Internship Report: Week 2

Author: Yahya Alnwsany
Period: Internship Week 2
Company: Cellula AI
Department: NLP Engineer Internship
Supervisor: Jannah Mahmoud
Week 2 Repo: Week 2 Repo
Live Demo: Hugging Face Space

Project Context

This report documents the second phase of the internship project: *Safe and Responsible Multi-Modal Toxic Content Moderation*.
Building on Week 1's text moderation foundation, this week focused on deploying a dual-stage, multi-modal moderation system as a production-ready Streamlit web app. The system now supports both text and image inputs, leverages state-of-the-art vision-language and transformer models, and implements a robust, research-driven moderation workflow.

Executive Summary

During Week 2 at Cellula AI, I transformed the research pipeline into a real-world, interactive moderation tool. The app integrates a hard safety filter (Llama Guard), a fine-tuned DistilBERT+LoRA classifier, and BLIP for image captioning. I addressed class imbalance, improved model robustness, and delivered a user-friendly, dual-stage moderation workflow. The result is a scalable, extensible, and transparent system ready for real-world deployment and further research.

1. Dual-Stage Moderation Pipeline

1.1 Stage 1: Hard Filter (Llama Guard)

- **API:** Llama Guard (Meta, via OpenRouter API)
- **Purpose:** Instantly blocks content that is legally or ethically unsafe (e.g., violence, hate, sexual exploitation).
- **Prompt:** Strict system prompt ensures only 'safe' or 'unsafe' is returned.
- **Logic:** If unsafe, user is notified and moderation stops. If safe, content proceeds to soft classifier.

1.2 Stage 2: Soft Classifier (DistilBERT+LoRA)

- **Model:** DistilBERT (transformer) fine-tuned with PEFT-LoRA for 9-class toxic content classification.
- **Categories:** Safe, Violent Crimes, Elections, Sex-Related Crimes, Unsafe, Non-Violent Crimes, Child Sexual Exploitation, Unknown S-Type, Suicide & Self-Harm.
- **Output:** Displays predicted category and class probabilities for transparency.
- **Improvements:** Addressed class imbalance with resampling/augmentation (SMOTE, class weights, oversampling).

1.3 Image Support (BLIP)

- **Model:** BLIP (Bootstrapped Language-Image Pre-training, Salesforce)
- **Purpose:** Generates captions for uploaded images, enabling moderation of visual content via the same pipeline.
- **Integration:** Caption is appended to text input and passed through both moderation stages.

2. Streamlit App Deployment

- **Interface:** Accepts raw text and/or image uploads. Displays moderation results with clear feedback and probabilities.
- **Workflow:** User input → BLIP caption (if image) → Llama Guard filter → DistilBERT+LoRA classifier (if safe).
- **Reproducibility:** All code, model weights, and requirements are versioned and documented.

- **Live Demo:** [Hugging Face Space](https://huggingface.co/spaces/NightPrince/Dual-Stage-Toxic-Moderation)

3. Model Selection & Class Imbalance

- Compared PEFT-LoRA DistilBERT and baseline CNN/LSTM on validation set (accuracy, F1-score, confusion matrix).
- Analyzed class distribution and addressed imbalance with SMOTE, class weights, and oversampling.
- Retrained and selected the best model for deployment in the app.

4. Reporting & Documentation

- Recorded results of class imbalance experiments and model selection.
- Documented Llama Guard API and BLIP integration.
- Summarized dual-stage logic and provided code documentation for reproducibility.
- All code and artifacts are available on GitHub and Hugging Face.

5. Folder Structure

```
Week2/
├── app_streamlit.py            # Main Streamlit app (UI, imports modular pipeline)
├── pipeline/                   # Modular pipeline for moderation logic
│   ├── __init__.py             # Pipeline package init
│   ├── blip_caption.py         # BLIP image captioning module
│   ├── llama_guard.py          # Llama Guard API filter module
│   └── toxic_classifier.py     # DistilBERT+LoRA classifier module
├── requirements.txt            # Python dependencies (transformers, streamlit, peft, torch, etc.)
├── .env.example                # Example environment file for API keys
├── README.md                   # Project overview, setup, and usage
├── internship_week2_report.html # This report (detailed, professional)
```

6. Key Code Snippets & Modular Pipeline

New in Week 2: The moderation logic is now fully modularized for clarity, maintainability, and extensibility. Each stage of the pipeline is encapsulated in its own Python module under `pipeline/` :

- `blip_caption.py` : Handles BLIP image captioning.
- `llama_guard.py` : Calls the Llama Guard API for hard filtering.
- `toxic_classifier.py` : Loads and runs the DistilBERT+LoRA classifier.

The main app (`app_streamlit.py`) imports and orchestrates these modules, providing a clean, professional codebase ready for production or research extension.

6.1 Llama Guard API Call (pipeline/llama_guard.py)

```
def llama_guard_filter(text):
    headers = {
        "Authorization": "Bearer ...",
        "Content-Type": "application/json"
    }
    data = {
        "model": "meta-llama/llama-guard-3-8b",
        "max_tokens": 25,
        "messages": [
            {"role": "system", "content": "You are a strict safety
classifier. Respond with only one word: either 'safe' or 'unsafe'. Do not
add punctuation, explanation, or formatting."},
            {"role": "user", "content": text}
        ]
    }
    response =
requests.post("https://openrouter.ai/api/v1/chat/completions",
headers=headers, json=data)
```

```
    ...
```

6.2 BLIP Captioning (pipeline/blip_caption.py)

```
def caption_image(img):
    processor, model = load_caption_model()
    inputs = processor(images=img, return_tensors="pt")
    pixel_values = inputs["pixel_values"]
    out = model.generate(pixel_values=pixel_values)
    caption = processor.decode(out[0], skip_special_tokens=True)
    return caption
```

6.3 DistilBERT+LoRA Classifier (pipeline/toxic_classifier.py)

```
def classify_toxicity(text_input, caption):
    pipe = load_toxic_classifier()
    full_input = text_input + " [SEP] " + caption
    preds = pipe(full_input)
    ...
```

7. Results, User Experience & Improvements

- **Modular Pipeline:** All moderation logic is now split into clear, reusable modules for each stage.
- **Professional UI:** Streamlit app provides instant feedback, clear error handling, and supports both text and image moderation.
- **Reproducibility:** All code, model weights, and requirements are versioned and documented for easy setup and extension.
- **Documentation:** README and this report have been expanded to match Week 1's quality, with full project context, setup, model details, and results.

- **Extensibility:** The modular structure allows for easy addition of new moderation stages, models, or features (e.g., logging, authentication, advanced analytics).

- App provides instant feedback on unsafe content (Stage 1) and detailed category probabilities (Stage 2).
- Supports both text and image moderation, with clear UI and error handling.
- All results, code, and models are reproducible and open source.

8. Next Steps

- Expand to multi-language support and more nuanced categories.
- Integrate user authentication and moderation logs.
- Deploy as a cloud service with REST API.
- Continue benchmarking and model improvements.

Appendix: References & Resources

- Hugging Face Space (Live Demo)
- GitHub Repo
- Week 1 Documentation
- Llama Guard (Meta)
- DistilBERT+LoRA (Hugging Face)
- BLIP (Salesforce)
- Author Portfolio

Prepared by:
Yahya Alnwsany
Cellula AI Intern – Week 2
My Portfolio
Week 2 on GitHub