Replication Guide

This document provides detailed instructions for replicating the results presented in the paper. Follow these steps exactly to ensure you can reproduce the evaluation results for the Enhanced Bug Report Classifier.

Step 1: Environment Setup

First, ensure you have the correct environment setup:

```
# Clone the repository
git clone <repository-url>
cd Tool-Building-Project-Task-1/final\ assignment

# Install dependencies
pip install -r requirements.txt

# Verify installation
python -c "import numpy, pandas, sklearn, matplotlib, seaborn, scipy, tqdm; print('All deper
```

Step 2: Dataset Preparation

The evaluation uses five datasets from different deep learning frameworks. Make sure they are properly placed:

```
# Create datasets directory if it doesn't exist
mkdir -p datasets

# Place datasets in the directory
# If you're using the original datasets from Lab 1:
cp /path/to/lab1/datasets/*.csv datasets/

# Verify dataset structure
ls datasets/
# Expected output: caffe.csv  incubator-mxnet.csv  keras.csv  pytorch.csv  tensorflow.csv
```

Step 3: Baseline Results

To compare against the baseline, you need the original baseline results:

```
# Create baseline_results directory if it doesn't exist
mkdir -p baseline_results

# Copy baseline results from Lab 1
cp /path/to/lab1/results/*.txt baseline_results/
cp /path/to/lab1/results/*.csv baseline_results/

# Verify baseline results
```

```
ls baseline_results/
# Expected output: Various result files from Lab 1
```

Step 4: Create Results Directory

Ensure the results directory exists:

```
mkdir -p src/results
```

Step 5: Run the Evaluation

Now you can run the full evaluation pipeline:

```
cd src
python main.py
```

This process will: 1. Load all datasets 2. Load baseline results 3. Train and evaluate the enhanced classifier on each dataset 4. Generate results files and visualizations

The execution may take a few minutes to complete depending on your system. Progress indicators will show the status.

Step 6: Verify Results

After the execution completes, verify that the results match what's reported:

```
# Check if result files were generated
ls results/
# Expected output: Various .txt, .json and .png files

# Examine the latest text results file
cat results/evaluation_results_*.txt | tail -n 15
```

You should see aggregate results showing improvements over the baseline for 4 out of 5 frameworks with an overall F1 score improvement of approximately 2.74%.

Step 7: View Visualizations

The visualizations are saved in the `results` directory:

```
# List visualization files
ls results/*.png
```

You should see four PNG files: - `f1_comparison.png` - `performance_change.png` - `precision_recall_comparison.png` - `summary_metrics.png`

These can be opened with any image viewer to visually verify the results.

Expected Results

If you've followed these steps correctly, you should observe:

1. The enhanced classifier beats the baseline in F1 score on 4 out of 5 frameworks
2. Overall F1 score improvement of approximately 2.74%
3. Most significant improvement on the Keras dataset (around 12.01%)
4. Slight decrease in performance on MXNet (around -5.81%)

The exact numbers may vary slightly due to: - Random initialization in classifier training - Train/test splits (although we use fixed random seeds) - System-specific floating-point computation differences

However, the overall trend and conclusion should remain the same.

Potential Issues and Resolutions

Inconsistent Dataset Format

If you're using datasets with different column names or formats, you'll need to modify the code in `main.py`:

```
# Locate the preprocessing section in main.py
data['Title'] = data['Title'].apply(preprocess_text)
data['Body'] = data['Body'].apply(preprocess_text)
data['Comments'] = data['Comments'].apply(preprocess_text)
```

Missing Baseline Results

If baseline results are missing, the code will use default values, but this may affect the comparison. Ensure proper baseline results are available or check the fallback values in `load_baseline_results()` in `main.py`.

Memory Issues

For large datasets, you might encounter memory errors. You can reduce memory usage by:

1. Reducing the `max_features` parameter in the TF-IDF vectorizer
2. Processing one dataset at a time by modifying the `main.py` file
3. Running on a system with more RAM

Running on a Specific Dataset

If you want to replicate results for a specific framework only, modify `main.py`:

```
# Locate this section in main.py
projects = ['tensorflow', 'pytorch', 'keras', 'incubator-mxnet', 'caffe']
```

```python
# Modify to include only the framework you want to test
projects = ['keras']  # For example, to run only on Keras
```

Then run the script as usual:

```
python main.py
```

Running Just the Visualization

If you already have result files and want to regenerate visualizations:

```
python visualization.py
```

This will read existing result files in the `results` directory and create new visualization plots.