Report 1: Multi-Label Toxicity Classification (LoRA Fine-Tuning)

File: `toxicity_bert_model.ipynb`

This report details the fine-tuning of a DistilBERT model for multi-label toxicity classification using Parameter-Efficient Fine-Tuning (PEFT) via LoRA, and the comprehensive evaluation methodology applied.

1. Fine-Tuning Setup

Data

- **Source:** The model was trained on the `thesofakillers/jigsaw-toxic-comment-classification-challenge` dataset, sourced from Hugging Face datasets (using only `train.csv`).
- **Target Labels:** This is a multi-label classification task with six binary targets: **"toxic"**, **"severe_toxic"**, **"obscene"**, **"threat"**, **"insult"**, and **"identity_hate"**.
- **Preprocessing & Splitting:** The notebook loads the dataset and performs a 95% / 5% stratified split (using `seed=42`), resulting in 151,592 training samples and 7,979 validation samples. Text is tokenized with a `MAX_LEN` of 256.

Method

- **Base Model:** `distilbert-base-uncased`.
- **Fine-Tuning Technique:** The model was trained using **LoRA (Low-Rank Adaptation)** via the `peft` library, rather than full fine-tuning. This dramatically reduces the computational burden.
    - **LoRA Configuration:** A rank (r) of **8** and alpha of **16** were used. LoRA adapters were specifically targeted only to the attention mechanism's linear layers: `q_lin`, `k_lin`, `v_lin`, and `out_lin`.
    - **Efficiency:** This approach trained only **890,118** parameters, representing just **1.3119%** of the model's total 67.8 million parameters. The classification head (`pre_classifier` and `classifier`) was also trained.
- **Training Process:** Training was conducted using a custom PyTorch loop for **2 epochs** with a batch size of 16.
    - **Optimizer:** `AdamW` (LR: 2e-5, Weight Decay: 0.01).
    - **Scheduler:** A linear schedule with a 6% warmup phase.
    - **Loss Function:** `BCEWithLogitsLoss`, which is the correct loss function for multi-label binary classification tasks.

Results (Training)

The model was evaluated against the validation set at the end of each epoch using the **micro-F1 score** (at a default 0.5 threshold).

- **Epoch 1:** Validation Micro-F1 = **0.7556**
- **Epoch 2:** Validation Micro-F1 = **0.7658** (Final selected metric at 0.5 threshold)

2. Evaluation Methodology and Outcomes

A deep, multi-faceted evaluation was performed on the validation set logits *after* training was complete.

Methodology

1. **Metric Selection:** While Micro-F1 (which aggregates all label predictions globally) was used during training, the post-hoc evaluation focused heavily on metrics sensitive to per-label performance and calibration, recognizing the severe class imbalance (e.g., "threat" prevalence was only 0.18%).
2. **Threshold Tuning (Primary Eval):** The default 0.5 prediction threshold is suboptimal for imbalanced multi-label tasks. This notebook calculates the full **Precision-Recall (PR) Curve** for each of the 6 labels *independently*. It then selects the specific threshold for each label that maximizes that individual label's F1 score.
3. **Probabilistic Evaluation:** Probabilistic accuracy was measured using:
    - **AUPRC (Area Under PR Curve):** Measures performance across all possible thresholds.
    - **Brier Score:** A measure of probabilistic calibration (lower is better).
4. **Error Analysis (Qualitative):** The notebook analyzes the Top 3 False Positives (FPs) and False Negatives (FNs) for each label, sorted by their margin (distance from the tuned threshold), to identify systematic model weaknesses.
5. **Performance Slicing:** Model performance was analyzed across data segments based on comment length (short, medium, long).

Outcomes (Quantitative)

- **Performance (Default 0.5 Threshold):**
    - Micro-F1: 0.7658
    - Macro-F1: 0.4550 (This low score highlights the failure of the 0.5 threshold on rare classes).
- **Performance (Per-Label Tuned Thresholds):**
    - Micro-F1: 0.7617
    - Macro-F1: **0.5879** (A 13.3% absolute improvement in Macro-F1, demonstrating the critical necessity of threshold tuning).
- **Tuned Thresholds (Key Artifact):** The optimal thresholds derived from the PR curves were:

  - `toxic`: 0.395
  - `severe_toxic`: 0.275
  - `obscene`: 0.448
  - `threat`: **0.068** (This very low threshold is required to identify the rare "threat" class).
  - `insult`: 0.451
  - `identity_hate`: 0.166
- **Probabilistic Metrics (Macro Mean):**
  - Macro Mean AUPRC: 0.5805
  - Macro Mean Brier Score: 0.0125 (This is a very low/good calibration score).

Outcomes (Qualitative)

- **Length-Based Performance:** The model performs well on short/medium comments but degrades significantly on longer ones (Micro-F1 drops to 0.6889 for comments > 40 words).
- **Error Analysis:** The Top FN analysis reveals the model fails to flag:
  - Obfuscated slurs (e.g., "n! gger!").
  - Implicit or veiled threats (e.g., "...unban this ip... you have been warned").
- The Top FP analysis shows the model incorrectly flags aggressively sarcastic comments as "threats" (e.g., "...please fucking die if you like this film. just die.").

Report 2: Binary Sarcasm Detection (Comparative Models)

File: `Untitled0.ipynb`

This report details a comparative study of two models (DistilBERT vs. BERTweet) fine-tuned for binary sarcasm detection on tweet data, including specialized preprocessing and class-weighting techniques.

1. Fine-Tuning Setup

Data

- **Source:** The `tweet_eval` dataset, `irony` subset (sourced from Hugging Face).
- **Target Labels:** Binary classification: 0 (Non-Sarcastic/Non-Ironic) or 1 (Sarcastic/Ironic).
- **Splitting:** The experiment uses the dataset's predefined splits: 2,862 (Train), 955 (Validation), and 784 (Test).
- **Class Imbalance:** The training set has a slight imbalance (approx. 1.01:0.99 ratio), which is addressed in the second experiment.

Method (Experiment 1: DistilBERT Baseline)

- **Base Model:** `distilbert-base-uncased`.
- **Training Process:** A standard full fine-tuning was performed using the Hugging Face `Trainer`.
- **Hyperparameters:** 3 epochs, LR 2e-5, Batch Size 32.
- **Results (Training):** Achieved a peak Validation Accuracy of **0.6454**.

Method (Experiment 2: BERTweet Refinement)

This experiment represents a sophisticated refinement to improve performance by using a domain-specific model and addressing data biases.

- **Preprocessing:** A custom `preprocess_tweet` function was applied to normalize URLs and replace user mentions (`@handle`) with a generic `@USER` token. This is a standard best practice for tweet-based models.
- **Base Model:** `vinai/bertweet-base`, a RoBERTa-based model pre-trained specifically on 850M English tweets, making it ideal for this domain.
- **Fine-Tuning Technique:** Full fine-tuning using a custom **`WeightedTrainer`**.
- **Class Weighting (Key Method):** The custom trainer calculates the class imbalance and passes a weight tensor—`[1.0098, 0.9903]`—to the `CrossEntropyLoss` function. This penalizes the model slightly more for misclassifying the (marginally) rarer negative class.
- **Hyperparameters:** 5 epochs, LR 2e-5, Weight Decay 0.01, and a 6% warmup.
- **Model Selection:** The methodology utilized **Early Stopping** (patience=1) monitoring the Validation **F1 score**, ensuring the model saved was the one that achieved the highest F1 (not just accuracy or lowest loss).

Results (Training)

- The BERTweet model significantly outperformed the DistilBERT baseline.
- Training logs show the best validation performance (F1: 0.7606) was achieved at **Epoch 4 (step 360)**, which was correctly identified by the `load_best_model_at_end=True` callback.

2. Evaluation Methodology and Outcomes

Both models were subjected to rigorous post-hoc quantitative analysis. The BERTweet evaluation (Experiment 2) is the primary focus.

Methodology

- **DistilBERT (Exp 1) Evaluation:**

- Metrics included Accuracy, F1, ROC-AUC, Brier Score (0.2109), and ECE (0.0567).
- Threshold Tuning: The notebook optimized the threshold by maximizing the **F1 score**, identifying a threshold of 0.340.

- **BERTweet (Exp 2) Evaluation:**
  - Metrics: A full classification report including Accuracy, Precision, Recall, F1, Specificity, and Balanced Accuracy.
  - Threshold Tuning (Key Difference): The threshold for the final BERTweet model was optimized by maximizing **Balanced Accuracy** (Average of Recall and Specificity) rather than F1. This is a robust choice when both classes (sarcastic and non-sarcastic) are equally important to identify correctly. The optimal threshold was found to be **0.780**.

Outcomes (Quantitative - BERTweet at T=0.780)

The final BERTweet model (evaluated on the 955-sample validation set) yielded strong, balanced results:

- **Optimal Threshold:** 0.780 (Note: This is significantly higher than the 0.340 required by the baseline DistilBERT, suggesting the BERTweet model produces much more confident positive predictions).
- **Overall Accuracy:** 0.760
- **Balanced Accuracy: 0.761**
- **Confusion Matrix:**
  - TN: 374 | FP: 125
  - FN: 104 | TP: 352
- **Class 1 (Sarcasm):** Precision: 0.738 | Recall (Sensitivity): **0.772** | F1: 0.755
- **Class 0 (Non-Sarcasm):** Specificity (Recall): **0.749**

Outcomes (Qualitative)

- **Latency:** The baseline DistilBERT model (Cell 19) averaged **53.1 ms** per inference call (on the Colab GPU).
- **Performance Slicing (DistilBERT):** Analysis showed the DistilBERT baseline performed relatively consistently across short, medium, and long tweets (Cell 17). (Note: This analysis was not repeated for the final BERTweet model, but the BERTweet model's overall quantitative superiority is definitive).