Toxic Classification - Project Documentation

Introduction

In this project, I worked on building a multi-class text classification system to detect and categorize various types of toxic content. The categories included a wide range of topics like Violent Crimes, Unsafe Content, Suicide & Self-Harm, Sexual Exploitation, and more.

I experimented with multiple deep learning models including LSTM, Bidirectional LSTM, and a fine-tuned DistilBERT model with LoRA (Low-Rank Adaptation) to see how different architectures handle toxicity classification.

Dataset

The dataset contains 8,955 samples after balancing and includes the following columns:

- query (main text)
- image descriptions (extra context - not used in modeling for now)
- Toxic Category (target class)

Original Class Distribution:

Safe: 995

Violent Crimes: 792

Non-Violent Crimes: 301

unsafe: 274

Unknown S-Type: 196

Sex-Related Crimes: 115

Suicide & Self-Harm: 114

Elections: 110

Child Sexual Exploitation: 103

Balancing:

To address class imbalance, I applied random oversampling using sklearn.utils.resample to bring all classes to 995 instances.

Toxic Classification - Project Documentation

Preprocessing Pipeline

I applied several NLP preprocessing techniques using NLTK:

1. Lowercased all text

2. Removed special characters and digits

3. Tokenized using word_tokenize

4. Removed stopwords

5. Lemmatized each token using WordNetLemmatizer

6. Encoded class labels with LabelEncoder

7. Applied TF-IDF for traditional vectorization (baseline)

For deep learning models, I used Keras Tokenizer and padding to convert text into padded sequences.

Data Splitting

- 70% training
- 15% validation
- 15% testing

(Split using stratify to maintain class distribution)

Model 1: LSTM

Architecture:

Embedding -> LSTM -> Dropout -> Dense -> Softmax

Loss: Categorical Crossentropy

Optimizer: Adam

Performance:

Accuracy: 11%

The model completely failed on all categories except "Safe".

Model 2: Bidirectional LSTM with Class Weights

Toxic Classification - Project Documentation

Improvements:

- Used Bidirectional LSTM
- Applied Class weights to emphasize minority classes
- Added EarlyStopping

Performance:

Accuracy: 94%

Precision/Recall/F1: High across all categories

Model 3: DistilBERT Fine-Tuned with LoRA (PEFT)

Setup:

- Tokenized text using DistilBertTokenizerFast
- Used max length = 128
- Applied LoRA targeting q_lin and v_lin modules
- Fine-tuned for 3 epochs with Trainer API

Performance:

Accuracy: ~94%

F1-score: Strong across all classes

Results Summary

Model	Accuracy	Macro F1	Notes
LSTM	11%	0.02	Poor generalization; only predicted Safe
Bidirectional LSTM	94%	0.94	Great results with class weights
DistilBERT + LoRA	~94%	~0.94	Transformer-based, best generalization

Observations

- Traditional RNNs (like LSTM) struggle with class imbalance unless tuned or weighted.

- BiLSTM performed very well with class weights.
- DistilBERT with LoRA was efficient and powerful.
- image descriptions column wasnt used but could be explored in future multimodal setups.

Tools & Libraries

- Python: Pandas, Numpy, Matplotlib, Seaborn
- NLP: NLTK, scikit-learn, HuggingFace Transformers
- Deep Learning: TensorFlow / Keras
- PEFT: Low-Rank Adaptation for BERT fine-tuning

Future Work

- Incorporate image descriptions as part of a multimodal model.
- Try other pre-trained transformer models like RoBERTa or BERTweet.
- Add explainability (e.g., attention visualization).
- Explore ensemble techniques to combine predictions from multiple models.