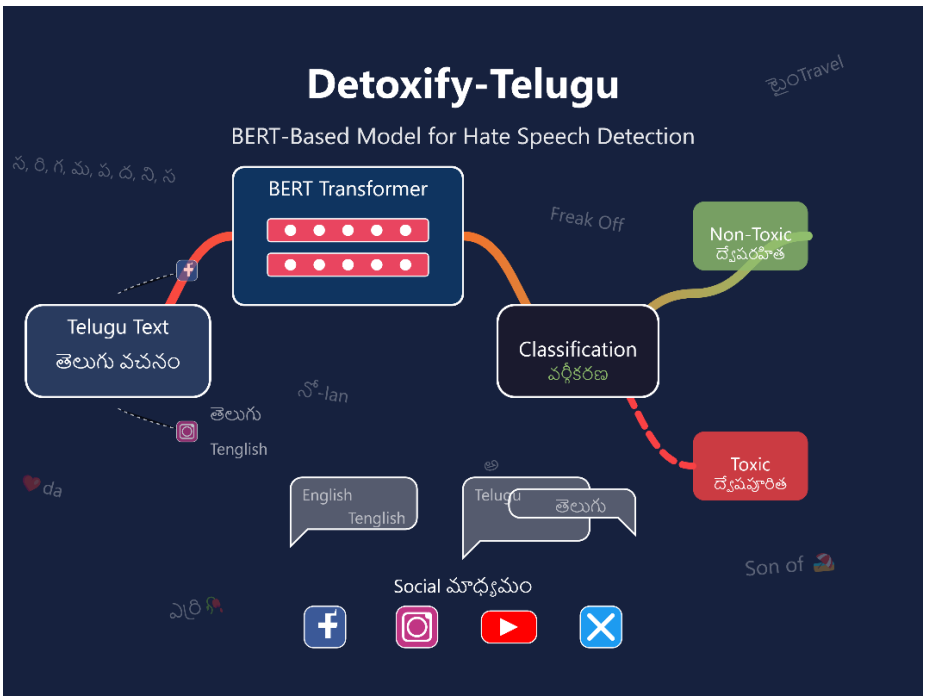
Detoxify-Telugu



A Fine-Tuned BERT-Based Language Models for Hate Speech Detection in Telugu

Author: Pavan Yellathakota
Institution: Clarkson University, Advanced Data Science Department
Supervisor: Prof. Boris Jukic

Table of Contents

Abstract

Social media platforms have become fertile grounds for fan wars, abuse, and targeted toxicity—particularly within regional entertainment spheres such as Telugu cinema. Languages like Telugu and Tenglish (Telugu in Roman script) often lack automated moderation tools due to their low-resource status in NLP. Detoxify-Telugu addresses this gap by delivering a fully modular, multilingual toxicity detection framework that enables binary and multi-class classification of hate speech across various toxicity types including caste slurs, threats, sexual abuse, and common insults.

Built upon Transformer-based models (like BERT) and a streamlined UI, the system empowers users—technical or otherwise—to annotate, clean, balance, train, evaluate, and deploy models seamlessly. It supports fine-tuning with customizable parameters, dynamic dataset balancing, and multilingual input handling. The platform demonstrates that even small-scale models (1M–50M parameters) can yield strong results with well-engineered pipelines and curated datasets. Evaluation reveals high accuracy in binary classification and actionable insights from multi-class predictions, validating the system's utility in both production and research contexts.

Keywords: Regional NLP, Toxicity Detection, Tenglish, Telugu, Streamlit, Hate Speech, BERT, Multi-class Classification, Generative AI

I. Introduction

Hate speech is a growing problem on the internet, especially with the huge amount of content being posted on social media. While many tools exist to detect toxic content in English and a few other major languages, there aren't many solutions for regional Indian languages like **Telugu**.

Telugu is one of the main languages in South India, spoken by over 96 million people. But there are very few tools or datasets available to detect hate speech in Telugu properly.

This gap has allowed harmful content like **online bullying, caste-based abuse, gender insults, and political hate** to spread without control. This issue is even more complicated in Telugu because people often mix English and Telugu (called **Tenglish**), use different regional slang, and write Telugu in English letters with different spellings.

To solve this, we are introducing **Detoxify-Telugu** an all-in-one system that helps detect hate speech in Telugu and Tenglish content. Inspired by the Detoxify tool for English, our system includes a simple interface where users can label data, clean it, train models, test results, and even make predictions.

Detoxify-Telugu uses smart language models like **BERT** to detect whether content is toxic or not. It can also classify toxic messages into types like bad language, caste slurs, threats, etc. By learning from real Telugu content, it helps make the internet safer for Telugu-speaking users.

II. Motivation & Problem Statement

Every day, tons of Telugu content is shared online, but current hate speech tools mostly focus on English or Hindi. As a result, harmful Telugu content often goes unchecked.

The problem is made worse because:

1. Mix of English and Telugu (Tenglish): People write Telugu words using English letters, which makes it hard for normal tools to understand. For example, “Poka chesava ?” or “repu cheddam ra ?” are typed in Roman script.

2. Regional Slang: Telugu is spoken differently in Telangana, Coastal Andhra, and Rayalaseema. Each region has its own slang and ways of saying things.

3. Spelling Differences: People write the same word in different ways based on sound. For example, “chetha” (“chetta”) for the Telugu word “చెత్త,” or use extra letters like “byaad” for “bad.”

4. Toxic Categories: It’s not enough to just say something is toxic or not. We also need to know if it’s a **threat**, **sexual abuse**, **caste insult**, **gender abuse**, or **political hate**.

5. **Lack of Good Data:** Most Telugu hate speech datasets are small, cleaned too much, or don’t have different categories. Some even remove the bad words, which makes it hard to train real-world models.

So, **Detoxify-Telugu** aims to:

- Provide a real dataset with unfiltered toxic content

- Detect different types of toxic messages, not just yes/no

- Handle Tenglish, spelling errors, and regional slang

- Offer a user-friendly interface so even non-tech users can use it

III. Literature Review

Thanks to recent advances in machine learning, especially **Transformer models** like **BERT**, we now have better tools for text classification (like detecting toxic content).

The **Detoxify** project showed that BERT-based models work well for detecting hate in English. But these models don't work well for Telugu because the language, script, and cultural context are different.

Some models, like **MuRIL**, **Indic BERT**, and **XLM-R**, are designed to work with many Indian languages. However, they still struggle with code-mixed Telugu (Tenglish) and informal writing styles.

There is a small dataset called **Telugu-Hatespeech** on HuggingFace, but it only labels texts as toxic or not. It doesn't provide more details like the type of toxicity.

In other languages like Tamil and Hindi, some researchers have used character-level models and back-translation techniques to deal with mixed-language input. But these methods don't always work well when there is very little labeled data, like in Telugu.

What makes **Detoxify-Telugu** different:

- It uses **real, unfiltered data** with actual abusive language and slang

- It supports **multi-type classification** (not just toxic or not)

- It understands **Tenglish, spelling mistakes, and regional slang**

- It comes with an easy-to-use system for labeling data, training models, and making predictions

In short, Detoxify-Telugu is designed to fill a serious gap in hate speech detection for Telugu and similar regional languages.

IV. Dataset Overview

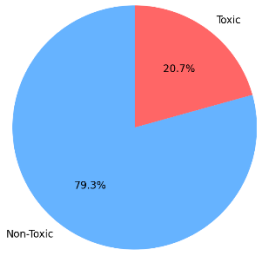
To train and evaluate the Detoxify-Telugu platform, we built a diverse dataset using multiple sources and methods to reflect real-world Telugu and Tenglish toxicity. Below is a summary of how the dataset was collected and structured.

Sources Used:

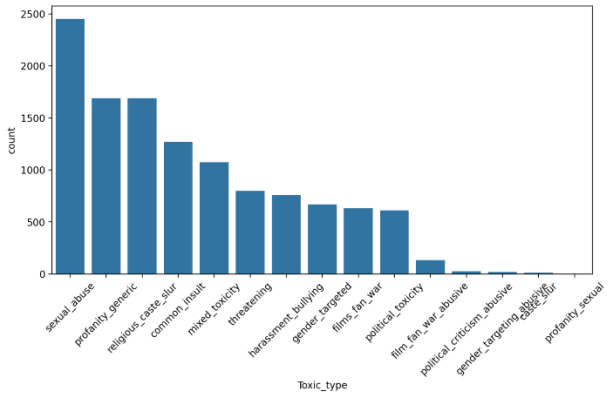
The Detoxify-Telugu dataset was created using a mix of real, synthetic, and curated data from the following sources:

- YouTube Comments collected using a custom Python script (Scrape.py) with Selenium
- Telugu-Hatespeech Dataset from HuggingFace (a publicly available dataset for Telugu)
- Manually Collected Toxic Word Lists — over 2,000 records labeled and verified
- Synthetic Data — around 7,000 to 8,000 examples were generated to balance and enrich the dataset

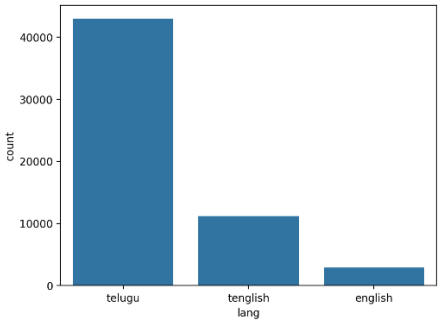
Toxic vs Non-Toxic Distribution



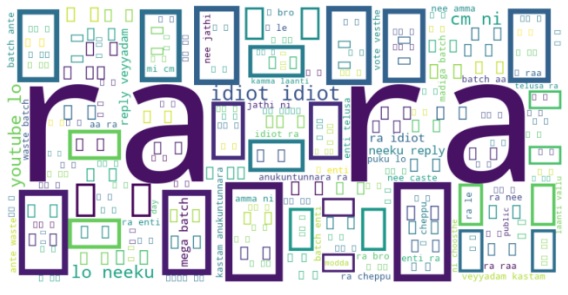
Toxic Type Distribution



Language Distribution



WordCloud of Toxic Texts



Class Distribution

The dataset is imbalanced, meaning there are more non-toxic examples than toxic ones.

Non-Toxic: 79.3% Toxic: 20.7%

This imbalance can be due to:

- A natural lower occurrence of toxic content in real posts
- Possible errors or limitations in the way data was labeled
- Platforms often filter or remove hate speech before it becomes visible

Visualizations Provided

To better understand the dataset, we include the following visual insights:

- Toxic vs Non-Toxic Distribution (Pie/Bar Chart)
- Toxic Type Distribution – shows how different types of toxicity (e.g., sexual abuse, caste slurs, threats) are represented
- Language Distribution – includes Telugu, English, and code-mixed (Tenglish) texts
- Word Cloud of Toxic Texts – shows the most frequently used offensive or harmful words.

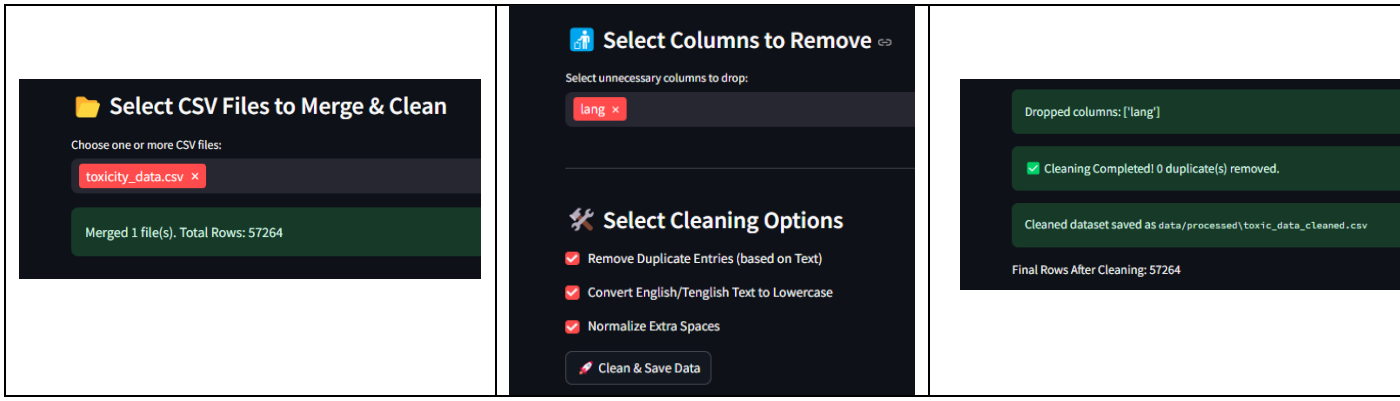
These visuals help in understanding patterns in toxic speech and guide model training and evaluation.

V. Data Preprocessing

Preprocessing plays a key role in building an accurate hate speech detection system, especially in a multilingual and code-mixed environment like Telugu and Tenglish. In the Detoxify-Telugu project, we designed a structured pipeline to clean, annotate, and balance the data before using it for training machine learning models.

Preprocessing Scripts Used

We developed three main Python scripts to handle different parts of the preprocessing workflow:

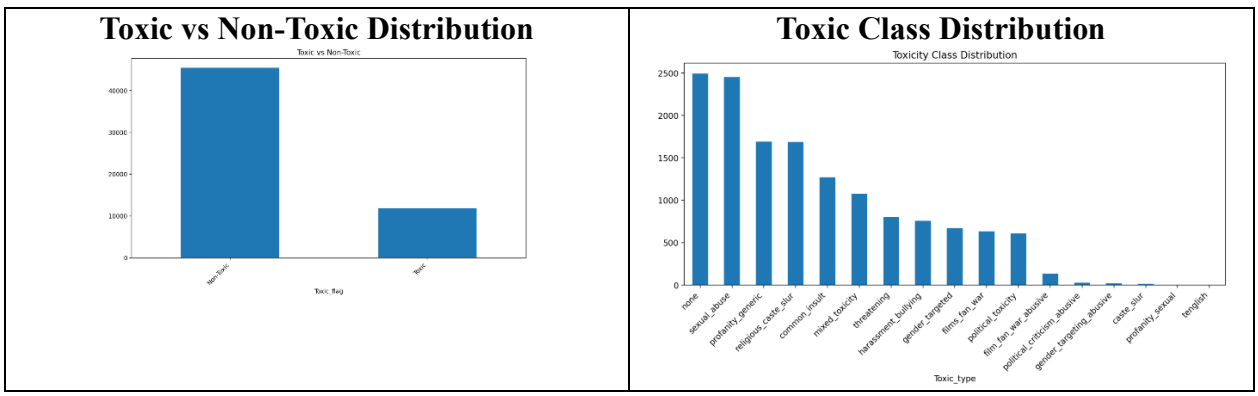


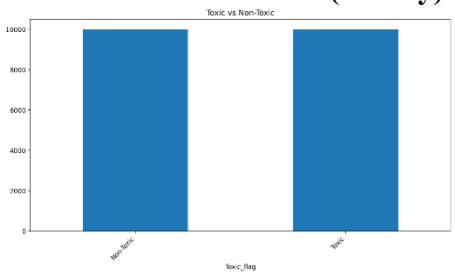
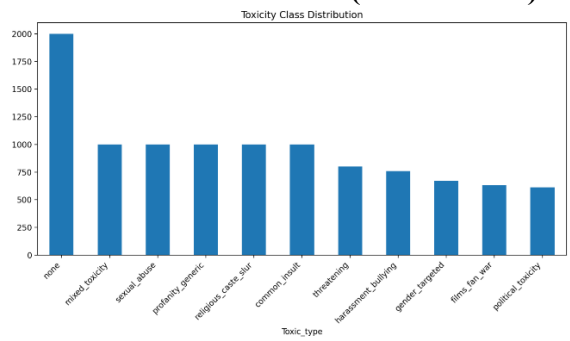
data_cleaning.py – Cleans the raw text by removing noise, special characters, and duplicate entries.

Note: Around 20% of the initially labeled data was removed during this step due to duplication or poor quality.

data_annotation.py –Automatically assigns labels to comments based on keyword matches and script detection (Telugu vs Romanized Telugu).

data_balancing.py – Balances the dataset to ensure equal representation of toxic and non-toxic samples (or toxic subcategories).



Balanced Distribution (Binary)	Balanced Distribution (Multi- Class)
	

Balanced Dataset Versions

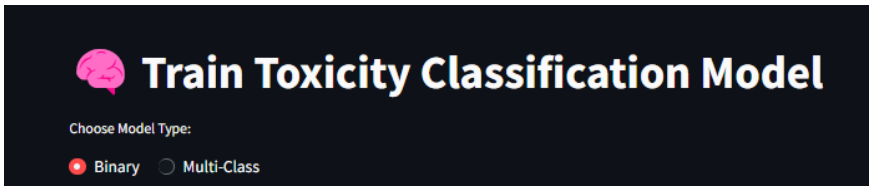
Due to the original imbalance (more non-toxic content), we created two balanced versions of the dataset:

- **Binary Classification Dataset** : Contains an equal number of toxic and non-toxic samples. (File saved as: data/training/binary/detoxify_binary.csv)
- **Multi-Class Classification Dataset** : Balanced across different toxicity types. (File saved as: data/training/multi/detoxify_multi.csv)

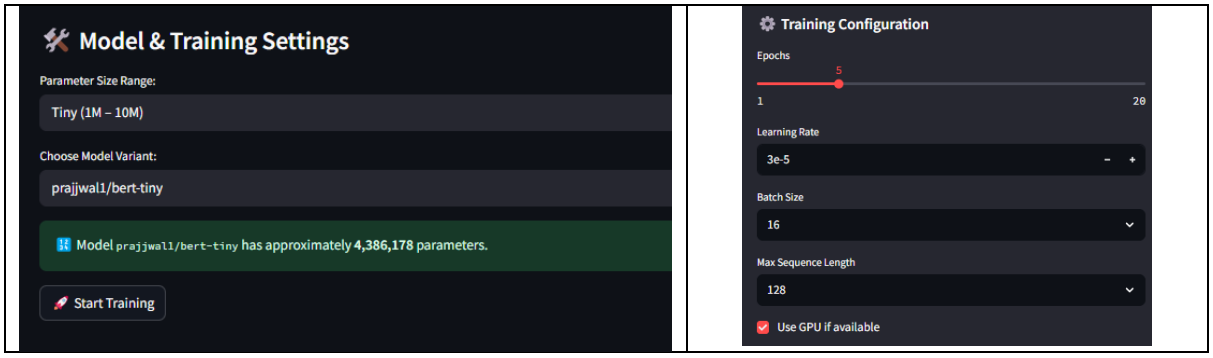
These cleaned and balanced datasets ensure better model performance during training and testing.

VI. Model Design

Detoxify-Telugu is built upon Transformer-based architectures, specifically variants of BERT, which have proven highly effective for text classification tasks.



The system supports both Binary Classification (toxic vs non-toxic) and Multi-Class Classification (categorizing types of toxicity such as threats, abuse, slurs, etc.).



1. Flexible Architecture Selection

Users can choose between:

- **Binary Classification:** A simple yes/no model indicating the presence of toxicity.
- **Multi-Class Classification:** A more granular model identifying specific types of toxicity (e.g., “profanity_generic”, “harassment_bullying”).

This flexibility allows the platform to adapt to different moderation needs and downstream use cases.

2. Pretrained BERT Variants

To optimize for accuracy and available computing resources, users can choose from a range of pre-integrated models:

- Tiny (~4M – 10M parameters): e.g., prajjwal1/bert-tiny, google/bert_uncased_L-2_H-128_A2
- Small (~10M – 25M): e.g., prajjwal1/bert-mini, microsoft/MiniLM-L6-v2
- Medium (~66M – 110M): e.g., distilbert-base-uncased, bert-base-uncased

This approach ensures the platform scales across environments — from laptops to GPU-backed servers.

3. Training Configuration Options

The user interface allows full control over key hyperparameters:

- **Epochs:** 1 to 20
- **Learning Rate:** Adjustable via numeric input (default 3e-5)
- **Batch Size:** Selectable (8, 16, 32)
- **Max Sequence Length:** Options include 64, 128, 256 tokens

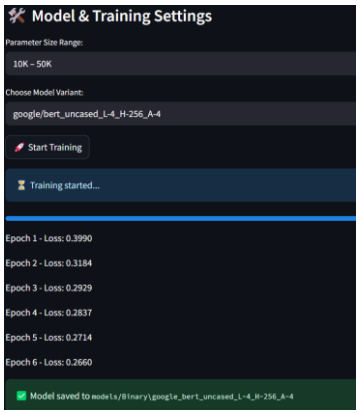
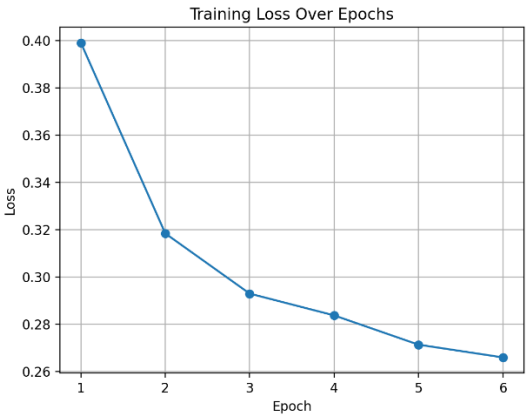
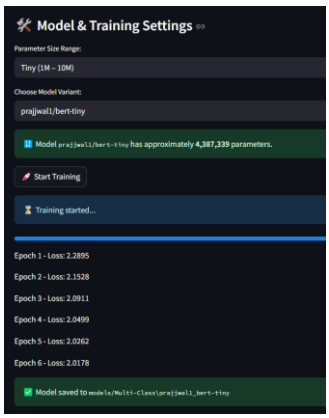
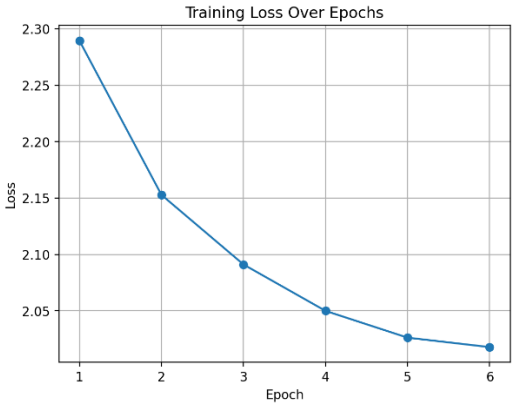
These parameters allow users to fine-tune their training strategy based on dataset size and available resources.

4. Seamless Training Flow

Once configurations are selected, pressing the “Start Training” button initializes the training loop:

- Loads the selected model and tokenizer
- Splits data into training/validation sets
- Begins fine-tuning with real-time loss feedback and a live progress bar
- Automatically saves the trained model and tokenizer to a structured output directory for future prediction or evaluation

VII. Training Results

Binary Model Training results	Binary Model Training loss
	
Multi-Class Model Training results	**Multi-Class Model Training loss**
	

VIII. Evaluation Metrics

The Detoxify-Telugu platform supports both Binary and Multi-Class classification pipelines. Evaluation metrics are used to quantify model effectiveness post-training. This section outlines comparative analysis and insights based on empirical evaluation.

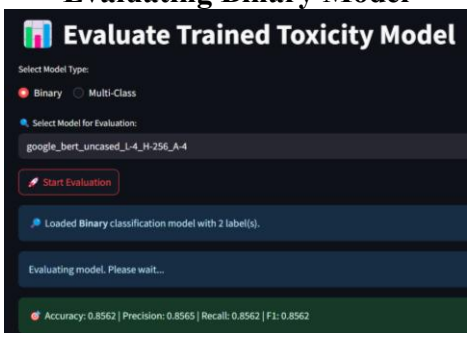
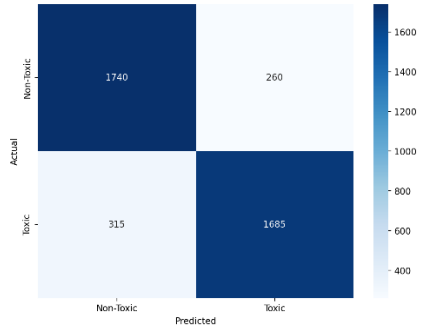
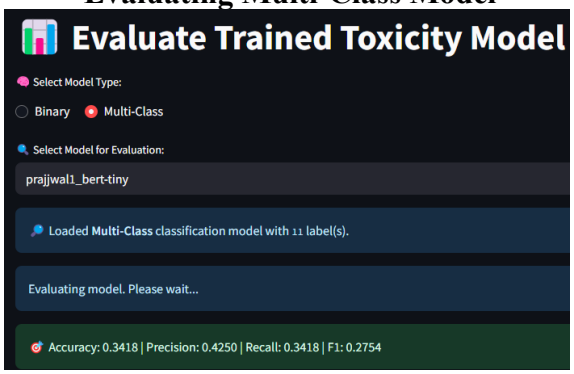
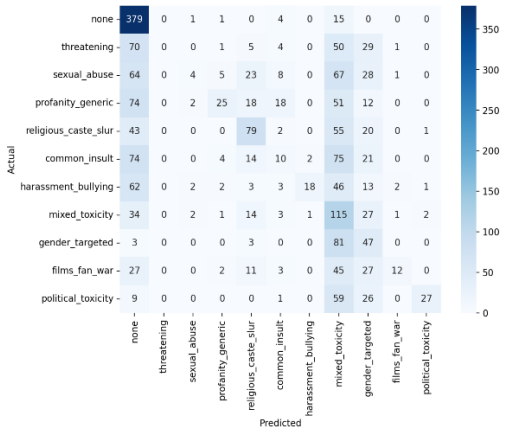
Accuracy: The proportion of total predictions that were correct.

Precision: The proportion of positive predictions that were actually correct.

Recall: The proportion of actual positives that were correctly identified.

F1-score: The harmonic mean of precision and recall, balancing both metrics.

Confusion matrix: A table that summarizes the performance of a classification model by showing true vs. predicted labels.

<table>
<tr>
<td>Evaluating Binary Model
 
</td>
<td>Confusion Matrix
 
</td>
</tr>
<tr>
<td>Evaluating Multi-Class Model
 
</td>
<td>Confusion Matrix
 
</td>
</tr>
</table>

Toxicity Detection: Comparative Analysis of BERT-based Models :

This report compares BERT-based models for toxicity detection in two tasks: binary classification (Toxic vs. Non-Toxic) and multi-class classification (11 toxicity types + "none"). The analysis evaluates model performance using standard metrics (Accuracy, Precision, Recall, F1-Score) and provides insights from confusion matrices.

1. Binary Classification: Toxic vs. Non-Toxic

Task : Predict whether a comment is "Toxic" or "Non-Toxic".

Models Compared :

Model Name	Parameters	Description
google/bert_uncased_L-4_H-256_A-4	~4.3M	Lightweight BERT with 4 layers, 256 hidden units
prajjwal1/bert-tiny	~4.3M	Tiny BERT optimized for efficiency

Evaluation Metrics :

Metric	google/bert_uncased_L-4_H-256_A-4	prajjwal1/bert-tiny	Comparison
Accuracy	0.8562	0.8472	google/bert_uncased slightly higher
Precision	0.857 (Toxic), 0.8565 (Non-Toxic)	0.938 (Non-Toxic)	prajjwal1/bert-tiny higher for Non-Toxic
Recall	0.934 (Toxic), 0.8562 (Non-Toxic)	0.861 (Toxic/Non-Toxic)	google/bert_uncased higher for Toxic
F1-Score	0.845 (Toxic), 0.8562 (Overall)	0.845 (Toxic/Non-Toxic)	google/bert_uncased better balanced

Confusion Matrix Insights

- **google/bert_uncased:** More True Positives and True Negatives, indicating balanced performance.
- **prajjwal1/bert-tiny:** Slightly higher False Positives and False Negatives, prioritizing Non-Toxic precision.

Observation: Accuracy discrepancy in prajjwal1/bert-tiny warrants further investigation.

Key Insights:

The google/bert_uncased_L-4_H-256_A-4 model slightly outperforms prajjwal1/bert-tiny in overall metrics. However, prajjwal1/bert-tiny excels in minimizing false positives for Non-Toxic comments, which may be preferable in specific use cases.

2. Multi-Class Classification: 11 Toxicity Types + "None"

Task : Predict one of 11 toxicity types (e.g., "mixed_toxicity", "gender_targeted", "threatening") or "none".

Models Compared :

Model Name	Parameters	Description
google/bert_uncased_L-4_H-256_A-4	~4.3M	Lightweight BERT with 4 layers
prajjwal1/bert-mini	~29M	Larger BERT with moderate complexity
prajjwal1/bert-tiny	~4.3M	Tiny BERT optimized for efficiency

Evaluation Metrics (Macro Averages) :

Metric	google/bert_uncased	prajjwal1/bert-mini	prajjwal1/bert-tiny
Accuracy	45.30%	47.64%	34.18%
Precision	0.516	0.571	0.442
Recall	0.408	0.476	0.342
F1-Score	0.404	0.491	0.237

Confusion Matrix Insights :

- **Common Trends:** All models perform best on the "none" class, likely due to class imbalance.
- **Confusion Patterns:** Frequent misclassifications among "mixed_toxicity", "gender_targeted", "films_fan_war", and "political_toxicity" due to semantic overlap.

Model-Specific:

- **prajjwal1/bert-tiny:** Significant misclassifications; zero precision/F1 for "threatening".
- **prajjwal1/bert-mini:** Improved performance but still exhibits confusion.
- **google/bert_uncased:** Moderate performance, struggles with minority classes.

Key Insights :

- Multi-class classification is significantly more challenging than binary due to task complexity and class imbalance.
- prajjwal1/bert-mini outperforms others, likely due to its larger parameter size (~29M).
- Semantic similarities between toxicity types contribute to misclassifications.
- Class imbalance (dominance of "none") skews model performance.

3. Binary vs. Multi-Class: Comparative Summary

Comparison Table :

Feature	Binary (google/bert_uncased)	Multi-Class (prajjwal1/bert-mini)
Task Complexity	Simple	Complex
Accuracy	~85.6%	~47.6%
Precision (Macro)	~85.7%	~0.571
Recall (Macro)	~89.5%	~0.476
F1-Score (Macro)	~85.6%	~0.491
Training Time	Fast	Moderate
Confusion Matrix	Balanced	Skewed (biased toward "none")

Key Conclusions

Binary Classification: High performance due to simpler task; google/bert_uncased is robust and efficient.

Multi-Class Classification: Lower performance due to complexity, class imbalance, and semantic overlap; prajjwal1/bert-mini benefits from larger size.

IX. Results & Insights

This section summarizes key findings from training dynamics and evaluation outcomes for binary and multi-class toxicity detection, focusing on lightweight BERT-based models.

Training Curve Insights

Binary Classification:

- google/bert_uncased_L-4_H-256_A-4 (~4.3M parameters) converged quickly with stable loss reduction, reflecting its efficiency for the simpler Toxic vs. Non-Toxic task.
- prajjwal1/bert-tiny (~4.3M parameters) showed similar convergence but slightly higher variance in loss, likely due to its optimized architecture.

Multi-Class Classification:

- prajjwal1/bert-tiny exhibited high initial loss and slower convergence, struggling with the complexity of 12 classes (11 toxicity types + "none").
- prajjwal1/bert-mini (~29M parameters) converged faster than bert-tiny, benefiting from increased capacity.
- google/bert_uncased_L-4_H-256_A-4 showed moderate convergence, balancing efficiency and performance.

Observation: Multi-class models provide granular toxicity labeling for analytical use cases but demand more data and computational resources.

Evaluation Highlights

Binary Classification:

- google/bert_uncased_L-4_H-256_A-4: Robust performance (Accuracy: 85.62%, Precision: 85.7% Toxic/85.65% Non-Toxic, Recall: 93.4% Toxic/85.62% Non-Toxic, F1: 84.5% Toxic/85.62% Overall).
- prajjwal1/bert-tiny: Slightly lower performance (Accuracy: 84.72%, Precision: 93.8% Non-Toxic, Recall: 86.1% Toxic/Non-Toxic, F1: 84.5% Toxic/Non-Toxic), prioritizing Non-Toxic precision.

Multi-Class Classification:

- prajjwal1/bert-tiny: Lowest performance (Accuracy: 34.18%, Precision: 0.442, Recall: 0.342, F1: 0.237), with zero precision for "threatening" and widespread misclassifications.
- prajjwal1/bert-mini: Best performance (Accuracy: 47.64%, Precision: 0.571, Recall: 0.476, F1: 0.491), but struggled with minority classes.

- google/bert_uncased_L-4_H-256_A-4: Moderate performance (Accuracy: 45.30%, Precision: 0.516, Recall: 0.408, F1: 0.404), with confusion among similar toxicity types.

Challenge: Multi-class models face difficulties due to class imbalance (prevalence of "none") and semantic overlap (e.g., "mixed_toxicity" vs. "political_toxicity").

Observational Learnings

- **Model Capacity:** Larger models (prajjwal1/bert-mini) outperform smaller ones (bert-tiny) in multi-class tasks, but binary tasks show comparable performance across sizes.
- **Data Requirements:** Multi-class classification requires balanced datasets and clear class definitions to reduce misclassifications.
- **Task Complexity:** Binary classification is robust and generalizes well, while multi-class classification is hindered by overlapping categories.

Practical Implications

- **Live Moderation:** Binary models (google/bert_uncased_L-4_H-256_A-4) are ideal for their speed and high accuracy.
- **Content Analysis:** Multi-class models (prajjwal1/bert-mini) provide detailed toxicity profiling for post-moderation analysis.
- **Resource Constraints:** Lightweight models (bert-tiny) are suitable for low-resource environments but sacrifice multi-class performance.

Areas for Improvement

- **Model Scaling:** Explore larger models like distilbert-base-uncased (~66M parameters) for multi-class tasks to improve accuracy and robustness.
- **Data Rebalancing:** Apply oversampling or weighted loss functions to address class imbalance, particularly for minority classes like "threatening."
- **Class Definition:** Refine annotations for overlapping categories (e.g., "mixed_toxicity" vs. "gender_targeted") or merge them (e.g., combine "gender_targeted" and "identity_attack").
- **Preprocessing:** Normalize text to handle slang, typos, and contextual nuances, improving model robustness.

X. Error Analysis

This section identifies key challenges observed during training and evaluation, offering directions for future refinements.

Key Challenges

Spelling Variants and Slang:

- Non-standard spellings (e.g., "h8" vs. "hate") and slang reduced model confidence and bypassed keyword-based filters.
- **Impact**: Increased false negatives, particularly in binary classification for Toxic comments.

Semantic Complexity:

- Models struggled with sarcasm, indirect hate, or contextual innuendo, especially in multi-class classification.
- **Example**: Comments with subtle insults (e.g., "Great work, genius") were misclassified as "none" instead of "sarcasm_toxicity."

Class Overlap:

- Semantic similarities between categories like "mixed_toxicity," "gender_targeted," "films_fan_war," and "political_toxicity" caused frequent misclassifications.
- **Impact**: Reduced precision and recall for minority classes in multi-class tasks.

Class Imbalance:

- Overrepresentation of the "none" class skewed predictions, particularly for prajjwal1/bert-tiny.
- **Impact**: Minority classes like "threatening" had near-zero precision and F1 scores.

Dataset-Specific Observations

Assumption: The dataset is primarily English, as no multilingual context was specified. Challenges like slang and sarcasm are language-agnostic but may be amplified in diverse datasets.

Note: If the dataset includes multilingual elements (e.g., code-mixing or non-standard scripts), additional preprocessing (e.g., Unicode normalization) would be required.

Proposed Solutions

1. **Preprocessing:**
    - Implement text normalization (e.g., lemmatization, slang dictionaries) to handle spelling variants and informal language.
    - Enhance tokenization to capture contextual nuances in ambiguous comments.

2. **Model Enhancements:**
    - Fine-tune models with datasets emphasizing sarcasm or indirect toxicity to improve multi-class performance.
    - Experiment with ensemble methods combining bert-mini and bert_uncased for balanced performance.
3. **Dataset Improvements:**
    - Rebalance classes using oversampling or synthetic data generation for minority toxicity types.
    - Refine annotations to clarify boundaries between overlapping categories (e.g., "mixed_toxicity" vs. "political_toxicity").

XI. Conclusion

Detoxify-Telugu v2.1 is a tailored NLP pipeline for toxicity detection in Telugu and Tenglish (Telugu-English code-mixed) content, designed for accessibility and effectiveness in resource-constrained environments. Key features include:

- **Transformer-based Classification**: Utilizes lightweight BERT models (google/bert_uncased_L-4_H-256_A-4, prajjwal1/bert-mini, prajjwal1/bert-tiny) for binary (Toxic vs. Non-Toxic) and multi-class (11 toxicity types + "none") classification, achieving up to 85.6% accuracy in binary tasks.
- **Streamlit-based GUI**: Provides an intuitive interface for data annotation, model training, and evaluation, bridging non-technical moderation needs with advanced NLP.
- **Modular Pipeline**: Supports end-to-end workflow from data scraping to real-time prediction, optimized for Telugu-language communities. This system demonstrates that region-specific NLP can effectively address toxicity in underrepresented languages like Telugu, despite challenges like data scarcity and code-mixing.

XII. Future Plans

To enhance Detoxify-Telugu v2.1, we have plans to extend the platform's capabilities which includes:

- **Synthetic Data Augmentation:** Use tools like synthetic_data_gen.py to generate adversarial toxic patterns, addressing class imbalance in multi-class tasks.
- **Data Expansion:** Collect and annotate more Telugu and Tenglish content (e.g., social media comments) to improve model robustness.
- **Multi-Class Optimization:** Implement class-weighted training and explore larger models (e.g., distilbert-base-uncased) to boost minority class accuracy.
- **Toxicity Scoring System:** Develop user profiling based on comment frequency and toxicity severity for moderation prioritization.
- **Sarcasm Detection:** Integrate sentence-embedding techniques (e.g., SBERT) to better capture implicit abuse and satire.
- **API Deployment:** Expose models via REST APIs or browser-based extensions for real-time toxicity filtering.

XIII. Workflow Overview

The entire model lifecycle is fully automated and user-friendly:

Workflow

[Scrape] → [Annotate] → [Clean] → [Balance] → [Train] → [Evaluate] → [Predict]

Pipeline Components:

- **scrape.py:** Extracts YouTube comments in Telugu and Tenglish for dataset creation.
- **data_annotation.py:** Facilitates keyword-based and manual labeling, supporting 11 toxicity types + "none."
- **data_cleaning.py:** Normalizes text (e.g., handles code-mixing, slang) for consistent input to BERT models.
- **data_balancing.py:** Applies class-balanced sampling to mitigate the dominance of the "none" class.
- **model_training.py:** Trains BERT models (google/bert_uncased_L modes, prajjwal1/bert-mini, prajjwal1/bert-tiny`) with configurable hyperparameters.
- **predict.py:** Enables real-time or batch toxicity predictions for moderation.
- **Streamlit UI:** Unifies all components into a single interface for data management, training, and evaluation.

XIV. References

- **HuggingFace Telugu HateSpeech Dataset:** Source of Telugu and Tenglish comments for training and evaluation.
- **Devlin, J., et al. (2018).** "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv:1810.04805.
- **Detoxify Library:** Reference for English toxicity detection, adapted for Telugu context.
- **Scikit-learn Documentation**: Used for evaluation metrics and data preprocessing.
- **Selenium WebDriver:** Enabled YouTube comment scraping via scrape.py.
- **Streamlit Documentation:** Supported development of the GUI for pipeline management.

XV. Appendices

Fun Facts & Reflections:

- **Custom Telugu Stopwords:** Developed a Telugu stopwords library to reduce noise, later replaced by BERT's contextual tokenization for better performance.
- **Tenglish Transliterator:** Built a rule-based tool to convert Tenglish to Telugu script, improving preprocessing for code-mixed inputs compared to general-purpose LLMs.

Reflection:

Developing Detoxify-Telugu v2.1 highlighted the challenges of NLP for underrepresented languages. Despite limited data and compute, careful engineering (e.g., lightweight BERT models, class balancing) enabled effective toxicity detection, emphasizing the value of community-driven NLP efforts (especially for underrepresented languages like Telugu).