Important Note About Models

Please be aware that the trained models are **not** included in the GitHub repository due to their large file sizes. The `models/` directory will be created automatically when running the code, and you will need to train your own models by following this replication guide.

This means that the first time you run the experiment for each project, it will take longer as the model needs to be trained from scratch. Once trained, the model will be saved to the `models/` directory and can be loaded for future use.# Replication Guide for Bug Report Classification with RoBERTa

This document provides comprehensive instructions to replicate the experimental results of the "Fine-Tuning RoBERTa for Performance-Related Bug Report Classification" research. By following these steps carefully, you should be able to reproduce the results reported in the paper with minimal deviation.

Table of Contents

Expected Benchmark Results

Performance Comparison

When replicating our experiments, you should expect results similar to those in Table 2 of the paper:

Model	Accuracy	Precision	Recall	F1-Score	AUC
Naive Bayes + TF-IDF	0.5747	0.6082	0.7066	0.5240	0.7066
RoBERTa (Our Model)	0.9155	0.8210	0.8297	0.8196	0.9202
Improvement	+59.30%	+34.99%	+17.42%	+56.41%	+30.23%

Cross-Project Generalization

For cross-project testing (training on four frameworks and testing on one), expect F1-scores similar to Table 3:

Test Project	Naive Bayes + TF-IDF	RoBERTa (Ours)	Improvement
TensorFlow	0.5406	0.8916	+64.93%
PyTorch	0.5519	0.7860	+42.42%
Keras	0.5369	0.8449	+57.37%
MXNet	0.5479	0.8533	+55.74%
Caffe	0.4428	0.7221	+63.08%

Confusion Matrix

The confusion matrices generated should show similar patterns to those in Figures 2 and 3 of the paper, demonstrating RoBERTa's significant reduction in false positives compared to the Naive Bayes approach.

Environment Setup

Option 1: Google Colab (Recommended for Exact Replication)

1. Access the notebook in Google Colab:
   - Upload `ISE.ipynb` to Google Colab
2. Configure the runtime settings:
   - Click "Runtime" → "Change runtime type"
   - Set Hardware accelerator to "GPU"
   - Set Runtime shape to "High-RAM" if available
   - Click "Save"
3. Verify GPU availability (this should match the paper's setup of NVIDIA L4):

```
import torch
print(f"CUDA available: {torch.cuda.is_available()}")
print(f"GPU model: {torch.cuda.get_device_name(0) if torch.cuda.is_available() else 'None'}")
```

Option 2: Local Setup

Hardware Requirements

- NVIDIA GPU with at least 8GB VRAM
- 16GB+ RAM
- 10GB+ free disk space

Software Setup

1. Clone the repository:

```
git clone https://github.com/Alva1103/ISE-Coursework.git
cd ISE-Coursework
```

2. Create and activate a virtual environment:

```
# Create virtual environment
python -m venv ise_env

# Activate on Windows
ise_env\Scripts\activate

# Activate on Linux/Mac
source ise_env/bin/activate
```

3. Install dependencies with exact versions to ensure reproducibility:

```
pip install transformers==4.36.2 datasets==2.15.0 scikit-learn==1.2.2 pandas==1.5.3 numpy==1.24.0 matplotlib==3.7.2 seaborn==0.
```

4. Create the directory structure:

```
mkdir -p BugReportClassification/data
mkdir -p BugReportClassification/models
mkdir -p BugReportClassification/results
```

Data Preparation

Dataset Verification

The datasets are already included in the `data/` directory of the repository. You can verify the dataset structure and class distribution:

```
# For a single dataset
import pandas as pd

dataset_path = 'BugReportClassification/data/pytorch.csv'  # Change as needed
df = pd.read_csv(dataset_path)

# Verify columns
print(f"Columns: {df.columns.tolist()}")

# Verify class distribution
if 'class' in df.columns:
    class_dist = df['class'].value_counts(normalize=True) * 100
    print(f"Class distribution (%):\n{class_dist}")

    # Should show approximately 16.4% for class 1 (performance-related bugs)
    # This matches the class imbalance mentioned in the paper
```

Step-by-Step Replication Process

1. Notebook Setup

For Google Colab, execute the first two cells to:

- Mount Google Drive
- Set up directory structure
- Install dependencies

For local setup, ensure your environment is configured as described above.

2. Set Random Seeds

Execute the cell containing random seed settings to ensure reproducibility:

```
import random
import numpy as np
import torch

# Set fixed random seeds
torch.manual_seed(42)
random.seed(42)
np.random.seed(42)
```

3. Data Loading and Preprocessing

Execute the cells that:

- Define text preprocessing methods
- Define the data loading and preprocessing function
- Define the RoBERTa dataset class

4. Define the Training Function

Execute the cell that defines the `train_roberta_model` function, which includes:

- Model creation
- Training loop
- Evaluation
- Results collection

5. Single Project Experiment

To run the experiment on a single project:

```
# Choose a project to run the experiment on
project_name = 'pytorch'  # Change to the desired dataset

# Load and preprocess data
data = load_and_preprocess_data(project_name)
print(f"Data loaded and preprocessed. Shape: {data.shape}")

# Train RoBERTa model with 10 repeats
roberta_metrics = train_roberta_model(
    data=data,
    project_name=project_name,
    epochs=10,
    batch_size=16,
    repeat=10
)
```

6. Repeating for Different Projects

To replicate experiments for different projects, you'll need to repeat step 5 for each project individually by changing the project_name variable:

```
# For each project you want to test, update the project name and run the experiment
project_name = 'tensorflow'  # Change to the desired dataset: 'pytorch', 'caffe', 'tensorflow', 'keras', or 'incubator-mxnet'

# Load and preprocess data
data = load_and_preprocess_data(project_name)
print(f"Data loaded and preprocessed. Shape: {data.shape}")

# Train RoBERTa model with 10 repeats
roberta_metrics = train_roberta_model(
    data=data,
    project_name=project_name,
    epochs=10,
    batch_size=16,
    repeat=10
)
```

You'll need to manually track results for each project to create a comprehensive comparison table.

7. Execution Time Expectation

- On Google Colab with NVIDIA L4 GPU:
  - Single project: ~1-2 hours
  - All five projects: ~5-10 hours
- On local setup with NVIDIA GPU:
  - Times will vary depending on GPU specifications
  - Allow for 1.5-3x longer than Colab estimates

Verification of Results

Verification of Results

When verifying your results, compare them with those reported in Table 2 of the paper:

Model	Accuracy	Precision	Recall	F1-Score	AUC
Naive Bayes + TF-IDF	0.5747	0.6082	0.7066	0.5240	0.7066
RoBERTa (Our Model)	0.9155	0.8210	0.8297	0.8196	0.9202
Improvement	+59.30%	+34.99%	+17.42%	+56.41%	+30.23%

The tool will automatically output these metrics after the training process completes. Look for the output section titled "=== RoBERTa Model Results (Average across repeats) ===" which will display the average metrics across the 10 repeats.

Minor variations (±5%) are acceptable due to hardware differences, library version differences, and inherent randomness in neural network training.

For confusion matrices, the tool will automatically generate and display a confusion matrix after the final repeat. Compare this with Figures 2 and 3 in the paper to verify that your model shows a similar pattern of improvements over the baseline.

Cross-Project Testing

The cross-project testing results (Table 3) presented in the paper were based on a modified version of the code. The current codebase in ISE.ipynb does not directly implement cross-project testing functionality.

To compare with the cross-project results reported in the paper, you would need to:

1. Train models individually on each dataset
2. Record the metrics reported in the paper's Table 3
3. Compare your results with the expected performance

The reported cross-project F1-scores from the paper are:

Test Project	Naive Bayes + TF-IDF	RoBERTa (Ours)	Improvement
TensorFlow	0.5406	0.8916	+64.93%
PyTorch	0.5519	0.7860	+42.42%
Keras	0.5369	0.8449	+57.37%
MXNet	0.5479	0.8533	+55.74%
Caffe	0.4428	0.7221	+63.08%

Troubleshooting

Common Issues and Solutions

1. Out of Memory Errors

- **Symptoms**: CUDA out of memory, RuntimeError during training
- **Solutions**:
    - Reduce batch size in the final cell (try 8 or 4 instead of 16)
    - If using Google Colab, restart the runtime and select a high-memory instance
    - Close other applications using GPU memory

2. Inconsistent or Lower Results

- **Symptoms**: Metrics significantly below those reported in the paper
- **Solutions**:
    - Verify that random seeds are correctly set to 42
    - Check that the exact package versions match those specified
    - Ensure all preprocessing steps are executed

3. Dataset Loading Issues

- **Symptoms**: FileNotFoundError, KeyError, or unexpected dataset structure
- **Solutions**:
    - Verify the dataset paths
    - Check if CSV files have the correct column names
    - Ensure files were uploaded to the correct location

4. Google Colab Specific Issues

- **Symptoms**: Session disconnection, lost progress
- **Solutions**:
    - Save intermediate results to Google Drive
    - Keep the browser tab active to avoid timeouts
    - Check "Keep GPU alive" in "Settings" → "Miscellaneous"