CleanSpeech:
Toxicity Detection
and Rewriting with Explainable AI

Final Project Report

Course: DS-Lab

Project: CleanSpeech

Institution: IIT Madras

Team: Team-10

CleanSpeech: 1
Toxicity Detection 1
and Rewriting with Explainable AI 1
Abstract 3
1. Introduction & Objectives 3
2. Literature Review (Milestone 1) 4
3. Dataset and Methodology (Milestone 2–3) 4
3.1. System Architecture 4
3.2. Dataset Preparation 5
3.3. Model Architecture 5
3.4. Loss Function 5
4. Model Development and Hyperparameter Tuning (Milestone 4) 6
4.1. Training Setup 6
4.2. Regularization and Optimization 6
4.3. Hyperparameter Experiments 7
5. Evaluation & Analysis (Milestone 5) 7
5.1. Quantitative Analysis (Detection) 7
5.2. Quantitative Analysis (Rewriting) 8
5.3. Qualitative Analysis (Explainability) 8
5.4. Error Analysis 9
6. Deployment & Documentation (Milestone 6) 9
6.1. Deployment 9
6.2. Documentation 9
7. Conclusion and Future Work 10
7.1. Limitations 10
7.2. Future Work 11
7.3. Conclusion 11
8. References and Appendix 11
References 11
Appendix 11

Abstract

This report details the design, implementation, and evaluation of CleanSpeech, a comprehensive system to address online toxicity. The project's primary goal is to detect, explain, and constructively rewrite harmful text. The system employs a modular architecture featuring a fine-tuned **microsoft/mdeBERTa-v3-base** model for multi-label toxicity classification, achieving a **Macro ROC-AUC of 0.983** and an **Optimized Macro F1-score of 0.68** on unseen test data. Explainability is integrated using **SHAP** (SHapley Additive exPlanations), providing token-level insights into model decisions. Finally, the **Gemini API** is leveraged for constructive text rewriting, neutralizing toxicity while preserving semantic intent, which achieved a **BERTScore F1 of 0.948**. This project successfully demonstrates a robust, transparent, and practical pipeline for fostering healthier online communication.

1. Introduction & Objectives

The proliferation of online toxicity, hate speech, and harassment poses a significant challenge to digital communities. Automated content moderation is essential, but traditional systems often lack transparency (acting as "black boxes") and are purely punitive.

The CleanSpeech project was initiated to create a more holistic solution. Instead of just flagging or deleting content, our system is designed to:

- **Detect:** Accurately classify text across multiple toxicity categories.
- **Explain:** Provide transparent, human-readable explanations for why content is flagged.
- **Rewrite:** Offer constructive, non-toxic alternatives to harmful messages, promoting user education and positive interaction.

The primary objective is to build a robust, modular, and explainable pipeline that serves as both a moderation tool and a constructive feedback mechanism.

2. Literature Review (Milestone 1)

The problem of toxicity detection has been extensively studied in Natural Language Processing (NLP). Early approaches relied on keyword-matching and linear models (e.g., TF-IDF with Logistic Regression), which struggle with context, sarcasm, and nuanced language.

The advent of transformer-based models (e.g., BERT, RoBERTa) significantly advanced the state-of-the-art by capturing deep contextual relationships. However, these models are often criticized for their "black box" nature, making it difficult to understand *why* a decision was made. This lack of transparency is a major drawback for moderation systems, where fairness and accountability are paramount.

In parallel, the field of Explainable AI (XAI) has produced methods like LIME and SHAP, which can provide token-level attributions for model predictions. Concurrently, the rise of large-scale Generative AI (like the Gemini models) has opened new possibilities not just for *detecting* harmful content, but for *correcting* it.

CleanSpeech is positioned at the intersection of these three fields. It addresses the gaps identified in the literature by:

1. Using a state-of-the-art transformer (**mDeBERTa-v3**) for robust detection.
2. Integrating **SHAP** directly into the pipeline for model transparency.
3. Employing a **Gemini API** for constructive rewriting, moving beyond simple censorship to user education.

3. Dataset and Methodology (Milestone 2–3)

The system is built on a modular pipeline that processes text from input to a constructive output.

3.1. System Architecture

The CleanSpeech architecture follows a linear flow, with each component designed for independent evaluation and improvement.

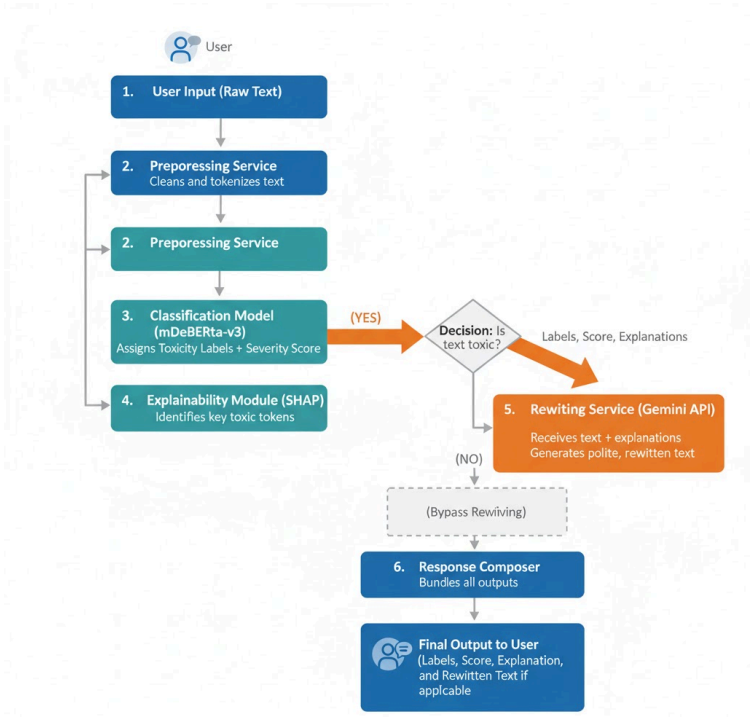

Figure 1: CleanSpeech System Architecture Overview

The flow is as follows:

1. **User Input:** Text is received by the system.
2. **Preprocessing:** The text is cleaned and tokenized.
3. **Classification:** The fine-tuned mDeBERTa-v3 model performs multi-label classification and assigns a severity score.
4. **Explainability:** SHAP-based analysis identifies the tokens most responsible for the toxicity score.
5. **Rewriting:** If toxic, the text and explanations are sent to the Gemini API, which generates a constructive, polite alternative.
6. **Response Composition:** The final output (labels, explanation, and rewritten text) is returned to the user.

3.2. Dataset Preparation

The **Jigsaw Toxic Comment Classification Challenge** dataset was selected as the primary and sole corpus for this project. An initial strategy involving a secondary dataset (HASOC) with Gemini-based label augmentation was explored but ultimately abandoned, as the augmentation results were not satisfactory.

- **Corpus:** The Jigsaw dataset provides a large scale (~160,000 samples) and a multi-label format with six classes: `toxic`, `severe_toxic`, `obscene`, `threat`, `insult`, `identity_hate`.
- **Preprocessing:** Standard preprocessing was applied, including removing URLs, HTML tags, emojis, and non-ASCII characters, and normalizing whitespace.
- **Data Splits:** The dataset was divided into training (127,397 samples), validation (31,850 samples), and unseen test sets to ensure robust model evaluation. A stratified split was used to maintain the proportion of toxic to non-toxic comments.

3.3. Model Architecture

Classifier (mDeBERTa-v3)

We selected microsoft/mdeBERTa-v3-base as our core classification model. Its disentangled attention mechanism, which separates content and position encodings, provides a superior understanding of context and nuance, crucial for identifying subtle or sarcastic forms of toxicity.

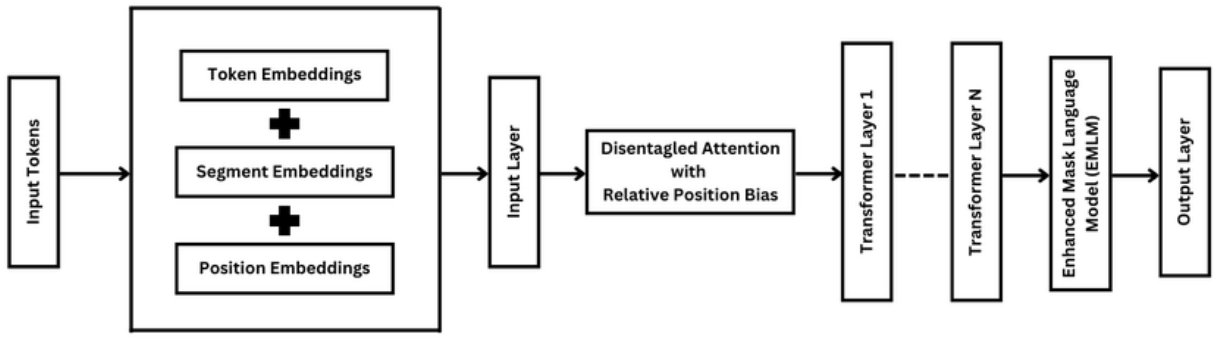

Figure 2: mDeBERTa-v3 Transformer Architecture

Explainability (SHAP)

SHAP (SHapley Additive exPlanations) was chosen to provide token-level explanations,

offering clear insights into which words or phrases contributed most to a toxicity prediction.

Rewriter (Gemini API)

The Gemini API was used for the text rewriting module due to its strong instruction-following capabilities and its ability to maintain semantic intent while neutralizing toxic language and adopting a polite tone.

3.4. Loss Function

To address the significant class imbalance in the Jigsaw dataset (where classes like threat are rare), a **Weighted Binary Cross-Entropy (WBCE) Loss** was implemented. This function assigns a higher weight to minority classes during training, ensuring the model learns to identify them effectively.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} w_c \left[y_{ic} \log(\hat{y}_{ic}) + (1 - y_{ic}) \log(1 - \hat{y}_{ic}) \right]$$

Where:

- y_{ic}: true label for class c
- $\hat{y}_{ic}$: predicted probability
- w_c: class-specific weight to address label imbalance

classification.

4.1. Training Setup

- **Framework:** Hugging Face transformers with accelerate for distributed training.
- **Hardware:** 2 × T4 GPUs.
- **Optimization: AdamW** optimizer with a learning rate of **2e-5** and weight decay of **0.01**.
- **Scheduler:** A **linear warm-up** scheduler (10% of steps) was used for stable convergence.
- **Configuration:** Training was performed with mixed precision (**fp16**), a per-GPU batch size of 16 (32 effective), and a max sequence length of **256 tokens**.
- **Loss:** Weighted BCEWithLogitsLoss (as defined in section 3.4).

4.2. Regularization and Optimization

Several techniques were employed to ensure stable training and prevent overfitting:

Technique	Purpose	Observed Effect

Weight Decay (AdamW)	Penalize large weights	Reduced variance; smoother training
Early Stopping (p=2)	Stop when val AUC plateaus	Best epoch = 3 (out of 6)
Gradient Clipping (1.0)	Avoid fp16 instability	No NaN losses
Warm-up LR	Gradual ramp-up	Prevented initial divergence
Class-Weighted Loss	Handle imbalance	Improved AUC for `threat` & `identity_hate`

4.3. Hyperparameter Experiments

Parameter	Values Tried	Observation
Learning Rate	2e-5 – 3e-5	2e-5 was most stable
Batch Size	8 – 16	16 balanced speed + gradient noise
Sequence Length	128 vs 256	256 slightly better AUC
Weight Decay	0 vs 0.01	0.01 reduced overfit

Warm-up Ratio	0.05 vs 0.10	0.10 yielded smoother curve

The best configuration (`lr = 2e-5`, `batch = 16`, `max_len = 256`, `weight_decay = 0.01`) was used for the final model. Initial training results on the validation set showed a strong Macro Average ROC-AUC of **0.982**, confirming the model was ready for final evaluation.

5. Evaluation & Analysis (Milestone 5)

The fine-tuned model was rigorously evaluated on the unseen test set.

5.1. Quantitative Analysis (Detection)

The model demonstrated excellent generalization. The primary metric, **Macro ROC-AUC**, reached **0.983**, indicating strong discriminative ability across all classes.

For practical application, decision thresholds were tuned for each label on the validation set to optimize the F1-score. This resulted in a final **Macro F1-score of 0.68** on the test set.

Table 1: Per-Label Performance (Optimized Thresholds)

Label	ROC-AUC	F1 (Tuned)	Best Threshold
toxic	0.987	0.74	0.42
severe_toxic	0.982	0.55	0.38
obscene	0.989	0.78	0.47
threat	0.984	0.61	0.33
insult	0.981	0.72	0.46
identity_hate	0.976	0.67	0.40
Macro Avg	0.983	0.68	—

5.2. Quantitative Analysis (Rewriting)

The quality of the constructive rewrites generated by the Gemini API was evaluated using **BERTScore**. This metric measures the semantic similarity between the original text's intent and the rewritten text. The rewriting module achieved a **BERTScore F1 of 0.948**, confirming

that the rewrites successfully preserve the original meaning while removing the toxicity.

5.3. Qualitative Analysis (Explainability)

Explainability was crucial for validating model behavior. SHAP visualizations and attention heatmaps confirmed that the model was "looking" at the right words.

SHAP Plots: As shown in Figure 4, the model correctly identifies words like "idiot" and "dum" as strong contributors (red) to the toxicity score, while non-toxic text ("Thanks for the clarification") is correctly pushed down (blue).

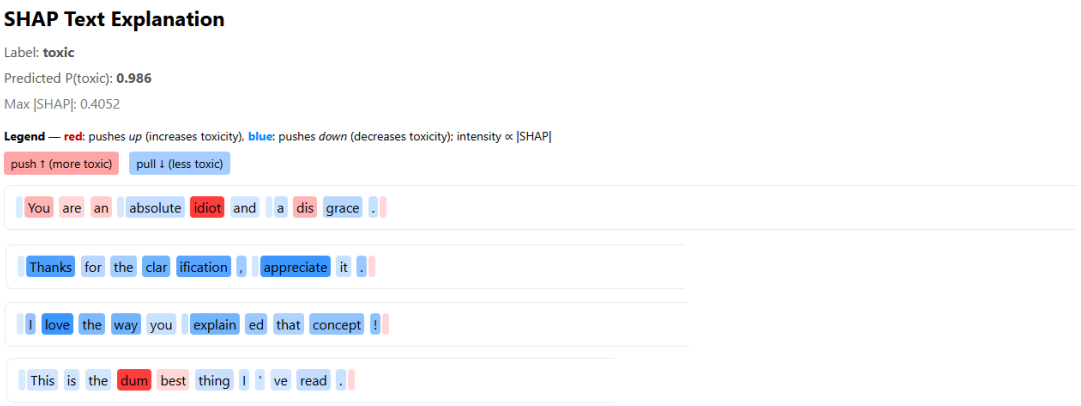

Figure 4: SHAP Text Explanation for Toxic and Non-Toxic Inputs

Attention Heatmaps: The attention mechanism (Figure 5) shows a high degree of focus on the toxic tokens ("idiot", "hate") when relating them to the [CLS] token (used for classification), confirming the model's focus.

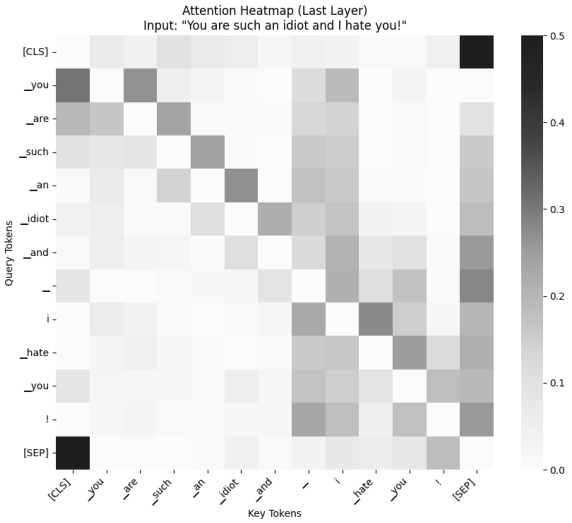

Figure 5: Attention Heatmap Visualization

5.4. Error Analysis

Evaluation revealed several key areas for improvement:

- **Sarcasm & Ambiguity:** The model struggled with nuanced sarcasm (e.g., "Nice job, genius.") which was sometimes misclassified as toxic.
- **Context Dependence:** Phrased-based context was sometimes missed (e.g., "That was a kill shot!" in a gaming context flagged as violent).
- **Mixed Language:** While mDeBERTa is multilingual, it showed some weakness with code-mixed slang not present in its training.

6. Deployment & Documentation (Milestone 6)

6.1. Deployment

The final CleanSpeech system is deployed as an interactive web application using **Streamlit**. This UI serves as a proof-of-concept and a tool for qualitative analysis.

The Streamlit interface allows a user to:

- Enter free-form text.
- View the multi-label toxicity predictions with probability bars.
- Inspect the SHAP-based model explanation, which highlights the contribution of each token.
- Generate a **non-toxic rewrite** using the integrated Gemini API.

This interactive deployment successfully brings all components of the project (detection, explainability, and rewriting) into a single, user-friendly interface.

6.2. Documentation

Project documentation is maintained in the `README.md` file and within the `doc/` folder, which contains detailed reports for each milestone. The repository is structured for clarity and reproducibility.

```
CleanSpeech/
│   .gitignore
│   README.md
│
├── doc/
```

```
│   └── milestone-1/ through milestone-5/
│       └── Contains detailed reports, diagrams, and analysis for each project phase.
│
├── src/
│   ├── mdeberta-v3-base/
│   │   ├── code/
│   │   │   └── Core notebooks for config, data processing, training, inference, and explainability.
│   │   ├── notebooks-test/
│   │   │   └── Experimental notebooks (e.g., HASOC data prep, alternate training runs).
│   │   └── reports/
│   │       ├── figs/
│   │       │   └── Generated SHAP HTML visualizations for qualitative analysis.
│   │       └── previews/
│   │           └── CSV previews of training & validation data splits.
│   │
│   ├── model-artifacts/
│   │   └── Saved baseline (TF-IDF + Logistic Regression) model files for comparison.
│   │
│   └── tf-idf-logistic-reg/
│       └── Notebooks used to create the baseline model (EDA, preprocessing, training).
│
└── ui/
    └── All source code for the Streamlit web application (app.py, components, etc.).
```

7. Conclusion and Future Work

7.1. Limitations

While successful, this project has several limitations that inform future work.

- **Imbalanced Data:** Rare-class recall (e.g., `threat`) is still lower than for common classes.
- **Token-Level XAI:** SHAP is token-based and can miss phrase-level or idiomatic meanings.
- **Rewrite Tone:** The Gemini rewrites can sometimes over-soften the original intent, losing the author's strong (but non-toxic) emotion.

7.2. Future Work

- **Improved Loss Function:** Experiment with class-balanced focal loss to further boost minority class performance.
- **Advanced XAI:** Explore gradient-based explainability methods (e.g., Integrated

Gradients) that may capture phrase-level context better.

- **Tone-Controlled Rewriting:** Engineer more sophisticated prompts for the Gemini API to allow for "tone preservation" (e.g., "rewrite this to be non-toxic but still 'angry' or 'frustrated'").

7.3. Conclusion

The CleanSpeech project successfully designed and implemented a modular, high-performing system for toxicity detection, explanation, and rewriting. The use of a state-of-the-art mDeBERTa-v3 classifier, validated by transparent SHAP explanations, provides a robust foundation for content moderation. Furthermore, the integration of the Gemini API for constructive rewriting (with a 0.948 BERTScore) moves beyond simple censorship to offer a practical tool for user education and community health. This project provides a strong and extensible framework for building fairer, safer, and more constructive online environments.

8. References and Appendix

References

Adams, C., Sorensen, J., Elliott, J., Dixon, L., McDonald, M., nithum, & Cukierski, W. (2017). Toxic Comment Classification Challenge. Kaggle. Retrieved from *https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge*

Gemini Team, et al. (2025). Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805. Retrieved from *https://arxiv.org/abs/2312.11805*

He, P., Liu, X., Gao, J., & Chen, W. (2021). DeBERTa: Decoding-enhanced BERT with Disentangled Attention. arXiv:2006.03654. Retrieved from *https://arxiv.org/abs/2006.03654*

Lundberg, S., & Lee, S. (2017). A Unified Approach to Interpreting Model Predictions. arXiv:1705.07874. Retrieved from *https://arxiv.org/abs/1705.07874*

Appendix

Appendix A: Full System Architecture Diagram

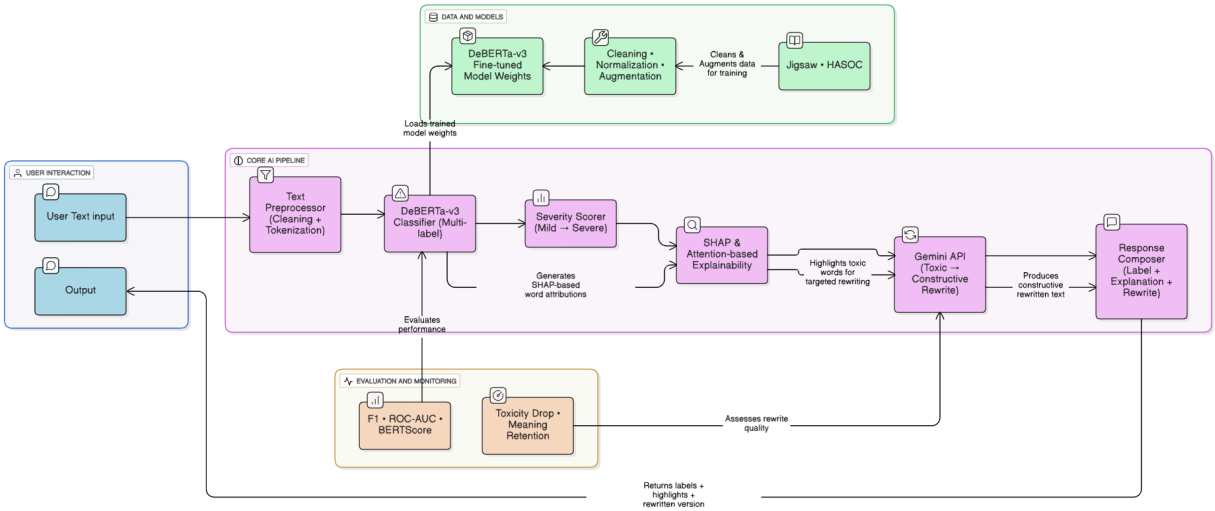

Figure 6: Detailed end-to-end system data and model flow.