Intelligent Bug Report Classification: A Hybrid Approach Combining Tf-IDF, GloVe, and Meta Features

Replication

1. Ensure all dependencies listed in *requirements.pdf* are installed.
2. Make sure glove.6B.100d.txt is in the same directory as the python scripts (download it if you don't have it by following *requirements.pdf*).
3. Make sure all the CSV datasets are in the same directory as the python scripts (it should already be).

Follow the steps in *manual.pdf* to evaluate datasets for the baseline / hybrid, and save the results to a CSV file.

A brief recap is as follows:

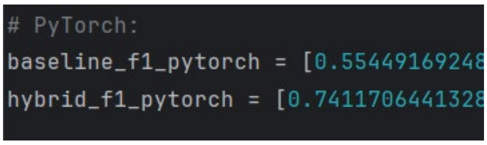
1. Open a terminal
2. Run *python improved_br_classification.py*
3. Select dataset from GUI
4. Click Run Evaluation
5. Wait for results to appear (20-60sec)
6. A results CSV file will be saved as *results_[dataset]_HYBRID.CSV*
7. Each CSV includes:
8. Accuracy, Precision, Recall, F1, AUC (averaged over 30 runs)
9. List of 30 individual AUC scores
10. List of 30 individual F1 scores
11. Time taken

Once this has been done for every dataset with both tools (baseline and hybrid), we have all the data necessary to reproduce Figure 3 from the report.

Reproducing Statistical Tests (Wilcoxon Signed-Rank)

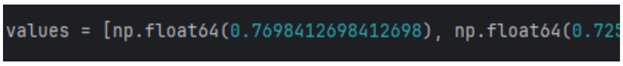
1. Run both the baseline and hybrid tools for all datasets and save the CSV files generated
2. Open the provided python code file *wilcoxon_test.py*
3. Open the file and copy the 30-run F1 (under CV_list(F1)) scores from:
    a. Results_[dataset]_HYBRID.csv (hybrid)
    b. Results_[dataset]_NB.csv (baseline)

4. Paste the results in their respective place in *wilcoxon_test.py.* E.g, for pytorch [baseline/hybrid]_f1_[dataset]::



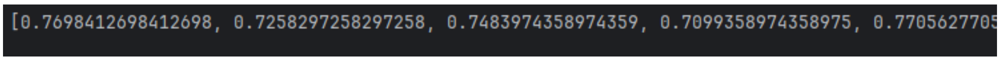
```
# PyTorch:
baseline_f1_pytorch = [0.5544916924
hybrid_f1_pytorch = [0.7411706441328
```

5. Then go back into the same CSV files and copy the 30-run AUC (under CV_list(AUC)) scores.
6. Open *wilcoxon_cleaner.py* and paste in the one of the results (either from baseline or hybrid CSV) here:



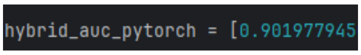
```
values = [np.float64(0.7698412698412698), np.float64(0.72
```

7. Now run the file with *python wilcoxon_cleaner.py,* to get the terminal result:



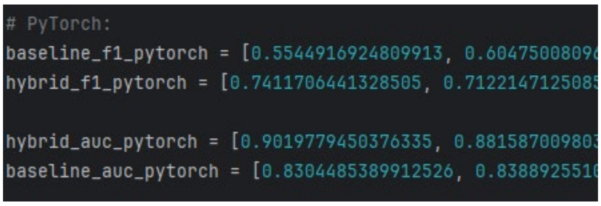
```
[0.7698412698412698, 0.7258297258297258, 0.7483974358974359, 0.7099358974358975, 0.770562770
```

8. Copy the entire output and open *wilcoxon_test.py* to paste in the result in its respective place [baseline/hybrid]_auc_[dataset]:



```
hybrid_auc_pytorch = [0.901977945
```

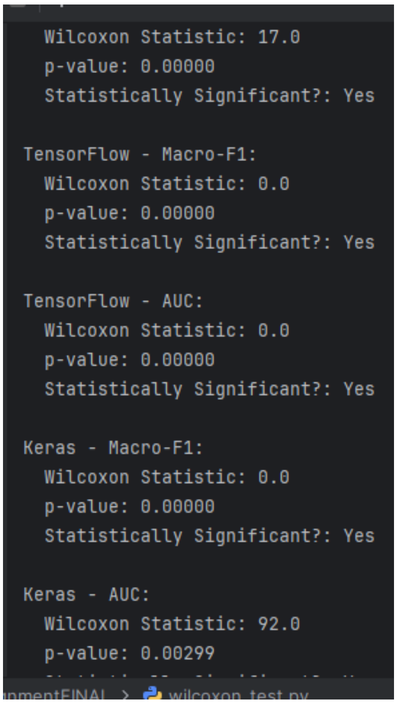
9. Repeat this process from step 5. For the other tool you haven't done yet (either baseline or hybrid) and paste the result accordingly in *wilcoxon_test.py*
10. You should now have the data stored for one dataset:



```
# PyTorch:
baseline_f1_pytorch = [0.5544916924809913, 0.6047500809
hybrid_f1_pytorch = [0.7411706441328505, 0.712214712508

hybrid_auc_pytorch = [0.9019779450376335, 0.88158700980
baseline_auc_pytorch = [0.8304485389912526, 0.838892551
```

11. Repeat the whole process again from step 3. For each dataset till you have stored data in *wilcoxon_test.py* for all of them.
12. Now run *python wilcoxon_test.py.*
13. You should see:
    a. Wilcoxon statistic
    b. p-value
    c. Significance flag



```
  Wilcoxon Statistic: 17.0
  p-value: 0.00000
  Statistically Significant?: Yes

TensorFlow - Macro-F1:
  Wilcoxon Statistic: 0.0
  p-value: 0.00000
  Statistically Significant?: Yes

TensorFlow - AUC:
  Wilcoxon Statistic: 0.0
  p-value: 0.00000
  Statistically Significant?: Yes

Keras - Macro-F1:
  Wilcoxon Statistic: 0.0
  p-value: 0.00000
  Statistically Significant?: Yes

Keras - AUC:
  Wilcoxon Statistic: 92.0
  p-value: 0.00299
```

14. This reproduces the p-values shown in Figure 4 of the report.