Replication guide for Bug Report Classifier

UNIVERSITY OF BIRMINGHAM

Elisa Ciocarlan
School of Computer Science

22nd March, 2025

Introduction

The Bug Report Classifier project aims to classify bug reports as to whether it is performance bug-related or not. In this project, we propose an approach using Support Vector Machine (SVM) paired with Term Frequency-Inverse Document Frequency (TF-IDF). According to the results displayed in the report of this project, the proposed approach beats the baseline tool that uses Naives-Bayes paired with TF-IDF. This guide provides detailed instructions to replicate the project's results.

Prerequisites

To run the project's code, Python 3.6+ is required along with the libraries listed in the **requirements.txt** file.

Datasets

The performance of the classifier is measured across datasets of different projects.

Datasets description

In this project, we use five datasets that consist of labelled bug reports across five projects : pytorch, tensorflow, keras, incubator-mxnet and caffe. Each report has the following field:

- Repository: This refers to the name of the repository where the bug report was posted.
- Number: This represents the unique identification number of the specific bug report or issue within the repository.
- State: This indicates the current status or state of the issue.
- Title: This is the title of the bug.
- Body: This column contains the detailed description of the issue, including steps to reproduce, error messages, and any additional context provided by the reporter.
- Labels: These are tags or categories that help classify the bug report based on its nature.
- Comments: This contains any comments made by users or maintainers related to the issue.
- Codes: This includes code snippets, error logs, or relevant technical details provided in the bug report.
- Commands: This could represent specific shell commands related to the bug report.
- Class: This column is used to categorize the bug report, indicating whether the bug is performance-related or not.

Datasets access

The datasets are stored in the directory named 'datasets'. Each CSV file corresponds to a project dataset.

Codebase Overview

File structure

/ISE_project
── datasets/
— ── caffe.csv
— ── incubator-mxnet.csv
— ── keras.csv
— ── pytorch.csv
— ── tensorflow.csv
── examples/
— ── Eclipse_bug_report.txt
— ── Firefox_bug_report.txt
── models/
— ── classifier.pkl
— ── vectorizer.pkl
── results/
— ── experiments.csv
— ── training_results.csv
── src/
— ── baseline.py
— ── bug_report_classifier.py
— ── svm.py
— ── train_model.py
── manual.pdf
── replication.pdf
── requirements.pdf
── requirements.txt

Key files to replicate the report's results

- 'baseline.py': this file contains the code of the baseline approach which uses TF-IDF and Naive-Bayes.
- 'svm.py': this file contains the code of the proposed approach which uses TF-IDF and SVM.

Instructions for replication

To replicate the results that are presented in the first table of the report, both 'baseline.py' and 'svm.py' files need to be run five times, one time for each project. Each time a file is run, the performance on different metrics such as accuracy, precision, recall, and F1-score is measured and stored in a csv file in the 'results' directory.

Instructions for running the baseline code

Step 1 : Choose the project dataset

The project dataset on which the classifier is trained can be selected by changing the value of *project* at **line 76** in the 'baseline.py' file :

```
# Choose the project ('pytorch', 'tensorflow', 'keras', 'incubator-mxnet', 'caffe')
project = 'pytorch'
```

Step 2 : Run the 'baseline.py' file with the following command

```
$ python src/baseline.py
```

At the end of the code execution, the computed results will be stored in the 'results/results.csv' file.

Instructions for running the code of the proposed approach

Step 1 : Choose the project dataset

The project dataset on which the classifier is trained can be selected by changing the value of *project* at **line 90** in the 'svm.py' file :

```
# Choose the project ('pytorch', 'tensorflow', 'keras', 'incubator-mxnet', 'caffe')
project = 'pytorch'
```

Step 2 : Run the 'svm.py' file with the following command

```
$ python src/svm.py
```

At the end of the code execution, the computed results will be stored in the 'results/results.csv' file.

Validation

After running each code on the different projects, the results file should contain the same values as in the table 1. The proposed classifier should achieve approximately 89% in accuracy while the baseline tool should only achieve approximately 58%.

Method	Project	repeated_times	Accuracy	Precision	Recall	F1
Baseline	pytorch	30	0.6406181015452538	0.613395432339462	0.7572401012890487	0.566318880205258
Baseline	tensorflow	30	0.5551454138702461	0.6348912520794088	0.710618869648387	0.5361699639722113
Baseline	keras	30	0.5634328358208955	0.630227718731005	0.6930823640375823	0.5446052206726558
Baseline	incubator-mxnet	30	0.5993589743589745	0.6075582518063971	0.7463047125970756	0.5363960690371622
Baseline	caffe	30	0.5385057471264367	0.564401890242322	0.6538785536994604	0.4629714352393971
SVM + TF-IDF	pytorch	30	0.8920529801324503	0.7596026217503605	0.7200594507147037	0.7320039360100773
SVM + TF-IDF	tensorflow	30	0.8951901565995526	0.8538044063678863	0.7822839031523061	0.8085303545637849
SVM + TF-IDF	keras	30	0.8718905472636815	0.8115735892725344	0.7754900223929456	0.7878038664164976
SVM + TF-IDF	incubator-mxnet	30	0.8926282051282051	0.7653269309314749	0.6974448593608205	0.7202435906567367
SVM + TF-IDF	caffe	30	0.910919540229885	0.7865847326104832	0.6654964516461466	0.6932068186494841

Table 1: Expected performance results