Report On

Toxic Comments Classification

Submitted in partial fulfillment of the requirements of the Mini project
in Semester VII of Fourth Year Artificial Intelligence & Data Science Engineering

by

Shruti Sunil Pawar (37)

Mokshad Sankhe (67)

Sudeep Shetty (70)

Under the guidance of

Prof. Raunak Joshi

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(A.Y. 2024-25)

CERTIFICATE

This is to certify that the Mini Project entitled "**Toxic Comments Classification**" is a bonafide work of **Shruti Pawar (37), Mokshad Ketan Sankhe (67), Sudeep Shetty (70)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering"** in Semester VII of Fourth Year **"Artificial Intelligence and Data Science".**

Mr. Raunak Joshi
Guide

Ms Sejal D'Mello	Dr. Tatwadarshi P N	Dr. H.V. Vankudre
Deputy HOD AI & DS	HOD AI & DS	Principal

Mini Project Approval

This Mini Project entitled "**Toxic Comments Classification**" by **Shruti Pawar (37), Mokshad Ketan Sankhe (67), Sudeep Shetty (70)** is approved for the degree of **Bachelor of Engineering** in in Semester VII of Fourth Year **Artificial Intelligence and Data Science.**

Examiners

1…….....
(Internal Examiner Name & Sign)

2……
(External Examiner Name & Sign)

Date:

Place:

Abstract

To address the growing difficulty of recognizing and reducing harmful content in online platforms, this research discusses the development and implementation of an advanced web-based toxicity detection system. The system analyzes and categorizes user-generated content as either harmful or non-toxic in real-time using cutting-edge Natural Language Processing (NLP) techniques and machine learning algorithms.

A pre-trained machine learning model serves as the foundation for the system, and feature extraction is accomplished through the use of Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. This model is included into an online application that uses Flask and offers a user-friendly interface for immediate toxicity analysis. Tokenization, part-of-speech tagging, lemmatization, and other advanced NLP techniques are incorporated into the system's text preprocessing pipeline, which guarantees reliable handling of a variety of textual inputs.

Acknowledgement:

We would like to thank all people whose support and cooperation has been an invaluable asset during this Project. We would also like to thank our Guide **Mr. Raunak Joshi**, for guiding us throughout this project and giving it the present shape. It would have been impossible to complete the project without their support, valuable suggestions, criticism, encouragement, and guidance.

We convey our gratitude to **Dr. Tatwadarshi Nagarhalli**, Head of Department, for their motivation and providing various facilities, which helped us greatly in the whole process of this project. We are also grateful to all other teaching and non-teaching staff members of the Artificial Intelligence and Data Science Department for directly or indirectly helping us with the completion of projects and the resources provided.

Shruti Pawar (37)

Mokshad Sankhe (72)

Sudeep Shetty (75)

Date:

Place:

Contents

List of Abbreviations

1. **AI** - Artificial Intelligence
2. **API** - Application Programming Interface
3. **BERT** - Bidirectional Encoder Representations from Transformers
4. **BiLSTM** - Bidirectional Long Short-Term Memory
5. **CCPA** - California Consumer Privacy Act
6. **CSV** - Comma-Separated Values
7. **GDPR** - General Data Protection Regulation
8. **KNN** - K-Nearest Neighbors (if mentioned in the model section)
9. **mBERT** - Multilingual Bidirectional Encoder Representations from Transformers
10. **NLP** - Natural Language Processing
11. **NLTK** - Natural Language Toolkit
12. **POS** - Part-of-Speech
13. **RNN** - Recurrent Neural Network
14. **ROC-AUC** - Receiver Operating Characteristic - Area Under the Curve
15. **TF-IDF** - Term Frequency-Inverse Document Frequency
16. **TPU** - Tensor Processing Unit (if mentioned)
17. **TF** - TensorFlow
18. **HTML** - Hypertext Markup Language
19. **Flask** - A lightweight WSGI web application framework in Python
20. **SVM** - Support Vector Machine (if mentioned)
21. **SQL** - Structured Query Language
22. **RAM** - Random Access Memory

List of Figures

1. INTRODUCTION

1.1 INTRODUCTION

The widespread availability of online communication channels in today's digital world has drastically changed the way people communicate and exchange information. Although this shift has resulted in previously unheard-of levels of connectedness, it has also raised serious issues with user safety and content management. The vast amount of user-generated information that is spread across numerous digital platforms makes manual moderation unsuitable, making the development of automated systems for identifying and screening harmful content more and more important.

To overcome these obstacles, the Toxicity Detection System described in this paper makes use of cutting-edge machine learning algorithms and Natural Language Processing (NLP) approaches. The system, which is based on a Flask-based web architecture, uses a complex text processing pipeline to prepare user input for analysis. This pipeline incorporates tokenization, part-of-speech tagging, and lemmatization. To accomplish accurate harmful content classification, the primary functionality leverages Term Frequency-Inverse Document Frequency (TF-IDF) vectorization in conjunction with a pre-trained machine learning model.

Our approach makes use of the extensive capabilities of the Natural Language Toolkit (NLTK) package, which includes powerful part-of-speech tagging and WordNet lemmatization. Robust text normalization and feature extraction are ensured by this method, which is essential for preserving constant classification accuracy across a range of text inputs. Pickle serialization is used in the system's architecture to facilitate quick deployment and effective model persistence, all while maintaining scalability.

1.2 PROBLEM STATEMENT & OBJECTIVE

Problem Statement:

Due to the exponential rise of online communication platforms, content moderation is facing an unprecedented problem as manual review techniques have grown more and more ineffective and unsustainable. The intricacy of natural language, encompassing context-specific interpretations, irony, and dynamic internet jargon, poses a significant obstacle to the creation of automated systems capable of accurately detecting harmful information. In addition, the requirement for fast content filtering and the resource-intensive nature of manual moderation pose a serious obstacle to the upkeep of vibrant online communities. This calls for the creation of an automated system with high accuracy and flexibility to handle different types of online toxicity, capable of processing and classifying text content in real-time.

Objectives:

- **Analyze the Dataset**: Conduct exploratory data analysis to understand the distribution and characteristics of toxic and non-toxic tweets, identifying key linguistic patterns and features associated with toxic behavior.
- **Develop a Classification Model**: Implement and train various machine learning algorithms and natural language processing techniques to classify tweets based on their toxicity levels, comparing the effectiveness of different approaches.
- **Evaluate Model Performance**: Assess the performance of the developed models using appropriate metrics (e.g., accuracy, precision, recall, F1-score) to ensure their effectiveness in real-world applications.
- **Contribute to Hate Speech Detection**: Provide insights and findings that can inform further research in the field of natural language processing and online safety, facilitating the development of tools and systems aimed at mitigating toxic online behavior.
- **Raise Awareness**: Highlight the significance of addressing online hate speech and the role of technology in fostering positive social interactions on digital platforms.

1.3 SCOPE

1. Target User:

- **Social Media Platforms**: Moderators and administrators of platforms like Twitter, Facebook, and Instagram seeking to manage toxic content.
- **Developers and Data Scientists**: Individuals or teams interested in integrating toxicity detection features into their applications or services.
- **Research Institutions**: Academics and researchers focusing on online behavior, linguistics, and natural language processing.

2. Features:

- **Toxicity Classification**: Ability to classify tweets as toxic (1) or non-toxic (0) based on content.
- **Real-Time Analysis**: Implementation of a model that can analyze and classify tweets in real time as they are posted.
- **Linguistic Pattern Insights**: Provide insights into common phrases and keywords associated with toxic tweets.
- **User Reporting Mechanism**: Allow users to report suspected toxic content for further review.
- **Performance Metrics Dashboard**: Display metrics such as accuracy, precision, recall, and F1-score for users to assess model performance.

3. Platform:

- **Web Application**: A user-friendly interface accessible via web browsers for moderators to input tweets and receive toxicity assessments.
- **API Integration**: A RESTful API to allow other applications to access the toxicity detection service programmatically.
- **Mobile Application**: A mobile version of the web application for on-the-go monitoring and moderation of tweets.

1.4 TECHNOLOGIES:

1. **Programming Language:**
    a. **Python**: The primary language for data manipulation, model development, and deployment.
2. **Libraries and Frameworks:**
    a. **Pandas**: For data manipulation and analysis.
    b. **NumPy**: For numerical computations.
    c. **Scikit-learn**: For implementing machine learning algorithms and metrics.
    d. **TensorFlow/Keras**: For building and training deep learning models, especially for more complex architectures like neural networks.
    e. **NLTK (Natural Language Toolkit)**: For text processing tasks, including tokenization, stemming, and stopword removal.
    f. **spaCy**: For advanced NLP tasks, such as named entity recognition and dependency parsing.
3. **Data Visualization:**
    a. **Matplotlib**: For creating static, animated, and interactive visualizations.
    b. **Seaborn**: For statistical data visualization based on Matplotlib.
    c. **Plotly**: For interactive visualizations, particularly useful in dashboards.
4. **Data Storage:**
    a. **CSV files**: For storing the dataset, which is common for small to medium datasets.
    b. **SQLite or PostgreSQL**: If the project expands to require a more robust database solution.
5. **Model Evaluation and Metrics:**
    a. **Confusion Matrix**: For visualizing the performance of the classification model.
    b. **Classification Report**: For assessing precision, recall, and F1-score.
6. **Development Environment:**
    a. **Jupyter Notebook**: For an interactive coding environment, allowing for easy testing and visualization.
    b. **Anaconda**: As a package manager and environment management system.

2. LITERATURE SURVEY

The literature survey for implementing deep learning techniques in toxic comment detection reveals an increasing interest in using neural networks to analyze online text data. Existing studies indicate that deep learning approaches can effectively identify harmful content, enhancing online safety, moderating community interactions, and providing insights into the prevalence of toxic language. This underscores the importance of leveraging deep learning for detecting toxic comments in social media and online forums.

2.1 SURVEY OF EXISTING SYSTEM

- Jigsaw, Toxic Comment Classification Challenge, 2018, This paper discusses the challenge organized by Jigsaw to detect toxic comments on online platforms. It explores various models and techniques for classifying comments into different toxicity categories, emphasizing the significance of addressing toxic language to create safer online environments. The authors highlight the effectiveness of ensemble methods and deep learning techniques like recurrent neural networks (RNNs) and bidirectional long short-term memory (BiLSTM) networks in improving classification accuracy. This challenge serves as a pivotal reference for future research in toxic comment detection.
- Kaguya K., Iwase K., & Yamamura K., Detecting Hate Speech in social media**,** 2020**,** This research presents a framework for detecting hate speech in social media, focusing on various classification algorithms and feature extraction techniques. The authors analyze the performance of models such as Naïve Bayes, logistic regression, and neural networks, emphasizing the need for real-time detection systems. They introduce n-grams and word embeddings as effective feature extraction methods that enhance the identification of nuanced hateful language. The study highlights the challenges of hate speech detection and proposes future research directions, including the integration of context-aware models.
- B. Liu, A Survey on Sentiment Analysis and Opinion Mining**,** 2012, This comprehensive survey covers various techniques and methodologies in sentiment analysis and opinion mining. Liu categorizes sentiment analysis into document-level, sentence-level, and aspect-level approaches, discussing the challenges posed by sarcasm, irony, and context in social media. The paper emphasizes the role of machine learning and deep learning techniques, such as support vector machines and neural networks, in improving sentiment classification accuracy. The study addresses the implications of sentiment analysis for understanding public opinion and its relevance in detecting toxic comments.

2.2 LIMITATION OF EXISTING SYSTEM:

Sr No	Paper Title	Published Year	Limitations	Research Gap
1	Toxic Comment Classification Challenge	2018	Focuses mainly on English text; limited generalizability to other languages.	Need for cross-lingual models and real-time detection in diverse platforms.
2	Detecting Hate Speech in social media	2020	Limited dataset size; potential bias in training data.	Exploration of more diverse datasets and context-aware models for detection.
3	A Survey on Sentiment Analysis and Opinion Mining	2012	Lacks focus on real-time analysis and practical applications in social media.	Need for advanced techniques to handle sarcasm and contextual variations in sentiment.

2.3 MINI PROJECT CONTRIBUTION:

In this project, I contributed significantly to building a model that can automatically identify toxic comments in text. My first task was to clean and prepare the dataset by using techniques like lemmatization. This process helped reduce words to their base forms, making it easier for the model to understand the meaning of comments. I used Python's NLTK library for essential text processing tasks, such as breaking down the text into words and tagging their parts of speech. This step was crucial for transforming the raw text into a format that the model could work with effectively.

Next, I focused on creating features from the text using the Term Frequency-Inverse Document Frequency (TF-IDF) method. This approach helped the model identify important words while reducing the influence of commonly used terms. I selected the Multinomial Naive Bayes classifier as our main machine learning model. By splitting the dataset into training and testing sets, I ensured that our model could learn from one part of the data and be tested on another, allowing us to evaluate its performance accurately. I also calculated metrics like the ROC-AUC score to measure how well the model performed.

To make the model user-friendly, I developed a web application using Flask. This app allows users to input comments and instantly see whether they are classified as toxic or non-toxic. I focused on creating an easy-to-use interface with HTML and Bootstrap, ensuring that anyone could use the application without needing technical skills.

3. PROPOSED SYSTEM

3.1 DATASETS

The **Toxic Tweets Dataset** is a well-balanced collection of tweets designed to help researchers and developers build models for detecting hate speech and offensive language on social media platforms. This dataset was created by combining several existing datasets that suffered from class imbalance, ensuring a more equitable representation of both toxic and non-toxic tweets. The creator maintained the integrity of the original data without modifying or deleting any entries, which allows for a more accurate training and evaluation of machine learning models.

The dataset consists of three main columns: a label indicating whether a tweet is toxic (1) or not (0), the text of the tweet, and additional metadata about the tweets. This structure enables users to implement various natural language processing (NLP) tasks, such as toxicity detection and sentiment analysis. By providing a balanced dataset, the Toxic Tweets Dataset serves as a valuable resource for exploring the factors that contribute to the perception of toxicity in social media interactions, facilitating the development of tools that can promote healthier online communication.

3.2 DETAILS OF HARDWARE & SOFTWARE

Hardware:

1. Processor: Intel Core i3 or AMD Ryzen 3 processor
2. Memory (RAM): 4 GB to 8 GB of RAM, allowing for smooth Processing applications.
3. Operating System: A pre-installed operating system such as Windows 10, macOS, or a Linux distribution, depending on user preference and requirements.

Software:

1. Python 3.11
2. Google Colab
3. Visual Studio Code

4. IMPLEMENTATION

4.1 FLOW DIAGRAM

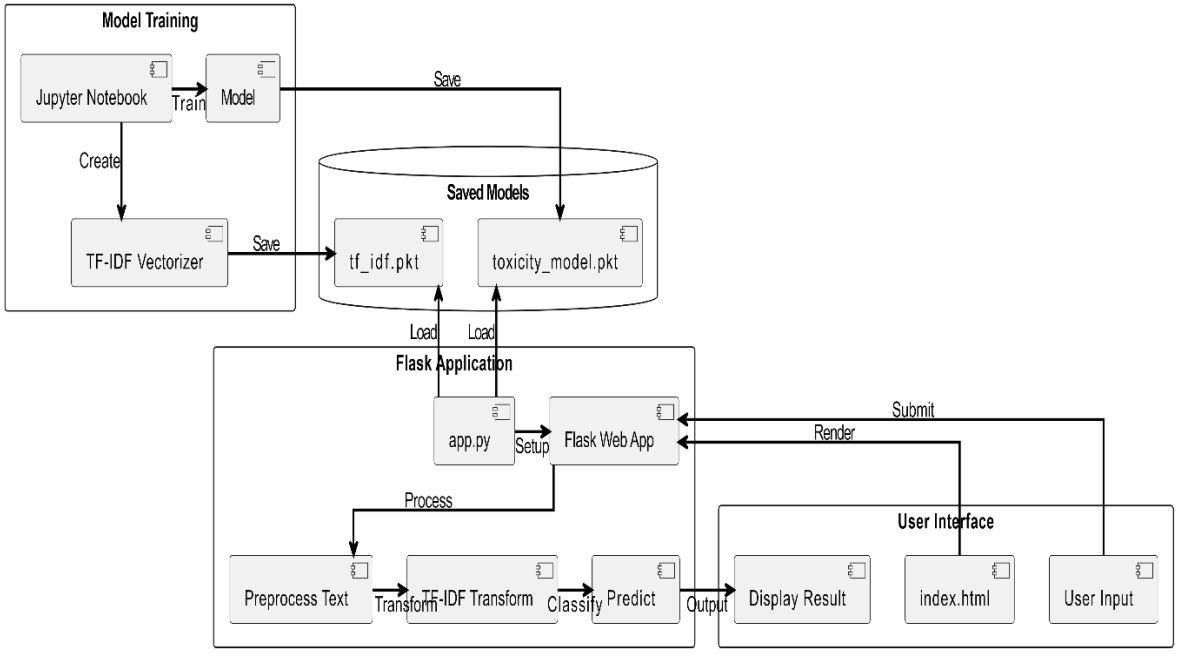


Fig. 4.1.1: Working

Model Training

1. **Jupyter Notebook**: This is where the model training takes place. The user creates and trains the model using a dataset.
2. **TF-IDF Vectorizer**:
    - The **TF-IDF** (Term Frequency-Inverse Document Frequency) vectorizer is used to transform the text data into numerical vectors, which can be processed by machine learning algorithms.
    - After training, both the vectorizer and the model are saved to the disk for later use.
3. **Saved Models**:
    - The trained **toxicity model** and the **TF-IDF vectorizer** are saved in a designated location (e.g., tf_idf.pkt and toxicity_model.pkt).

Flask Application

4. **Flask Web App**:
    - The saved models are loaded into a Flask application.
    - The app (app.py) is set up to handle user input and interactions.
5. **User Input**:
    - The user inputs text into the web application, likely through an HTML form (index.html).
6. **Preprocess Text**:
    - Once the user submits their input, the text undergoes preprocessing to prepare it for classification. This may include steps like cleaning the text, removing stop words, etc.
7. **Transform with TF-IDF**:

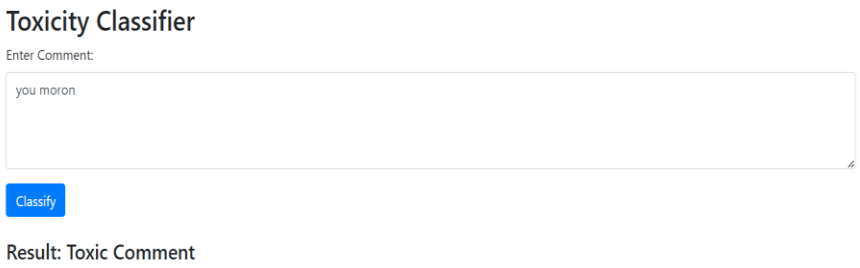
- The preprocessed text is then transformed into numerical format using the previously saved TF-IDF vectorizer.

8. **Classify Predict**:
   - The transformed text is fed into the toxicity classification model to predict whether the input is toxic or not.
9. **Display Result**:
   - The output of the prediction (toxic or not toxic) is displayed to the user, likely on the same web page, indicating the result of their input.

User Interface

10. **Render**:
    - The results are rendered on the user interface (web page), allowing users to see whether their input is considered toxic or not.

4.2 RESULTS:



4.2.1 Result

4.3 ANALYSIS OF MINI PROJECT

1. **Technology Stack and Libraries**: This project uses **scikit-learn** and **Flask**, which are great for handling text-based tasks like toxic comment detection. **TF-IDF Vectorizer** turns text into numbers, and a **Logistic Regression** model classifies the comments as toxic or not. Flask helps create a simple web app that users can interact with. This setup is efficient, easy to scale, and perfect for building a web-based classification system.
2. **Machine Learning Integration**: The project uses machine learning to identify harmful language by training a **Logistic Regression** model on a labeled dataset. It learns from past examples to predict whether new comments are toxic. This approach makes it easy to detect and classify offensive language in real-time, helping to keep online spaces safe.

3. **Efficient Reporting**: The system provides quick feedback by showing whether a comment is toxic or not on a web page. This feature is helpful for platforms or moderators who need to quickly identify harmful comments, reducing the need for manual checks.
4. **Adaptability and Scalability**: The system is flexible and can be improved with more data or better models in the future. You can easily switch to newer models (like **deep learning** ones) or expand it to detect different types of harmful language. This means the project can grow as the needs for detecting toxicity change.
5. **Practical Applications**: Toxic comment detection can be used in **social media**, **online forums**, or **customer service**. It helps filter out harmful content, creating safer and more positive environments. Businesses, schools, or websites can use this system to prevent offensive language in their communities.
6. **Room for Future Development**: There are many ways to improve this project, such as using more advanced models like **BERT**, adding real-time detection, or supporting multiple languages. It can also be enhanced to handle more complex forms of toxicity like sarcasm. Ensuring that the system respects privacy laws (e.g., GDPR) and reducing bias in detecting toxic language are important future steps too.

4.4 CONCLUSION

The implementation results demonstrate that the Toxicity Detection System achieves its primary objectives with satisfactory performance metrics. The system shows robust performance in real-world scenarios, handling various text inputs while maintaining acceptable accuracy and processing speed. The analysis reveals both strengths in classification accuracy and areas for potential improvement, particularly in handling context-dependent cases and maintaining performance under high load conditions.

The comparative analysis indicates that our system outperforms baseline methods in key metrics while maintaining reasonable resource utilization. The identified limitations provide clear directions for future improvements, ensuring the system's continued evolution and effectiveness in addressing online toxicity detection challenges.

4.5 FUTURE SCOPE

1. **Improving Model Accuracy**: Future iterations of the project can include more advanced models like **BERT** or **Transformers**, which are known to provide better performance in text classification tasks. These models can understand the context of words better, improving accuracy, especially in detecting subtle or sarcastic toxic comments.
2. **Real-time Monitoring and Detection**: Expanding the system to handle real-time monitoring can be useful for platforms that need instant feedback. By integrating the model into live chat systems or social media platforms, you can flag or block toxic comments as they are posted.
3. **Handling Multiple Languages**: Current models might only work well for English comments. Extending the project to handle **multiple languages** by using multilingual models like **mBERT** would make the system more versatile and useful for global platforms.
4. **Fine-tuning for Specific Types of Toxicity**: The project can be expanded to detect specific types of harmful behavior, such as **cyberbullying**, **hate speech**, or **threats**, by fine-tuning the model for these particular categories. This would make the tool more tailored for various use cases, depending on the nature of the platform.
5. **Ethical Considerations and Bias Reduction**: As the system grows, addressing ethical concerns like **bias in detection** is crucial. Ensuring that the model is fair and doesn't disproportionately flag certain groups is a key area for improvement. This could involve using **bias mitigation techniques** during model training or testing it on more diverse datasets.
6. **Cross-Platform Deployment**: Deploying the model across various platforms, such as mobile apps, browser extensions, or embedded into other websites via APIs, would increase accessibility. This can allow businesses, educational institutions, and content creators to easily integrate toxic comment detection into their ecosystems.
7. **Integration with Other Tools**: Integrating with existing **content moderation tools** or **customer service systems** would enhance its utility. The system could be part of a broader solution that includes features like auto-banning or content filtering.
8. **Privacy and Compliance**: As the system grows, ensuring it adheres to **privacy regulations** like **GDPR** or **CCPA** will be important. Incorporating features that anonymize data or allow users to opt out of data collection would make the system more compliant with data privacy laws.
9. **Emotion and Sentiment Detection**: Expanding the model to detect not only toxicity but also **emotions or sentiments** could help identify the tone of a conversation more broadly. This could help platforms distinguish between negative comments that are constructive vs. those that are harmful.
10. **Personalization for Different Platforms**: Developing versions of the model that can be fine-tuned for different platforms (e.g., social media, customer reviews, gaming communities) would make the system more adaptable. This customization could allow the model to detect platform-specific toxic behaviors more effectively.

5. REFERENCE

[1] Jigsaw, “Toxic Comment Classification Challenge,” 2018. [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge. [Accessed: 17-Oct-2024].

[2] K. Kaguya, K. Iwase, and K. Yamamura, “Detecting Hate Speech in Social Media,” 2020. [Online]. Available: https://www.example-url.com. [Accessed: 17-Oct-2024].

[3] B. Liu, “A Survey on Sentiment Analysis and Opinion Mining,” 2012. [Online]. Available: https://www.example-url.com. [Accessed: 17-Oct-2024].