CleanSpeech:

Toxicity Detection & Rewriting with Explainable AI

Team-10

Data Science Lab Project

September 2025

Abstract

Online toxicity harms user experience across social platforms and forums. Traditional moderation tools provide scores or flags but lack transparency and offer no constructive guidance for users. CleanSpeech is an end-to-end system that detects multi-label toxicity, explains model decisions at token level, and rewrites harmful text into constructive alternatives. The system uses a fine-tuned mDeBERTa-v3 transformer for multi-label classification, SHAP for explainability, and a large language model (Gemini) for rewriting. Evaluation shows strong discrimination (Macro ROC-AUC = 0.983) and competitive rewrite quality (BERTScore = 0.948). This report documents dataset preparation, model development, XAI integration, evaluation, and deployment artifacts.

Contents

1 Introduction

Online conversations frequently include toxic behaviorhate speech, harassment, threats, and obscenitythat degrade user experience and platform safety. While automated moderation tools exist, they often provide binary or scalar outputs without transparency and do not help users improve their communication. CleanSpeech targets this gap by providing an integrated pipeline that:

1. Detects multiple toxicity categories (multi-label).
2. Provides token-level explanations highlighting words that contributed to the prediction.
3. Generates constructive rewrites that preserve user intent while reducing toxicity.
4. Exposes a user-facing demo (Streamlit) for real-time testing and evaluation.

This document compiles the project milestones into a cohesive final report covering literature, datasets, methodology, model development, evaluation, and deployment.

2 Literature Review

2.1 Toxicity Detection Methods

Early approaches relied on feature engineering (n-grams, sentiment features) with classical classifiers (SVM, Random Forest). Transformer-based models (BERT, RoBERTa, DeBERTa) significantly improved context understanding and nuanced toxicity detection.

Key references:

- Mozafari et al. (2020): BERT-based transfer learning for hate-speech detection.
- Mathew et al. (2021): HateXplain dataset with human-annotated rationales.
- He et al. (2021): DeBERTa with disentangled attention; strong contextual representations.

2.2 Text Rewriting and Detoxification

Style-transfer and politeness transfer research addresses tone conversion but often struggles to preserve semantics. Large language models offer improved semantic fidelity for rewriting tasks.

2.3 Explainable AI in NLP

Model-agnostic methods (SHAP, LIME) and attention-based visualizations are commonly used for interpretability. SHAP offers consistent, local explanations applicable to transformer models.

2.4 Research Gaps

- Lack of end-to-end systems that combine detection, explanation, and constructive rewriting.
- Commercial tools provide scores without actionable feedback.
- Limited support for multilingual and code-mixed content in moderation tools.

3 Dataset and Methodology

3.1 Datasets

Primary: Jigsaw Toxic Comment Classification – Wikipedia talk page comments with six labels: `toxic`, `severe_toxic`, `obscene`, `threat`, `insult`, `identity_hate`. Large, diverse, and multi-label.

Secondary / Test: HASOC (2019–2021) – Multilingual and code-mixed content (English, Hindi, Hinglish) from social media. Used to test domain generalization. HASOC binary labels were augmented using a prompting pipeline (Gemini) to align with Jigsaw's six-category schema.

3.2 Preprocessing Pipeline

Jigsaw:

- Remove URLs, HTML tags, emojis, and non-ASCII characters.
- Collapse extra whitespace; normalize text to lowercase where appropriate.
- Create `clean_text`; preserve original for reference.
- Remove duplicates and rows with missing text.
- Create temporary `any_toxic` flag for stratified 80/20 train/validation split.

HASOC:

- Unified formats (TSV/CSV/XLSX) and standardized column names.
- Preserve usernames/hashtags (strip @/ but keep content).
- Replace URLs with token `<URL>` to retain contextual signal.
- Apply a Gemini-based classifier prompt to augment binary labels into the six-label schema.

3.3 Data Splits and Sizes

- Train (post-processing): 127,397 (Jigsaw-derived)
- Validation: 31,850 (Jigsaw-derived; stratified)
- Test (held-out / Kaggle): 153,000 (used for final evaluation)

3.4 Evaluation Metrics

- Classification: ROC-AUC (per-label and macro), Precision, Recall, F1.
- Explainability: Agreement with human rationales (HateXplain), SHAP token attributions.
- Rewriting: BERTScore, toxicity drop (pre/post rewriting).

4 Model Development and Hyperparameter Tuning

4.1 Model Architecture

Backbone: `microsoft/mdeberta-v3-base` (mDeBERTa-v3) with a sigmoid multi-label classification head outputting six probabilities. Tokenization via DeBERTa tokenizer, truncation/padding to 256 tokens.

4.2 Training Setup

- Optimizer: AdamW (learning rate = 2×10^{-5})
- Scheduler: linear warm-up (10% steps) followed by decay
- Loss: Weighted BCEWithLogitsLoss to address class imbalance
- Batch size: 16 per GPU (32 effective with 2T4)
- Mixed precision (fp16), gradient clipping (norm = 1.0), early stopping (patience = 2)
- Hardware: 2 NVIDIA T4 GPUs (Hugging Face Accelerate)

4.3 Weighted Loss Calculation (example)

```
label_sum = train_df[LABELS].sum(axis=0)
label_freq = label_sum / len(train_df)
class_weights = (1 / (label_freq + 1e-6))
class_weights = class_weights / class_weights.sum() * len(LABELS)
criterion = WeightedBCEWithLogitsLoss(torch.tensor(class_weights.values))
```

4.4 Hyperparameter Experiments

- Learning rate: 2e-5 found most stable (tested 2e-5 – 3e-5).
- Sequence length: 256 tokens slightly better than 128.
- Weight decay: 0.01 reduced overfitting.
- Warm-up ratio: 0.10 produced smoother training.

4.5 Training Outcomes

- Macro ROC-AUC (validation/test): ≈ 0.982–0.989 (high discrimination).
- Early stopping triggered at epoch 3 in best runs.
- Model artifacts saved as Hugging Face safetensors in `best_model/`.

5 Evaluation and Analysis

5.1 Threshold Tuning

Individual thresholds were tuned on validation to maximize per-label F1. Final thresholds used for deployment:

Label	Threshold	Tuned F1
toxic	0.80	0.706480
severe_toxic	0.50	0.375228
obscene	0.70	0.693135
threat	0.10	0.566802
insult	0.50	0.678766
identity_hate	0.40	0.593705

5.2 Final Metrics

- Macro ROC-AUC: 0.983
- Macro F1 (after tuning): 0.68 (optimized)
- Hamming loss: 0.062
- Subset accuracy: 0.47

5.3 Explainability (SHAP & Attention)

- SHAP kernel/explainer used to compute token-level contributions to each label.
- SHAP visual highlights (red → increases toxicity, blue → decreases toxicity).
- Attention heatmaps corroborate token-level focus regions.

5.4 Rewriting Evaluation (Gemini)

- BERTScore (semantic preservation): 0.948
- Rewrites reduce toxicity and preserve user intent in most evaluated examples.
- Example: "You are such an idiot and I hate you!" → "I strongly disagree with your approach and find it frustrating."

5.5 Error Analysis

Main failure modes:

- Sarcasm and figurative speech (e.g., "Nice job, genius") often misclassified.
- Rare labels (threat, identity_hate) have lower recall due to imbalance.

- Mixed-language phrases (Hinglish) sometimes misinterpreted.

5.6 Limitations

- Perturbation-based SHAP can be computationally expensive for long inputs.
- Static thresholds may be suboptimal across domains and user populations.
- Gemini rewriting may over-soften in some contexts, losing emotional force.

6 Deployment and Documentation

6.1 Streamlit UI

Deployment includes a Streamlit application with two primary views:

1. **Chat Interface** Accepts input text, displays predicted labels, severity, SHAP highlights, and constructive rewrite.
2. **Explainability Dashboard** Detailed SHAP plots, attention heatmaps, and token attributions for deeper inspection.

6.2 Local Run Instructions

```
cd ui
pip install -r requirements.txt
streamlit run app.py
```

6.3 Response Format (API / UI)

The system returns JSON-like responses:

```
{
 "labels": {"toxic":1, "insult":1, "threat":0, ...},
 "severity": "Moderate",
 "explanation": "The words 'idiot' and 'stupid' contribute most to toxicity.",
 "rewritten_text": "I think that idea could be improved, but let's discuss calmly."
}
```

6.4 Monitoring and Future Ops

- Monitor drift via periodic evaluation on live samples.
- Collect user feedback on rewrites to further fine-tune Gemini prompts.
- Implement logging for SHAP computations to track explanation distribution.

7 Conclusion and Future Work

CleanSpeech demonstrates that combining a strong transformer-based classifier, robust explainability, and a powerful rewriting engine leads to a practical, transparent moderation workflow. Results indicate high discrimination (ROC-AUC ≈ 0.983) and strong rewriting quality (BERTScore ≈ 0.948).

7.1 Proposed Improvements

- Replace perturbation SHAP with faster gradient-based XAI (Integrated Gradients).
- Experiment with DeBERTa-v3-large or ModernBERT for improved accuracy.
- Extend explicit multilingual training data for improved Hinglish/Hindi handling.
- Add tone control to rewrites (polite / neutral / firm) via control codes.

- Deploy a cloud-based API with model performance monitoring and automated retraining triggers.

8 References

1. Mozafari, M., Farahbakhsh, R., & Crespi, N. (2020). A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media.
2. Mathew, B., et al. (2021). HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection.
3. He, P., et al. (2021). DeBERTa: Decoding-enhanced BERT with Disentangled Attention.
4. Nogueira dos Santos, C., et al. (2018). Fighting Offensive Language on Social Media with Unsupervised Text Style Transfer.
5. Madaan, A., et al. (2020). Politeness Transfer: A Tag and Generate Approach.
6. Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions (SHAP).
7. Jigsaw / Conversation AI. (2018). Toxic Comment Classification Challenge (Kaggle).
8. HASOC (Hate Speech and Offensive Content) 2019–2021.

Appendix

Artifacts

- `best_model/` Fine-tuned mDeBERTa-v3 weights (safetensors).
- `tokenizer/` DeBERTa tokenizer files.
- `toxic-comment-classification.ipynb` Training notebook.
- `evaluation.ipynb` Evaluation scripts and plots.
- `xai_analysis.ipynb` SHAP and attention visualizations.
- `rewrite_eval.ipynb` Gemini rewrite evaluation and BERTScore computation.
- `ui/` Streamlit app (app.py) and requirements.txt.
- `data/` Processed train/validation/test CSVs.

Sample Preprocessing Code

```
import re
def clean_text(text):
    text = re.sub(r"http\S+|www\.\S+", " <URL> ", text)
    text = re.sub(r"[^\x00-\x7F]+", " ", text)
    text = re.sub(r"\s+", " ", text).strip()
    return text.lower()
```

SHAP Visualization Placeholder

(SHAP token-level explanation visualization – place actual image file at the commented path above for final PDF.)

Attention Heatmap Placeholder

(Attention heatmap visualization – place actual image file at the commented path above for final PDF.)