School of Computer Science

UNIVERSITY OF BIRMINGHAM

Intelligent Software Engineering
Lab1: Bug Report Classification

Dr. Tao Chen

This is lab1 for the module, which can be chosen as the problem for the coursework. If you need specific support in terms of programming, please attend one of the lab sessions and ask the TAs for help.

The Content

For Lab1, you will build a **baseline tool** using Naive Bayes paired with TF-IDF ($\mathcal{M}$) to automatically classify a bug report as to whether it is performance bug-related or not (e.g., if the report is about accuracy/inference speed, or anything else):

$$\mathbf{c} = \mathcal{M}(\mathbf{r}) \tag{1}$$

whereby $\mathbf{r}$ is the parsed representation of the texts from a report; $\mathbf{c}$ is the classification decision of whether it is performance bug related, which can be either 0 or 1. To resolve the above, the baseline will use TF-IDF to encode the texts into numeric representation while training/using Naive Bayes as the classifier to process those numeric representations.

You are welcome to research the internal working mechanism of Naive Bayes and TF-IDF, but some existing implementations are available from the well-known library: `https://scikit-learn.org/stable/modules/naive_bayes.html`; `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`.

The Datasets of Projects

The datasets cover 5 DL projects with a large number of reports. Table 1 summarises the collected total of 3,712 GitHub reports for the lab.

Table 1: Characteristics of the GitHub reports. ($\mathcal{P}$ and $\mathcal{N}$ denote the number of performance bug related reports and those that are not performance bug-related, respectively; $\mathcal{P}$% is the percentage of performance bug related reports.)

Project	$\mathcal{P}$	$\mathcal{N}$	**Total**	$\mathcal{P}$%
`TensorFlow`	279	1211	1490	18.7%
`PyTorch`	95	657	752	12.6%
`Keras`	135	533	668	20.2%
`MXNet`	65	451	516	12.6%
`Caffe`	33	253	286	11.5%
TOTAL	607	3105	3712	16.4%

The link and details of the datasets can be accessed here: `https://github.com/ideas-labo/ISE/tree/main/lab1`.

How to use the Datasets?

The normal procedure is that, for each project, you would randomly select 70% report to train the baseline and use the remaining 30% for testing (make sure that, when training and testing, you include a good number of positive samples). The above process is repeated, e.g., 30 times, to avoid stochastic bias. Note that in each repeat, the training/testing sample of reports is randomly selected for all approaches you wish to compare, if any. You can then examine the results against the metrics below.

The Metrics

Commonly, for binary classification problems, we can use three different metrics:

- **Percision** is the ability of a classification model to identify only the relevant data points:

$$\frac{\#\text{ture positive samples}}{\#\text{ture positive samples} + \#\text{false positive samples}} \tag{2}$$

- **Recall** is the ability of a classification model to find all the relevant data points:

$$\frac{\#\text{ture positive samples}}{\#\text{ture positive samples} + \#\text{false negative samples}} \tag{3}$$

- **F1 Score** is simply the combination of the above:

$$2 \times \frac{\text{recall} \times \text{percision}}{\text{recall} + \text{percision}} \tag{4}$$

Some existing implementations of the metrics can be found here:

- `https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.precision_score.html#sklearn.metrics.precision_score`
- `https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.recall_score.html#sklearn.metrics.recall_score`
- `https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.f1_score.html`

The Statistical Test

If there is nothing to compare, then you might not need the statistical test. The link to an existing statistical test library can be found here: `https://docs.scipy.org/doc/scipy/reference/stats.html`. You will find implementations of the tests mentioned in the module and those that we have not talked about.