

ISE Tool Building Project: Manual

Setup

To run the tool, firstly clone the [GitHub repository](https://github.com/MXP356/ISE-Tool-Building-Project) by running the following commands:

```
git clone https://github.com/MXP356/ISE-Tool-Building-Project.git
cd ISE-Tool-Building-Project
```

Dependencies

Ensure that you have all the dependencies installed by running:

```
pip install pandas numpy torch scikit-learn torch transformers nltk tk
```

Config

On lines 2-19 of main.py, you can modify different configurations of the tool. The variables which you can configure are as follows:

Variable	Possible Values	Description
project	'pytorch' 'tensorflow' 'keras' 'incubator-mxnet' 'caffe'	Specifies which dataset of bug reports to use for training & validation
REPEAT_TIMES	[integer array]	Specifies the repeated_times for which we would like to see average evaluation metrics for
IncludeCommentsFromBugReports	True False	Toggles the inclusion of different fields from the dataset
IncludeCodeSnippetsAndErrorLogsFromBugReports	True False	
IncludeLabelsFromBugReports	True False	
Method	'TFIDFNaiveBayes' 'BERTLogisticRegression'	Whether to use the TF-IDF + NB baseline or BERT + LR model
RemoveHTMLTags	True False	Toggles different text pre-processing techniques
RemoveEmoji	True False	
RemoveStopWords	True False	
CleanString	True False	
UseGridSearchCVForBERTLR	True False	Whether to use GridSearchCV on BERT + LR
USE_LAB1_BASELINE_CONFIG	True False	Resets all configs to restore the configuration of the TF-IDF baseline model

Running the Tool

Run the main.py file by running the command:

```
python main.py
```

The status of the tool's execution will be printed in the terminal, and then a CSV file containing evaluation metrics of the model will be saved into the /outputs folder.





















CSV Output

The CSV output contains 12 columns, including the average Accuracy, Precision, Recall, F1 and AUC metrics of the model at a different number of repeated counts, as defined in the REPEAT_TIMES config.

There are also CV_list columns for each metric, which contain the individual metrics calculated at each interval. These can be used to perform statistical tests between two models.

	A	B	C	D	E	F	G	H	I	J	K	L
1	experiment	repeated_times	Accuracy	Precision	Recall	F1	AUC	CV_list(AU	CV_list(Ac	CV_list(Pri	CV_list(Re	CV_list(F1)
2	lab1/outputs/pytorch_NB.csv	5	0.62781457	0.622343253	0.746022849	0.571306065	0.746022849	[np.float64	[0.635761	[0.613468	[0.769214	[0.5619032
3	lab1/outputs/pytorch_NB.csv	10	0.625827815	0.609021933	0.747575464	0.555144187	0.747575464	[np.float64	[0.635761	[0.613468	[0.769214	[0.5619032
4	lab1/outputs/pytorch_NB.csv	30	0.640838852	0.6137774	0.758098166	0.5665952	0.758098166	[np.float64	[0.635761	[0.613468	[0.769214	[0.5619032

The name of the output CSV file is based on the configuration values which you have set.

-  caffe_BERT-LR
-  caffe_NB
-  incubator-mxnet_BERT-LR
-  incubator-mxnet_NB
-  keras_BERT-LR
-  keras_NB
-  pytorch_BERT-LR
-  pytorch_BERT-LR_CodeSnippetsAndError...
-  pytorch_BERT-LR_CommentsIncluded
-  pytorch_BERT-LR_KeepStopwords
-  pytorch_BERT-LR_LabelsIncluded
-  pytorch_NB
-  pytorch_NB_CodeSnippetsAndErrorLogs...
-  pytorch_NB_CodeSnippetsAndErrorLogs...
-  pytorch_NB_CommentsIncluded
-  pytorch_NB_CommentsIncluded_KeepSt...
-  pytorch_NB_KeepStopwords
-  pytorch_NB_LabelsIncluded
-  tensorflow_BERT-LR
-  tensorflow_NB

Performing Wilcoxon Signed-Rank Statistical Tests

Open the CSV output file for the two models you'd like to compare. Copy and paste the CV_list metrics for each model into their corresponding variable within the statisticalTest.py file, save the file, and then run the command:

```
python statisticalTest.py
```

The result for each metric will be output into the terminal, indicating whether there has been a statistically significant change or not:

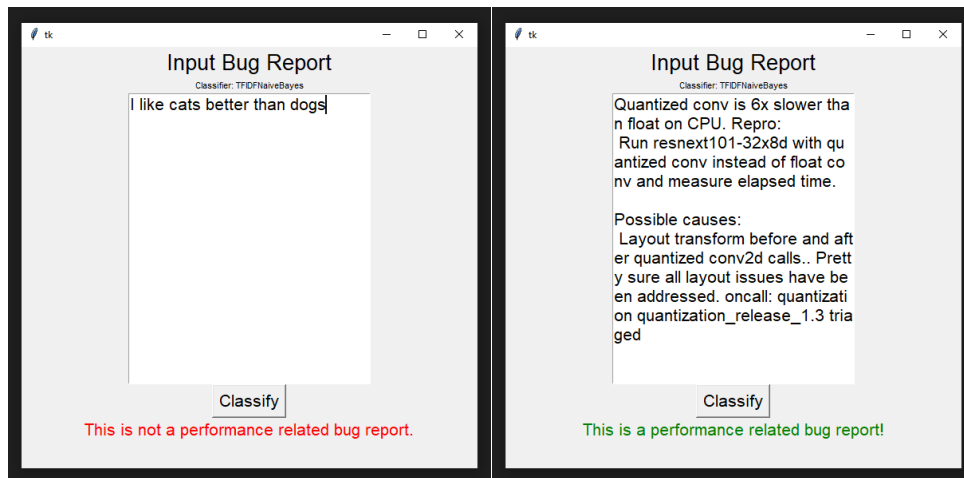
```

Test Results:
Test Statistic: 23.0000
P-value: 0.6953
● No statistically significant difference in Auc ( $p \geq 0.05$ )
The difference could be due to random chance.
Test Statistic: 0.0000
P-value: 0.0020
● The difference in Accuracy is statistically significant ( $p < 0.05$ )
The new model significantly outperforms the Baseline
Test Statistic: 0.0000
P-value: 0.0020
● The difference in Precision is statistically significant ( $p < 0.05$ )
The new model significantly outperforms the Baseline
Test Statistic: 23.0000
P-value: 0.6953
● No statistically significant difference in Recall ( $p \geq 0.05$ )
The difference could be due to random chance.
Test Statistic: 0.0000
P-value: 0.0020
● The difference in F1 is statistically significant ( $p < 0.05$ )
The new model significantly outperforms the Baseline

```

Using the UI

At the end of main.py, a Tkinter UI is launched which allows you to classify new bug reports on the model which you have just configured, trained and tested.



Simply input a bug report into the text field, and then click on the Classify button to view the class label which has been predicted by the model.

Deleting Cached Word Embeddings

If you would like to delete the cache of generated BERT word embeddings, you can do so by deleting the file: storage/cached_bert_embeddings.pkl

This will require the tool to regenerate word embeddings all over again when it is next re-run, which is not recommended as this process takes some time. Only do this if you would like to observe the tool generating new word embeddings: you can view the progress of this in the output when run.