Hate Speech Detection (Filipino-English) — Project Documentation

1. Overview

- Goal: Detect hate speech in bilingual (Filipino/English) text using a BiLSTM model and deploy in a Tkinter GUI.
- Highlights:
  - Unified multiple datasets (TikTok Filipino hate speech + additional CSVs)
  - Baseline BiLSTM achieved ~84.3% validation accuracy, ~0.885 F1
  - GUI provides real-time moderation with adjustable threshold

2. Dataset & Preprocessing

- Sources:
  - `hatespeech/train.csv`, `valid.csv`, `test.csv`
  - `filipino-tiktok-hatespeech-main/data/*.csv`
  - `aggression_parsed_dataset.csv`, `cyberbullying_tweets.csv`
- Unified loader: `load_unified_dataset.py` merges sources, standardizes `text,label,source`.
- Preprocessing: `text_preprocessing.py`
  - Lowercasing; normalize URLs/mentions/hashtags
  - Remove excess punctuation; keep language-specific tokens
  - Tokenization via custom `Vocabulary` class
- Vocabulary
  - Size used in current run: 10,000 tokens (`vocabulary.pkl`)
  - Special tokens: `<PAD>`, `<UNK>`, `<SOS>`, `<EOS>`

3. Model Design

- Architecture: `rnn_model.py` → `BiLSTMHateSpeechClassifier`
  - Embedding: `vocab_size x 128`
  - BiLSTM: 2 layers, hidden size 128, bidirectional
  - Dropout: 0.3
  - Dense + Sigmoid for binary classification
- Forward pass:
  - Inputs: padded sequences (`LongTensor`) and lengths
  - Packed sequence for LSTM efficiency
  - Concatenate final forward/backward hidden states → dropout → dense → logits
- Parameter Count: printed in notebook (Cell 13) via `count_parameters()`

4. Training Setup

- Files: `train_bilstm.py`, Notebook Cells 15–18
- Optimizer: Adam (`lr=0.001`)
- Loss: `BCEWithLogitsLoss`
- Batching: `torch.utils.data.DataLoader` with custom `collate_fn`
- Splits:
  - Train/Val/Test = 60/20/20 via stratified splits on label
- Imbalance handling:
  - Observed 67% hate, 33% non-hate in train
  - Recommended: `pos_weight = non_hate/hate` in `BCEWithLogitsLoss`

5. Hyperparameter Tuning

- Configs explored (see `hyperparameter_tuning.py`, notebook Section 6):
  - Baseline (current): 128 embed, 128 hidden, 2 layers, dropout 0.3, Adam, batch 64
  - Extended/Deep/Fast/Regularized variants (see `hyperparameter_comparison.png`)
- Findings:
  - Baseline best F1/accuracy
  - Excessive dropout + SGD led to failure (Config 5)

6. Evaluation

- Metrics: Accuracy, Precision, Recall, F1 (`sklearn.metrics`)
- Plots: `training_history.png`, `confusion_matrix.png`
- Current run (10k vocab, epoch 10):
  - Val Acc: 0.8434; Val F1: 0.8854
- Error Analysis:
  - Class imbalance: model biased to predict hate (~69.5% of test predicted as hate)
  - Vocabulary coverage: English polite phrases map partially to `<UNK>` → misclassification
  - Threshold tuning helps: moving from 0.5 → 0.75 reduces false positives

7. Deployment (GUI)

- App: `social_media_app.py` and `gui_app.py`
- Features:
  - Real-time text preprocessing + inference
  - Adjustable threshold slider (recommended 0.7–0.8 for current model)
  - Visual feedback: safe vs violation dialogs
- Files used: `best_bilstm_model.pt`, `vocabulary.pkl`

8. Reproducible Pipeline

1. Prepare environment (Python 3.10+, PyTorch, scikit-learn, seaborn)
2. Load datasets via `load_unified_dataset.py`
3. Preprocess & build/load `vocabulary.pkl`
4. Split data (train/val/test)
5. Initialize model (`rnn_model.py`)
6. Train with Adam; optionally set `pos_weight` in `BCEWithLogitsLoss`
7. Save best model to `best_bilstm_model.pt`
8. Evaluate; generate plots and reports
9. Run GUI for live moderation

9. Recommendations & Next Steps

- Mitigate imbalance:
  - Use class weights (`pos_weight`) or weighted sampler
  - Oversample non-hate or undersample hate for training
- Improve vocabulary:
  - Increase to 20k tokens; include common English polite phrases
  - Consider subword tokenization (BPE/WordPiece) for robustness
- Threshold calibration:
  - Calibrate on validation set for optimal precision/recall tradeoff
  - Persist recommended threshold in GUI defaults
- Logging & Monitoring:
  - Add prediction logging; track false positives in GUI
  - Periodically retrain with feedback data

10. Results Snapshot

- Best Val Acc (current run): 0.8434

- Best Val F1: 0.8854
- Confusion Matrix and Classification Report saved

11. Appendix

- Code Files:
  - Notebook: `HateSpeech_Detection_Complete.ipynb`
  - Model: `rnn_model.py`
  - Preprocessing: `text_preprocessing.py`
  - Training: `train_bilstm.py`, `hyperparameter_tuning.py`
  - Data Loader: `load_unified_dataset.py`
- Artifacts:
  - `best_bilstm_model.pt`, `vocabulary.pkl`
  - `training_history.png`, `confusion_matrix.png`, `hyperparameter_comparison.png`

Export to PDF

If you want a polished PDF: 1. Open `ANN_Documentation.md` in VS Code. 2. Install the "Markdown PDF" extension. 3. Use `Ctrl+Shift+P` → "Markdown PDF: Export (pdf)".

Alternatively, via Pandoc (if installed):

```
pandoc ANN_Documentation.md -o ANN_Documentation.pdf
```