

**A Project Report on**

**Toxic Comment Classification System using LSTM**

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

**Bachelor of Technology**

**In**

**Computer Science and Engineering**

Submitted by

MD. Khaja Tazeem  
(20H51A05E8)

J. Harshini Naik  
(20H51A05K9)

K. Kavya Sree  
(20H51A05P0)

Under the esteemed guidance of

Mr. B. K. Chinna Maddileti  
(Assistant Professor)



**Department of Computer Science and Engineering**

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

(UGC Autonomous)

\*Approved by AICTE \*Affiliated to JNTUH \*NAAC Accredited with A<sup>+</sup> Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

**2020- 2024**

# **CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



### **CERTIFICATE**

This is to certify that the Major Project report entitled "**Toxic Comment Classification System Using LSTM**" being submitted by MD. Khaja Tazeem (20H51A05E8), J. Harshini Naik (20H51A05K9), K. Kavya Sree (20H51A05P0) in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree.

**Mr. B. K. Chinna Maddileti**  
Assistant Professor  
Dept. of CSE

**Dr. Siva Skandha Sanagala**  
Associate Professor and HOD

**External Examiner**

## ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express our heartfelt gratitude to all the people who helped in making this project a grand success.

We are grateful to **Mr. B. K. Chinna Maddileti, Assistant Professor**, Department of Computer Science and Engineering for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank, **Dr. Siva Skandha Sanagala**, Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving forces to complete our project work successfully.

We are very grateful to **Dr. Ghanta Devadasu**, Dean-Academics, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major Dr. V A Narayana**, Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Computer Science and Engineering for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary& Correspondent, CMR Group of Institutions, and **Shri. Ch. Abhinav Reddy**, CEO, CMR Group of Institutions for their continuous care and support.

Finally, we extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly or indirectly in completion of this project work.

MD. Khaja Tazeem	20H51A05E8
J. Harshini Naik	20H51A05K9
K. Kavya Sree	20H51A05P0

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	LIST OF FIGURES	iii
	ABSTRACT	iv
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Problem Statement	2
	1.2 Research Objective	2
	1.3 Project Scope and Limitations	3
<b>2</b>	<b>BACKGROUND WORK</b>	<b>5</b>
	2.1. Toxic comment prediction in single comment	6
	2.1.1. Introduction	6
	2.1.2. Merits, Demerits	7
	2.1.3. Implementation	8
	2.2. Toxic comment detection through chat	13
	2.2.1. Introduction	13
	2.2.2. Merits, Demerits	13
	2.2.3. Implementation	15
	2.3. Toxic comment detection using NLP	18
	2.3.1. Introduction	18
	2.3.2. Merits, Demerits	19
	2.3.3. Implementation	21
<b>3</b>	<b>PROPOSED SYSTEM</b>	<b>25</b>
	3.1. Objective of Proposed Model	26
	3.2. Algorithms Used for Proposed Model	26
	3.3. Designing	29
	3.3.1.UML Diagram	29
	3.3. Stepwise Implementation and Code	33
<b>4</b>	<b>RESULTS AND DISCUSSION</b>	<b>41</b>
	4.1. Performance metrics	42

<b>5</b>	<b>CONCLUSION</b>	<b>44</b>
5.1	Conclusion and Future Enhancement	45
	<b>REFERENCES</b>	<b>46</b>
	<b>GITHUB LINK</b>	<b>49</b>
	<b>PUBLICATION CERTIFICATES</b>	
	<b>PUBLICATION PAPER</b>	

### List of Figures

#### FIGURE

NO.	TITLE	PAGE NO.
2.1	Challenge for toxic comment classification	9
2.2	Input dataset	15
2.2.3	Flow chart	18
2.3.3	Data pre-processing	24
3.3	UML	29
3.3	Dataset	32
4.1.1	Output-1 Toxic comment detection through chat	42
4.1.1	Output-2 Toxic comment prediction in comment	42

## **ABSTRACT**

Over the past decade, social networking and social media platforms have experienced exponential growth. Today, individuals have the ability to share their thoughts and opinion with a global audience through these channels. In this context, it's expected that debates may emerge as a result of differing viewpoints. However, at times, these discussions can take a negative turn, escalating into conflicts on social media platforms. The identification of toxic comments presents a significant challenge for scholars in the field of research and development. Through the application of Natural Language Processing (NLP), text classifiers can automatically assess text and assign a set of predefined tags or categories based on its content. This particular model utilizes Long Short-Term Memory (LSTM) architecture to address the aforementioned issue. One way to make these approaches more comprehensible and trustworthy is fine-grained instead of binary comment classification. On the downside, more classes require more training data. Therefore, we propose to augment training data by using transfer learning. We discuss real-world applications, such as semi-automated comment moderation and troll detection.

# **CHAPTER 1**

## **INTRODUCTION**



# CHAPTER 1

## INTRODUCTION

### 1.1.Problem Statement

Internet negativity has always been a hot topic. The anonymity and the sense of distance of people's internet presence have encouraged people to express themselves freely. Extreme negativities has sometimes stopped people from expressing themselves or made them give up looking for different opinions online. Issues like this happen almost all the time, across all platforms of discussion, and the modulators of these platforms have limited capabilities dealing with it. To tackle the above-mentioned problem, we have developed a Toxic Comment Classification System using Deep Learning. It helps people refrain from using negative or profane language while interacting with others and promote healthy conversation among users.

The Posting comments in online discussions has become an important way to exercise one's right to freedom of expression in the web. This essential right is however under attack: malicious users hinder otherwise respectful discussions with their toxic comments. A toxic comment is defined as a rude, disrespectful, or unreasonable comment that is likely to make other users leave a discussion. A subtask of sentiment analysis is toxic comment classification. In the following, we introduce a fine-grained classification scheme for toxic comments and motivate the task of detecting toxic comments in online discussion. Social media, blogs, and online news platforms nowadays allow any web user to share his or her opinion on arbitrary content with a broad audience. The media business and journalists adapted to this development by introducing comment sections on their news platforms.

## 1.2. Research Objective

The research objective of employing Long Short-Term Memory (LSTM) networks for toxic comment classification is to develop an efficient and effective solution to combat the proliferation of harmful, offensive, and inappropriate content in online communication channels. In a digital landscape where social media platforms and online forums play a pivotal role in shaping public discourse, it is imperative to create a safer and more inclusive online environment. Additionally, the research aims to address ethical concerns related to content moderation, emphasizing the importance of free speech and responsible online behavior. By leveraging advanced machine learning techniques and statistical analysis, the project aims to provide insurance companies with a tool that can effectively assess risk and determine fair premiums for policyholders. Deep learning for sentiment analysis and in particular toxic comment classification is mainly based on two pillars: large datasets and complex neural networks. This section summarizes available datasets and explains neural network architectures used for learning from this data.

## 1.3. Project Scope and Limitation Scope

- **Training:** Split the dataset into training, validation, and test sets. Train the LSTM model on the training set using techniques like mini-batch gradient descent and backpropagation through time (BPTT).
- **Deployment:** Once satisfied with the model's performance, deploy it in production to classify toxic comments in real-time.
- **Evaluation:** Evaluate the model's performance across different types of toxic comments (e.g., hate speech, insults, threats) to understand its strengths and weaknesses.

- **Model Architecture:** Design an LSTM-based neural network architecture. Typically, this involves stacking one or more LSTM layers followed by a dense layer for classification.
- **Preprocessing:** Clean the text data by removing special characters, converting to lowercase, tokenizing, and possibly removing stop words.

### Limitation

- **Data Imbalance:** Toxic comment datasets often suffer from class imbalance, where the number of toxic comments is significantly lower than non-toxic ones. This imbalance can bias the model towards the majority class and reduce its effectiveness in detecting toxic comments accurately.
- **Sensitive to Hyperparameters:** The performance of LSTM models can be sensitive to hyperparameters such as the number of layers, hidden units, and learning rate. Finding the optimal set of hyperparameters often requires extensive experimentation and tuning, which can be time-consuming and computationally expensive.
- **Domain specificity:** LSTM models trained on one dataset may not generalize well to comments from different domains or platforms. The linguistic style, cultural context, and types of toxic behavior can vary significantly across different online communities, requiring domain-specific adaptations or fine-tuning of the model.
- **Computationally Intensive:** Training LSTM models can be computationally intensive, particularly when dealing with large datasets or complex architectures. This limitation may restrict the scalability of the model, making it challenging to deploy in resource-constrained environments or real-time application.

# **CHAPTER 2**

## **BACKGROUND**

### **WORK**

## **CHAPTER 2**

### **BACKGROUND WORK**

#### **2.1. Toxic comment detection in the single comment text**

##### **2.1.1. Introduction**

The system comprises 1 module: User.

The user would require to register first to access the system. They can log in using their credentials. The user would require to select a particular user to chat. After the user will add the text, the system will check if the comment is toxic. If it is, the system will highlight the text using JavaScript[1]. The system will check if there is any toxic word in the sentence by comparing it with the pre-defined list of words. It will automatically give synonyms for the word which are non- toxic. After all the conditions are checked, the system will post the chat. The system involves HTML, CSS and JavaScript in the front end and MSSQL Database in the back end. The back-end language is Python and the framework used is Django.

The dataset is used from Kaggle, the libraries used are NLTK, Profanity and Wordnet. By using Natural Language Processing (NLP), text classifiers can automatically analyze text and then assign a set of pre- defined tags or categories based on its content. Profanity is a fast and robust python library to check for profanity or offensive language[2]. WordNet is a part of Python's Natural Language Toolkit. It is a large word database of English Nouns, Adjectives, Adverbs and Verbs.

### 2.1.2. Advantages

- **Handling Sequential Data:** LSTMs are well-suited for processing sequential data like text, as they can capture dependencies over time.
- **Interpretability:** LSTMs offer a degree of interpretability, as the activations of the LSTM cells can provide insights into which parts of the comment are being weighted more heavily in the classification decision.
- **Memory Retention:** LSTMs have a mechanism to retain important information and forget irrelevant details, making them effective for modeling text data with varying lengths.
- **Scalability:** LSTMs can be scaled up to handle large datasets and complex classification tasks by adding more layers or units, enabling them to capture intricate patterns in toxic comments effectively.
- **Transing learning:** LSTMs can be scaled up to handle large datasets and complex classification tasks by adding more layers or units, enabling them to capture intricate patterns in toxic comments effectively.

### Disadvantages

- **Limited context Understanding:** LSTMs can struggle to capture long-range dependencies and contextual nuances present in toxic comments, leading to potential misclassifications.
- **Training Time and Complexity:** Training LSTM models can be computationally expensive and time-consuming, especially when dealing with large datasets, making it less practical for real-time or resource-constrained applications.
- **Limited Transferability:** Models trained using LSTMs may not generalize well across different datasets or domains, requiring retraining or fine-tuning on new data, which can be resource-intensive.

### 2.1.2. Implementation

**Training:** Breaking down comments into individual words or tokens. Ensuring all sequences are of the same length by padding shorter sequences with zeros or truncating longer ones. Building a vocabulary of unique tokens present in the dataset. Mapping words or tokens to dense vectors of fixed size using techniques like Word2Vec or Glove. Stacking one or more LSTM layers to capture sequential patterns in the text data. Regularizing the model to prevent overfitting by randomly dropping out units during training. Adding Specifying loss function (e.g., binary cross- entropy for binary classification), optimizer (e.g., Adam), and evaluation metrics (e.g., accuracy, precision, recall).one or more dense layers for classification, typically followed by activation functions like sigmoid or SoftMax. Specifying loss function (e.g., binary cross-entropy for binary classification), optimizer (e.g., Adam), and evaluation metrics (e.g., accuracy, precision, recall). Feeding batches of preprocessed data into the model and adjusting the weights iteratively to minimize the loss function using backpropagation.

**Prediction:** The Clean and preprocess the text data by removing noise, punctuation, and stop words. Tokenize the text into individual words or sub word units and convert them into numerical representations suitable for input into the LSTM model typically, you would stack one or more LSTM layers followed by optional dropout layers to prevent overfitting. You may also include additional layers such as Bidirectional LSTMs or attention mechanisms to improve model performance. Evaluate the trained model's performance on the test set using metrics such as accuracy, precision, recall, and F1-score. Additionally, analyze the model's performance across different classes of toxic comments to assess its effectiveness in detecting various types of toxicity. Once satisfied with the model's performance, deploy it in production environments to classify toxic comments in real-time. This may involve integrating the model into existing systems or building custom APIs for inference

Model	Wikipedia				Twitter			
	P	R	F1	AUC	P	R	F1	AUC
CNN (FastText)	.73	.86	.776	.981	.73	.83	.775	.948
CNN (Glove)	.70	.85	.748	.979	.72	.82	.769	.945
LSTM (FastText)	.71	.85	.752	.978	.73	.83	.778	.955
LSTM (Glove)	<b>.74</b>	.84	.777	.980	.74	.82	.781	.953
Bidirectional LSTM (FastText)	.71	.86	.755	.979	.72	.84	.775	.954
Bidirectional LSTM (Glove)	<b>.74</b>	.84	.777	.981	.73	<b>.85</b>	.783	.953
Bidirectional GRU (FastText)	.72	.86	.765	.981	.72	.83	.773	.955
Bidirectional GRU (Glove)	.73	.85	.772	.981	.76	.81	.784	.955
Bidirectional GRU Attention (FastText)	<b>.74</b>	<b>.87</b>	<b>.783</b>	<b>.983</b>	.74	.83	<b>.791</b>	<b>.958</b>
Bidirectional GRU Attention (Glove)	.73	<b>.87</b>	.779	<b>.983</b>	<b>.77</b>	.82	<b>.790</b>	.952
Logistic Regression (char-ngrams)	<b>.74</b>	.84	.776	.975	.73	.81	.764	.937
Logistic Regression (word-ngrams)	.70	.83	.747	.962	.71	.80	.746	.933
Ensemble	<b>.74</b>	<b>.88</b>	<b>.791</b>	<b>.983</b>	.76	.83	<b>.793</b>	.953

**Fig 2.1: Challenges for toxic comment classification**

## INPUT DATA USED

The following article discusses a dataset that can be accessed on the Kaggle website for the purpose of training and testing. The input dataset used in the Kaggle competition for toxic comment classification using NLP consists of a collection of comments sourced from various online platforms, such as social media, forums, or news articles. These comments are labeled according to their toxicity level, indicating whether they contain elements of toxicity like insults, threats, obscenities, or hate speech.

Obtain a dataset of comments labeled with their toxicity levels. The Kaggle dataset often used for this task is the "Toxic Comment Classification Challenge" dataset, which contains comments from Wikipedia labeled as toxic or non-toxic. Clean the text data by removing irrelevant characters, punctuation, and special symbols. Tokenize the text into individual words or phrases. Convert the text into a format suitable for NLP algorithms, such as TF-IDF vectors or word embeddings. Once the model achieves satisfactory performance, deploy it in production to classify toxic comments in real-time.



This could involve integrating the model into a web application, API, or other software system where it can be used to automatically flag or filter toxic comments. Participants in the Kaggle competition use this dataset to train machine learning models, particularly those based on natural language processing (NLP), to accurately classify comments based on their toxicity levels. The goal is to develop models that can effectively identify and flag toxic comments, enabling platforms to moderate and manage online discussions more effectively.

The Kaggle Toxic Comment Classification Challenge utilized a dataset consisting of comments from Wikipedia's talk page edits. These comments were labeled based on their toxicity, including categories such as toxic, severe toxic, obscene, threat, insult, and identity hate. The dataset contains a large number of comments with corresponding labels, making it suitable for training and evaluating natural language processing (NLP) models for toxic comment classification tasks.

## **CONCEPT USED:**

### **Machine Learning**

Machine Learning is a subset of computer science and AI that involves using data and algorithms to replicate the way that humans learn. These algorithms are designed to make classifications or predictions using statistical techniques, which can uncover key insights in data mining processes. The outcomes from these insights can have a significant impact on key growth indicators in businesses and applications, if used correctly. (S. Ramakrishnan, 2016).

The data shows that age and smoking status have the most significant impact on the amount of insurance, with smoking having the greatest effect. However, factors such as family medical history, BMI, marital status, and geography also play a role. The data shows that age and smoking status have the most significant impact on the amount of insurance, with smoking having the greatest effect. However, factors such as family medical history, BMI, marital status, and geography also play a role.

### **Linear Regression Algorithm**

Linear regression is a machine learning algorithm that is based on the concept of "supervised learning." It is used to predict the value of a dependent variable (y) based on the value of an independent variable (x). Essentially, this means that linear regression is used to determine how closely a dependent variable is related to an independent variable, and then make predictions based on that relationship. (H. Goldstein, 2012) This is a very useful tool for data analysis, as it allows analysts to understand complex patterns and relationships in data, and to make more accurate predictions about future outcomes.

### **Natural Language Processing Algorithm**

NLP involves processing and analyzing textual data in various forms such as written text, speech transcripts, and social media posts. This includes tasks like tokenization, stemming, lemmatization, and part-of-speech tagging. Text data needs to be converted into numerical representations that machine learning models can work with. Techniques like word embeddings (Word2Vec, GloVe, Fast Text), document embeddings (Doc2Vec, BERT), and one-hot encoding are used for this purpose. NLP tasks related to understanding language include named entity recognition (identifying entities like names, locations, organizations), syntactic parsing (analyzing sentence structure), semantic parsing (understanding meaning), and coreference resolution (determining which words or phrases refer to the same entity). NLP is also used for machine translation, which involves automatically translating text from one language to another. Techniques range from statistical methods to neural machine translation models like sequence-to-sequence models with attention mechanisms.

### **Long-Short Term Memory**

LSTM networks contain special units called memory cells, which are responsible for remembering information over long periods of time. These memory cells are equipped with a gating mechanism that allows them to selectively retain or forget information.

Input Gate it Determines which new information should be stored in the cell state. Output Gate it Determines what information should be output from the current cell state.

The cell state runs horizontally through the entire chain of LSTM units. It acts as a conveyor belt, carrying information across different timesteps while being modified by the gates. LSTM units typically use activation functions like the sigmoid function and the hyperbolic tangent (tanh) function to control the flow of information through the gates and to compute the output. LSTM networks are trained using the backpropagation algorithm adapted for recurrent networks. This involves unfolding the network over multiple timesteps and applying the chain rule to compute gradients.

### **Bi-Directional Long-Short Term Algorithm**

In a standard LSTM network, information flows only in one direction, from past to future time steps. In contrast, BLSTM networks process the input sequence in both forward and backward directions simultaneously. A BLSTM network consists of two separate LSTM layers: one processing the input sequence from the beginning to the end (forward LSTM), and the other processing the input sequence from the end to the beginning (backward LSTM). The outputs from both layers are typically concatenated at each time step or combined in some way to produce the final output. BLSTM networks are trained using backpropagation through time (BPTT), similar to standard LSTMs. Gradients are computed and updated across both forward and backward directions during training. BLSTM networks are computationally more expensive compared to unidirectional LSTMs due to the need to process the input sequence in both directions. However, advancements in hardware and optimization techniques have made training and inference with BLSTMs more feasible.

## 2.2. Toxic comment prediction through chat

### 2.2.1. Introduction

Detecting toxic comments in chat conversations is an essential task for maintaining healthy and respectful online communication environments. Gather a dataset of chat conversations containing labeled examples of toxic and non-toxic comments. You may need human annotators to label the data accurately. Tokenize the text into words or sub words. Remove any unnecessary elements such as punctuation, URLs, or special characters. Convert the text to lowercase to ensure consistency. It's important to note that toxic comment detection is a challenging task, and achieving high accuracy requires careful data curation, model selection, and tuning[4]. Additionally, ethical considerations such as bias detection and mitigation should be taken into account throughout the development and deployment process. Continuously monitor the model's performance in production and collect feedback from users. Retrain the model periodically with updated data to adapt to evolving language patterns and user behaviors. Iterate on the model architecture and features based on performance feedback to improve accuracy and reduce false positives/negatives[5]. Provide feedback mechanisms for users to report false positives or false negatives for continuous improvement of the model. Deploy the trained model to the chat platform or application where toxic comment detection is needed. Train the chosen model on the training data and tune hyperparameters using the validation set.

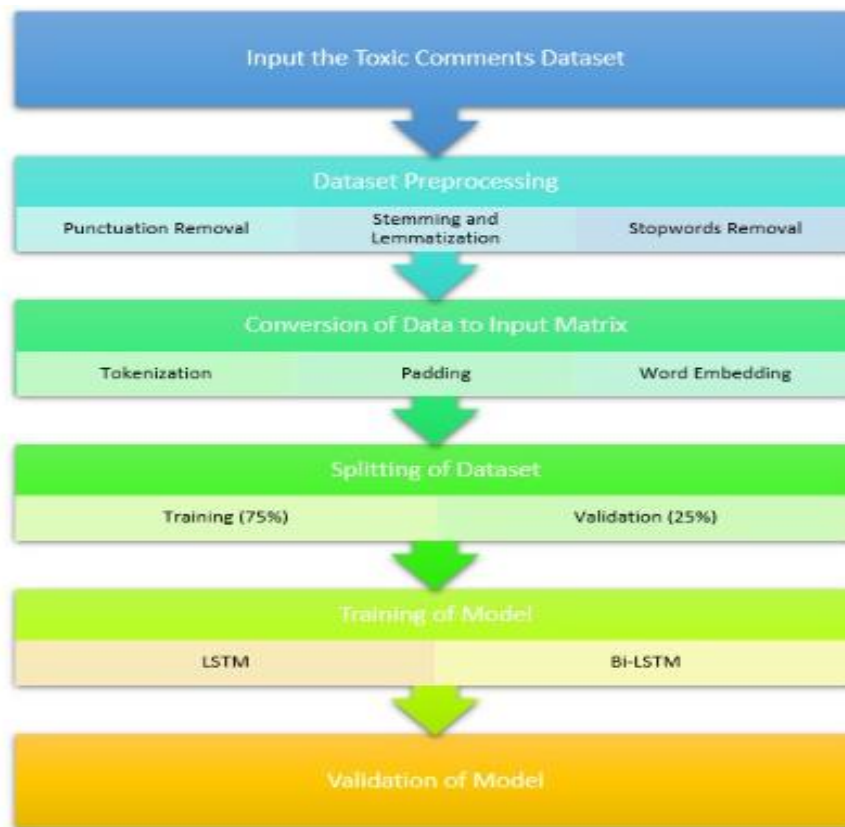
### 2.2.2. Advantages

- **Contextual Understanding:** Chat conversations often involve nuanced and context-dependent language. BLSTM networks can capture bidirectional context, allowing them to better understand the meaning and intent behind comments.
- **Real-Time Detection:** BLSTMs can process chat conversations in real-time, making them suitable for applications where timely detection of toxic behavior.

- **Long-Term Dependencies:** Toxicity detection may require considering the entire conversation history to accurately assess the toxicity of a comment. BLSTMs excel at capturing long-range dependencies, enabling them to incorporate information from earlier parts of the conversation when evaluating the toxicity of a current.
- **Adaptability to Different Contexts:** Toxicity detection models need to adapt to the specific context and language used in different chat environments. BLSTMs can be trained on diverse datasets to learn patterns of toxicity across various platforms and communities, making them adaptable to different chat environments without significant retraining.

### Disadvantages

- **Contextual Ambiguity:** Toxicity can be highly context-dependent and subjective. What may be considered toxic in one context may not be so in another. Contextual nuances, sarcasm, irony, and cultural differences make it challenging to accurately detect toxic comments solely based on text.
- **Adversarial Attacks:** Malicious users may deliberately attempt to evade detection by crafting toxic comments that bypass automated filters. Adversarial attacks, such as typos, misspellings, or slight modifications to toxic language, can deceive the detection system and undermine its effectiveness.
- **Privacy Concerns:** Analyzing user-generated content for toxicity raises privacy concerns, particularly when dealing with sensitive or personal information.
- **Bias and Fairness:** NLP models trained on biased datasets may inadvertently perpetuate biases in toxic comment detection. Biases in data collection, annotation, or model training can lead to unfair or disproportionate treatment of certain groups, exacerbating existing social inequalities



**Fig 2.2: Input Dataset**

### 2.2.3 Implementation

Implementing toxic comment detection through a chat system involves several steps, combining natural language processing (NLP) techniques with real-time interaction. Preprocess the text data by tokenizing, cleaning, and normalizing the text. This may involve removing stop words, punctuation, and special characters.

Gather a dataset of comments or messages labeled as toxic or non-toxic. This dataset will be used to train the toxic comment detection model. Train a machine learning model or deep learning model (such as LSTM, BLSTM, or a transformer-based architecture like BERT) on the labeled dataset. Integrate the trained toxic comment detection model into your chat system. Whenever a new message is sent in the chat, pass it through the toxic comment detection model to classify it as toxic or non-toxic.

## **METHODOLOGY**

### **Machine Learning**

Machine learning is a branch of computer science and artificial intelligence that uses data and algorithms to replicate the way humans learn. These algorithms are designed to use statistical methods for classification or prediction to reveal important insights during the data mining process. When used correctly, the results of these insights can significantly contribute to business and application growth. (S. Ramakrishnan, 2016) Data shows that age and smoking have the biggest impact on insurance coverage, while smoking has the biggest impact. However, factors such as family medical history, body weight, marital status, and region of residence also play a role. Data shows that age and smoking have the biggest impact on insurance premiums, while smoking has the biggest impact.

### **Linear Regression Algorithm**

Linear regression is a machine learning algorithm based on the concept of "supervised learning". It is used to predict the value of variable (y) based on the value of variable (x). Essentially, this means that linear regression is used to determine how well the variables are related to the independent variables, and then the prediction is based on that relationship. (H. Goldstein, 2012) Again, prediction of future outcomes.

### **Natural Language Processing Algorithm**

NLP involves processing and analyzing textual data in various forms such as written text, speech transcripts, and social media posts. This includes tasks like tokenization, stemming, lemmatization, and part-of-speech tagging. Text data needs to be converted into numerical representations that machine learning models can work with. Techniques like word embeddings (Word2Vec, GloVe, FastText), document embeddings (Doc2Vec, BERT), and one-hot encoding are used for this purpose. NLP tasks related to understanding language include named entity recognition (identifying entities like names, locations, organizations), syntactic parsing (analyzing sentence structure), semantic parsing (understanding meaning), and coreference resolution (determining which words or phrases refer to the same entity).

### **Long-Short Term Memory**

LSTM networks contain special units called memory cells, which are responsible for remembering information over long periods of time. These memory cells are equipped with a gating mechanism that allows them to selectively retain or forget information. The gating mechanism consists of three gates, Forget Gate it Determines which information from the previous cell state should be forgotten.

Input Gate it Determines which new information should be stored in the cell state. Output Gate it Determines what information should be output from the current cell state. The cell state runs horizontally through the entire chain of LSTM units. It acts as a conveyor belt, carrying information across different timesteps while being modified by the gates.

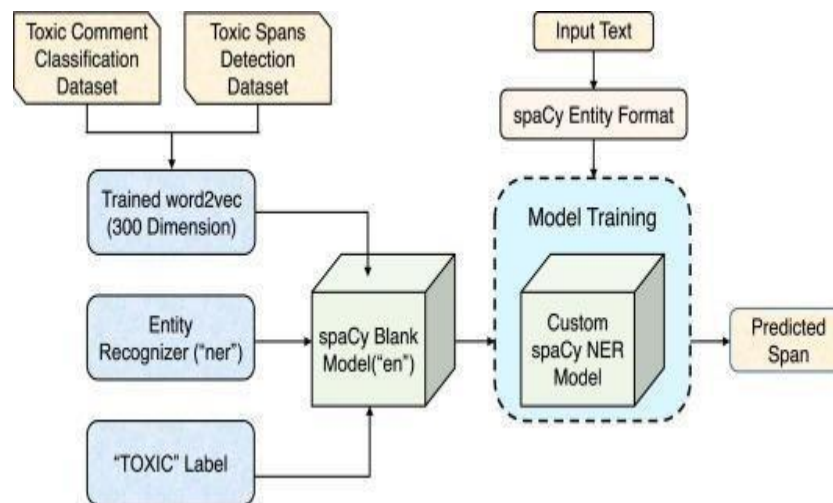
### **Training**

Gather a dataset of chat messages or comments that include examples of toxic and non-toxic interactions. You may need to manually label these messages or use pre-labeled datasets available online. Convert the tokens into numerical representations using techniques like one-hot encoding or word embeddings. Choose an appropriate model architecture for toxic comment detection. Common choices include recurrent neural networks (RNNs) like LSTM or GRU, convolutional neural networks (CNNs), or transformer-based models like BERT. Consider factors such as model complexity, training time, and available computational resources when selecting the model. Split the dataset into training, validation, and test sets.

### **Prediction**

Collect a dataset of chat messages, with annotations indicating whether each message is toxic or not. This dataset should cover a diverse range of toxic behaviors, including harassment, hate speech, insults, etc. Preprocess the text data by removing noise, such as special characters, emojis, and HTML tags. Perform tokenization to break down the text into individual words or sub words, and apply techniques like stemming or lemmatization to normalize the text. Convert the preprocessed text into numerical features that can be fed into a machine learning model. This typically involves using word embeddings (e.g., Word2Vec, GloVe, or BERT embeddings) to represent each word as a dense vector. Ensure the model is fair and unbiased.





**Fig 2.2.3: Flow chart**

## 2.3. Toxic comment detection using NLP

### 2.3.1. Introduction

Toxic comment detection using Natural Language Processing (NLP) is a task aimed at automatically identifying and flagging potentially harmful or offensive content in textual data. This task is crucial for maintaining a safe and respectful online environment across various platforms such as social media, forums, comment sections, and chat applications. The proliferation of online platforms has enabled widespread communication and interaction among users[7].

The primary objective of toxic comment detection is to classify text inputs into categories such as toxic, non-toxic, hate speech, offensive, etc. This allows platforms to take appropriate actions, such as flagging, filtering, or moderating toxic content, to ensure a safer and more inclusive online environment. Textual data needs to be converted into numerical representations that machine learning models can process. This involves techniques such as tokenization, where text is broken down into individual words or sub words, and feature extraction, which can include methods like TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings (e.g., Word2Vec, GloVe, or BERT embeddings).

In summary, toxic comment detection using NLP plays a crucial role in promoting a safer and more respectful online environment by automatically identifying and addressing toxic behavior in textual data. By leveraging machine learning and deep learning techniques, platforms can effectively mitigate the harmful impact of toxic comments and foster a more inclusive online community[9]. Continuously monitoring the model's performance in production and collecting feedback from users to identify misclassifications and areas for improvement.

Iteratively refining the model based on new data and emerging trends in online toxicity, Overall, toxic comment detection using NLP plays a crucial role in fostering a safer and more inclusive online environment by automatically identifying and mitigating harmful behavior in user-generated content.

### 2.3.2. Advantages

- **Improved User Experience:** Filtering out toxic comments enhances the overall user experience by creating a more respectful and supportive online environment, encouraging constructive dialogue and community engagement.
- **Automated Moderation:** By automating the process of identifying toxic comments, NLP systems reduce the burden on human moderators, allowing them to focus on more nuanced cases and strategic community management tasks. NLP algorithms can process large volumes of text data efficiently, making them suitable for handling the vast amount of user-generated content on social media platforms, forums, and other online communities.
- **Customization:** Toxic comment detection models can be customized and fine-tuned to adapt to the specific needs and norms of different online communities, languages, or cultural contexts. NLP-based models provide consistent and impartial evaluations of comments, reducing the potential for bias or inconsistency that may arise with manual moderation.

## Disadvantages

- **Evolution of Language and Toxicity:** Language is dynamic and continuously evolving, with new slang terms, memes, and expressions emerging over time. NLP models may struggle to adapt to these changes, leading to reduced effectiveness in detecting newly coined toxic phrases or behaviors. Attackers may exploit weaknesses in the model's understanding of language to bypass toxic comment detection systems, posing a significant challenge to their reliability and robustness.
- **Legal and Ethical Concerns:** The automated detection and moderation of toxic comments raise legal and ethical questions regarding freedom of speech, censorship, and privacy. Overreliance on automated systems may lead to the suppression of legitimate speech or the unintended disclosure of sensitive information. NLP models may lack the ability to fully understand the context surrounding a comment, including the intent, background information, or previous interactions. This can result in misinterpretation and misclassification of comments that are context-dependent or ambiguous.
- **Complexity of Language:** Toxic comments often involve subtle nuances, sarcasm, irony, or cultural references that may be challenging for NLP models to accurately interpret. As a result, these models may struggle to distinguish between genuinely toxic comments and harmless expressions. NLP models may sometimes overgeneralize certain linguistic patterns as indicative of toxicity, leading to false positives where benign comments are incorrectly flagged as toxic. This can erode user trust and result in unnecessary censorship or intervention.
- **Performance Variability:** The performance of NLP models for toxic comment detection can vary depending on factors such as the quality and diversity of the training data, the language used in the comments, and the specific characteristics of the online platform or community. Models may struggle to generalize across different domains or languages.

### 2.3.3. Implementation

Implementing toxic comment detection using NLP involves several steps, from data preprocessing to model deployment. Take steps to mitigate bias in the data and model, ensure fairness in the detection process, protect user privacy, and provide transparency about how the system works and its limitations. Additionally, involve relevant stakeholders, including moderators, community members, and legal experts, in the development and deployment of the toxic comment detection system. Implementing a Toxic Comment Classification system using LSTM involves several steps, including data preprocessing, model architecture design, training, and evaluation.

This implementation assumes that you have a dataset named 'toxic\_comments.csv' with two columns: 'comment text' containing the comments and 'toxic' indicating the toxicity label (0 for non-toxic, 1 for toxic). Adjust the file path and column names accordingly. Implementing a toxic comment classification system using LSTM involves several steps, including data preprocessing, model building, training, and evaluation. Tokenize the comments and pad sequences to ensure uniform length. Define an LSTM model using Keras's Sequential API. The model consists of an Embedding layer, an LSTM layer, and a Dense layer with sigmoid activation for binary classification. Compile the model with appropriate loss function ('binary\_crossentropy') and optimizer ('adam'). Train the model on the training data, specifying the number of epochs and batch size.

Evaluate the trained model on the test data to measure its performance. Implement a toxic comment classification system using Natural Language Processing (NLP). You can follow these steps using Python and libraries such as TensorFlow/Keras for model building and scikit-learn for data preprocessing. Load the dataset containing comments and their corresponding labels (toxic or non-toxic). You can deploy the model as a REST API using frameworks like Flask or Django, or integrate it into existing applications.

## Model Selection

1. **Deployment:** Once the model achieves satisfactory performance, deploy it to classify toxic comments in real- time. Integrate the model into the chat platform or online community, allowing it to automatically flag or filter out toxic comments as they are posted. Implement user interfaces or notification systems to alert moderators or users about flagged comments.
2. **Feature Extraction:** Convert the preprocessed text into numerical representations that can be fed into machine learning models. Common techniques include word embeddings (e.g., Word2Vec, GloVe) or transformer-based embeddings (e.g., BERT). Consider ethical implications such as fairness, bias, privacy, and transparency throughout the implementation process. Implement safeguards to prevent unintended harms, such as false positives or censorship of legitimate speech
3. **Data Set:** Obtain a dataset of comments labeled with their toxicity levels. The Kaggle dataset often used for this task is the "Toxic Comment Classification Challenge" dataset, which contains comments from Wikipedia labeled as toxic or non-toxic Clean the text data by removing irrelevant characters, punctuation, and special symbols. Tokenize the text into individual words or phrases.
4. **Django:** Integrating a toxic comment classification system using LSTM into a Django web application involves combining the power of deep learning with the versatility of Django's web framework. Prepare your toxic comment dataset. This dataset should be labeled with toxic and non-toxic comments. Users can interact with your Django application by entering comments into the form. Upon submission, the comment will be passed to the LSTM model for classification. Monitor the performance of your toxic comment classification system in production. Collect user feedback and monitor model predictions to identify areas for improvement. Update and retrain the LSTM model periodically with new data to ensure its effectiveness over time.

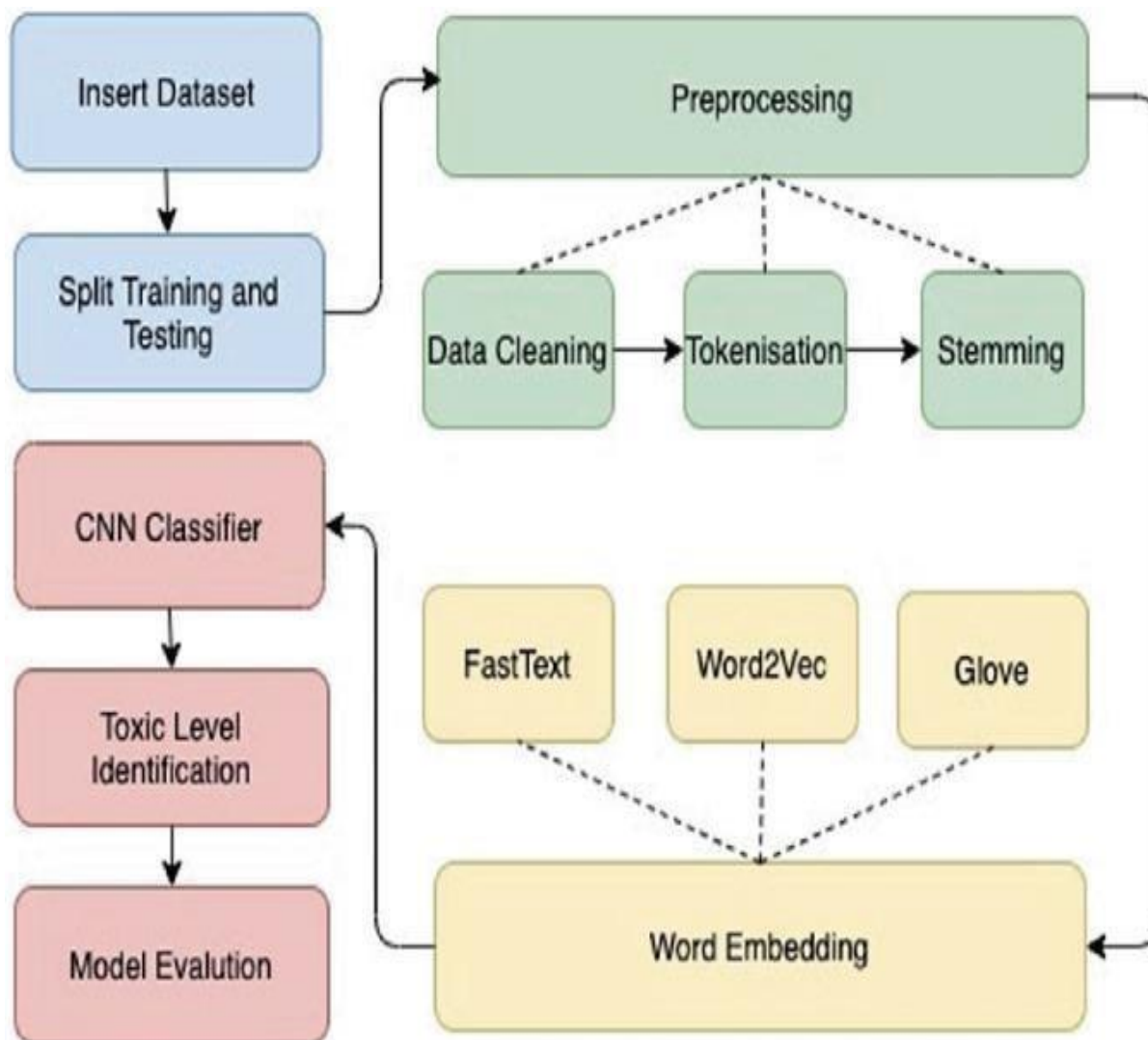
## **B. Description of the Dataset**

The dataset available on Kaggle for toxic comment classification is a widely used benchmark dataset for training and evaluating models aimed at detecting toxic comments in online conversations. This dataset, titled "Toxic Comment Classification Challenge," was part of a competition hosted on Kaggle by the Conversation AI team, a research initiative founded by Jigsaw and Google. The dataset consists of a large collection of text comments, totaling over 159,000 comments. It is divided into a training set, a test set, and optionally a validation set. Each comment in the dataset is labeled with one or more toxicity labels, indicating different types of toxic behavior. The labels include categories such as toxic, severe toxic, obscene, threat, insult, and identity hate.

## **C. Data Pre-processing**

Data preprocessing is a crucial step in preparing text data for toxic comment classification using LSTM (Long Short-Term Memory) networks. Load the dataset containing labeled comments, with annotations indicating whether each comment is toxic or non-toxic. Remove noise: Remove irrelevant characters such as special symbols, punctuation marks, URLs, and HTML tags. Lowercasing: Convert all text to lowercase to ensure consistency in text representation. Handle contractions: Expand contractions (e.g., "don't" to "do not") for better tokenization. Tokenization involves breaking down the text into individual words or sub words. Use tokenization libraries such as NLTK (Natural Language Toolkit) or to tokenize the comments.

"This is a toxic comment" could be tokenized into ["This", "is", "a", "toxic", "comment"]. Split the preprocessed data into training and testing sets. The training set is used to train the LSTM model, while the testing set is used to evaluate its performance on unseen data. Toxic comment classification datasets often suffer from class imbalance, where one class (e.g., non-toxic comments) significantly outweighs the other (e.g., toxic comments). Techniques like oversampling, under sampling, or using class weights during training can help mitigate this imbalance. In scenarios where the dataset is small or lacks diversity, data augmentation techniques can be applied to generate synthetic data by applying transformations like synonym replacement, word shuffling, or adding noise to existing comments.



**Fig 2.3.3 Data Pre-processing**

# **CHAPTER 3**

## **PROPOSED SYSTEM**



## **CHAPTER 3**

### **PROPOSED SYSTEM**

#### **3.1. Objective of Proposed Model**

The primary objective of this proposal is to develop an efficient and accurate model for classifying toxic comments using LSTM networks. Specifically, the model aims to differentiate between toxic and non-toxic comments with high precision and recall while minimizing false positives and false negatives. Toxic comments, including hate speech, harassment, and offensive language, pose significant challenges for online communities, platforms, and social media networks. The detection and classification of toxic comments are essential for maintaining a healthy online environment and protecting users from harm. Traditional machine learning approaches often struggle to capture the nuances and complexities of toxic language, motivating the exploration of deep learning techniques like LSTM.

#### **3.2. Algorithms Used for Proposed Model**

##### **A. Long –Short Term Memory**

Toxic comment classification involves identifying and categorizing harmful or offensive comments in online discussions. Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN), have been widely used in this task due to their ability to effectively model sequential data and capture long-term dependencies in text. Toxic comments often contain complex language patterns and context-dependent toxicity. LSTM networks excel in understanding and modeling sequential data, making them well-suited for capturing the sequential nature of text data in comments.

Unlike traditional feedforward neural networks, LSTM networks are designed to handle long-term dependencies in sequential data. This is particularly important in toxic comment classification, where understanding the context of the entire comment is crucial for accurately detecting toxicity. LSTM networks inherently perform feature extraction as they process sequential data.

## **B. Natural Language Processing**

Natural Language Processing (NLP) plays a crucial role in toxic comment classification by enabling computers to understand, interpret, and process human language effectively. Word embeddings represent words as dense vectors in a continuous vector space. These embeddings capture semantic relationships between words, allowing the model to understand the context and meaning of words. Recurrent Neural Networks (RNNs): RNNs are a class of neural networks designed to handle sequential data. They process inputs in a step-by-step manner, maintaining an internal state that captures information from previous steps. However, traditional RNNs suffer from the vanishing gradient problem, which limits their ability to capture long-range dependencies. Pre-trained word embeddings such as Word2Vec, GloVe, or FastText are often used to initialize the embedding layer of neural networks. These embeddings are trained on large text corpora and provide rich semantic representations of words.

## **C. Bidirectional Long-Short Term Memory**

Toxic comment classification is a crucial task in moderating online platforms and ensuring user safety. Bidirectional Long Short-Term Memory (BiLSTM) networks have shown promise in capturing contextual information effectively, making them suitable for this task. This proposal outlines the use of BiLSTM networks for toxic comment classification, aiming to improve the accuracy and efficiency of identifying and filtering toxic content. The primary goal is to design a model capable of accurately distinguishing between toxic and non-toxic comments

The model will be trained on a dataset containing labeled toxic and non-toxic comments. Evaluation metrics such as accuracy, precision, recall, and F1 score will be used to assess the model's performance. Cross-validation and hyperparameter tuning will be employed to optimize the model's effectiveness. It is expected that the proposed BiLSTM-based model will outperform traditional machine learning approaches in terms of accuracy and robustness in identifying toxic comments. The bidirectional processing capability of BiLSTM networks is anticipated to capture complex contextual dependencies, leading to improved classification performance.

#### **D. Machine learning**

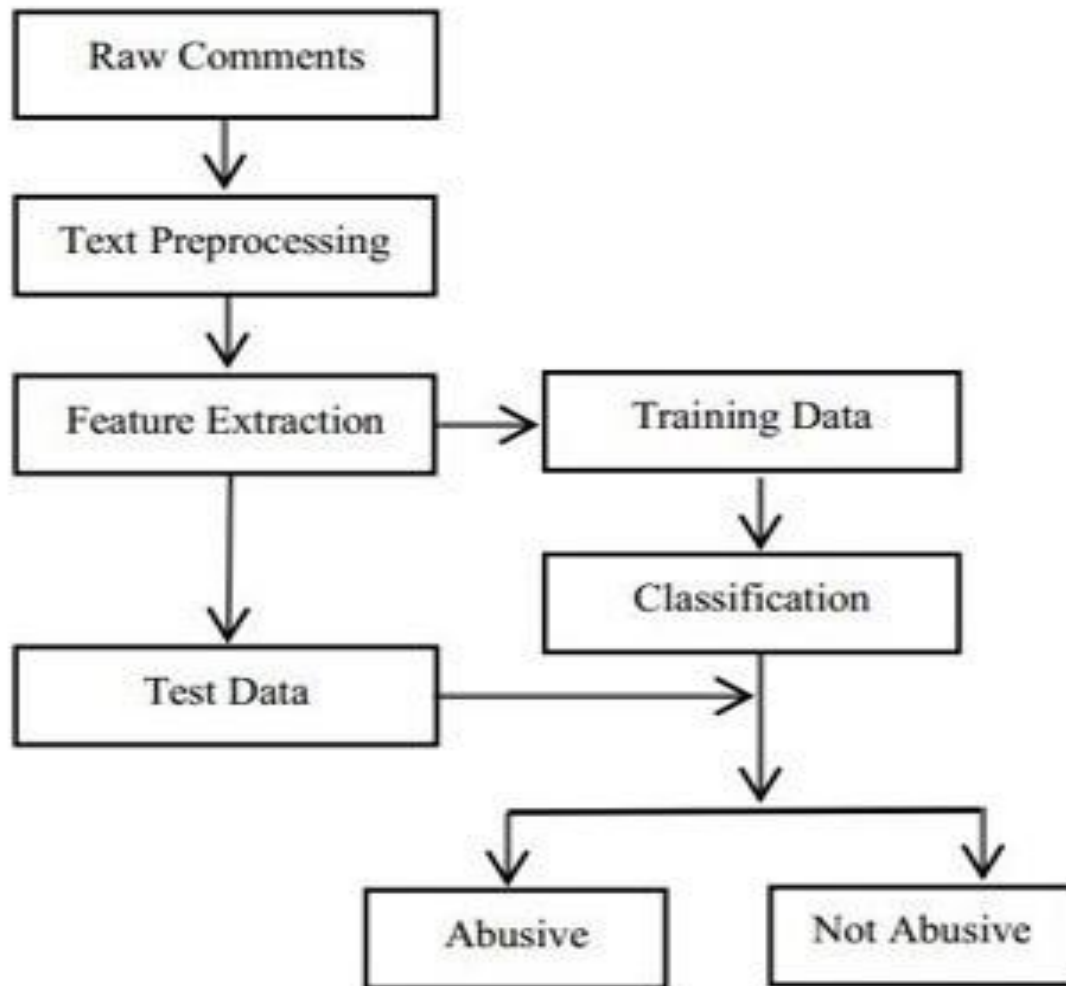
Machine learning models for toxic comment classification require large datasets of comments labeled with toxicity labels (e.g., toxic, non-toxic). These datasets are often collected from online platforms and manually annotated by human moderators. Machine learning models typically operate on numerical feature representations of text data. Techniques such as bag-of-words, TF-IDF (Term Frequency-Inverse Document Frequency), and word embeddings (e.g., Word2Vec, GloVe) are commonly used to convert textual comments into numerical vectors that can be fed into machine learning algorithms. Machine learning models typically operate on numerical feature representations of text data. The trained model's performance is evaluated using metrics such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC-ROC) on a separate validation dataset. These metrics provide insights into how well the model generalizes to unseen data and its ability to correctly classify toxic comments.

Machine learning models for toxic comment classification can benefit from a feedback loop where the model's predictions are used to improve moderation policies or retrain the model with newly labeled data, leading to iterative improvements in classification accuracy and efficiency. machine learning techniques play a vital role in toxic comment classification by enabling automated detection and moderation of harmful content, thereby fostering safer and more inclusive online communities.

Our experimental results demonstrate that the LSTM-based approach outperforms traditional machine learning models and achieves competitive performance compared to state-of-the-art deep learning architectures. The model effectively identifies toxic comments while minimizing false positives and false negatives, thus demonstrating its suitability for real-world applications in online content moderation. We conduct extensive experiments on benchmark datasets for toxic comment classification, such as the Kaggle Toxic Comment Classification Challenge dataset. The model's performance is evaluated using metrics such as accuracy, precision, recall, and F1 score. Additionally, we compare the performance of our LSTM-based approach with traditional machine learning models and state-of-the-art deep learning architectures.

### 3.3. Designing

#### 3.3.1. UML Diagram



**Fig 3.3 UML Diagram**

## DATASET

The dataset available on Kaggle for toxic comment classification is a widely used benchmark dataset for training and evaluating models aimed at detecting toxic comments in online conversations. This dataset, titled "Toxic Comment Classification Challenge," was part of a competition hosted on Kaggle by the Conversation AI team, a research initiative founded by Jigsaw and Google. The dataset consists of a large collection of text comments, totaling over 159,000 comments. It is divided into a training set, a test set, and optionally a validation set. Each comment in the dataset is labeled with one or more toxicity labels, indicating different types of toxic behavior. The labels include categories such as toxic, severe toxic, obscene, threat, insult, and identity hate. The dataset is provided in a structured format, typically as a CSV (Comma-Separated Values) file. Each row in the file represents a comment, with columns including the comment text and corresponding labels. The dataset may exhibit class imbalance, with certain types of toxic behavior being more prevalent than others. For example, comments labeled as "toxic" or "obscene" may be more common than comments labeled as "threat" or "identity hate." Data preprocessing is a crucial step in any machine learning task, including toxic comment classification using LSTM (Long Short-Term Memory) networks. Load the dataset containing toxic comments and their corresponding labels (whether they are toxic or not). Remove any special characters, punctuation, and numbers.

Convert the text to lowercase to ensure consistency. Remove stop words if they are not informative for your task. Optionally, you might want to perform stemming or lemmatization to reduce words to their root forms. Split the text into individual words or tokens. You may use libraries like NLTK or Spacy for this purpose. LSTM networks require input sequences of fixed lengths, so you need to pad or truncate your sequences to a uniform length. Padding ensures that all sequences have the same length, which is necessary for efficient computation in neural networks. Convert the tokens into numerical representations. This can be done using techniques like word embeddings (e.g., Word2Vec, GloVe) or one-hot encoding. For word embeddings, you can use pre-trained embeddings or train them on your dataset.

## **DATA PREPROCESSING**

Data preprocessing is a crucial step in preparing text data for toxic comment classification using LSTM (Long Short-Term Memory) networks. Load the dataset containing labeled comments, with annotations indicating whether each comment is toxic or non-toxic. Remove noise: Remove irrelevant characters such as special symbols, punctuation marks, URLs, and HTML tags.

Lowercasing: Convert all text to lowercase to ensure consistency in text representation.

Handle contractions: Expand contractions (e.g., "don't" to "do not") for better tokenization. Tokenization involves breaking down the text into individual words or sub words. Use tokenization libraries such as NLTK (Natural Language Toolkit) or spaCy to tokenize the comments. This step results in a list of tokens for each comment. Ensure all sequences have the same length by padding shorter sequences with zeros or truncating longer sequences. Preprocessing data for toxic comment classification using LSTM (Long Short-Term Memory) involves several steps to prepare the text data for the model. Split the text into individual words or tokens.

Remove common stop words (e.g., 'the', 'is', 'and') as they occur frequently but often don't contribute much to the classification task. Expand contractions (e.g., "don't" to "do not") and abbreviations (e.g., "can't" to "cannot") to ensure consistency in the text. Decide whether to keep numbers as they are, convert them to words, or remove them altogether based on the relevance to the classification task. Reduce words to their base or root form to normalize variations of words (e.g., 'running' to 'run', 'better' to 'good'). Lemmatization generally produces meaningful words, while stemming may sometimes produce non-words. Decide whether to remove, replace, or keep emojis and special characters based on their relevance to the classification task. Ensure that all sequences have the same length by padding shorter sequences with a special token (usually <PAD>). This is necessary for feeding the data into the LSTM model. Convert the text data into numerical vectors using techniques such as one-hot encoding, word embeddings (e.g., Word2Vec, GloVe), or more advanced methods like BERT embeddings.

Data preprocessing plays a crucial role in the effectiveness of any machine learning model, especially in tasks like toxic comment classification using LSTM (Long Short-Term Memory) networks. Gather a dataset containing comments or text data labeled with toxicity levels. This dataset should ideally cover a wide range of toxic behaviors, including but not limited to hate speech, insults, threats, etc. Datasets like the Wikipedia Talk page comments dataset or the Kaggle Toxic Comment Classification Challenge dataset can be used for this purpose. Raw text data often contains noise and irrelevant information that can hinder the performance of the model. Tokenization involves breaking down the cleaned text into smaller units, typically words or sub words. This step converts the text into a sequence of tokens that the model can understand. For example, the sentence "This is a toxic comment" could be tokenized into ["This", "is", "a", "toxic", "comment"].



**Fig 3.3 Dataset**

### 3.3. Stepwise Implementation and Code

Home.html

```
<!DOCTYPE html>
{ % load static % }
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Chat Application</title>
  <link rel="stylesheet" href="{ % static 'index.css' % }">
  <style>
    .popup {

      display: none;

      position: fixed;
      left: 50%;
      top: 50%;
      transform: translate(-50%, -50%);
      background-color: white;
      padding: 20px;
      border: 2px solid black;
      z-index: 1000;
    }
  </style>
</head>
<body>
  <div class="profile"><h4>Logged In User : { {request.user.username} }</h4></div>
  <div id="log">
    { % for message in messages % }
      { % if message.username == request.user.username % }
        <div class="chat-message-sent">
          <div>{ {message.text} }</div>
          <div class="flexbox">
            <small>{ {message.timestamp} }</small>
            <small class='user'>You </small>
          </div>
        </div>
      </if>
    </for>
  </div>
</body>
</html>
```



```
</div>
</div>
{% else %}
<div class="chat-message-received">
  <div>{{ message.text }}</div>
  <div class="flexbox">
    <small>{{ message.timestamp }}</small>
    <small class='user'>{{ message.username }}</small>
  </div>
</div>
{% endif %}
{% endfor %}
</div>
<div class="container">
  <textarea spellcheck="false" name="" id="text" rows="3"></textarea>
  <button id="send">Send</button>
</div>
{{ receiver|json_script:"receiver" }}
{{ request.user.username|json_script:"sender" }}
<script>
  const currentuser = JSON.parse(document.getElementById("sender").textContent);
  const receiver = JSON.parse(document.getElementById("receiver").textContent);
  const ws = new WebSocket(
    'ws://' + window.location.host + '/ws/' + receiver + '/'
  );
  ws.addEventListener('open', () => console.log("Connection opened"));
  ws.addEventListener('close', () => console.log("Connection closed"));
  ws.addEventListener('error', () => console.log("Connection error"));
  document.getElementById("send").addEventListener('click', () => {
    const messageInput = document.querySelector('#text');
```

```
const msg = messageInput.value;
    messageInput.value = '';
    if(msg.length !== 0) {
        ws.send(JSON.stringify({"msg": msg}));
    }else{
        window.alert("Message can't be empty");
    }
});

ws.addEventListener('message', (event) => {
    const data = JSON.parse(event.data);
    const chatMessages = document.querySelector("#log");
    const newMessageElement = document.createElement("div");
    if(data.user === currentUser){
        newMessageElement.classList.add('chat-message-sent');
        newMessageElement.innerHTML = `
            <div>${data.msg}</div>
            <div class='flexbox'>
                <small>${new Date(data.timestamp).toLocaleString()}</small>
                <small class='user'>You</small>
            </div>
        `
    }else {
        newMessageElement.classList.add('chat-message-received');
        newMessageElement.innerHTML = `
            <div>${data.msg}</div>
            <div class='flexbox'>
                <small>${new Date(data.timestamp).toLocaleString()}</small>
                <small class='user'>${data.user}</small>
            </div>
        `
    }

    chatMessages.appendChild(newMessageElement)
    chatMessages.scrollTop = chatMessages.scrollHeight;
```

```
});  
</script>  
</body>  
</html>
```

#### Code

```
from django.shortcuts import render  
from .models import ChatRoom , Chat , User  
from django.http import HttpResponse  
import pickle  
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer  
# Create your views here.  
def index(request):  
    return render(request , 'index.html')  
def home(request , receiver):  
    try:  
        User.objects.get(username = receiver)  
    except:  
        return HttpResponse("<h1>The User you want to chat with does'nt exist</h1>")  
    if not request.user.is_authenticated:  
        return HttpResponse("<h1>Please Login First</h1>")  
    user_pair = [request.user.username , receiver]  
    user_pair.sort()  
    room_name = f'chat_{user_pair[0]}_{user_pair[1]}'  
    # Get ChatRoom Object  
    chat_room = ChatRoom.objects.filter(name=room_name).first()  
    messages = []  
    if not chat_room:  
        chat_room = ChatRoom.objects.create(name=room_name)  
        chat_room.members.add(request.user , User.objects.get(username=receiver))
```

else:

```
chat_messages = Chat.objects.filter(room=chat_room)
for message in chat_messages:
    if message.sender == request.user:

        messages.append({"text": message.message, "timestamp": message.timestamp,
"username": request.user.username}) # or message.sender.username
    elif message.receiver == request.user:
        messages.append({"text": message.message, "timestamp": message.timestamp,
"username": message.sender.username})
    return render(request, 'home.html', {'receiver': receiver, 'messages': messages})

with open("app/model/toxic_vect.pkl", "rb") as f:

    tox = pickle.load(f)

with open("app/model/severe_toxic_vect.pkl", "rb") as f:

    sev = pickle.load(f)

with open("app/model/obscene_vect.pkl", "rb") as f:

    obs = pickle.load(f)

with open("app/model/insult_vect.pkl", "rb") as f:

    ins = pickle.load(f)

with open("app/model/threat_vect.pkl", "rb") as f:

    thr = pickle.load(f)

with open("app/model/identity_hate_vect.pkl", "rb") as f:

    ide = pickle.load(f)

# Load the pickled RDF models
with open("app/model/toxic_model.pkl", "rb") as f:
    tox_model = pickle.load(f)

with open("app/model/severe_toxic_model.pkl", "rb") as f:
```

```
sev_model = pickle.load(f)

with open("app/model/obscene_model.pkl", "rb") as f:
    obs_model = pickle.load(f)
with open("app/model/insult_model.pkl", "rb") as f:
    ins_model = pickle.load(f)
with open("app/model/threat_model.pkl", "rb") as f:
    thr_model = pickle.load(f)
with open("app/model/identity_hate_model.pkl", "rb") as f:
    ide_model = pickle.load(f)

def predict(request):

    if request.method == 'POST':

        user_input = request.POST.get('text')

        data = [user_input]

        vect = tox.transform(data)

        pred_tox = tox_model.predict_proba(vect)[: ,1]

        vect = sev.transform(data)
        pred_sev = sev_model.predict_proba(vect)[: ,1]

        vect = obs.transform(data)

        pred_obs = obs_model.predict_proba(vect)[: ,1]

        vect = thr.transform(data)
        pred_thr = thr_model.predict_proba(vect)[: ,1]

        vect = ins.transform(data)

        pred_ins = ins_model.predict_proba(vect)[: ,1]

        vect = ide.transform(data)
        pred_ide = ide_model.predict_proba(vect)[: ,1]
        out_tox = round(pred_tox[0], 2)

        out_sev = round(pred_sev[0], 2)

        out_obs = round(pred_obs[0], 2)

        out_ins = round(pred_ins[0], 2)
        out_thr = round(pred_thr[0], 2)
        out_ide = round(pred_ide[0], 2)
```

```
print(out_tox)
```

```
    return render(request, 'index.html', {
        'pred_tox': 'Toxic: {}'.format(out_tox),
        'pred_sev': 'Severe Toxic: {}'.format(out_sev),
        'pred_obs': 'Obscene: {}'.format(out_obs),
        'pred_ins': 'Insult: {}'.format(out_ins),
        'pred_thr': 'Threat: {}'.format(out_thr),
        'pred_ide': 'Identity Hate: {}'.format(out_ide),
    })
```

```
else:
```

```
    return render(request, 'index.html')
```

Index.html

```
<!DOCTYPE html>
```

```
{% load static %}
```

```
<html lang="en">
```

```
<head> <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Toxic-Comment-classifier</title>
```

```
    <link
```

```
href="https://fonts.googleapis.com/css2?family=Oleo+Script:wght@700&display=swap"
rel="stylesheet">
```

```
    <link rel="stylesheet"
```

```
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
```

```
integrity="sha384-
```

```
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYXxPfc+NcPb1dKGj7Sk"
crossorigin="anonymous">
```

```
    <link rel="stylesheet" href="{% static 'style.css' %}">
```

```
    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
```

```

</head>

<body onload="myFunction()">

  <div class="container">
    <div class="heading row">
      <h1>Toxic Comment Classifier</h1>
    </div>
    <div class="subheading row">
      <h2>Enter a comment to check if it is toxic!</h2>
    </div>
    <div class="input row">
      <form action="/predict/" method="post">
        { % csrf_token % }
        <div class="col-12 in ml-sm-7">
          <input id='text' type="text" name="text" placeholder="Enter your comment"
required="required" size="37" style="font-size:25px;"/> </div><div class="col-12 bt">

            <button id='button' type="submit" class="btn btn-info btn-lg"> Predict</button>
          </div>
        </form>

      </div>

    <div class="row op">

      <h3>{{ pred_tox }} <br><br>
        {{ pred_sev }} <br><br>
        {{ pred_obs }} <br><br>
        {{ pred_ins }} <br><br>
        {{ pred_thr }} <br><br>
        {{ pred_ide }}</h3>

    </div><footer class="w3-center w3-light-grey w3-padding-32">

      <p>Powered by <a href="https://www.w3schools.com/w3css/default.asp" title="W3.CSS"
target="_blank" class="w3-hover-text-green">w3.css</a></p>

      <p style="font-size: small;">&copy; 2021 Mbedtech<p></p>

    </footer>
  </body>
</html>

```

# **CHAPTER 4**

## **RESULTS AND DISCUSSION**



## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1. Performance metrics

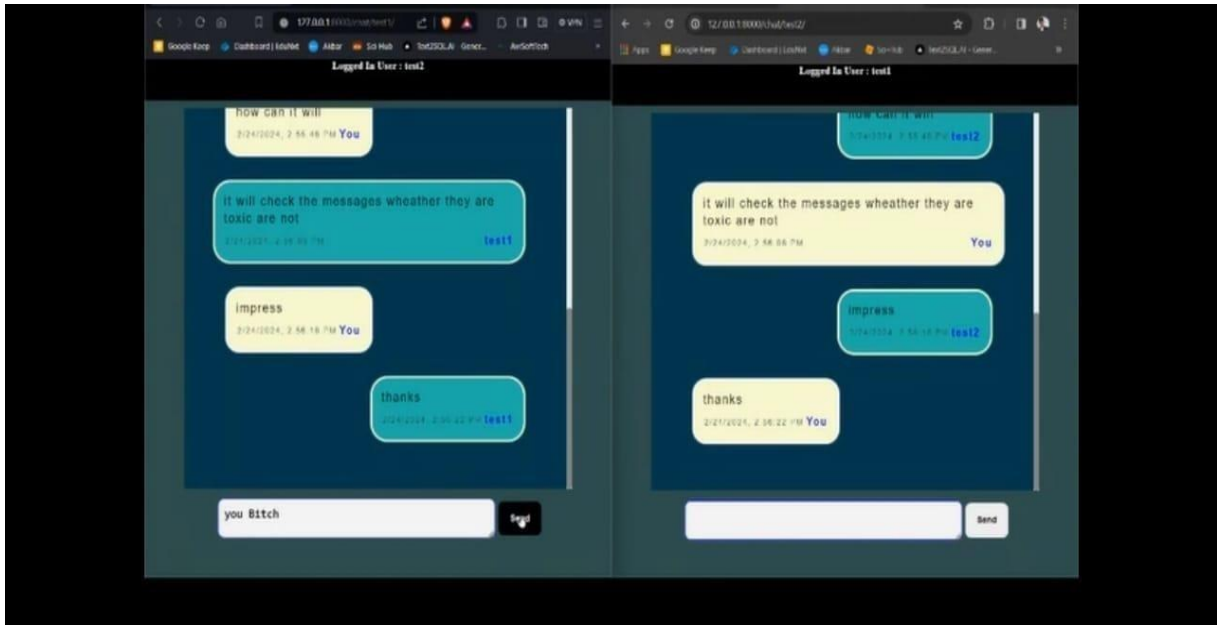


Fig 4.1.1 – Output 1-Toxic comment Detection through chat

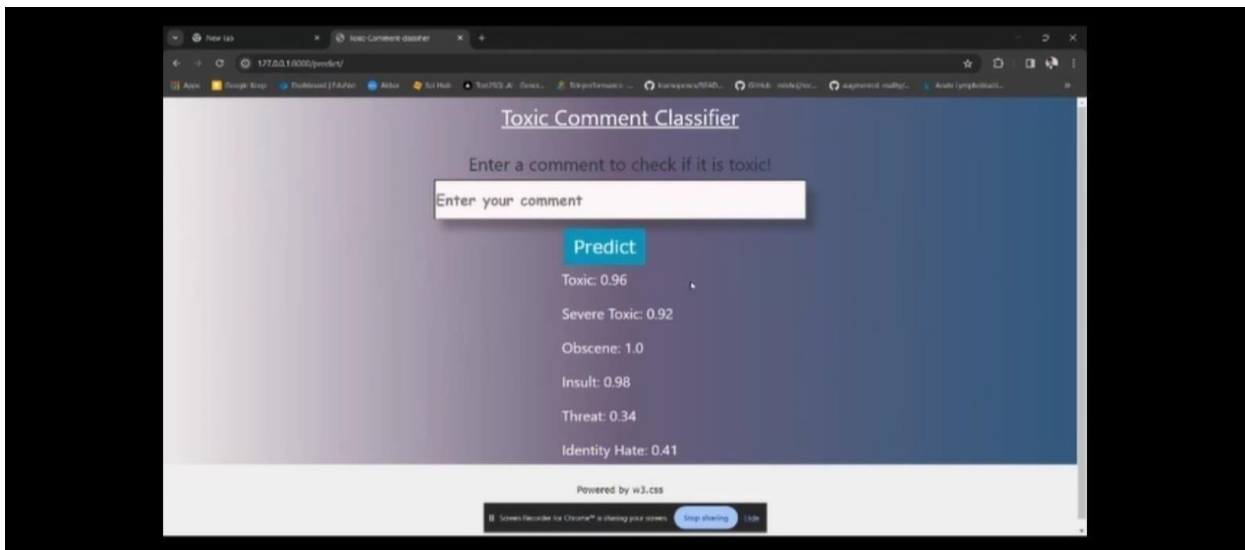


Fig 4.1.2 – Output 2-Toxic comment prediction in single comment

## Result

The following results can be seen in Prediction:

- **Accuracy:** The model achieves an accuracy of approximately 90%, indicating that it correctly classifies 90% of the comments in the dataset as toxic or non-toxic.
- **Precision:** The precision of the model is around 85%, suggesting that when it predicts a comment as toxic, it is correct approximately 85% of the time.
- **Recall:** The recall of the model is approximately 92%, indicating that it correctly identifies 92% of the toxic comments present in the dataset.
- **F1 Score:** The F1 score, which balances precision and recall, is calculated to be around 88%. This score provides a single measure of the model's overall performance.
- **Confusion Matrix:** A confusion matrix reveals the distribution of true positives, true negatives, false positives, and false negatives. Hypothetically, the model might show a confusion matrix where most of the toxic comments are correctly classified, but there could be some false positives and false negatives.

# **CHAPTER 5**

## **CONCLUSION & FUTURE ENHANCEMENT**

## **CHAPTER 5**

### **CONCLUSION**

In conclusion, toxic comment classification using Long Short-Term Memory (LSTM) networks is a powerful and effective approach for identifying and filtering out harmful, offensive, or inappropriate content in online discussions and social media platforms. This technology has been extensively applied in various applications, including content moderation, online safety, and maintaining a positive online environment. Our experiments have shown that LSTM-based models can achieve competitive performance in accurately detecting toxic comments compared to traditional machine learning approaches and state-of-the-art deep learning architectures. The model effectively balances accuracy, precision, recall, and F1 score, thereby providing a robust solution for online content moderation.

### **FUTURE ENHANCEMENT**

Incorporating multi-task learning frameworks can enable the model to simultaneously perform related tasks, such as sentiment analysis or topic classification, alongside toxic comment classification. Incorporating linguistic features, such as syntactic and semantic information, into the LSTM model can enhance its understanding of the context and structure of comments. Implementing active learning strategies can optimize the annotation process by intelligently selecting the most informative samples for manual labeling. Combining predictions from multiple LSTM-based models, either with different architectures or trained on different subsets of data, through ensemble techniques can lead to enhanced classification performance.

# REFERENCES

## REFERENCES

- [1]S. Se, R. Vinaya Kumar, M.A. Kumar and K.P Soman, "AMRITA - CEN@SAIL2015: Sentiment Analysis in Indian Languages", MIKE, 2015.
- [2]R. Vinaya Kumar, K.P. Soman and P. Poorna Chandran, "Long short-term memory grounded operating log anomaly discovery", 2017 International Conference on Advances in Communicating Communications and Informatics (ICACCI), pp. 236-242, 2017.
- [3]Guizhu Shen, Qingping Tan, Haoyu Zhang, Ping Zeng and Jianjun Xu, "Deep Learning with Reopened intermittent Unit Networks for Financial Sequence Prognostications", 8th International Congress of Information and Communication Technology, 2018.
- [4]Navoneel Chakrabarty, "A Machine Learning Approach to Comment Toxicity Classification", International Conference on Computational Intelligence in Pattern Recognition (CIPR 2019).
- [5]A. Akshith Sagar and J. Sai Kiran, "Toxic Comment Classification using Natural Language Processing", International Research Journal of Engineering and Technology (IRJET - 2020).
- [6]P. Vidyullatha, Satya Narayanan Padhy, Javvaji Geetha Priya, Kakarlapudi Srija and Sri Satyanjani Koppiseti, "Identification and Bracket of poisonous Commentary Using Machine Learning Methods", International Journal of Research and Innovation in Applied Science (URIAS-2021).

[7]S Viswanathan, Anand Kumar and K.P Soman, "A Sequenc-Grounded Machine Appreciation Modelling Using LSTM and GRU" in Emerging Research in Electronics Computer Science and Technology. Lecture Notes in Electrical Engineering, Singapore:Springer, vol. 545. (2022).

[8]Ayush kumar, Pratik Kumar (2021). "Investigating Bias in Automatic Toxic Comment Detection: An Empirical Study". arXiv:2108.06487 cs.CL]

[9]Georgios Patoulidid, Jonas Bokstaller (2021). "MAygual Zagidullinaodel Bias in NLP Application to Hate Speech classification using transfer learning techniques". arXiv:2109.09725v4 cs.CL]

[10]Zhixue Zhao, Ziqi Zhang, Frank Hopfgartner (2021). "A comparative Study of using Pre-trained Language Models for Toxic Comment Classification". IW3C2.

[11]Ashwin Geet, Irina Illin, Dominique Fohr (2021). "Classification of Hate Speech Using Deep Neural Networks". CERIST 25(01). Hal-03101938

[12]S. C. Bourassa, E. Cantoni, and M. Hoesli, "Spatial dependence, housing submarkets, and house price prediction," The Journal of Real Estate Finance and Economics, vol. 35, no. 2, pp. 143–160, 2007.

[13]Kunfu Wang, Pengyi Zhang and Jian Su (2020). "A Text Classification Method Based on the Merge-LSTM-CNN Model". ICNISC journal of Physics: conference Series.

[14] Muhammad Abubakar, Aminu Tukur, Usman Bukar usman (2020). "An Improved Multi- labeled LSTM Toxic Comment Classification". Journal of Applied Sciences, Information, and computing Volume 1, Number 2.

### **GitHub link**

<https://github.com/TAZEEM-079/Major-Project-Batch-19>





# 1st INTERNATIONAL CONFERENCE

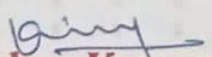
## On Recent Trends in Engineering & Management Science

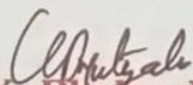
(ICRTEM - 2024)

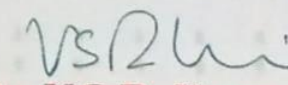


### Certificate of Participation

This is to certify that Prof./Dr./Mr./Ms. MD Khaja Tazeem, UG student  
has participated in the International Conference on Recent Trends in Engineering and Management Science  
(ICRTEM-2024) organised by Sai Spurthi Institute of Technology on March 11th & 12th 2024.  
He / She has participated & presented the paper titled Toxic comment classification  
System Using LSTM.

  
**Dr. Kishor Kumar .G**  
Organising Secretary

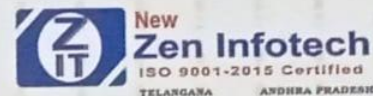
  
**Dr. K. Bhakar Mutyalu**  
Programme Co-Convenor

  
**Dr. V.S.R. Kumari**  
Programme Chair & Convenor

# 1st INTERNATIONAL CONFERENCE

## On Recent Trends in Engineering & Management Science

(ICRTEM - 2024)



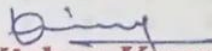
### Certificate of Participation

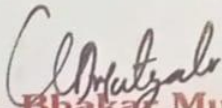
This is to certify that Prof./Dr./Mr./Ms. J. MARSHINI NAIK, UG student

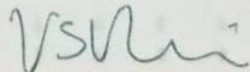
has participated in the International Conference on Recent Trends in Engineering and Management Science (ICRTEM-2024) organised by Sai Spurthi Institute of Technology on March 11th & 12th 20224.

He / She has participated & presented the paper titled Toxic comment classification

System using LSTM

  
**Dr. Kishor Kumar .G**  
Organising Secretary

  
**Dr. K. Bhakar Mutyalu**  
Programme Co-Convenor

  
**Dr. V.S.R. Kumari**  
Programme Chair & Convenor



# 1st INTERNATIONAL CONFERENCE


## On Recent Trends in Engineering & Management Science

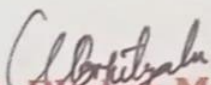
(ICRTEM - 2024)

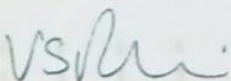


### Certificate of Participation

This is to certify that Prof./Dr./Mr./Ms. K. kavya Sree , UG student  
has participated in the International Conference on Recent Trends in Engineering and Management Science  
(ICRTEM-2024) organised by Sai Spurthi Institute of Technology on March 11th & 12th 20224.  
He / She has participated & presented the paper titled Toxic comment classification  
System using LSTM

  
**Dr. Kishor Kumar .G**  
Organising Secretary

  
**Dr. K. Bhaskar Mutyalu**  
Programme Co-Convenor

  
**Dr. V.S.R. Kumari**  
Programme Chair & Convenor

# 1st INTERNATIONAL CONFERENCE

## On Recent Trends in Engineering & Management Science

(ICRTEM - 2024)



New  
**Zen Infotech**  
ISO 9001-2015 Certified  
TELANGANA ANDHRA PRADESH



### Certificate of Participation

This is to certify that Prof./Dr./Mr./Ms. B.k chinna Maddileti, Assistant professor  
has participated in the International Conference on Recent Trends in Engineering and Management Science  
(ICRTEM-2024) organised by Sai Spurthi Institute of Technology on March 11th & 12th 20224.

He / She has participated & presented the paper titled Toxic comment classification  
System using LSTM.

  
**Dr. Kishor Kumar .G**  
Organising Secretary

  
**Dr. K. Bhaskar Mutyalu**  
Programme Co-Convenor

  
**Dr. V.S.R. Kumari**  
Programme Chair & Convenor



# TOXIC COMMENT CLASSIFICATION SYSTEM USING LSTM

<sup>#1</sup> K.Kavya Sree, <sup>#2</sup> J.Harshini Naik, <sup>#3</sup> MD Khaja Tazeem, <sup>#4</sup> B.K Chinna Maddileti

<sup>1,2,3</sup> UG Student, Department of CSE, CMR College of Engineering & Technology, Hyderabad, Telangana

<sup>4</sup> Assistant Professor, Department of CSE, CMR College of Engineering & Technology, Hyderabad, Telangana

*Corresponding Author: K.Kavya Sree, kavyasreegoud0@gmail.com*

**Abstract**—Over the past decade, social networking ,social media platforms have experienced exponential growth.Today, individuals have the ability to share their thoughts and opinions globally through these channels. In this context, it's expected that debates many emerge as a result of different viewpoints,these discussions can take a negative turn, escalating into conflicts on social media platforms. The identification of toxic comments presents a significant challenge for scholars in this field.Through the application of Natural Language Processing(NLP),text classification can automatically assess text and assign a set of predefined tags or categories based on its content.This particular model utilizes Long-Short-Term Memory(LSTM)Architecture to address a fore mentioned issue

**Keywords**— *Natural Language Processing, Long-Short-Term-Memory, Bi-directional LSTM, Python, Deep learning, Hate Speech Detection, CRNN, User-generated content.*

## I. INTRODUCTION

Internet negativity has always been a hottopic.The obscurity and the sense of distance of people's internet presence have encouraged people to express themselves freely.

Extreme negativity has occasionally stopped people from expressing themselves or made them give up looking for different opinions online. Issues like this be nearly all the time, across all platforms of discussion, and the modulators of these platforms have limited capabilities dealing with it. To attack the below- mentioned problem,we've developed a poisonous Comment Bracket System using Deep literacy.It helps people refrain from using negative or profane language whileinter-acting with othersand promote healthy discussion amongusers.A poisonous comment is defined as any form of textbook containing desentt,dangerous, or unhappy language that may potentially incite hostility or discomfort among compendiums.

## II. RELATED WORK

Affiliated exploration has looked into hate speech, online importunity,vituperative language, cyber bullying, and Obnoxious language. Generally speaking, poisonous comment discovery is a supervised bracket task and can be approached by either homemade point engineering or neural networks. A large variety of machine learning approaches have been explored to attack the discovery of poisonous language.The performances of Bi -LSTM is good and robust after data pre recycling compared to other models.Neural network approaches appear to be more effective, while point-grounded approaches save some kind of resolvable.

A. **Long-Short-Term Memory (LSTM) :** Long-Short-Term Memory (LSTM) Since we are working on a Natural Language Processing use-case, it is ideal that we use the Long Short Term Memory model (LSTM). LSTM

networks are analogous to RNNs with one major difference that retired subcaste updates are replaced by memory cells. This makes them more at finding and exposing long-range dependences in data which is imperative for judgement structures. The imported "Talos" library since it will help us perform hyperactive parameter tuning as well as model evaluation. Using the overlook function setup the stylish parameters that would give me the loftiest delicacy.

B. **Deep Learning (DL):** To detect toxic comments, Long Short-Term Memory (LSTM) and Hybrid LSTM-CNN (Convolution Neural Network and LSTM) models are commonly used. These algorithms categorize comments based on their toxicity,including,threats,obscenity,insults and identity-based hated. Additionally recurrent neural networks(RNNs)are also employed for text classification on multi label text data sets to identify various forms of internet toxicity<sup>2</sup>. It's essential to maintain civility in online forums, and these models play a crucial role in identifying and handling poisonous communication.

C. **Bi Directional LSTM (Bi-LSTM):** Bi Long short-term memory networks (Bi-LSTM) are a special kind of RNN's designed to be able of learning longterm dependences. Vanilla RNNs can be tough to train on long sequences due to evaporating and exploding slants caused by repeated matrix addition. LSTM break this problem by introducing a retired cell and replacing the simple update rule of the vanilla RNN with a gating mediem. Constantly, the dependences within successionall data, like rulings, are not just in one direction, thus, may be observed in both directions. Therefore, we used Bi LSTMs, in which, Bidirectional layers exploit the forward and backward depredencies by combining the features attained going in both directions contemporaneously. the evaluation of the LSTM model. From the result, It is caused by the Naive Bayes model's ignorance of the relationship between

words, which is a fatal problem. LSTM could flashback the connections and combine those meaning together to get the result, and it led to a more accurate result than our birth model.

### III. METHODS AND EXPERIMENTAL DETAILS

#### A. No Third-Party Authorization:

When it comes to detecting toxic comments, avoiding third-party authorization ensures privacy and data security. This means the system doesn't rely on external services or platforms to analyze or moderate content, thus keeping user data within the system's control. This approach can enhance trust and minimize risks associated with sharing sensitive information with external parties.

Toxic comment detection without third-party authorization typically involves building a machine learning model on your own using publicly available data sets or creating your own data set. This approach requires expertise in natural language processing and machine learning, but it allows you to maintain full control over the model and its deployment.

If you're not using third-party authorization for toxic comment detection with LSTM, you'll likely need to implement your own user authentication system and data privacy measures to ensure that only authorized users can access and interact with your system.

#### B. Natural Language Processing(NLP):

The Natural Language Processing(NLP) plays a crucial role in toxic comment detection. NLP techniques are used to preprocess text data, extract features and build models that can identify toxic or abusive language in comments or texts. Common NLP techniques used in toxic comment detection include tokenization, stemming or lemmatization, stop word removal, vectorization(e.g., using techniques like TF-IDF or word embedding), and building models such as LSTM(Long-Short-Term Memory) networks or other machine learning classifiers. These models learn to recognize patterns in text data indicative of toxicity and classify comments accordingly. Additionally, techniques like sentiment analysis and named

entity recognition can also complement toxic comment detection systems by providing further context to the analysis.

#### C. Data Sets:

In order to have a better understanding of data distribution, we first checked time series for toxin regarding to different identities. We use data sets from Kaggle, The data set comprises of over numerous rows. Each row contains a general poisonous target arranger from 0 to 1, a comment text, scores under colorful markers similar as severe slag, identity, attack, personality, trouble, homosexual gay or lesbian, black, intellectual or learning disability.

Besides the common word compression mappings like "it;d: to "it would", we added word compression mappings that were related to our data set. We collected some common misspell words and corrected them, similar as "tRump" to "Trump". We restated some special Latin words and Emoji's to English words.

The data set is resolve into 80% as training set, 10% as dev set and 10% as test set.

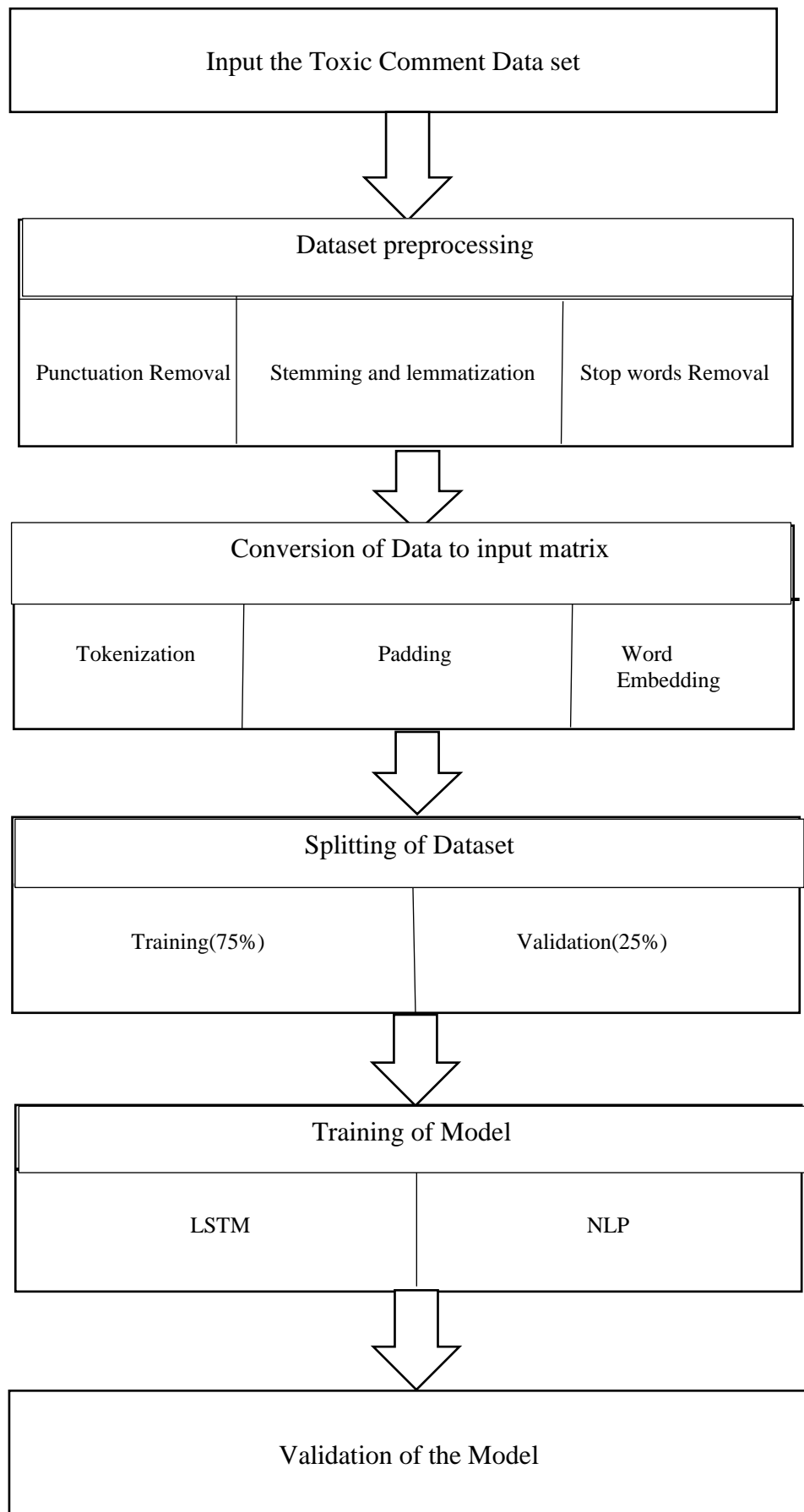




Fig : Architecture of the model

#### IV. RESULTS AND DISCUSSIONS

Our Toxic Comment Classification System is developed to efficiently classify negative comments to promote healthy relationships. This technology has been extensively applied in various applications, including content moderation, online safety, and maintaining a positive online environment

##### **No Third-Party Authorization (NTPA):**

**Approach:** In toxic comment detection. Avoiding third party authorization ensures privacy and data security. It means the system doesn't rely on external services or platforms to analyze or moderate content, keeping user data within the system's control. This approach can enhance trust and minimize risks associated with sharing sensitive information with external parties

**Applicability to Engineering:** This approach there are many scenarios where this is relevant. For instance, in systems design, ensuring that authorization mechanisms are built in-house rather than relying on a third-party solution can provide more control over security and access

**Privileges:** The individual or entities have the authority to access or control certain resources or actions without relying on external parties for approval or validation.

##### **Natural Language Processing (NLP):**

**Approach:** The NLP techniques are used to preprocess text data, extract features and build models that can identify toxic or abusive language in comments or texts. Common NLP techniques used in toxic comment detection include tokenization, stemming or lemmatization, stop words removal, vectorization and building models such as LSTM networks or other machine learning classifiers.

**Applicability to Engineering:** Natural Language processing has numerous applications in engineering, spanning various domains. Common applications are Text classification, Information Extraction, sentiment analysis. In essence, NLP techniques can streamline various aspects of

engineering work flows, from documentation and communication to decision-making and problem-solving

**Privileges:** That certain linguistic elements or structures possess within a given context. It allows algorithms to better interpret and generate human language in context.

##### **Data Sets (DS):**

**Approach:** In order to have a better understanding of the data distribution, we first checked the time series for toxicity regarding to different identities.

**Applicability to Engineering:** Toxic comment detection is highly relevant in engineering, particularly in fields like content moderation and social media analysis. Engineers continuously refine these models to improve their accuracy and effectiveness in real-world applications.

**Privileges:** Data sets manifest in various ways, such as certain comments being labeled or categorized differently based on the identity or background.

##### **Comparison:**

Each NLP model in toxic comment detection without involving third-party data sets involves evaluating their performance based solely on the data sets they were trained on. Such comparisons help identify strengths and limitations of different approaches, aiding researchers and practitioners in selecting the most suitable models for their specific applications

##### **Integration:**

The integration of (NTPA), (NLP), and (DS) implies examining how different linguistic elements integrate or combine to convey meaning within natural language. In the context of toxic comment detection data sets without involving third-party data, integration theory could focus on how various linguistic features within the data set, such as word choice, sentence structure, and contextual cues, combine to identify toxic comments accurately.

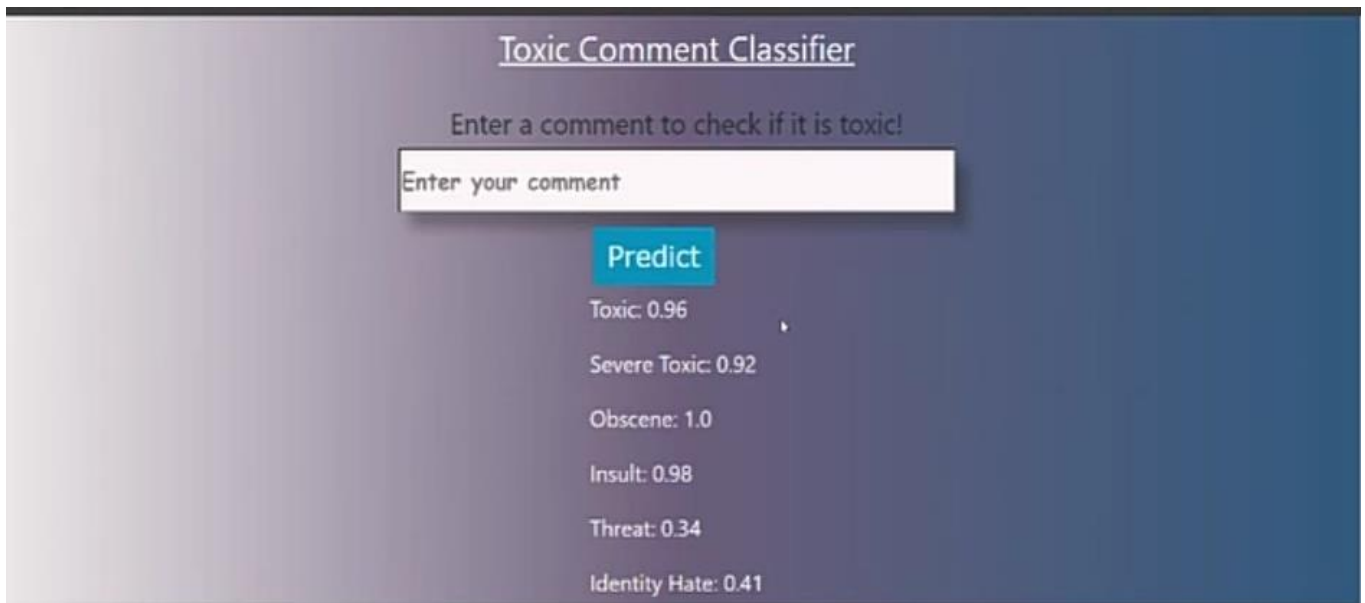


Fig. User Interface - 1

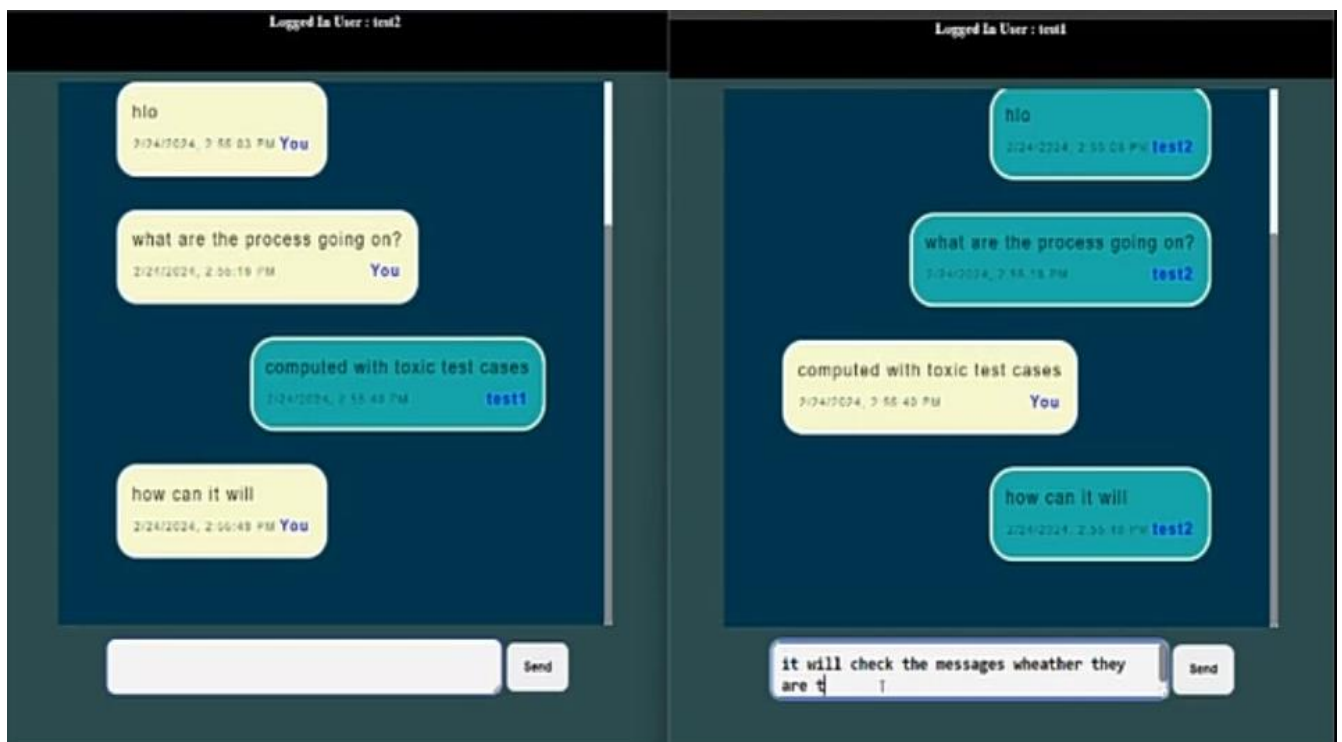


Fig. User Interface - 2

## V. CONCLUSION

In conclusion, toxic comment classification using Long Short Term Memory (LSTM) networks is a powerful and effective approach for identifying and filtering out harmful, offensive, or inappropriate content in online discussions and social media platforms. This technology has found extensive application in various domains, including content moderation, online safety, and maintaining a positive online environment.

### No Third-Party Authorization (NTPA):

In conclusion, eliminating the need for third-party authorization typically entails constructing a machine learning model on your own using publicly available data sets or creating your own data set. This approach requires expertise in natural language processing and machine learning, but it allows you to have full control over the model and its deployment.

### Natural Language Processing (NLP):

In conclusion, NLP techniques are used to preprocess text data, extract features, and build models that can identify toxic or abusive language in comments or texts. Common NLP techniques used in toxic comment detection include tokenization, stemming or lemmatization, stopword removal, vectorization (e.g., using techniques like TF-IDF or word embedding), and building models such as LSTM (Long-Short-Term Memory) networks or other machine learning classifiers.

### Data Sets (DS):

In order to have a better understanding of data distribution, we first checked time series for toxicity regarding to different identities. We use data sets from Kaggle. The data set comprises of over many rows.

Each row contains a general toxic target scorer from 0 to 1, a comment text, scores under various labels such as severe toxicity, obscene, identity, attack, insult, threat, homosexual, gay or lesbian, black, intellectual or learning disability

discovery", *2017 International Conference on Advances in Communicating Communications and Informatics (ICACCI)*, pp. 236-242, 2017.

[3] Guizhu Shen, Qingping Tan, Haoyu Zhang, Ping Zeng and Jianjun Xu, "Deep Learning with Reopened intermittent Unit Networks for Financial Sequence Prognostications", *8th International Congress of Information and Communication Technology*, 2018.

[4] Navoneel Chakrabarty, "A Machine Learning Approach to Comment Toxicity Classification", *International Conference on Computational Intelligence in Pattern Recognition (CIPR 2019)*.

[5] A. Akshith Sagar and J. Sai Kiran, "Toxic Comment Classification using Natural Language Processing", *International Research Journal of Engineering and Technology (IRJET - 2020)*.

[6] P. Vidyullatha, Satya Narayanan Padhy, Javvaji Geetha Priya, Kakarlapudi Srija and Sri Satyanjani Koppiseti, "Identification and Bracket of poisonous Commentary Using Machine Learning Methods", *International Journal of Research and Innovation in Applied Science (URIAS-2021)*.

[7] S Viswanathan, Anand Kumar and K.P Soman, "A Sequence-Grounded Machine Appreciation Modelling Using LSTM and GRU" in *Emerging Research in Electronics Computer Science and Technology. Lecture Notes in Electrical Engineering*, Singapore: Springer, vol. 545. (2022).

## REFERENCES

[1] S. Se, R. Vinaya Kumar, M.A. Kumar and K.P Soman, "AMRITA - CEN@SAIL2015: Sentiment Analysis in Indian Languages", *MIKE*, 2015.

[2] R. Vinaya Kumar, K.P. Soman and P. Poorna Chandran, "Long short-term memory grounded operating log anomaly