Enhanced Bug Report Classifier - Replication Guide

This document provides detailed instructions on how to replicate the results of the Enhanced Bug Report Classifier project. Following these steps will ensure that you can reproduce the experiments and verify the improvements over the baseline.

Table of Contents

Environment Setup

Requirements

- Python 3.6 or higher
- 8+ GB RAM (16+ GB recommended, especially for word embeddings)
- Operating System: Windows, macOS, or Linux

Step 1: Clone the Repository

```
git clone
https://github.com/your-username/bug-report-classifier.git
cd bug-report-classifier
```

Step 2: Create a Virtual Environment (Recommended)

```
# For Windows
python -m venv venv
venv\Scripts\activate
# For macOS/Linux
python -m venv venv
source venv/bin/activate
```

Step 3: Install Dependencies

```
pip install -r requirements.txt
```

Step 4: Download NLTK Resources

```
python -c "import nltk; nltk.download('stopwords');
nltk.download('punkt'); nltk.download('wordnet')"
```

Step 5: Download Word Embeddings (Optional)

This step is optional but recommended to avoid delays during the first run:

```
python -c "import gensim.downloader as api;
api.load('glove-wiki-gigaword-100')"
```

Dataset Preparation

Option 1: Use Provided Datasets

The datasets should be placed in the `data/` directory with the following structure:

```
data/
  ··· pytorch.csv
  ··· tensorflow.csv
  ··· keras.csv
  ··· incubator-mxnet.csv
  ··· caffe.csv
```

Option 2: Download Datasets

If the datasets are not included, you can download them from the original source:

[Link to datasets repository]

After downloading, place them in the `data/` directory.

Expected Dataset Format

Each dataset should have the following columns:

- `Title`: The title of the bug report
- `Body`: The description of the bug report
- `class`: The binary label (0 or 1) indicating whether the bug is performance-related

Running the Baseline

Step 1: Navigate to the Lab 1 Directory

```
cd "Lab 1"
```

Step 2: Run the Baseline Classifier on All Datasets

```
# For PyTorch dataset
python br_classification.py
# For other datasets, modify the code to use a different dataset
# Open br_classification.py and change the 'project' variable to
one of:
# 'tensorflow', 'keras', 'incubator-mxnet', 'caffe'
```

Step 3: Collect Baseline Results

The baseline results will be saved in files named `[dataset]_NB.csv` in the parent directory. For example, `../pytorch_NB.csv`.

Running the Enhanced Classifier

Step 1: Navigate to the Final Assignment Directory

```
cd "../Final Assignment"
```

Step 2: Create Results Directory

```
mkdir -p results
```

Step 3: Run the Enhanced Classifier

To run with default settings (SVM, stemming, and word embeddings):

```
python run_experiments.py
```

This will:

1. Process all five datasets
2. Run 10 repeated experiments for each dataset
3. Calculate and save the average metrics
4. Compare with the baseline results

Alternative Configurations

You can experiment with different configurations:

```
# Use Random Forest instead of SVM
python run_experiments.py --classifier rf
# Disable word embeddings for faster processing
python run_experiments.py --embeddings false
# Use lemmatization instead of stemming
python run_experiments.py --stemming false --lemmatization
# Run on specific datasets only
python run_experiments.py --datasets pytorch tensorflow
# Increase TF-IDF features
python run_experiments.py --max-features 3000
```

Comparing Results

Automated Comparison

The `run_experiments.py` script automatically compares the enhanced classifier results with the baseline when the `--compare-baseline` flag is enabled (default).

Manual Comparison

You can also manually compare the results:

1. Open the baseline results file (e.g., `../pytorch_NB.csv`)
2. Open the enhanced classifier results file (e.g., `results/pytorch_SVM.csv`)
3. Compare the metrics (Accuracy, Precision, Recall, F1, AUC)

Experimental Validation

To ensure that the results are robust and not due to chance, we run each experiment multiple times with different random seeds. Here's how to validate the statistical significance of the improvements:

Statistical Testing

Run the following script to perform statistical testing on the results:

```
python statistical_test.py
```

This script:

1. Loads the baseline and enhanced results
2. Performs a paired t-test to determine if the improvements are statistically significant
3. Reports the p-values for each metric

Cross-Dataset Validation

To verify that the improvements are consistent across datasets:

```
python cross_validation.py
```

This script:

1. Performs k-fold cross-validation on each dataset
2. Reports the average performance across folds
3. Checks if the improvements are consistent across different partitions of the data

Expected Results

If everything is set up correctly, you should observe the following improvements over the baseline Naive Bayes model:

1. **Accuracy**: 2-5% improvement
2. **Precision**: 5-10% improvement
3. **Recall**: 10-20% improvement
4. **F1 Score**: 10-20% improvement
5. **AUC**: 3-8% improvement

The exact numbers may vary depending on the random seed, but the trends should be consistent.

Troubleshooting Replication Issues

If you encounter issues during replication:

1. **Inconsistent Results**:
    - Check if you're using the same random seeds (controlled by the `--repeat` argument)
    - Ensure you're using the correct versions of all dependencies
2. **Performance Issues**:
    - Try disabling word embeddings if processing is too slow
    - Reduce the number of TF-IDF features
    - Process one dataset at a time

3. **Memory Issues**:
    - Disable word embeddings to reduce memory usage
    - Process datasets in smaller chunks
    - Use a machine with more RAM
4. **Dataset Issues**:
    - Verify that the datasets have the correct format
    - Check for any preprocessing differences between the baseline and enhanced classifier

If you continue to experience issues, please open an issue in the repository with detailed information about the problem.