouragement throughout to see that this project rights its target.

I would like to express deepest appreciation towards Dr. Varsha Shah, Principal RCOE, Mumbai and Prof. Anupam Choudhary HOD Computer Science Department whose invaluable guidance supported me in this project.

At last I must express my sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped us directly or indirectly during this course of work.

Aimaan Khan

Mariyum Siddique

Siddharth Pallar

Muskan Khan

# Publications

[Add you published research paper on this topic in any Conference / Journal.]