This is the accepted version of a paper presented at 2nd IEEE/ACM International Workshop on Natural Language-Based Software Engineering, NLBSE 2023, Melbourne, 20 May 2023.

Citation for the original published paper:

Laiq, M. (2023)
An Intelligent Tool for Classifying Issue Reports
In: Proceedings - 2023 IEEE/ACM 2nd International Workshop on Natural Language-Based Software Engineering, NLBSE 2023 (pp. 13-15). Institute of Electrical and Electronics Engineers (IEEE)
https://doi.org/10.1109/NLBSE59153.2023.00010

N.B. When citing this work, cite the original published paper.

©2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:bth-25325
An Intelligent Tool for Classifying Issue Reports

Muhammad Laiq
Blekinge Institute of Technology
muhammad.laiq@bth.se

Abstract—A considerable amount of issue reports are submitted daily in large-scale software development. Manually reviewing and classifying each issue report is challenging and error-prone. Thus, to assist practitioners, in this paper, we propose and evaluate an automatic supervised machine learning-based approach that can automatically predict the newly submitted issue report type (i.e., bug, feature, question, or documentation). We applied the supervised machine learning-based approach to over 1.4 million issue reports data from real open-source projects. We performed our experiments using Stochastic Gradient Descent (SGD)-based classifier and achieved an F1 micro average score of 0.8523.

Index Terms—Software Analytics, Software Maintenance, Issue Classification, Bug Reports, Natural Language Processing

I. INTRODUCTION

In large-scale software development, a large number of issue reports are submitted daily. Thus, manually reviewing and classifying each issue report is challenging, tedious, and error-prone. The approach that can automatically identify the type of newly submitted issue report will assist practitioners in reducing the effort spent on the manual classification, which will further help in issue assignment, prioritization, and resolution.

In literature, a number of studies have investigated the use of automatic approaches [1]–[11] to assist practitioners in identifying the type of newly submitted issue reports. In this paper, we classify issue reports using a classical supervised machine learning classifier, i.e., a Stochastic Gradient Descent (SGD) based classifier. SGD is a simple yet efficient approach, which is better suited for text classification and natural language processing tasks [12].

Using SGD, we achieve comparable performance (i.e., an F1 micro average score of 0.8523) to previously BERT (Bidirectional Encoder Representations from Transformers) based approaches [13].

The detail of our tool can be found on the tool’s page 1. In the following sections, we describe the dataset, the tool’s architecture, pre-processing steps, implementation details, and the results of our study.

II. ISSUE CLASSIFICATION WITH SGD

In this section, we describe the data set used for issue reports classification, the architecture of our approach, the steps used in pre-processing, and the values of the performance evaluation metrics.

A. Data Set

We use issue reports data made available for ‘The NLBSE’23 Tool Competition’ [14], i.e., see Table I. Each issue report contains the following fields:

- **id**: This field contains a unique issue report identifier.
- **labels**: This field contains the labels for each report that indicates the type of the issue report, i.e., bug, feature, question, or documentation.
- **title**: This field contains a brief description of a submitted issue report.
- **body**: This field contains a detailed description of the issue. For instance, for the type bug, it may contain a detailed description of the problem that happened on the reporter side with steps to reproduce or code where an error occurred.
- **author_association**: This field shows the association of an issue submitter, such as member, contributor, collaborator, or owner.

<table>
<thead>
<tr>
<th>Set</th>
<th>Bug</th>
<th>Feature</th>
<th>Question</th>
<th>Documentation</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>670,951 (52.6%)</td>
<td>472,216 (37%)</td>
<td>76,048 (6%)</td>
<td>56,666 (4.4%)</td>
<td>1,275,881</td>
</tr>
<tr>
<td>Test Set</td>
<td>74,781 (52.5%)</td>
<td>52,707 (37.1%)</td>
<td>8,490 (6%)</td>
<td>6,252 (4.4%)</td>
<td>142,320</td>
</tr>
</tbody>
</table>

B. Our Approach

Figure 1 presents the architecture of our approach. At first, we train our model using past issue reports. Then in the second step, we use the trained model to predict the type of a newly submitted issue report.

*Stochastic Gradient Descent*: This study used a classical Stochastic Gradient Descent (SGD) based classifier. SGD is simple yet efficient and has been utilized in large-scale and sparse machine-learning problems usually encountered in natural language processing and text classification problems [12]. SGD is an efficient technique for fitting linear models under convex loss functions such as Logistic Regression and linear Support Vector Machines (SVM). For instance, \( SGDClassifier(loss='hinge', penalty='l2') \) results in an SVM model fitted via SGD. We implemented an SGDClassifier using the hinge loss function (i.e., an equivalent of linear SVM). SGD comprises various parameters that can be tuned.

to improve classification performance, such as alpha, penalty, and max iterations; see [12] for more details.

C. Data Pre-processing

We apply the following standard pre-processing steps mainly using Gensim library [15]:

- At first, we merge the title and body fields of each issue report.
- In the second step, we remove special characters, HTML tags, punctuations, numbers, consecutive whitespaces, and stopwords.
- Then, we make stems of words and convert labels of each issue report to numbers, i.e., bug:0, feature:1, questions:2, and documentation:3.
- Finally, we apply Tfidf [16] on our cleaned data and generate sparse matrices for both the training and testing data sets.

D. Performance Evaluation Metrics

To evaluate the performance of the issue classification tool, we measure micro average Precision (P), Recall (R), and F1 using the sklearn library [17]

\[
R = \frac{TP}{TP + FN} \quad (1)
\]

\[
P = \frac{TP}{TP + FP} \quad (2)
\]

\[
F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3)
\]

\[
mP = precision\_score(y\_true, y\_pred, average = \text{\textquoteleft}micro\textquoteright) \quad (4)
\]

\[
mR = recall\_score(y\_true, y\_pred, average = \text{\textquoteleft}micro\textquoteright) \quad (5)
\]

\[
mF1 = f1\_score(y\_true, y\_pred, average = \text{\textquoteleft}micro\textquoteright) \quad (6)
\]

E. Implementation Details

We used the Scikit-learn [18] machine learning library for implementing our approach. The classifier was trained with following settings: SGDClassifier(loss=’hinge’, penalty=’l2’, alpha=0.000001, random_state=42, max_iter=20, tol=0.001). The tuning was performed by adjusting the max_iter (number of passes over the training data, aka epochs), alpha (value to control the regularization), and tol (stopping criteria) parameters. All other parameters were set to default values.

We performed our experiments on a machine equipped with Windows 10, 64-bit as the operating system, Intel (R) Core (TM) i7, and 16GB RAM. It takes approximately 2.53 minutes to train the classifier, excluding the data pre-processing time of approximately 25 minutes.
Table II presents our results and their comparison to the baseline approaches. The results of our approach are comparable to previous BERT-based complex approaches [13]. We achieve an F1 micro average score of 0.8523 using a simple and classical machine learning approach based on SGD, and our approach outperforms FastText.

We also tried RidgeClassifier and achieved almost identical results, i.e., micro-precision: 0.8486, micro-recall: 0.8486 and micro-F1: 0.8486. The settings for both classifiers can be found on our tool page2.

Based on our results, we can see that the performance of the tool for minority classes (i.e., documentation and question) is not that good when compared to majority classes (i.e., bug and feature). Therefore, we suggest, when evaluating the performance of classification tools applied on an imbalanced dataset, these numbers (i.e., micro average results) should be considered with caution, and other performance evaluation metrics, such as the Mathew correlation coefficient (MCC) [19], should be utilized together with micro averages.

<table>
<thead>
<tr>
<th>Model</th>
<th>Metrics</th>
<th>Bug</th>
<th>Feature</th>
<th>Question</th>
<th>Documentation</th>
<th>Micro Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastText</td>
<td>Precision</td>
<td>0.8771</td>
<td>0.8415</td>
<td>0.6702</td>
<td>0.7363</td>
<td>0.8510</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>0.9173</td>
<td>0.8621</td>
<td>0.4555</td>
<td>0.5011</td>
<td>0.8510</td>
</tr>
<tr>
<td></td>
<td>F1</td>
<td>0.8967</td>
<td>0.8517</td>
<td>0.5424</td>
<td>0.5964</td>
<td>0.8560</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>Precision</td>
<td>0.9113</td>
<td>0.8950</td>
<td>0.7309</td>
<td>0.7594</td>
<td>0.8906</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>0.9390</td>
<td>0.8967</td>
<td>0.5684</td>
<td>0.6975</td>
<td>0.8906</td>
</tr>
<tr>
<td></td>
<td>F1</td>
<td>0.9248</td>
<td>0.8958</td>
<td>0.6395</td>
<td>0.7271</td>
<td>0.8906</td>
</tr>
<tr>
<td>Our</td>
<td>Precision</td>
<td>0.87</td>
<td>0.84</td>
<td>0.77</td>
<td>0.78</td>
<td>0.8523</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>0.92</td>
<td>0.86</td>
<td>0.38</td>
<td>0.57</td>
<td>0.8523</td>
</tr>
<tr>
<td></td>
<td>F1</td>
<td>0.90</td>
<td>0.85</td>
<td>0.51</td>
<td>0.66</td>
<td>0.8523</td>
</tr>
</tbody>
</table>

IV. CONCLUSION AND FUTURE WORK

In this paper, we report the details of our tool for participating in ’The NLBSE’23 Tool Competition’ [14]. We applied the SGD-based classifier to predict the issue type. Despite using a simple and classical supervised machine learning-based approach, we achieved comparable results to previous complex approaches, i.e., an F1 micro average score of 0.8523. This study applied an SGD-based classifier to the archival data from open-source projects. However, in the future, we aim to utilize it to classify the issue reports in real-industrial settings.

ACKNOWLEDGMENT

This work has been supported by ELLIIIT. ELLIIIT is a strategic research environment in information technology and mobile communications funded by the Swedish government.