Internship Report: Week 1

Author: Yahya Alnwsany
Period: Internship Week 1
Company: Cellula AI
Department: NLP Engineer Internship
Supervisor: Jannah Mahmoud
Week 1 Repo: Week 1 Repo

Project Context

This report documents the first phase of the internship project: *Safe and Responsible Multi-Modal Toxic Content Moderation*.
The overall goal is to build a dual-stage, multi-modal moderation system for both text and images, combining state-of-the-art NLP and vision models. This week's work lays the foundation for the text moderation pipeline, which will be extended and integrated into the full system in subsequent weeks.

Executive Summary

During my first week at Cellula AI, I initiated the development of a robust toxic comment classification pipeline. The main goal was to explore the performance trade-offs between a classical deep learning architecture and a transformer-based model utilizing advanced fine-tuning methods. Key accomplishments included full data preprocessing, designing and training two models (custom LSTM-based and DistilBERT + LoRA), and generating comparative metrics to guide deployment decisions. This foundational week ensures scalability, reproducibility, and future extensibility.

Note: This is the first part of a larger project that will include a hard moderation filter (Llama Guard), image captioning and moderation (BLIP), and a Streamlit-based deployment in the coming weeks.

1. Data Pipeline Overview

1.1 Data Source and Loading

- **Dataset:** `data/cellula-toxic.csv` – includes multi-class labeled user comments with toxicity annotations.
- **Tool:** `pandas` was used to load and inspect the dataset.

1.2 Cleaning and Normalization

- Lowercased all comments for consistency.
- Removed emojis, special characters, and HTML entities.
- Applied whitespace normalization.
- Optional: stopword removal and lemmatization (NLTK, spaCy).

1.3 Tokenization

A custom tokenizer was created using either HuggingFace's `Tokenizer` or Keras tokenizer. The tokenizer was fitted on the cleaned corpus to map words into integer sequences, crucial for deep learning models.

1.4 Label Encoding

Multi-class labels were mapped using `data/label_map.json`, ensuring consistency during training and evaluation.

1.5 Dataset Splitting

- Stratified split into train, validation, and test sets (60/20/20).
- Maintained proportional representation of classes.
- Used `train_test_split` from sklearn with `stratify=y`.

1.6 Saved Artifacts

- `data/cleaned.csv`, `tokenizer.pkl`, `label_map.json`
- `train.csv`, `eval.csv`, `test.csv`

2. Modeling Approaches

2.1 Deep Learning Baseline – Why LSTM?

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to capture long-range dependencies in sequential data. In the context of toxic comment classification, LSTMs are well-suited because they can model the order and context of words, which is crucial for understanding nuanced language and detecting subtle forms of toxicity.

Why use LSTM?

- **Context Awareness:** LSTMs can remember information over long sequences, making them effective for text where context matters.
- **Bidirectionality:** Using a Bidirectional LSTM allows the model to consider both past and future context in a sentence, improving classification accuracy.
- **Efficiency:** LSTMs are less computationally intensive than transformers, making them suitable for rapid prototyping and deployment on resource-constrained systems.
- **Interpretability:** The architecture is relatively simple and easy to debug compared to more complex models.

```
Embedding(vocab_size, 128, input_length=max_len),
Bidirectional(LSTM(64, return_sequences=True)),
GlobalMaxPooling1D(),
Dropout(0.3),
Dense(64, activation='relu'),
Dropout(0.3),
Dense(num_classes, activation='softmax')
```

Explanation: The embedding layer converts words to dense vectors. The Bidirectional LSTM captures context from both directions. GlobalMaxPooling1D reduces the sequence to a fixed-length vector. Dense and Dropout layers add non-linearity and regularization, and the final Dense layer outputs class probabilities.

Performance:

- Accuracy: **94%**
- Macro F1: **82%**, Weighted F1: **94%**

```
          precision    recall  f1-score   support
2       1.00         0.91      0.95       11
3       0.94         0.98      0.96       45
6       0.33         0.25      0.29        4
7       0.97         1.00      0.99       35
8       1.00         0.86      0.92        7
Accuracy: 0.94 (n=102)
```

2.2 Transformer-Based Model – DistilBERT with PEFT (LoRA)

Transformers have revolutionized NLP by enabling models to learn contextual relationships between words in a sentence using self-attention mechanisms.

DistilBERT is a distilled (compressed) version of BERT, offering nearly the same performance as BERT but with fewer parameters and faster inference.

Parameter-Efficient Fine-Tuning (PEFT) is a family of techniques that allow large pre-trained models to be adapted to new tasks by training only a small subset of parameters, rather than the entire model. This is especially important for deploying transformer models in production, where memory and compute resources may be limited.

What is LoRA?

LoRA (Low-Rank Adaptation) is a PEFT method that injects small, trainable low-rank matrices into each layer of a transformer model. Instead of updating all the weights in the model, LoRA only updates these additional matrices, drastically reducing the number of trainable parameters.

Benefits of LoRA:

- **Efficiency:** Requires less memory and compute, making fine-tuning feasible on modest hardware.
- **Speed:** Faster training and inference compared to full fine-tuning.
- **Performance:** Achieves results comparable to full fine-tuning in many tasks.
- **Modularity:** LoRA adapters can be swapped in and out, allowing for easy experimentation and deployment.

Why use PEFT/LoRA in this project? The toxic comment classification task benefits from the language understanding of large models like DistilBERT, but full fine-tuning is resource-intensive. LoRA enables efficient adaptation of DistilBERT to our specific dataset, making it practical to deploy high-performing models even with limited resources.

- **Base Model:** distilbert-base-uncased
- **Fine-Tuning Method:** LoRA adapters via PEFT
- **Epochs:** 3

- **Learning Rate:** 5e-5
- **Optimizer:** AdamW

```
"epoch": 3.0,
"eval_loss": 0.4127,
"eval_runtime": 1.82,
"eval_samples_per_second": 167.0,
"eval_steps_per_second": 10.44
```

Artifacts Produced:

- adapter_model.safetensors (LoRA adapter weights)
- all_results.json (training and evaluation metrics)
- training_args.bin (training configuration)

3. Comparative Summary

Aspect	LSTM Baseline	DistilBERT + LoRA
Performance	Strong (94% Acc)	Strong (Eval Loss: 0.41)
Training Cost	Low	Medium
Inference Speed	High	Medium
Flexibility	Good for Edge	Better for NLP Stack
Next Steps	Hyperparam Tuning	RoBERTa / DeBERTa PEFT

4. Next Week Objectives

- Hyperparameter grid search for both models
- Model quantization for deployment
- Inference pipeline and REST API setup
- Try RoBERTa or ALBERT under PEFT
- Build a Streamlit dashboard for live demo

Appendix: Folder Layout

```
.\
├── data/
│   ├── cellula-toxic.csv
│   ├── cleaned.csv
│   ├── tokenizer.json
│   ├── label_map.json
│   ├── train.csv / eval.csv / test.csv
├── models/
│   ├── toxic_classifier.keras
│   ├── toxic_classifier_v3.keras
│   └── LSTM Model (Hugging Face)
│   └── LSTM Model (GitHub)
├── src/
│   ├── preprocess.py
│   ├── tokenize_and_split.py
├── FineTuned-DB-ToxicClassifier/
│   ├── adapter_model.safetensors
│   ├── all_results.json
│   ├── training_args.bin
│   └── DistilBERT+LoRA (Hugging Face)
│   └── DistilBERT+LoRA (GitHub)
```

Prepared by:
Yahya Alnwsany
Cellula AI Intern – Week 1

[My Portfolio]

[Week 1 on GitHub]