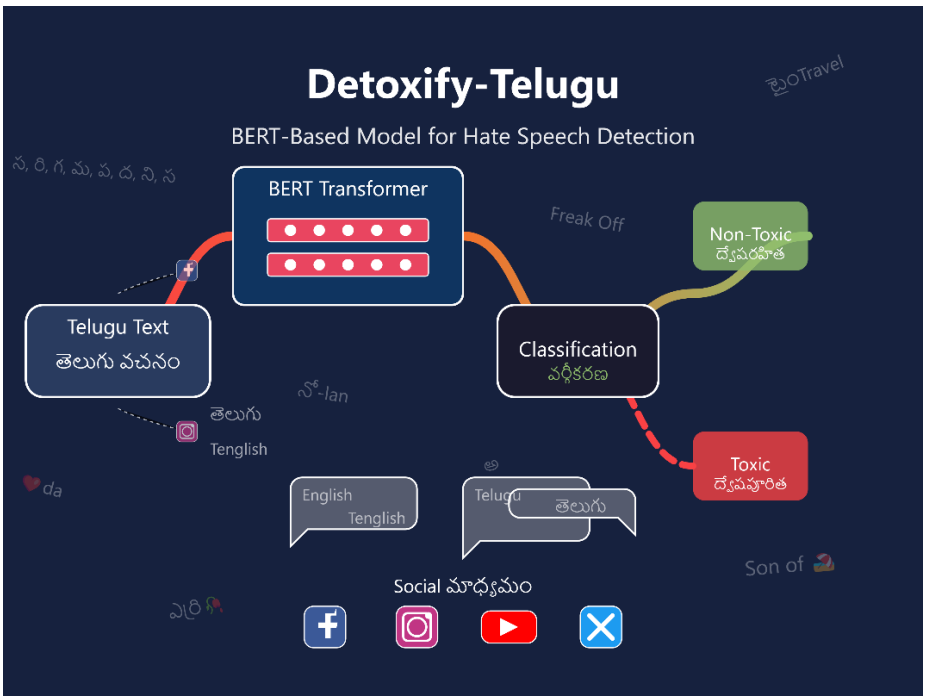
Detoxify-Telugu



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

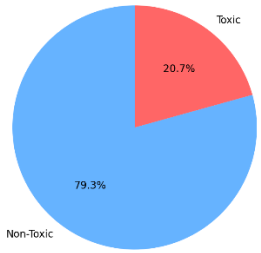
To train and evaluate the Detoxify-Telugu platform, we built a diverse dataset using multiple sources and methods to reflect real-world Telugu and Tenglish toxicity. Below is a summary of how the dataset was collected and structured.

Sources Used:

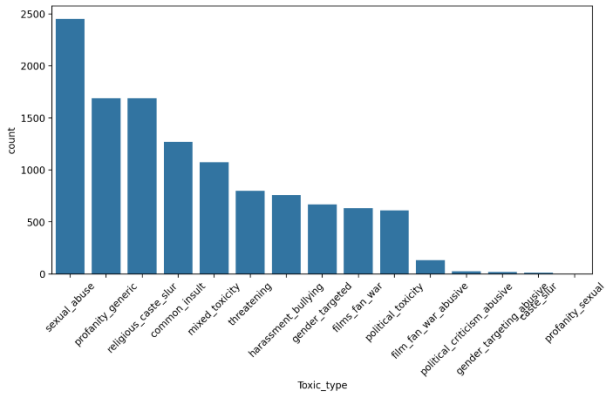
The Detoxify-Telugu dataset was created using a mix of real, synthetic, and curated data from the following sources:

- YouTube Comments collected using a custom Python script (YT_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

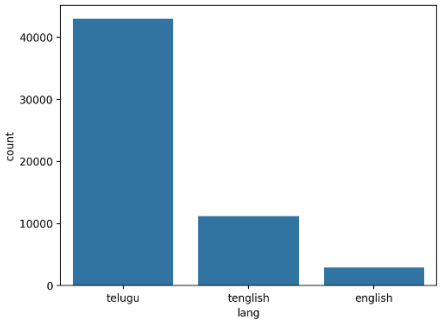
Toxic vs Non-Toxic Distribution



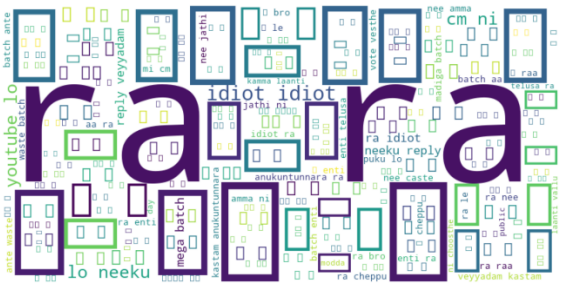
Toxic Type Distribution



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.

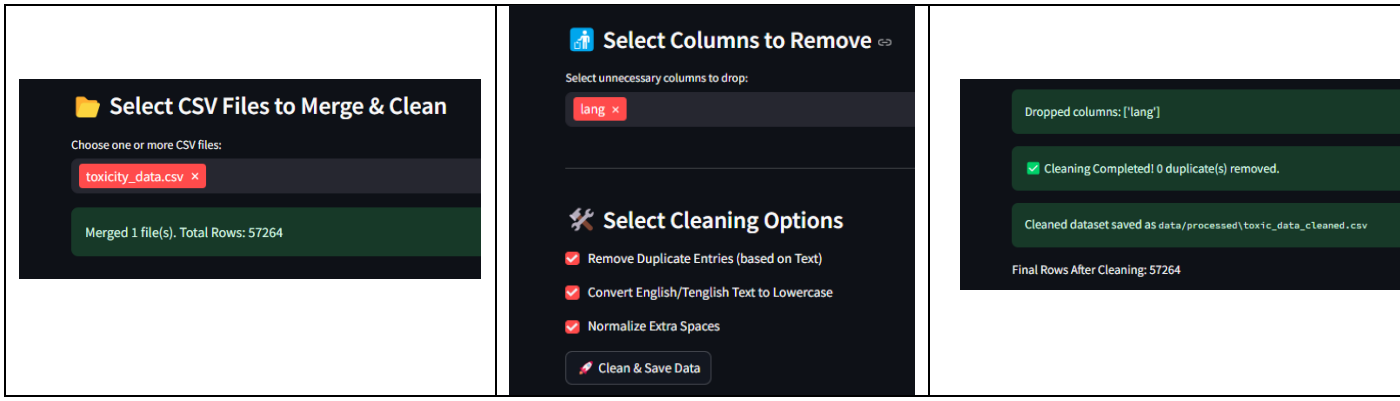
These visuals help in understanding patterns in toxic speech and guide model training and evaluation.

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:

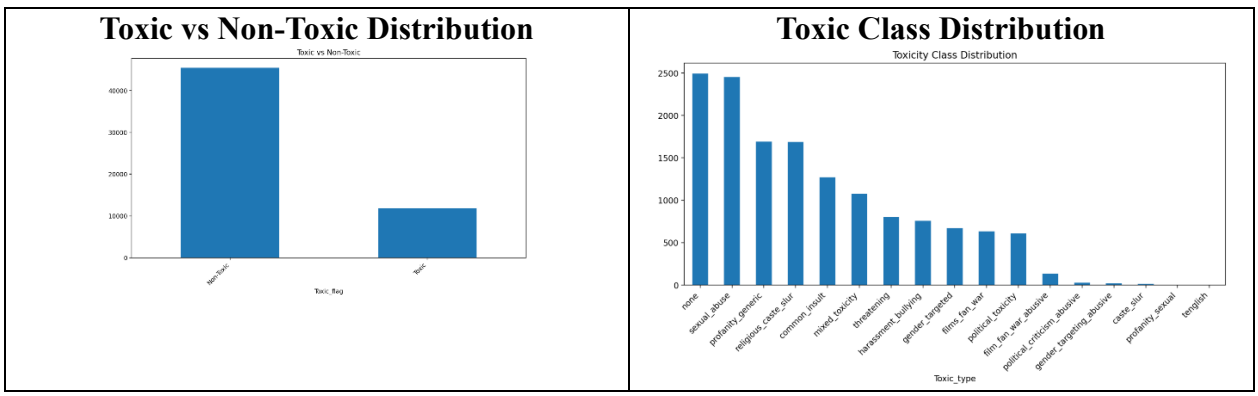


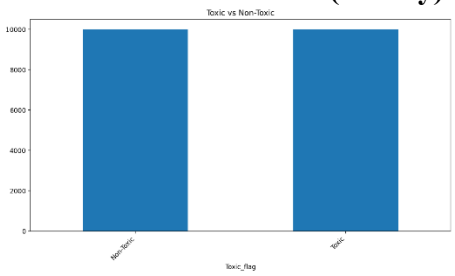
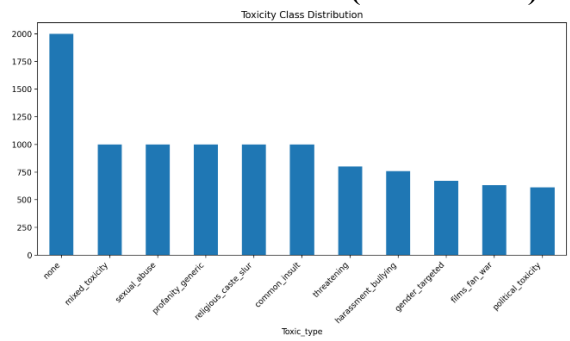
data_cleaning.py – Cleans the raw text by removing noise, special characters, and duplicate entries.

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

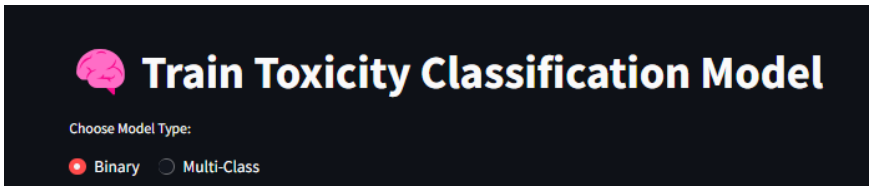
Due to the original imbalance (more non-toxic content), we created two balanced versions of the dataset:

- **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)

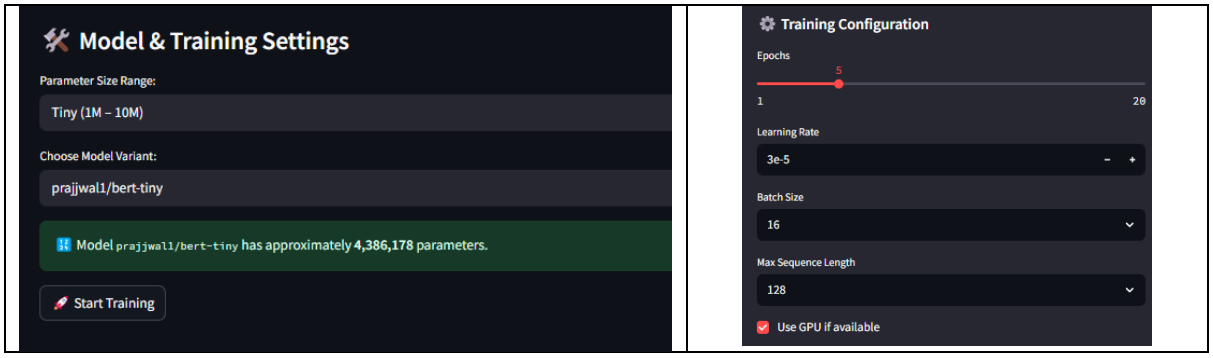
These cleaned and balanced datasets ensure better model performance during training and testing.

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

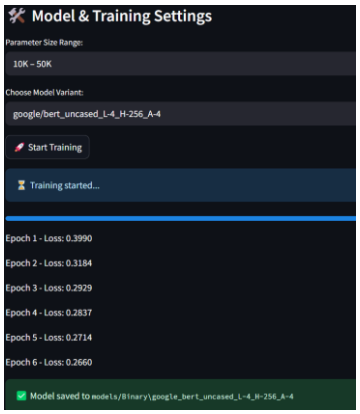
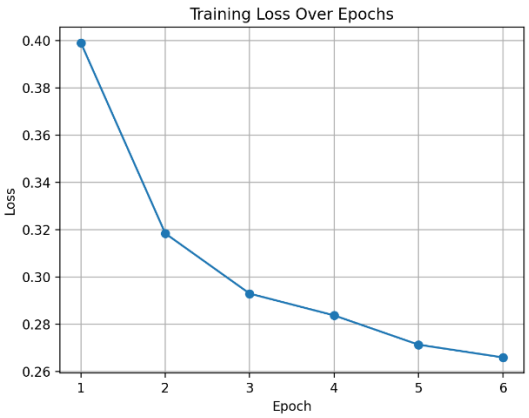
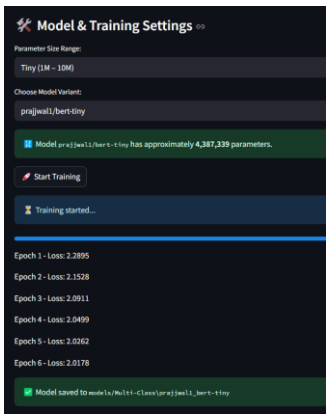
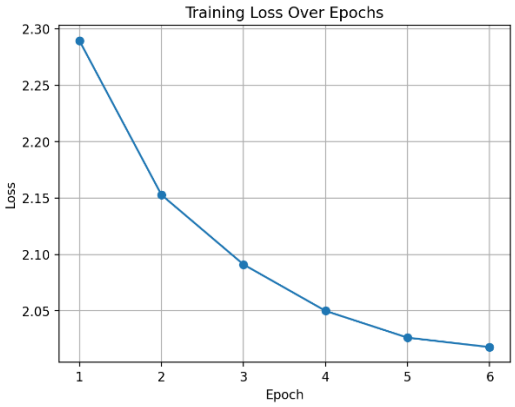
These parameters allow users to fine-tune their training strategy based on dataset size and available resources.

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.

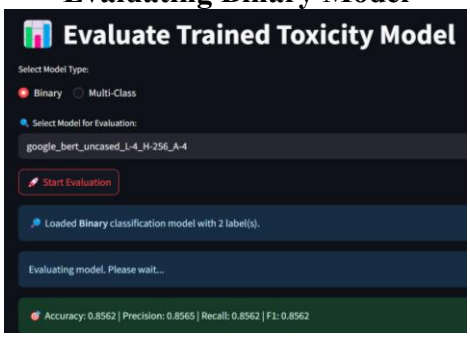
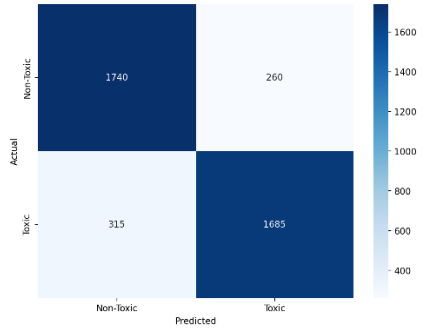
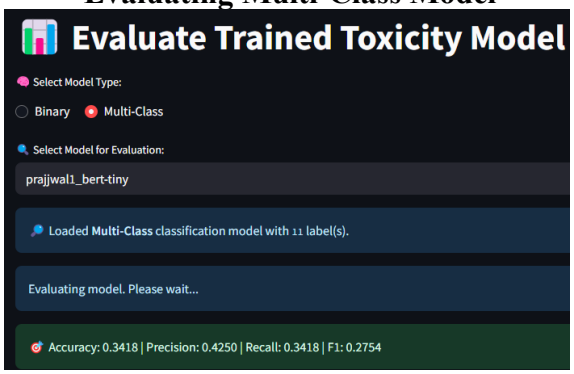
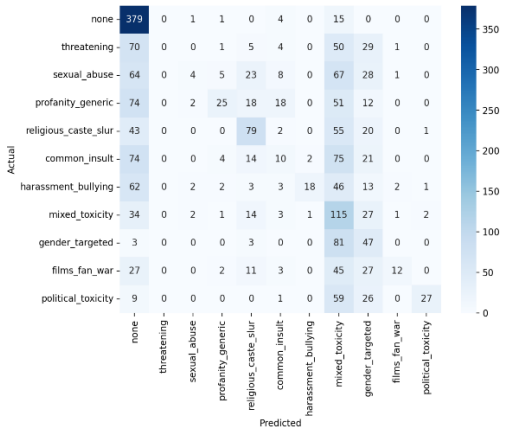
Accuracy: The proportion of total predictions that were correct.

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>

1. Binary Classification Model

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

Model Used: `google/bert_uncased_L-4_H-256_A-4`

Parameter Range: ~10K-50K

Training Observations:

- Initial Loss: 0.40 (low)
- Convergence: Fast convergence in <6 epochs
- Loss Trend: Steady decline per epoch

Evaluation Results:

- Accuracy: 85.62%
- Precision: 85.65%
- Recall: 85.62%
- F1 Score: 85.62%

Confusion Matrix:

- True Positives: 1685
- True Negatives: 1740
- False Positives: 260
- False Negatives: 315

Key Insights:

- Balanced trade-off between precision and recall.
- Effective at flagging both toxic and non-toxic comments.
- Suitable for moderation systems requiring fast, reliable toxicity screening.

2. Multi-Class Classification Model

Task: Predict one of 11 toxicity types or "none"

Model Used: `prajjwal1/bert-tiny`

Parameter Range: ~4.3M (Tiny BERT)

Training Observations:

- Initial Loss: 2.29 (significantly higher than binary)
- Loss Pattern: Sharp drop initially, slower thereafter
- Challenge: More complex classification boundaries

Evaluation Results:

- Accuracy: 34.18%
- Precision: 42.50%
- Recall: 34.18%
- F1 Score: 27.54%

Confusion Matrix Observations:

- Strongest prediction for "none" class (379 correct)
- Misclassifications common in classes like "sexual_abuse", "common_insult", and "mixed_toxicity"
- Notable confusion across closely related toxicity types

Key Insights:

- Multi-class classification is significantly harder.
- Model struggles to generalize between overlapping categories.
- High data imbalance likely contributes to underperformance.

Summary: Binary vs Multi-Class

Feature	Binary Model (`google/bert_uncased_L-4_H-256_A-4`)	Multi-Class Model (`prajjwal1/bert-tiny`)
Task Complexity	Simple (2 classes)	Complex (11 classes)
Accuracy	85.62%	34.18%
Precision	85.65%	42.50%
F1 Score	85.62%	27.54%
Training Time	Fast	Faster (but lower generalization)
Confusion Matrix	Balanced	Highly skewed

- Binary models offer higher accuracy, generalize well, and are easier to train.
- Multi-class models suffer from class confusion and data sparsity but provide more granular toxicity insights.

IX. Results & Insights

This section presents key takeaways based on training dynamics and evaluation outcomes across both binary and multi-class models.

Training Curve Insights

- Binary Model (`google/bert_uncased_L-4_H-256_A-4`) converged quickly, demonstrating stable and consistent loss reduction.
- Multi-Class Model (`prajjwal1/bert-tiny`) exhibited higher starting loss and slower convergence—common in high-class, low-capacity tasks.
- Despite lower accuracy, multi-class models offer deeper toxicity labeling, which is ideal for analytical or post-moderation use cases.

Evaluation Highlights

- Binary model performed robustly across all metrics (85%+), demonstrating strong generalization and balanced classification.
- Multi-class model struggled with minority classes and overlap among toxicity categories, especially in "mixed_toxicity" and "sexual_abuse".

Observational Learnings

- Parameter size matters: smaller models converge faster but might underperform on complex tasks.
- Multi-class classification inherently demands more data and balanced representation.
- Data-driven misclassifications point to opportunities for improving class definitions and annotations.

Practical Implications

- For live moderation: Binary is preferable (speed + accuracy).
- For content analysis: Multi-Class adds interpretability.

Areas for Improvement

- Model scaling: Consider switching to `distilbert-base-uncased` for multi-class tasks.
- Better annotation: Improve class definitions and clean label noise.
- Rebalancing: Equalize samples per toxicity type.
- Class merge: Consider merging overlapping toxic categories.

X. Error Analysis

During training and evaluation, several limitations and challenges were identified:

- **Unicode Challenges:** Telugu characters with diacritics or alternate forms often caused token mismatches, leading to missed keyword detections.
- **Slang & Spelling Variants:** Toxic phrases with spelling deviations (e.g., “suudhi” vs. “soodhi”) bypassed both regex and keyword-based filters.
- **Semantic Complexity:** Transformer-based models failed to consistently detect sarcasm, indirect hate, or contextual innuendo, especially in multi-class classification.
- **Code-mixing:** Mixed-language inputs (Tenglish + Telugu) degraded model confidence due to inconsistent embedding patterns.

These insights provide direction for future refinements in preprocessing and model design.

XI. Conclusion

Detoxify-Telugu v2.1 demonstrates that region-specific NLP pipelines can be both accessible and effective, even with limited computational resources. The system integrates:

- Transformer-based classification for Telugu and Tenglish

- Streamlit-based GUI for annotation, training, and evaluation.
- Modular pipeline support for dataset curation and real-time prediction
- This framework bridges the gap between non-technical moderation needs and advanced NLP capabilities, tailored specifically for Telugu-language communities.

XII. Future Plans

Planned enhancements to extend the platform's capabilities include:

- Synthetic Data Augmentation: Leverage synthetic_data_gen.py to introduce adversarial toxic patterns
- Data Expansion: Collect and label more real-world, code-mixed and native Telugu content
- Multi-Class Optimization: Apply class-weighted training and deeper models to boost minority class accuracy
- Toxicity Scoring System: Implement user profiling based on comment frequency and toxicity distribution
- Sarcasm Detection: Introduce sentence-embedding techniques to model implicit abuse and satire
- API Deployment: Expose models as REST APIs or browser-based toxicity filters

XIII. Workflow Overview

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

Workflow

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

Pipeline Components:

- YT_scrape.py: YouTube comment extractor
- data_annotation.py: Smart keyword-based labeling tool
- data_balancing.py: Class-balanced sampling
- model_training.py: Dynamic model builder with config controls
- predict.py: Real-time or batch toxicity prediction
- Streamlit UI: Unified interface to manage the full pipeline

XIV. References

- HuggingFace Telugu HateSpeech Dataset
- Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers"
- Detoxify Library (English Toxicity)
- Scikit-learn Documentation
- Selenium WebDriver (YouTube Scraping)
- OpenAI ChatGPT, Grok, Anthropic Claude — for code assistance and UI prototyping

XV. Appendices

Fun Facts & Reflections:

- Developed a custom Telugu stopwords library, later dropped for BERT-based tokenizers.
- Built a rule-based Tenglish → Telugu transliterator that outperformed general-purpose LLMs like Mistral, LLaMA2, and Qwen2 in accuracy.

Reflection:

Working on Detoxify-Telugu reinforced the importance of language-specific model training. Despite data scarcity and limited compute, the outcomes highlight how far careful engineering and community-driven NLP efforts can go—especially for underrepresented languages like Telugu.