Manual

Project Overview

My tool classifies bug reports from deep learning frameworks as either performance-related or not. I developed in an iterative way:

1. **Baseline Model**: Naive Bayes with TF-IDF
2. **Intermediate Model**: SVM with Word2Vec embeddings
3. **Hybrid Model**: Ensemble approach with multiple feature types (main tool) - final iteration

Project Structure

```
CG-ISE/
├── lab1/                      # Main project directory
│   ├── __pycache__/           # Python cache files
│   ├── comprehensive_results/ # Generated results from testing
│   │   ├── plots/             # Visualisations from the evaluation
│   │   ├── caffe_results.csv  # Results for Caffe dataset
│   │   ├── keras_results.csv  # Results for Keras dataset
│   │   ├── pytorch_results.csv # Results for PyTorch dataset
│   │   ├── tensorflow_results.csv # Results for TensorFlow dataset
│   │   └── evaluation_summary.md # Summary report of all evaluations
│   ├── datasets/              # Contains bug report datasets for each framework
│   ├── baseline_model.py      # Implementation of the Naive Bayes + TF-IDF model - Note -
this is the baseline model from lab1, but adapted so that it fits with my test scripts and
visualisations.
│   ├── intermediate_model.py  # Implementation of the SVM + Word2Vec model
│   ├── hybrid_model.py        # Implementation of the ensemble model (main tool) - this
is my final model - please use this when marking the model on how it beats the baseline
│   ├── preprocessing.py       # Text preprocessing utilities
│   ├── test_all_models.py     # Comprehensive evaluation script for all models - this is
the main testing file for comprehensive summary and visualisations
│   ├── test_hybrid_model.py   # Script to test hybrid model against baseline
│   └── test_intermediate_model.py # Script to test intermediate model against baseline
├── download_nltk_resources.py # Script to download required NLTK resources
├── .gitignore                 # Git ignore file
├── requirements.txt           # Project dependencies
├── README.md                  # Project documentation
├── manual.pdf                 # User manual
├── replication.pdf            # Instructions for replicating results
└── requirements.pdf           # Detailed project requirements
```

Module Descriptions

Core Models

1. **`baseline_model.py`**
   - Implements a Naive Bayes classifier with TF-IDF features
   - Serves as the baseline for comparison
   - Includes options for SMOTE to handle class imbalance
2. **`intermediate_model.py`**
   - Implements an SVM classifier with Word2Vec embeddings
   - Enhances semantic understanding of bug reports
   - Includes Word2Vec training and vectorisation
3. **`hybrid_model.py`**

    - Main tool with ensemble approach
    - Combines multiple feature types:
        - TF-IDF with domain-specific term weighting
        - Pattern-based regex features
        - Code-aware tokenisation
        - Meta-features from report structure
    - Uses multiple classifiers combined through voting

Utility Modules

4. **`preprocessing.py`**
    - Text preprocessing utilities
    - Handles code blocks, HTML, emojis
    - Extracts code-related features
    - Framework-specific technical term preservation

Testing Scripts

5. **`test_all_models.py`**
    - Comprehensive evaluation of all three models
    - Tests on all available frameworks
    - Performs statistical analysis
    - Generates visualisations and summary report
6. **`test_hybrid_model.py`**
    - Tests the hybrid model against the baseline
    - Can be run on any individual framework
7. **`test_intermediate_model.py`**
    - Tests the intermediate model against the baseline
    - Can be run on any individual framework

Using the Models

Testing Individual Models

Testing Intermediate Model against Baseline

```
python test_intermediate_model.py --framework tensorflow
```

You can replace `tensorflow` with any of: `pytorch`, `keras`, `mxnet`, or `caffe`.

Testing Hybrid Model against Baseline

```
python test_hybrid_model.py --framework pytorch
```

Comprehensive Evaluation

To run a complete evaluation of all models across all frameworks:

```
python test_all_models.py
```

This will: 1. Load datasets for all available frameworks 2. Train and evaluate all three models on each dataset 3. Perform statistical tests to compare model performance 4. Generate visualisations and a summary report 5. Save all results to the `comprehensive_results` directory

Understanding the Results

Comprehensive Results Directory

After running `test_all_models.py`, the `comprehensive_results` directory will contain:

1. **CSV Results Files**:
   - `tensorflow_results.csv`, `pytorch_results.csv`, etc.
   - Contains detailed metrics for each model/framework combination
2. **Visualisation Plots**:
   - Framework-specific plots showing precision, recall, F1 score, and training time
   - Cross-framework comparison plots showing performance across all frameworks
3. **Evaluation Summary**:
   - `evaluation_summary.md`: A markdown file with tables showing performance metrics - the question mark means +/-
   - Shows means and standard deviations for each metric
   - Includes both per-framework results and overall average performance

Interpreting the Metrics

- **Precision**: Proportion of performance bug predictions that are correct
- **Recall**: Proportion of actual performance bugs that are correctly identified
- **F1 Score**: Harmonic mean of precision and recall
- **Training Time**: Time taken to train the model (in seconds)
- **Prediction Time**: Time taken to make predictions (in seconds)

Visualisations

The generated plots help you understand model performance - these will be in your comprehesive_results directory.

1. **Per-Framework Plots**:
   - F1 Score comparison
   - Precision comparison
   - Recall comparison
   - Training time comparison
2. **Cross-Framework Plots**:
   - Show performance of each model across all frameworks
   - Helpful for identifying which model performs best overall

Example Use Cases

1. **Individual Framework Evaluation for the intermediate and hybrid model**:

   ```
   python test_intermediate_model.py --framework pytorch
   ```

   ```
   python test_hybrid_model.py --framework keras
   ```

2. **Complete Comprehensive Evaluation with full analysis and comparison with visualisations**:

   ```
   python test_all_models.py
   ```