Replication Instructions

Initial Setup

1. **Clone the Repository**

   ```
   # Using HTTPS
   git clone https://github.com/cg212/CG-ISE.git

   # OR using SSH
   git clone git@github.com:cg212/CG-ISE.git

   # Navigate to the project directory
   cd CG-ISE
   ```

2. **Set Up the Environment**

   ```
   # Install dependencies
   pip install -r requirements.txt

   # Download NLTK resources
   python download_nltk_resources.py
   ```

3. **Verify Dataset Availability**

   Ensure the following datasets are in the `lab1/datasets` directory:

   - `tensorflow.csv`
   - `pytorch.csv`
   - `keras.csv`
   - `caffe.csv`

Replication Scenarios

Scenario 1: Quick Replication of Main Results

This is the fastest way to replicate the key findings using a subset of the data:

```
# Navigate to the lab1 directory
cd lab1

# Run the comprehensive evaluation with smaller samples
python test_all_models.py
```

This will: - Sample a portion of each dataset (controlled by `SAMPLE_RATIO` in the script) - Run all three models across all available frameworks - Generate all visualisations - Create a summary report in evaluation_summary.md

Expected completion time: ~5-10 minutes depending on your hardware

Scenario 2: Full Replication of Individual Framework Results

For detailed results on specific frameworks:

```
# Make sure you're in the lab1 directory
cd lab1

# Test all models on TensorFlow dataset
python test_hybrid_model.py --framework tensorflow
```

```bash
# Test all models on PyTorch dataset
python test_hybrid_model.py --framework pytorch

# Test all models on Keras dataset
python test_hybrid_model.py --framework keras

# Test all models on Caffe dataset
python test_hybrid_model.py --framework caffe
```

Each command will: - Train the baseline and hybrid models on the specified framework - Report precision, recall, F1 score, and timing metrics - Print a classification report and improvement percentages

Expected completion time: ~2-3 minutes per framework

Scenario 3: Comprehensive Evaluation with Custom Parameters

For customised testing:

```bash
# Make sure you're in the lab1 directory
cd lab1

# Test with custom parameters
python test_hybrid_model.py --framework tensorflow --test_size 0.25 --random_state 123 --
        sample_size 400
```

Parameters you can adjust: - `--framework`: The framework dataset to use - `--test_size`: Proportion of data to use for testing (default: 0.3) - `--random_state`: Random seed for reproducibility (default: 42) - `--sample_size`: Number of samples to use (default: None = all samples) - `--fast`: Enable fast mode with simplified features (default: False)

Scenario 4: Comparison of Intermediate Model

To specifically evaluate the intermediate model against the baseline:

```bash
# Make sure you're in the lab1 directory
cd lab1

python test_intermediate_model.py --framework tensorflow
```

Examining Results

1. CSV Result Files

After running `test_all_models.py`, you can view the CSV files in the `lab1/comprehensive_results` directory by running:

```bash
cat lab1/comprehensive_results/tensorflow_results.csv
cat lab1/comprehensive_results/pytorch_results.csv
# etc.
```

These files contain detailed metrics for each model run, including: - Precision, recall, and F1 score - Training and prediction times - Framework and model identifiers - Run number (for statistical aggregation)

2. Evaluation Summary Report

The `evaluation_summary.md` file contains tables summarising the performance of each model:

```bash
cat lab1/comprehensive_results/evaluation_summary.md
```

The report is structured as follows: - Section for each framework with performance metrics - Standard deviations to indicate result stability - Overall model performance across all frameworks - Training and prediction time comparisons

3. Visualisation Plots

The `lab1/comprehensive_results/plots` directory contains several visualisation types:

1. **Per-Framework Metric Plots**:
    - `[framework]_f1_score.png`: F1 score comparison for each model
    - `[framework]_precision.png`: Precision comparison
    - `[framework]_recall.png`: Recall comparison
    - `[framework]_training_time.png`: Training time comparison (log scale)
2. **Cross-Framework Comparison Plots**:
    - `all_frameworks_f1_comparison.png`: F1 scores across all frameworks
    - `all_frameworks_precision_comparison.png`: Precision across frameworks
    - `all_frameworks_recall_comparison.png`: Recall across frameworks

How to interpret the plots: - Bar height represents the mean metric value - Higher bars for precision, recall, and F1 score indicate better performance - The Hybrid Model (main tool) should generally show taller bars than the Baseline - For training time plots (log scale), shorter bars indicate faster training

Verifying Specific Results

Key Result 1: Hybrid Model Outperforms Baseline

To verify that my Hybrid Model outperforms the Baseline in terms of F1 score:

1. Run the comprehensive evaluation:

    ```
    python test_all_models.py
    ```

2. Check the F1 score comparison in `lab1/comprehensive_results/evaluation_summary.md`

    - The Hybrid Model should show higher F1 scores than the Baseline across most frameworks
    - Look for the "Overall Model Performance" section to see average improvement

3. Examine the visualisation:

    ```
    lab1/comprehensive_results/plots/all_frameworks_f1_comparison.png
    ```

    - The Hybrid Model bars should be taller than the Baseline bars for most frameworks

Key Result 2: Statistical Significance

During execution of `test_all_models.py`, my script performs Mann-Whitney U tests to determine if the improvements are statistically significant:

- Look for output lines like:

    ```
    Mann-Whitney U test: Baseline vs HybridModel
    U statistic: [value]
    P-value: [value]
    Effect size r: [value]
    Significant difference: Yes/No
    ```

- A p-value less than 0.05 indicates a statistically significant difference

- The effect size r indicates the magnitude of the difference (larger is better)

Key Result 3: Precision and Recall Trade-offs

To verify the precision/recall characteristics:

1. Check `lab1/comprehensive_results/plots/all_frameworks_precision_comparison.png` and `lab1/comprehensive_results/plots/all_frameworks_recall_comparison.png`

2. The Hybrid Model generally shows better precision than the Baseline

    - This demonstrates its ability to reduce false positives

3. The Intermediate Model may show higher recall but lower precision

    - This highlights the trade-offs between different approaches

Troubleshooting Replication Issues

If you encounter issues during replication:

1. **NLTK Resource Issues**:
    - Run `python download_nltk_resources.py` again to ensure all resources are downloaded
    - Check for internet connectivity issues
2. **Memory Errors**:
    - Reduce the sample size: `python test_hybrid_model.py --framework tensorflow --sample_size 200`
    - Close other memory-intensive applications
3. **Unexpected Results**:
    - Check if you're using the correct framework name
    - Verify that the datasets are properly loaded
    - Try with a different random seed: `--random_state 456`
4. **Visualisation Errors**:
    - Ensure matplotlib and seaborn are properly installed
    - Try upgrading: `pip install --upgrade matplotlib seaborn`

Expected Results

When successfully replicated, you should observe:

1. The Hybrid Model consistently outperforms the Baseline model in terms of F1 score across most frameworks

2. The Hybrid Model shows higher precision than the Baseline model

3. The Intermediate Model may show higher recall in some cases but generally lower precision and F1 score

4. Training times are highest for the Hybrid Model, but the performance improvement justifies the additional computational cost

5. Statistical tests should confirm significant differences between the models