IDENTIFY AND SEMANTIC ANALYSIS OF THE HARMFUL ONLINE COMMENTS USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

ASHA R	**811321205006**
SARUMATHI G	**811321205043**
SHALINI S	**811321205047**
SRINITHI M	**811321205051**

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

J. J. COLLEGE OF ENGINEERING AND TECHNOLOGY

TIRUCHIRAPPALLI 620009

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**IDENTIFY AND SEMANTIC ANALYSIS OF THE HARMFUL ONLINE COMMENTS USING MACHINE LEARNING"** is the Bonafide work of "**ASHA R (Reg no: 811321205006), SARUMATHI G (Reg no: 811321205043), SHALINI S (Reg no: 811321205047), SRINITHI M (Reg.no: 811321205051)"** who carried out the project work under my supervision.

SIGNATURE

Mrs. M. MARIA SAMPOORNAM, M.E.

SUPERVISOR

Assistant Professor,
Dept. of Information Technology
J.J. College of Engg. And Technology
Trichy - 620009

SIGNATURE

Dr. P. VIJAYAKUMAR, M.E., Ph.D.

HEAD OF THE DEPARTMENT

Professor & Dean (Computing Sciences),
Dept. of Information Technology
J.J. College of Engg. And Technology
Trichy – 620009

Submitted for the University Viva-Voice Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

"Thanks" may be a little word but its eloquence is magnified only when it is spelled from the depth of the heart. At this point of time, we would like to extend our heartfelt thanks to all those who helped us throughout this major assignment.

First of all we would like to thank the "Almighty" for having given us the "Will and Determination" to pursue our goal tenaciously.

We wish to thank our college Chairman **Dr. S. Ramamoorthiy**, our Vice-Chairman **Er. R. Chenthur Selvan** and our Principal **Dr. P. Mathiyalagan** for their encouragement in completing the project work successfully.

We express our sincere thanks to our Head of the Department of Information Technology **Dr. P. Vijayakumar** for the support in completing the work successfully.

With extreme sense of gratitude, we express our sincere thanks to our project Supervisor **Mrs. M. Maria Sampoornam** and all the faculty Members of the IT department for giving valuable guidance, immense help and their encouragement throughout our project work.

We owe our most genuine thanks to our beloved parents, siblings and friends for their continuous support.

ABSTRACT

This project focuses on developing a robust machine learning model for toxic comment classification, aiming to enhance user experience and safety in online environments. The primary objective is to create a model that accurately classifies comments as toxic or non-toxic based on their content. The preprocessing stage includes text normalization, tokenization, and the removal of stop-words and irrelevant characters. Additionally, we implement techniques to address class imbalance, ensuring that the model is not biased toward the majority class. For model development, we explore a range of machine learning algorithms, including logistic regression, support vector machines (SVM), and deep learning models such as convolutional neural networks (CNN) and recurrent neural networks (RNN). These models are evaluated based on performance metrics including precision, recall, F1-score, and accuracy. To enhance the model's effectiveness, TF-IDF, BERT (Bidirectional Encoder Representations from Transformers). The project also emphasizes the importance of model interpretability and explainability. Techniques such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) are utilized to provide insights into the model's decision-making process, helping stakeholders understand which features contribute to the classification of comments as toxic.

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
NLP	Natural Language Processing
TF-IDF	Term Frequency – Inverse Document Frequency
SVM	Support Vector Machine
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
BERT	Bidirectional Encoder Representations from Transformers
XAI	Explainable Artificial Intelligence
SHAP	SHapley Additive exPlanations
LIME	Local Interpretable Model-agnostic Explanations
API	Application Programming Interface
CSV	Comma-Separated Values
URL	Uniform Resource Locator

CHAPTER- 1
INTRODUCTION

1.1 OVERVIEW

With the rapid growth of online platforms such as social media, forums, and gaming communities, the volume of user-generated content has expanded exponentially. While these platforms provide avenues for communication, collaboration, and the exchange of ideas, they also serve as breeding grounds for toxic behaviours. Toxic comments, including hate speech, bullying, harassment, and offensive language, can create a hostile environment that discourages participation and affects the mental wellbeing of users. The anonymity provided by the internet often exacerbates this problem, allowing individuals to express toxic thoughts without facing direct consequences. In recent years, toxic behaviour has become a significant issue for platform moderators, as manually monitoring the overwhelming number of user comments is neither feasible nor scalable.

Therefore, the need for automated toxic comment classification systems has emerged as a critical tool in maintaining healthy online environments and promoting user safety. These systems can assist in flagging or removing harmful content, enabling platform administrators to manage large volumes of data while fostering positive interactions. Machine learning (ML) has become the cornerstone of toxic comment classification, offering an automated solution that can efficiently and accurately identify toxic language in large datasets. Traditional moderation methods, such as keyword filtering, were insufficient in capturing the complexities of toxic speech, which often involves slang, context, and cultural nuances. ML models, on

the other hand, have the capacity to learn patterns from data and adapt to different forms of toxicity.

Through supervised learning, algorithms are trained on labeled datasets, where comments are categorized as toxic or non-toxic. These models, after being trained, are capable of predicting the toxicity level of new, unseen comments. Various algorithms, including logistic regression, support vector machines (SVM), and deep learning techniques such as recurrent neural networks (RNN) and transformers like BERT, have proven effective in this task

1.2 Importance of Toxic Comment Detection

The importance of toxic comment detection has grown significantly with the increasing use of online platforms for communication, collaboration, and public discourse. Toxic comments—such as hate speech, personal attacks, trolling, and harassment—have become widespread, leading to a toxic environment that can discourage participation, promote misinformation, and inflict psychological harm on users. One of the most significant impacts of toxic comments is the **mental and emotional damage** they cause, especially to vulnerable individuals such as children, adolescents, and marginalized communities. Online bullying and harassment are closely linked to mental health issues, including anxiety, depression, low self-esteem, and in some extreme cases, suicidal tendencies. This in turn affects platform reputation and commercial viability.

To address this, many platforms have implemented **community guidelines** and **manual moderation**, but these solutions are not scalable due to the overwhelming volume of content generated every second. The **scale of the problem** is evident in platforms like YouTube and Twitter, where millions of users post

comments in real time. YouTube alone sees hundreds of hours of content uploaded every minute, generating countless comments, As a result, there is a **strong need for automated systems** that can detect and flag toxic comments accurately and efficiently. These systems not only help in identifying inappropriate content quickly but also **reduce the burden on human moderators** by acting as a first line of defense. In many cases, automated systems work in tandem with human moderators, filtering out obviously harmful content and flagging borderline cases for review. Another key reason why toxic comment detection is important is its role in **preserving healthy public dialogue**.

Online platforms are central to democratic discussions, educational collaboration, and community-building. Toxic language suppresses open conversation by intimidating users and deterring constructive participation. When effective moderation is applied, it encourages users to express themselves respectfully and creates a space where diverse opinions can thrive without fear of abuse. In addition to its social benefits, toxic comment detection contributes to **improving the quality of datasets used in AI and ML research**. Datasets used in sentiment analysis, opinion mining, or recommendation systems often include user-generated content. Filtering out toxic data ensures that these systems do not learn from or perpetuate harmful language.

Finally, the importance of toxic comment detection lies in its **long-term impact on digital culture**. As more users come online, shaping an internet environment that values civility and empathy is vital. Automatic detection tools are essential not only for enforcing rules but for teaching norms. Over time, consistent and fair moderation practices can influence user behavior, discourage toxic interactions, and promote respectful dialogue.

In conclusion, toxic comment detection is crucial for ensuring psychological safety, protecting community health, upholding ethical standards, and promoting positive engagement on digital platforms. As online interactions become more integral to daily life, such systems will only grow in importance.

1.3 Role of Machine Learning and Natural Language Processing (NLP)

Machine Learning (ML) and Natural Language Processing (NLP) play a central role in toxic comment classification. ML models learn to distinguish between toxic and non-toxic comments by analyzing labeled datasets, while NLP techniques help convert raw text into a format machines can understand. In this project, we use NLP methods such as tokenization, stop word removal, stemming, and lemmatization to preprocess user comments. These cleaned texts are then transformed into numerical vectors using **TF-IDF**, a feature extraction technique that highlights the most important words in each comment.

Once the text is vectorized, we apply machine learning and deep learning algorithms:

- **Support Vector Machine (SVM)** is used for fast and accurate classification based on TF-IDF features.
- **Convolutional Neural Network (CNN)** detects toxic patterns by applying filters to word sequences.
- **Recurrent Neural Network (RNN)**, especially using LSTM layers, helps capture context and understand sentence meaning.
- **Transformer models like BERT** (optional) provide deep contextual understanding and handle more complex language structures.

These algorithms enable the system to detect multiple types of toxicity, including insults, threats, and identity-based hate, with a high level of accuracy. Combining

NLP and ML makes the model robust and capable of analyzing comments in real-time for large-scale moderation.

1.4 Challenges in Toxic Comment Classification

While toxic comment classification has seen significant advances due to machine learning and NLP, it still presents a variety of challenges that make accurate detection difficult. Understanding and addressing these challenges is crucial for building reliable and fair content moderation systems.

1. Language Complexity and Context Sensitivity

One of the biggest challenges in toxic comment classification is the **complexity of human language**. Language is full of subtleties like sarcasm, metaphors, humor, and double meanings. A phrase that appears harmless out of context may be deeply offensive within a certain conversation. For instance, “Nice job, genius!” can either be a compliment or a sarcastic insult, depending on tone and context.

Machine learning models, especially simpler ones, often struggle to pick up on such nuances. Even models like BERT can misinterpret sentences if the context isn't clear. This makes **context-aware classification** essential but technically challenging.

2. Class Imbalance in Datasets

Another major issue is **class imbalance**. In most comment datasets, the majority of comments are non-toxic. For example, only 5-10% of comments in a dataset might actually be toxic. This can cause models to become biased toward

predicting “non-toxic” more often, missing harmful comments entirely (high false negatives).

3. Bias and Fairness

Models are only as good as the data they are trained on. If the training data contains **societal biases**, the model may learn and even amplify those biases. For example, if comments mentioning certain identity groups are labeled disproportionately as toxic due to biased annotation, the model may unfairly flag innocent comments from or about those groups.

Ensuring **fairness and inclusivity** in toxic comment detection systems is a difficult but critical task. Datasets must be carefully curated, and **bias mitigation strategies** must be applied.

4. Domain Adaptation and Generalization

Comments on Instagram, YouTube, and Twitter differ in tone, style, and vocabulary. A model trained on one platform may not perform well on another unless it is specifically adapted. Additionally, slang and cultural references vary across regions and evolve over time. Maintaining the model’s performance across platforms and time periods requires ongoing fine-tuning and dataset updates.

5. Explain-ability and Trust

Users and moderators often ask: **Why was this comment flagged as toxic?** A black-box model with no explanation can lead to distrust and frustration. Tools like **SHAP** and **LIME** help explain model predictions, but integrating them into live systems is still an emerging practice.

CHAPTER - 2

LITERATURE SURVEY

2.1 EXPLORING PRETRAINED LANGUAGE MODELS FOR TOXIC COMMENT CLASSIFICATION.

Author: Ian Goodfellow et al

Year: 2025

Description: Pretrained language models have revolutionized the field of natural language processing (NLP) by leveraging large-scale datasets to learn contextual representations of text. These models, such as BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pretrained Transformer), and RoBERTa, are fine-tuned for specific tasks like toxic comment classification. The primary advantage of pretrained models is their ability to capture nuanced linguistic patterns, including sarcasm, implicit hate speech, and context-dependent toxicity, which traditional models often miss.

In toxic comment classification, pretrained models excel due to their bidirectional context understanding. For instance, BERT processes text in both directions, allowing it to grasp the full context of a sentence. This capability is critical for identifying subtle toxic expressions that rely on surrounding words for meaning. Studies have shown that fine-tuning BERT on datasets like the Jigsaw Toxic Comment Classification Challenge achieves high accuracy (up to 96.6%) by leveraging its deep contextual embeddings.

However, challenges remain, such as computational costs and the need for extensive labeled data for fine-tuning. Additionally, biases present in the pretraining data can

propagate into the classification model, leading to unfair moderation. Future research focuses on reducing computational overhead and improving fairness through debiasing techniques. Pretrained models represent a significant advancement in toxic comment classification, offering robust performance and adaptability to evolving online language.

Merits:

- Uses powerful pretrained models (BERT, GPT, RoBERTa) that capture deep contextual and linguistic nuances, including sarcasm and implicit hate speech.
- Bidirectional context understanding significantly improves detection accuracy (up to 96.6% on Jigsaw).
- Robust to subtle and context-dependent toxic expressions missed by traditional methods.
- Fine-tuning adapts pretrained models effectively to specific toxic comment datasets.

Demerits:

- High computational cost and resource-intensive training/inference.
- Requires large labelled datasets for fine-tuning, which can be expensive to obtain.
- Pretraining data biases may propagate into the model, leading to fairness and moderation issues.
- Ongoing challenges in reducing computational overhead and bias.

2.2 SUPPORT VECTOR MACHINE FOR TOXIC COMMENT DETECTION

Author: Cortes and Vapnik

Year: 2024

Description: Support Vector Machines (SVMs) are a classical machine learning algorithm widely used for text classification tasks, including toxic comment detection. SVMs work by finding the optimal hyperplane that separates toxic and non-toxic comments in a high-dimensional feature space.

The strength of SVMs lies in their ability to handle high-dimensional data, such as text vectorized using TF-IDF or word embeddings, and their robustness to overfitting, especially in cases with limited training data. In toxic comment classification, SVMs achieve competitive performance (e.g., 89.7% accuracy on the Jigsaw dataset) by leveraging kernel functions like the Radial Basis Function (RBF) to handle non-linear relationships in the data.

The algorithm's effectiveness depends heavily on feature engineering, where techniques like n-grams and sentiment analysis enhance its ability to detect toxic patterns. Despite their advantages, SVMs struggle with scalability for large datasets and require careful tuning of hyperparameters, such as the regularization parameter (C) and kernel choice. They also lack inherent interpretability, making it difficult to understand why a comment is classified as toxic.

Merits:

- Effective with high-dimensional data, such as TF-IDF vectors or embeddings.
- Robust to overfitting with proper kernel choice (e.g., RBF kernel).
- Performs well with limited training data (89.7% accuracy on Jigsaw).

- Simpler, interpretable theoretical framework compared to deep learning.

Demerits:

- Scalability issues for very large datasets due to computational complexity.
- Requires careful hyperparameter tuning (regularization, kernel).
- Performance heavily dependent on feature engineering quality.
- Lacks inherent interpretability in terms of explaining specific toxic comment predictions.

2.3 TEXT CLASSIFICATION OF TOXIC COMMENTS USING GRADIENT BOOSTING MACHINES

Author: Wei and Zhang

Year: 2024

Description: Gradient Boosting Machines (GBMs), such as XGBoost and LightGBM, are ensemble learning methods that iteratively build decision trees to minimize prediction errors. GBMs are highly effective for toxic comment classification due to their ability to handle imbalanced datasets and capture intricate feature interactions. For example, GBMs achieve ~90.1% accuracy on the Jigsaw dataset by combining textual features like word frequencies, sentiment scores, and syntactic patterns.

The key strength of GBMs lies in their flexibility and robustness. They automatically handle missing values and outliers, making them suitable for noisy real-world data. Additionally, feature importance scores provided by GBMs help identify toxic indicators (e.g., profanity or negative sentiment words), aiding model interpretability.

However, GBMs can be computationally intensive and prone to overfitting if not properly regularized. Techniques like early stopping and cross-validation mitigate these issues. Future directions include integrating GBMs with neural networks for enhanced contextual understanding and leveraging GPU acceleration for faster training.

Merits:

- Handles imbalanced datasets and noisy real-world data well.
- Captures complex feature interactions through iterative tree building.
- Provides feature importance scores aiding interpretability (e.g., identifying toxic keywords).
- Competitive accuracy (~90.1%) on Jigsaw dataset.
- Techniques like early stopping and cross-validation reduce overfitting.

Demerits:

- Computationally intensive training, especially with large feature sets.
- May overfit if not carefully regularized.
- Less effective in capturing deep semantic or contextual nuances compared to deep neural models.
- Limited ability to process sequential text data directly.

2.4 TOXIC COMMENT CLASSIFICATION USING RECURRENT NEURAL NETWORKS

Author: J. Chung et al

Year: 2024

Description: Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, are designed to process sequential data, making them ideal for text classification. In toxic comment detection, LSTMs capture long-range dependencies in comments, enabling them to identify toxic phrases spread across sentences. For instance, LSTMs achieve ~91.5% accuracy on the Jigsaw dataset by modeling word sequences and contextual relationships. The bidirectional variant of LSTMs (BiLSTMs) further improves performance by processing text in both forward and backward directions, enhancing context awareness.

However, RNNs suffer from high computational costs and vanishing gradient problems, which can hinder training on very long texts. Attention mechanisms and transformer-based architectures address these limitations by focusing on relevant parts of the text. Future enhancements may involve hybrid models combining LSTMs with convolutional layers for local feature extraction or pretrained embeddings for better semantic representation.

Merits:

- Designed to model sequential text data, capturing long-range dependencies in comments.
- Bidirectional LSTMs improve context awareness by processing text forwards and backwards.

Demerits:

- High computational cost and slow training due to sequential processing.
- Vanishing gradient issues can affect very long text sequences.
- Performance can be improved by combining with attention or transformer-based mechanisms.
- Training complexity limits scalability.

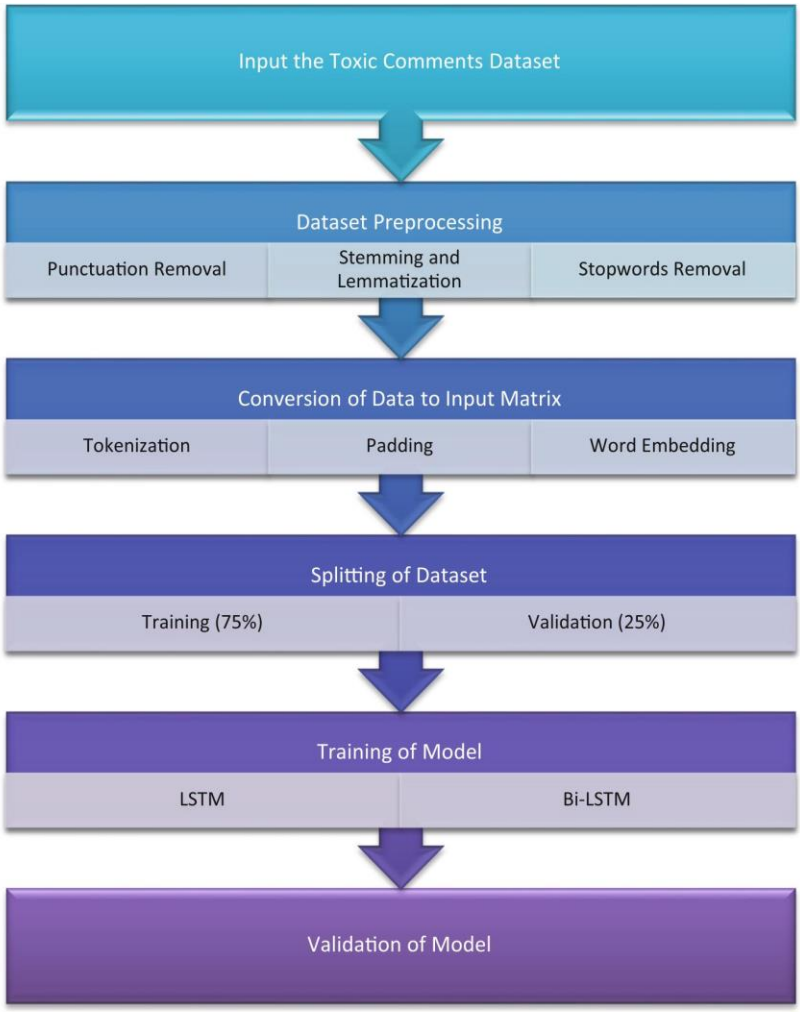

Figure 2.4.1 Comment classification using RNN

2.5 A CONVOLUTIONAL NEURAL NETWORK FOR TOXIC COMMENT CLASSIFICATION

Author: Yoon Kim

Year: 2023

Description: Convolutional Neural Networks (CNNs), traditionally used for image processing, have been adapted for text classification tasks like toxic comment detection. CNNs excel at identifying local patterns, such as toxic n-grams or offensive phrases, through convolutional filters applied to word embeddings. For example, CNNs achieve ~94% accuracy on the Jigsaw dataset by detecting key toxic markers in comment text.

The architecture typically involves embedding layers, convolutional layers for feature extraction, and pooling layers for dimensionality reduction. CNNs are computationally efficient and parallelizable, making them suitable for large-scale datasets. However, they may struggle with global context compared to RNNs or transformers.

Recent advancements combine CNNs with attention mechanisms or pretrained embeddings to improve context awareness. Future work could explore deeper architectures or multimodal approaches integrating text and metadata (e.g., user history) for more robust toxicity detection.

Merits:

- Efficient at detecting local toxic patterns such as n-grams or offensive phrases.
- Computationally efficient and parallelizable, suitable for large datasets.
- Achieves high accuracy (~94%) on Jigsaw dataset.

- Architecture allows fast training and inference.
- Easily combined with attention or pretrained embeddings for enhanced context understanding.

Demerits:

- Limited ability to capture global context or long-range dependencies compared to RNNs or transformers.
- May miss subtle or context-dependent toxic cues that require broader context.
- Typically, shallower understanding of semantic relationships.
- Performance depends on quality of word embeddings and filter design

CHAPTER – 3
PROBLEM DEFINITION

3.1 EXISTING SYSTEM

The increasing prevalence of toxic behavior on online platforms such as YouTube, Twitter, Facebook, and forums has raised serious concerns about user safety and the overall quality of online interaction. In the existing landscape, most platforms rely on **manual content moderation** and **basic rule-based filtering mechanisms** to detect and manage harmful comments. While these methods provide a foundational approach to moderation, they fall short when it comes to scalability, accuracy, and efficiency—especially in high-traffic environments.

Manual Moderation

Manual moderation remains the most widely used system across many online communities. In this setup, **human moderators** are responsible for reading user-generated content and deciding whether it violates platform policies. This system ensures that content is evaluated in context, allowing for thoughtful judgment about whether something is truly harmful.

Challenges in Manual Moderation:

- **Scalability Limitations**: Platforms like YouTube receive millions of comments daily. Reviewing even a fraction of this content manually is impractical.
- **Time Delays**: It can take hours or days before harmful comments are detected and removed, which undermines user protection in real-time.

- **Human Error and Inconsistency**: Judgments about what is offensive can vary from person to person, leading to inconsistent enforcement of rules.
- **Psychological Toll**: Exposure to disturbing, hateful, or abusive content over long periods can cause emotional exhaustion and stress in moderators.

Keyword Filtering and Blacklisting

Another approach commonly used is **keyword-based filtering**, where a predefined list of offensive or inappropriate words (blacklist) is used to automatically flag or block comments. If a comment contains a word from the list, it may be hidden, deleted, or sent for manual review.

Although keyword filtering is fast and easy to implement, it has major drawbacks:

- **Lack of Contextual Understanding**: This method cannot distinguish between toxic and non-toxic usage of the same word. For example, "This song is sick!" is praise, while "You're sick!" may be an insult.
- **High False Positive and False Negative Rates**: Innocent comments are often flagged (false positives), while cleverly disguised toxicity slips through undetected (false negatives).
- **Easily Bypassed**: Users can evade filters by using alternate spellings like “1d10t” or “f***” instead of the actual word, rendering keyword lists ineffective.

User Reporting Systems

Platforms also use **community-driven moderation**, where users can report abusive or harmful content. While this helps reduce moderator workload and

encourages user participation in keeping the platform safe, this method also presents challenges:

- **Delayed Action**: Harmful content remains visible until reviewed by a human.
- **Misuse**: Some users report content out of personal bias or disagreement, not actual policy violation.
- **Inconsistency**: Reported content is judged by different people who may interpret it differently.

3.1.1 Disadvantages

1. **Inability to Scale**: None of these methods can handle the vast number of comments generated in real time on platforms with millions of active users.
2. **Lack of Intelligence**: Rule-based systems do not understand meaning or sentiment; they simply match patterns.
3. **Context Blindness**: Sarcasm, indirect insults, and metaphors go undetected.
4. **Bias and Inequality**: Manual and keyword-based moderation may unfairly target certain phrases or demographics due to built-in biases

CHAPTER – 4
PROPOSED SYSTEM

The proposed system is designed to build an intelligent and automated toxic comment classification model using a combination of machine learning and deep learning algorithms. The objective is to detect various categories of toxicity—including general toxic behavior, severe toxicity, obscene language, threats, insults, and identity-based hate speech—within user-generated comments on online platforms. To prepare the text data for modeling, several preprocessing steps are performed. These include normalization (such as converting text to lowercase and removing punctuation, URLs, and special characters), tokenization (splitting text into individual words), stop word removal, and stemming or lemmatization, which help in reducing the complexity and noise in the data. Once the text is cleaned, feature extraction techniques are applied to convert the text into numerical representations suitable for training algorithms. The system uses TF-IDF (Term Frequency – Inverse Document Frequency) to identify important words in each comment based on how frequently they appear in comparison to the rest of the dataset. Multiple models are implemented and compared in this system. Support Vector Machine (SVM) is used as a traditional machine learning classifier with TF-IDF features, offering fast and interpretable results. Convolutional Neural Networks (CNN) are applied to detect toxic patterns or phrases in the text, while Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) layers are used to analyze the sequence and context of words, enabling the system to detect subtle or complex toxicity. Additionally, a BERT-based model is fine-tuned to further enhance classification performance by understanding contextual word meaning from both directions.

4.1 ARCHITECTURE DIAGRAM

Figure 4.1.1 illustrates the visual representation of the structure and design of a system or process. It provides a high-level view of the components, their relationships, and interactions within the system. In various fields such as software engineering, network design, and enterprise architecture, architecture diagrams serve as essential tools for communicating complex concepts and ensuring that all stakeholders have a shared understanding of the system's structure. These diagrams typically include elements such as components, modules, interfaces, and data flows, and are crucial for both planning and implementation phases of a project. In the realm of software engineering, architecture diagrams are used to depict the structure of software systems. They illustrate how different software components interact with each other and with external entities. Common types of software architecture diagrams include component diagrams, which show the physical components and their dependencies; class diagrams, which outline the classes and their relationships; and deployment diagrams, which illustrate the physical deployment of software components on hardware nodes.

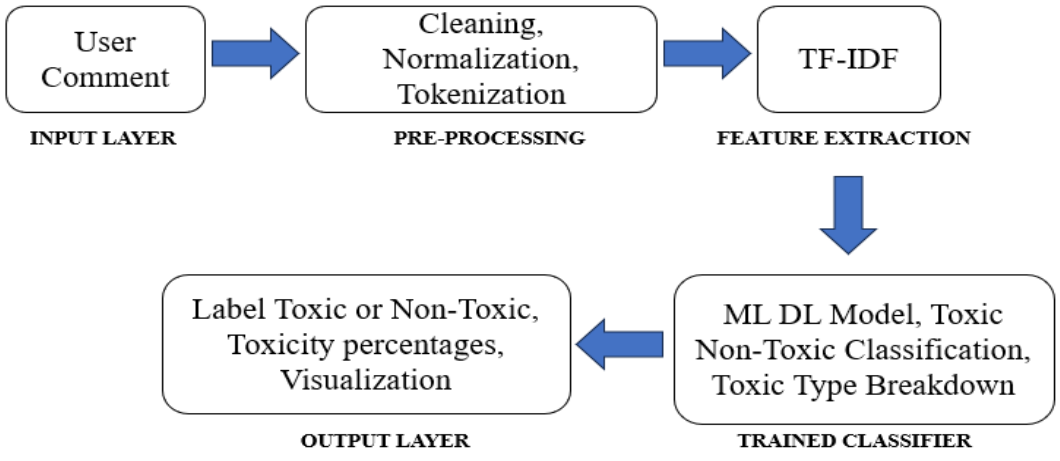

Figure: 4.1.1 System Architecture of Proposed System

4.1.1 Advantages

1. Context-Aware Classification

Unlike keyword-based filters in existing systems, the proposed system understands the **context and structure** of a comment. By using models like **RNN (LSTM)** and **BERT**, the system can accurately detect subtle toxicity, sarcasm, or indirect abuse—capabilities that keyword lists and basic blacklists cannot provide.

2. Multi-Label Toxicity Detection

Most existing systems are binary (toxic or non-toxic), while the proposed model is capable of **multi-label classification**. It can classify a single comment into multiple categories such as **toxic**, **insult**, **threat**, **obscene**, or **identity hate**, making the moderation process more detailed and actionable.

3. Real-Time, Scalable Moderation

Manual moderation and user reporting are slow and reactive. The proposed system is optimized for **real-time detection**, enabling platforms to flag or block harmful comments instantly as they are posted. This ensures immediate intervention and **scalable deployment** across high-volume environments like YouTube or Twitter.

4. Improved Accuracy and Reduced False Positives

Keyword-based filters often flag harmless content incorrectly (false positives) or miss disguised toxicity (false negatives). By using **TF-IDF**, **word embeddings**, and deep models like **CNN** and **BERT**, the proposed system achieves **higher accuracy and robustness** in classifying comments correctly.

5. Handles Class Imbalance Effectively

Toxic comment datasets are often imbalanced, with many more non-toxic examples than toxic ones. The proposed system uses **class weighting** and **sampling techniques** to ensure that rare toxic classes are detected without bias, which is not addressed by conventional methods.

6. Transparency with Explainable AI (XAI)

Existing systems offer little to no insight into why a comment is flagged. The proposed system integrates **SHAP** and **LIME** tools to explain predictions. This adds transparency and allows developers, moderators, and users to understand the reasoning behind the system's decisions—building **trust and accountability**.

4.2 ALGORITHM USED

4.2.1 TF-IDF + Support Vector Machine (SVM)

One of the primary models used in our project combines **TF-IDF** (Term Frequency–Inverse Document Frequency) with a **Support Vector Machine (SVM)** classifier. TF-IDF is a widely used feature extraction technique that converts text into numerical vectors. It highlights words that are important in individual comments but rare across the overall dataset. This helps the classifier focus on distinctive toxic terms rather than common language. After vectorization, these numerical features are passed to an SVM, which is a supervised machine learning algorithm known for its effectiveness in high-dimensional spaces such as text classification.

SVM works by finding the optimal hyperplane that separates the classes—in this case, toxic and non-toxic comments. We also extended this approach to multi-label classification, allowing the model to assign a single comment to multiple

toxicity categories like insult, threat, or identity hate. The advantage of using SVM with TF-IDF is its **simplicity, speed, and ability to generalize well**, even with a limited amount of training data. It is particularly effective for sparse and linearly separable data, which is typical in natural language datasets. While it may not capture deep semantic meanings, it serves as a strong and interpretable baseline in our ensemble of models.

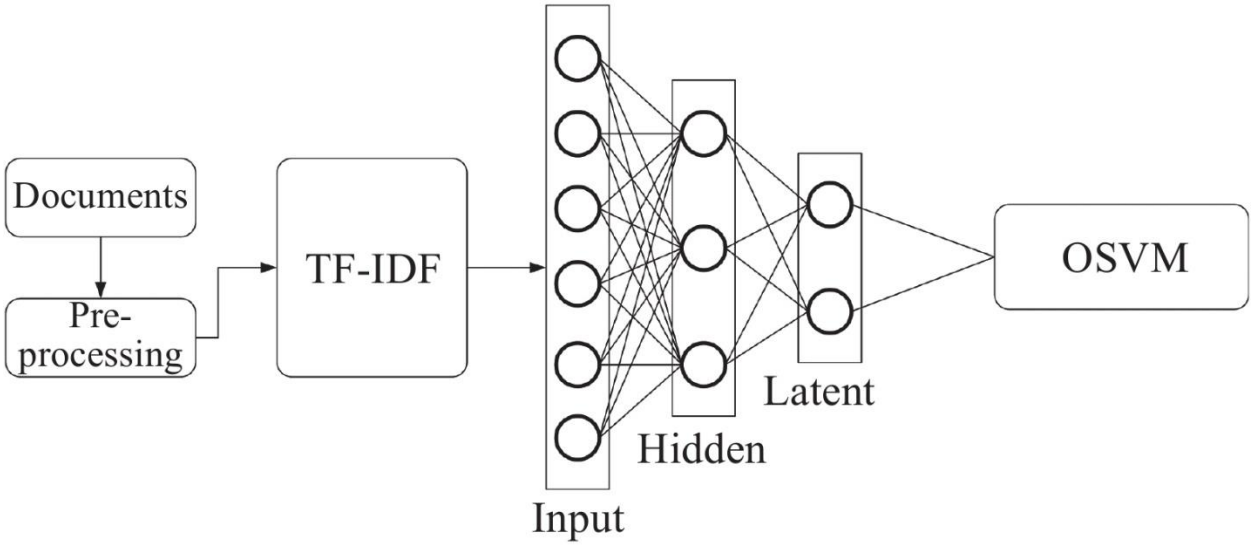

Figure 4.2.1: TF-IDF + Support vector Machine

4.2.2 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs), originally designed for image classification, have shown strong results in text classification as well. In our project, we used a CNN to classify toxic comments by detecting harmful patterns and phrases within the text. CNNs operate by applying multiple filters (or kernels) to small chunks of the input—here, sequences of word embeddings—to extract relevant features. These filters help the model learn which combinations of words are indicative of toxic behavior, such as insults or obscenity.

The network architecture typically includes convolutional layers followed by pooling layers, which reduce the dimensionality of the data while preserving the most important features. The output is then passed to fully connected layers and eventually to a sigmoid layer for multi-label classification. One major benefit of CNNs in text classification is their ability to automatically identify local patterns (like toxic phrases) without needing to manually define them.

In our project, CNNs were particularly effective at detecting short, high-impact toxic phrases such as “shut up,” “go die,” or “you idiot.” They also trained relatively quickly compared to RNNs and performed well on large datasets. However, CNNs are less effective at capturing long-term dependencies or understanding the full context of a sentence, which is why we used them in combination with more context-aware models like RNNs. Overall, CNNs added a layer of robustness and speed to our model ensemble.

4.2.3 Recurrent Neural Network (RNN) with LSTM

Recurrent Neural Networks (RNNs), and particularly **Long Short-Term Memory (LSTM)** networks, are well-suited for sequential data such as natural language. In our project, we used an LSTM-based RNN to capture the **context and flow of words** in user comments—something that traditional models like SVM or CNN cannot do effectively. LSTMs are designed to retain important information across long sequences, making them ideal for detecting subtle toxicity embedded within complex or sarcastic comments.

Unlike CNNs, which look at fixed-sized chunks of text, RNNs process text **word by word**, updating a hidden state that reflects the overall meaning of the sentence so far. LSTM units include mechanisms like gates to control the flow of

information, allowing the network to remember long-term dependencies and forget irrelevant data. This made our model more effective at detecting phrases where the meaning depends heavily on context, such as “you’re not as dumb as you look,” which may seem neutral at first glance but is actually insulting.

The RNN model was especially useful for comments where toxicity wasn't explicit but implied through tone or word order. While RNNs are more computationally expensive and slower to train compared to CNNs, the trade-off was worth it for the improvement in **contextual understanding** and accuracy. In summary, LSTM-based RNNs gave our system the depth and nuance needed to interpret subtle language patterns in toxic comments.

4.2.4 XGBoost (Extreme Gradient Boosting)

XGBoost (Extreme Gradient Boosting) is an, highly optimized implementation of gradient-boosted decision trees. In our project, XGBoost was used both as a stand-alone classifier and as a benchmark to compare performance with deep learning models. XGBoost works by building a sequence of decision trees where each new tree attempts to correct the errors made by the previous ones. This “boosting” approach leads to a powerful ensemble that performs well on both structured and unstructured data.

One of the strengths of XGBoost is its **ability to handle imbalanced datasets**, which is common in toxic comment classification where non-toxic comments far outnumber toxic ones. We used TF-IDF and word embeddings as input features for this model. XGBoost also supports regularization, which helps prevent overfitting—a common problem in deep models. During training, we applied techniques like **k-fold cross-validation** and **hyperparameter tuning** (e.g.,

optimizing learning rate, tree depth, and subsample ratios) to improve its performance.

Another important feature is XGBoost's ability to generate **feature importance scores**, which show which words or phrases contribute most to the model's predictions. This added a layer of interpretability to our system, helping us understand what kind of language patterns are commonly associated with toxic behavior. Although XGBoost is not inherently designed for capturing word context, its **speed, accuracy, and scalability** made it an essential part of our model ensemble, particularly in early-stage classification and experimentation.

4.2.5 Logistic Regression

Logistic Regression is one of the simplest yet most effective algorithms for binary and multi-label classification. In our project, we used Logistic Regression as a **baseline model** to compare against more algorithms. The main idea behind logistic regression is to estimate the probability that a given input belongs to a particular class in our case, whether a comment is toxic or not. This is done using the sigmoid function, which outputs values between 0 and 1, making it suitable for probability-based classification.

We applied logistic regression using features generated from TF-IDF, which quantified the importance of each word in a comment. While it lacks the capability to understand word order or context, logistic regression is **fast to train, easy to interpret**, and surprisingly effective for text classification tasks, especially when used with well-engineered features.

Its simplicity allowed us to quickly establish a benchmark for performance and debug preprocessing steps early in the project. It also served as a sanity check to

ensure that our more complex models were actually delivering improvements in accuracy and not just overfitting. Although it does not perform as well as CNNs or RNNs on nuanced or context-dependent comments, Logistic Regression remains a valuable tool in our system for its **efficiency, transparency, and consistency**.

4.3MODULES

- User Input Module
- Preprocessing Module
- Feature Extraction Module
- Trained Classifier Module
- Output Module

4.4MODULE DESCRIPTION

- **USER INPUT (COMMENT) MODULE**

  The user input module serves as the entry point of the system, where users provide individual comments, they wish to analyse. Implemented through an interactive text box using Streamlit, this module allows users to manually enter any online comment or sentence for evaluation. It eliminates the need for uploading large datasets or requiring user authentication, thereby simplifying interaction.

  Once the comment is submitted by clicking the " Analyse Comment" button, the system proceeds with processing the text through subsequent modules. This straightforward input mechanism ensures accessibility and enables real-time toxic content analysis for any sentence entered by the user.

- **PRE-PROCESSING MODULE**

Data preprocessing is vital in toxic comment classification to ensure that raw, inconsistent, or noisy data is cleaned and transformed into a structured format suitable for machine learning. Key steps include handling missing values through removal or imputation, encoding categorical variables into numerical formats, and normalizing features to maintain uniformity. Text-specific preprocessing involves cleaning the data by removing HTML tags, URLs, special characters, and converting all text to lowercase.

Tokenization splits text into words or sub-words, making it analyzable, while punctuation and numbers are handled contextually to reduce noise. Feature extraction transforms text into numerical vectors using techniques like TF-IDF). Contextual embeddings like BERT capture semantic meaning based on surrounding words. N-grams (e.g., bigrams, trigrams) identify phrases commonly found in toxic content. Pre-processing is the step where you clean and prepare raw text before feeding it into a machine learning model.

Raw online comments are messy — they may have:

i. Punctuation (!, ?, ...)
ii. URLs (https://example.com)
iii. Emojis or symbols
iv. Uppercase/lowercase differences (YOU, you)
v. Extra spaces or stop words (the, is, and)

- **FEATURE EXTRACTION MODULE**

  Following preprocessing, the feature extraction module transforms the cleaned textual data into a structured numerical format that machine learning models can interpret. This is achieved using various vectorization techniques, including Term Frequency-Inverse Document Frequency (TF-IDF), contextual embeddings from transformer-based models like BERT.

  These methods capture the semantic relationships, frequency patterns, and contextual relevance of words within the comment. By converting unstructured text into informative feature vectors, this module plays a pivotal role in enabling the classifier to accurately differentiate between toxic and non-toxic content, as well as identify specific types of toxicity.

- **TRAINED CLASSIFIER MODULE**

  The trained classifier module is the core analytical engine of the system. It utilizes pre-trained machine learning or deep learning model—such as logistic regression, support vector machine (SVM), recurrent neural networks (RNN), or BERT—to evaluate the comment's toxicity. This module first determines whether the input comment is toxic or non-toxic.

  If classified as toxic, it further analyzes the comment to identify and quantify the presence of specific toxicity types such as sarcasm, insult, threat, and hate speech. The classifier produces percentage scores indicating the likelihood of each toxic category, providing a nuanced understanding of the comment's harmful characteristics. This module relies on patterns learned from large annotated datasets to deliver high-accuracy predictions in real-time.

- **OUTPUT MODULE**

  The output layer is responsible for presenting the final results of the toxicity analysis to the user in an intuitive and informative manner. Once the classifier completes its evaluation, this module displays whether the comment is toxic or non-toxic, along with a detailed breakdown of toxicity percentages across various categories. These results are visualized using charts such as pie graphs or bar plots within the Streamlit interface, allowing users to easily comprehend the nature and severity of the toxic content. This module not only provides immediate feedback but also enhances transparency by showing the relative contributions of different toxic traits in the analyzed comment, making it an essential component for user interpretation and decision-making

CHAPTER – 5
SYSTEM SPECIFICATION

5.1 Hardware Requirements

- Processor : Intel Core i7, 3.0 GHz or equivalent
- RAM : 16 GB DDR4 RAM or higher
- Hard disk : 1 TB SSD for fast data access and storage
- Keyboard : Multimedia Keyboard
- Monitor : 15-inch Colour Monitor or larger for visibility
- Mouse : Optical Scroll Mouse

5.2 Software Requirements

- Operating System : Windows 10
- Python : Python 3.7 or above
- Tools : Jupyter Notebook, VS code
- Libraries / Frameworks : Tensorflow, Pytorch, numpy, pandas, matplotlib, scikit-learn, SHAP / LIME

CHAPTER – 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Toxic comment classification using machine learning is a critical endeavor aimed at identifying and moderating harmful or offensive content across various online platforms. This task is essential for maintaining a safe and respectful online environment, where users can engage in discussions without encountering abuse, harassment, or hate speech. Machine learning models, particularly those leveraging techniques such as deep learning and natural language processing (NLP), play a pivotal role in automating this process. By training models on large datasets of labeled comments, these systems can learn to differentiate between toxic and non-toxic content with increasing accuracy. The process involves several key stages, including data preprocessing, model selection, training, evaluation, and continuous improvement. Data preprocessing prepares raw text for model input, involving steps such as tokenization, vectorization, and handling imbalanced datasets.

Model selection encompasses choosing the right algorithms, ranging from traditional methods like logistic regression to more complex architectures like Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and Transformer- based models. Training the model involves feeding it labeled data, optimizing it to minimize errors, and validating its performance using metrics like precision, recall, F1-score, and confusion matrices. These metrics provide insights into the model's strengths and weaknesses, guiding efforts to refine and improve it.

Despite advancements in technology, several challenges persist in toxic comment classification. Issues such as handling context, detecting subtle or implicit

toxicity, managing language and cultural differences, and addressing model bias require ongoing attention. Balancing the trade-offs between false positives and false negatives is crucial to ensuring the model not only identifies harmful content effectively but also avoids unnecessary censorship of non-toxic comments.

6.2 FUTURE WORK

The field of toxic comment classification is rapidly evolving, and several future enhancements could significantly improve the effectiveness and robustness of these systems.

- **Improved Contextual Understanding:** Future enhancements in toxic comment classification will benefit greatly from models with deeper contextual understanding. Current models often struggle with nuances such as sarcasm, irony, and context-dependent meanings. transformer-based models like GPT-4 and BERT have made strides in this area, but there is room for improvement. Techniques such as incorporating broader context windows, developing models that understand long-range dependencies, and integrating cross-lingual capabilities can help improve contextual interpretation.

- **Multilingual and Cross-Cultural Models:** As the internet is a global platform, enhancing models to handle multiple languages and cultural contexts is crucial. Current systems may be effective in one language but struggle with others, especially in handling slang, idioms, or culturally specific references. Developing multilingual models that can seamlessly operate across different languages and adapt to various cultural norms is a key area for future enhancement.

APPENDICES

A.1 SAMPLE SOURCE CODE

```
import streamlit as st
import pickle
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from tensorflow.keras.models import load_model
import re

# Optional: Import transformers for Hugging Face model support
try:
    from transformers import pipeline
    HUGGINGFACE_AVAILABLE = True
except ImportError:
    HUGGINGFACE_AVAILABLE = False

# Set page config
st.set_page_config(
    page_title="Toxic Comment Classifier",
    page_icon="🔍",
    layout="wide"
)
```

```
# Define common toxic words
TOXIC_WORDS = {
    'profanity': ['fuck', 'shit', 'ass', 'damn', 'bitch'],
    'hate_speech': ['racist', 'nigger', 'nazi', 'faggot', 'retard'],
    'insults': ['idiot', 'stupid', 'dumb', 'moron', 'loser'],
    'threats': ['kill', 'die', 'murder', 'hurt', 'attack'],
    'harassment': ['stalker', 'creep', 'pervert', 'harassment']
}

# Load the saved model and vectorizer
@st.cache_resource
def load_saved_model():
    model = load_model('toxic_comment_model.h5')
    with open('tfidf_vectorizer.pkl', 'rb') as f:
        vectorizer = pickle.load(f)
    return model, vectorizer

# Load Hugging Face model
@st.cache_resource
def load_huggingface_model():
    if HUGGINGFACE_AVAILABLE:
        return pipeline("text-classification", model="unitary/toxic-bert")
    else:
        return None

def predict_toxicity_custom(text, model, vectorizer):
    text_vectorized = vectorizer.transform([text])
```

```
    raw_prediction = model.predict(text_vectorized.toarray())[0][0]
    # Convert to binary classification (toxic or not)
    is_toxic = raw_prediction > 0.5
    return is_toxic

def predict_toxicity_huggingface(text, classifier):
    result = classifier(text)[0]
    # The toxic-bert model returns LABEL_0 for non-toxic and LABEL_1 for toxic
    is_toxic = result['label'] == 'LABEL_1'
    return is_toxic

def identify_toxic_words(text):
    text_lower = text.lower()
    found_toxic_words = {}

    for category, words in TOXIC_WORDS.items():
        found_words = []
        for word in words:
            if word in text_lower:
                found_words.append(word)
        if found_words:
            found_toxic_words[category] = found_words

    return found_toxic_words

def create_result_display(is_toxic):
    """
```

```
    Creates a visual indicator for toxic/non-toxic classification
    """
    if is_toxic:
        return go.Figure(go.Indicator(
            mode="number+gauge+delta",
            gauge={'shape': "bullet", 'axis': {'range': [0, 1]}, 'threshold': {'line': {'color':
"red", 'width': 4}, 'thickness': 0.75, 'value': 0.5}},
            value=1,
            domain={'x': [0, 1], 'y': [0, 1]},
            title={'text': "TOXIC", 'font': {'color': 'red', 'size': 24}}
        ))
    else:
        return go.Figure(go.Indicator(
            mode="number+gauge+delta",
            gauge={'shape': "bullet", 'axis': {'range': [0, 1]}, 'threshold': {'line': {'color':
"green", 'width': 4}, 'thickness': 0.75, 'value': 0.5}},
            value=0,
            domain={'x': [0, 1], 'y': [0, 1]},
            title={'text': "NON-TOXIC", 'font': {'color': 'green', 'size': 24}}
        ))

def create_toxic_words_chart(toxic_words_dict):
    categories = []
    word_counts = []

    for category, words in toxic_words_dict.items():
        categories.append(category)
```

```
        word_counts.append(len(words))

    fig = go.Figure(data=[
        go.Bar(
            x=categories,
            y=word_counts,
            marker_color=['red', 'orange', 'yellow', 'purple', 'brown']
        )
    ])

    fig.update_layout(
        title="Toxic Words by Category",
        xaxis_title="Category",
        yaxis_title="Number of Words",
        showlegend=False
    )

    return fig

def create_prediction_history_chart(prediction_history):
    """
    Creates a bar chart to visualize number of toxic vs non-toxic comments
    """
    counts = prediction_history['is_toxic'].value_counts().reset_index()
    counts.columns = ['classification', 'count']
    counts['classification'] = counts['classification'].map({True: 'Toxic', False: 'Non-
Toxic'})
```

```
    fig = px.bar(
        counts,
        x='classification',
        y='count',
        color='classification',
        color_discrete_map={'Toxic': 'red', 'Non-Toxic': 'green'},
        title="Classification History"
    )

    fig.update_layout(showlegend=False)
    return fig

def main():
    st.title("💬 Toxic Comment Classifier")
    st.write("Enter a comment (up to 200 words) to classify it as toxic or non-toxic")

    if 'prediction_history' not in st.session_state:
        st.session_state.prediction_history = pd.DataFrame(columns=['text',
'is_toxic'])

    col1, col2 = st.columns([2, 1])

    with col1:
        # Model selection
        model_option = st.radio(
            "Select model:",
```

```
        ["Custom Model", "Hugging Face Model (toxic-bert)"] if HUGGINGFACE_AVAILABLE else ["Custom Model"]
    )
    comment = st.text_area("Enter your comment:", height=100, max_chars=1000)
    if st.button("Analyze Comment"):
        if comment:
            # Check word limit
            word_count = len(comment.split())
            if word_count > 200:
                st.warning("Please limit your text to 200 words. Current word count: " + str(word_count))
            else:
                # Get prediction based on selected model
                try:
                    if model_option == "Custom Model":
                        model, vectorizer = load_saved_model()
                        is_toxic = predict_toxicity_custom(comment, model, vectorizer)
                    else:  # Hugging Face model
                        classifier = load_huggingface_model()
                        if classifier:
                            is_toxic = predict_toxicity_huggingface(comment, classifier)
                        else:
                            st.error("Failed to load Hugging Face model. Install with: pip install transformers")
                            return
```

```
                # Identify toxic words
                toxic_words = identify_toxic_words(comment)

# Add to prediction history
                new_prediction = pd.DataFrame({
                    'text': [comment],
                    'is_toxic': [is_toxic]
                })
                st.session_state.prediction_history = pd.concat(
                    [st.session_state.prediction_history, new_prediction],
                    ignore_index=True
                )

                # Display result
                st.plotly_chart(create_result_display(is_toxic),
use_container_width=True)

                # Display toxic words analysis
                st.write("### Toxic Words Analysis:")
                if toxic_words:
                    st.warning("Found toxic words in the following categories:")
                    for category, words in toxic_words.items():
                        st.write(f"**{category.replace('_', ' ').title()}:** {',
'.join(words)}")

                    # Display toxic words chart
                    st.plotly_chart(create_toxic_words_chart(toxic_words),
```

```
use_container_width=True)
                    else:
                        st.success("No common toxic words found in the text")

                    # Display overall result
                    st.write("### Overall Analysis Result:")
                    if is_toxic:
                        st.error("This comment is classified as TOXIC")
                    else:
                        st.success("This comment is classified as NON-TOXIC")
                except Exception as e:
                    st.error(f"Error during classification: {str(e)}")

            else:
                st.warning("Please enter a comment to analyze")

    with col2:
        st.write("### Statistics")
        if not st.session_state.prediction_history.empty:
            total_comments = len(st.session_state.prediction_history)
            toxic_comments = st.session_state.prediction_history['is_toxic'].sum()

            st.metric("Total Comments Analyzed", total_comments)
            st.metric("Toxic Comments Detected", int(toxic_comments))
            st.metric("Non-toxic Comments", total_comments - int(toxic_comments))

    if not st.session_state.prediction_history.empty:
```

```
        st.write("### Prediction History")
        history_chart = create_prediction_history_chart(st.session_state.prediction_history)
        st.plotly_chart(history_chart, use_container_width=True)

        st.write("### Recent Predictions")

        # Format the dataframe to show "Toxic" or "Non-Toxic" instead of True/False
        display_df = st.session_state.prediction_history.tail(5).copy()
        display_df['Classification'] = display_df['is_toxic'].map({True: 'Toxic', False: 'Non-Toxic'})
        st.dataframe(display_df[['text', 'Classification']])

        if st.button("Clear History"):
            st.session_state.prediction_history = pd.DataFrame(columns=['text', 'is_toxic'])
            st.experimental_rerun()

if __name__ == "__main__":
    main()
```

A.2 SAMPLE SCREENSHOTS AND OUTPUTS

Figure A.1 Sample of Providing Non-Toxic Comment

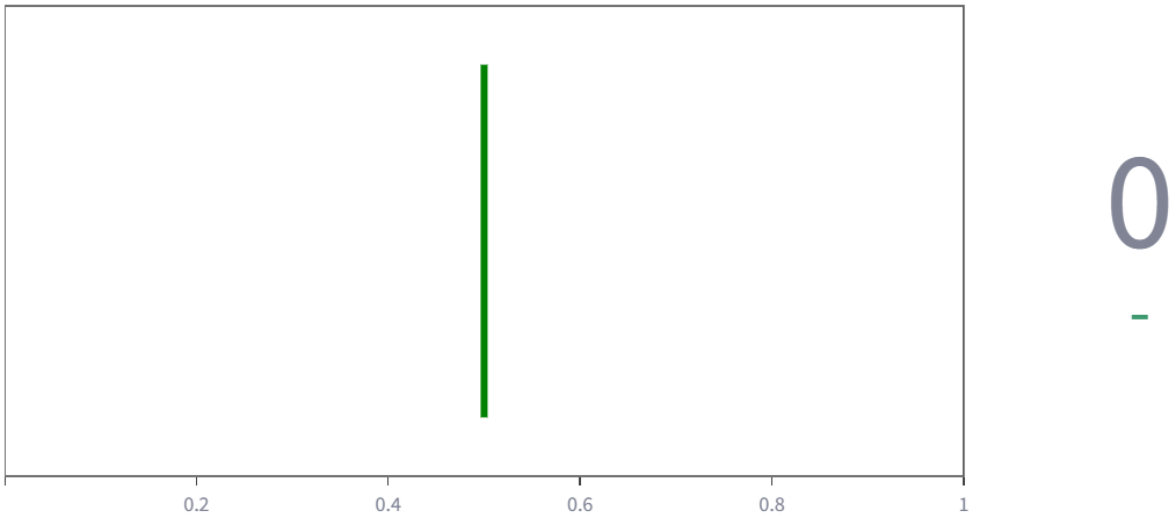

Figure A.2 Graph of Prediction

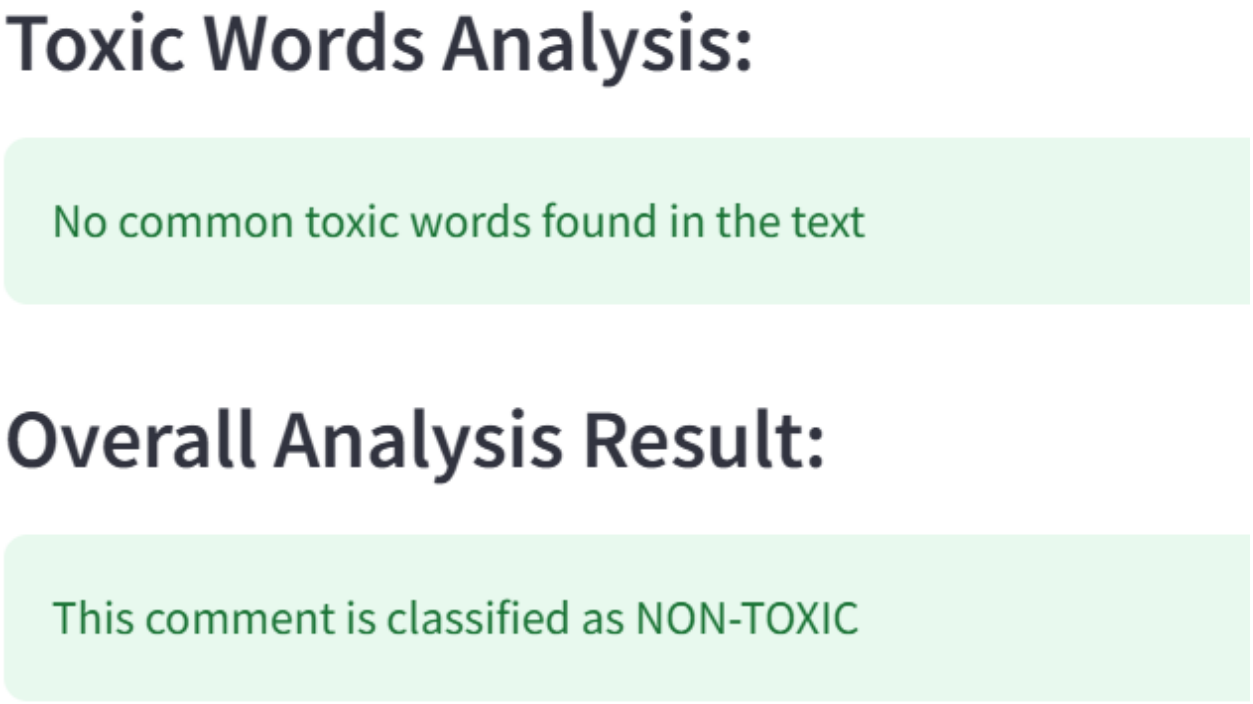

Figure A.3 Analysis Result

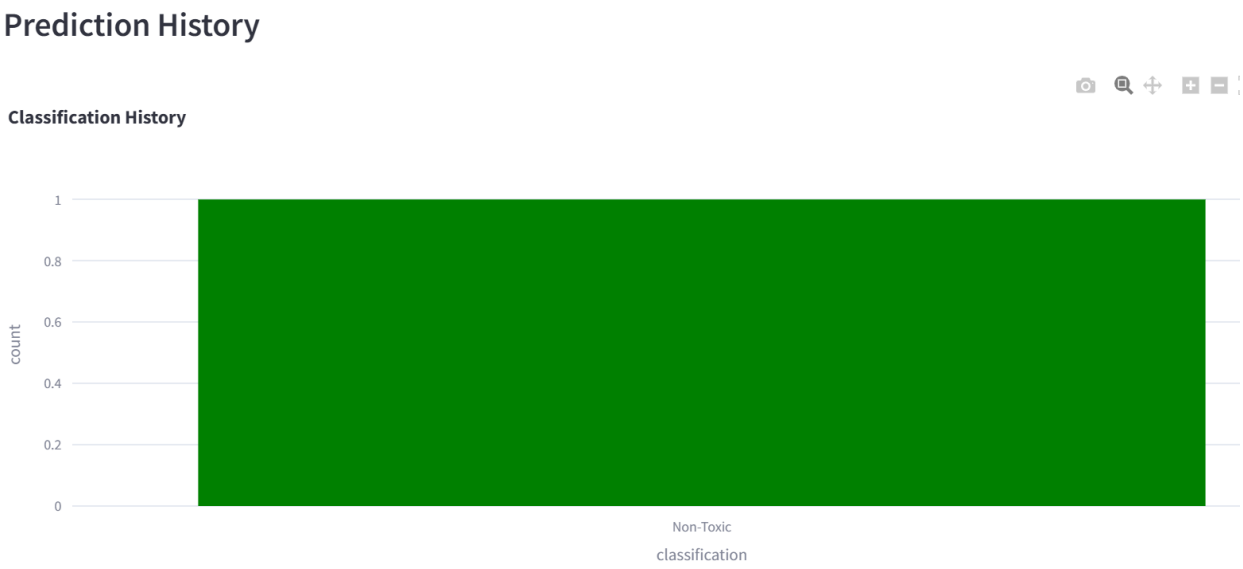

Figure A.4 Bar Chart of Prediction

Figure A.5 Providing Non-Toxic Sentence as an Input

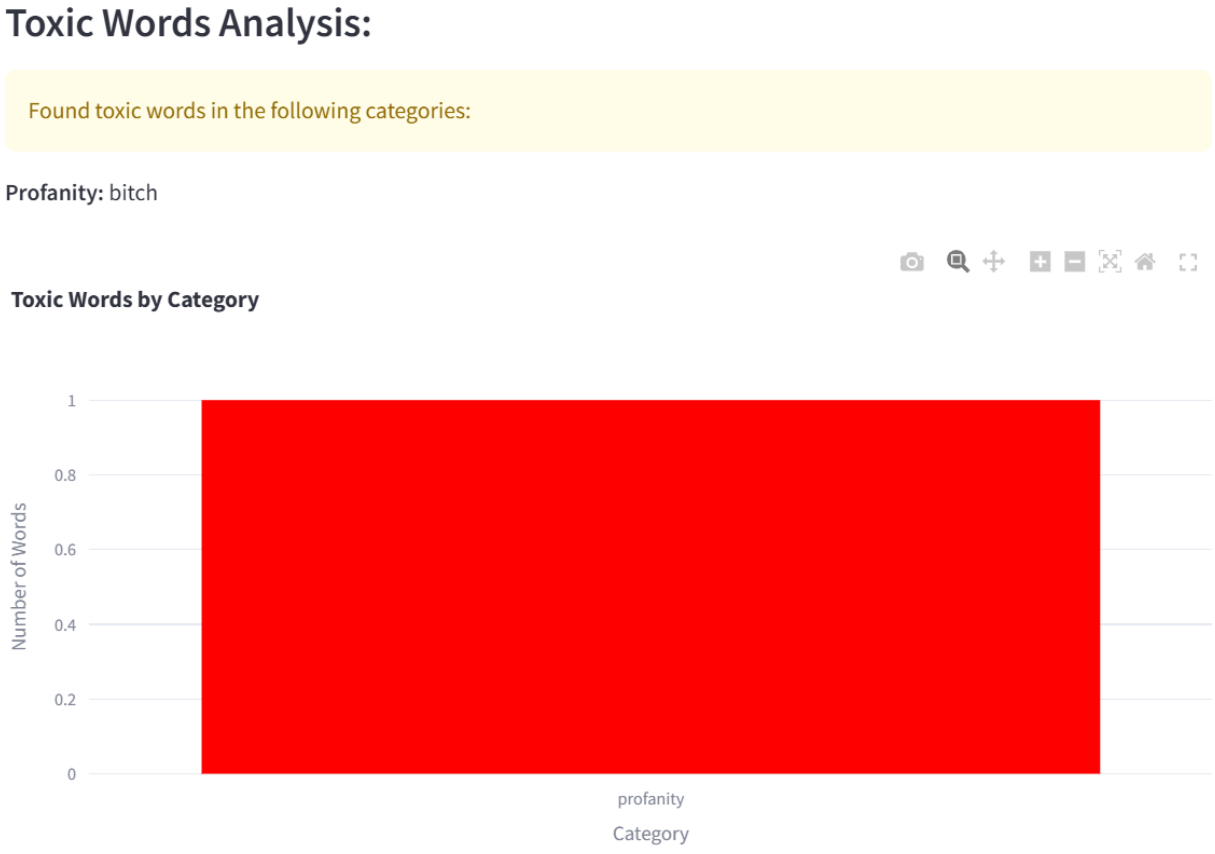

Figure A.6 Analysis and Prediction Of Comment

Figure A.7 Providing Toxic Sentence as an Input

Figure A.8 Classification of Toxic Comment

Figure A.9 Graph of Classified Toxic Comment

REFERENCE

1. Alissa de Bruijn, Vesa Muhonen, Tommaso Albinonistraat, Wan Fokkink, Peter Bloem, and Business Analytics. Detecting offensive language using transfer learning. 2019.
2. Estela Saquete, David Tomas, Paloma Moreda, Patricio Martinez-Barco, and Manuel Palomar. Fighting post-truth using natural language processing: A review and open challenges. Expert Systems with Applications, 141:112943, 2020.
3. Maria Koutamanis, Helen GM Vossen, and Patti M Valkenburg. Adolescents' comments in social media: Why do adolescents receive negative feedback and who is most at risk? Computers in Human Behavior, 53:486–494, 2015.
4. Mark Hsueh, Kumar Yogeeswaran, and Sanna Malinen. Leave your comment below: Can biased online comments influence our own prejudicial attitudes and behaviors? Human communication research, 41(4):557–576, 2015.
5. Moon J Lee and Jung Won Chun. Reading others' comments and public opinion poll results on social media: Social judgment and spiral of empowerment. Computers in Human Behavior, 65:479–487, 2016. 11.
6. Saleem Alhabash, Anna R. McAlister, Chen Lou, and Amy Hagerstrom. From clicks to behaviors: The mediating effect of intentions to like, share, and comment on the relationship between message evaluations and offline behavioral intentions. Journal of Interactive Advertising, 15(2):82–96, 2015.
7. Sameer Hinduja and Justin W Patchin. Bullying, cyberbullying, and suicide. Archives of suicide research, 14(3):206–221, 2010.

8. Shannon D Bailey and Lina A Ricciardelli. Social comparisons, appearance related comments, contingent self-esteem and their relationships with body dissatisfaction and eating disturbance among women. Eating behaviors, 11(2):107–112, 2010.
9. Thomas A Birkland. An introduction to the policy process: Theories, concepts, and models of public policy making. Routledge, 2019.
10. Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In Advances in neural information processing systems, pages 5754–5764, 2019.