Efficient and Dynamic Bug Report Prediction Using Mining Techniques

K. Sulopriya¹, P. Rajkumar², N. Mohan Prabhu³

M.E. Computer Science and Engineering, Mount Zion College of Engineering and Technology, Lena Vilakku, Thirumayam Tk, Pudukkottai, Tamil Nadu 622507, India.
Assistant Professor, Department of Computer Science and Engineering, Mount Zion College of Engineering and Technology, Lena Vilakku, Thirumayam Tk, Pudukkottai, Tamil Nadu 622507, India.
Assistant Professor, Department of Computer Science and Engineering, Mount Zion College of Engineering and Technology, Lena Vilakku, Thirumayam Tk, Pudukkottai, Tamil Nadu 622507, India.

ABSTRACT: Arrangement bugs are one of the overwhelming reasons for programming disappointments. Past reviews demonstrate that a design bug could bring about enormous money related misfortunes in a product framework. The significance of arrangement bugs has pulled in different research thinks about, e.g., to identify, analyze, and settle setup bugs. Given a bug report, an approach that can distinguish whether the bug is an arrangement bug could help designers diminish investigating exertion. We allude to this issue as arrangement bug reports forecast. To address this issue, we build up another mechanized structure that applies content mining advancements on the natural language portrayal of bug reports to prepare a factual model on chronicled bug reports with known names (i.e., design or non-arrangement), and the measurable model is then used to anticipate a name for another bug report. Designers could apply our model to naturally foresee names of bug reports to enhance their profitability. Our instrument first applies include choice methods (e.g., data pick up and Chi-square) to preprocess the literary data in bug reports, and afterward applies different content mining systems (e.g., innocent Bayes, SVM, guileless Bayes multinomial) to construct factual models. We assess our answer on 5 bug report datasets including accumulo, activemq, camel, flume, and wicket. We demonstrate that credulous Bayes multinomial with data picks up accomplishes the best execution. By and large over the 5 extends, its precision, arrangement F-measure and non-setup F-measure are 0.811, 0.450, and 0.880, separately. We likewise contrast our answer and the technique proposed by Arshad et al.. The outcomes demonstrate that our proposed approach that utilizations credulous Bayes multinomial with data pick up all things considered enhances precision, design F-measure and non-setup F-measure scores of Arshad et al.'s technique by 8.34%, 103.7%, and 4.24%, separately.

KEYWORDS: bug Configuration, Bug Detection and Prediction, Selection of Features, Mining Principles.

I. INTRODUCTION

Modern software systems allow users to customize the systems behaviors via configuration options. However, the flexibility of configuration options could potentially affect the reliability of the software systems. Previous studies show that configuration bugs (i.e., mis-configuration) are one of the major causes for the downtime of large-scale software systems.

Scenario-1: Without Tool: Bob is a junior developer in a large software project which contains many configuration files. One day, he was asked to fix a newly reported bug.
Due to his lack of experience, he tried to fix this bug by modifying various source code files. And he wasted much time and effort to locate the relevant source code files, however, he still did not find the root cause of the bug.

Then, he asked a senior developer Alice for help. After reading the description of the bug report, Alice told him that this bug was caused by a wrong setting in a configuration file, and only one parameter in the configuration file needs to be modified. Finally, Bob fixed this bug by making that small change in the configuration file, however much time and effort had been wasted.

Scenario-2: With Tool: Bob is a junior developer in a large software project which contains many configuration files. One day, he was asked to fix a newly reported bug. Bob initially thought that the root cause of this bug was in one of the source code files. To confirm his initial analysis, he used our tool. However, our tool told Bob that this bug is highly likely to be a configuration bug, and thus he should look into the configuration files. He followed the direction of our tool and soon he was able to locate the root cause which is a wrong setting in a parameter in a configuration file. The total bug fixing process only took him a short period of time.

<table>
<thead>
<tr>
<th>Project</th>
<th># Bugs</th>
<th>Time</th>
<th># Conf</th>
<th># Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>accumulo</td>
<td>181</td>
<td>2011.10 - 2013.06</td>
<td>33</td>
<td>227</td>
</tr>
<tr>
<td>activemq</td>
<td>175</td>
<td>2005.12 - 2007.12</td>
<td>29</td>
<td>327</td>
</tr>
<tr>
<td>camel</td>
<td>1,189</td>
<td>2007.07 - 2013.09</td>
<td>333</td>
<td>1,261</td>
</tr>
<tr>
<td>flume</td>
<td>279</td>
<td>2010.07 - 2013.05</td>
<td>83</td>
<td>341</td>
</tr>
<tr>
<td>wicket</td>
<td>1,379</td>
<td>2006.11 - 2013.09</td>
<td>46</td>
<td>1,340</td>
</tr>
</tbody>
</table>

Fig.1 System Architecture Design

<table>
<thead>
<tr>
<th>Confusion Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classified as</td>
</tr>
<tr>
<td>Configuration</td>
</tr>
<tr>
<td>TP</td>
</tr>
<tr>
<td>FN</td>
</tr>
</tbody>
</table>

Copyright to IJIRSET
DOI:10.15680/IJIRSET.2017.0602116
II. PRE-PROCESSING

In this module, we can eliminate stop words and stemming words. In corpus linguistics, part-of-speech, also called grammatical tagging or word-category disambiguation, is the process of marking a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph. In computing, stop words which are filtered before or after processing of natural language data (text).

Though stop words usually refer to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools, and indeed not all tools even when morphologically inflected, and a lemma is the base form of the word. Stemming words are also removed from user reviews.

- Pre-processing is used to construct the structured data.
- In this module removing stop words from data collection.
- Eliminating stemming words from review database.
- The reviews are imported then removing the stop words and stemming words.

Feature Selection

Feature selection techniques should be distinguished from feature extraction. Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of the features. Feature selection techniques are often used in domains where there are many features and comparatively few samples (or data points).

Feature selection techniques are classified into two categories. They are

- Information Gain
- Chi-square

Classifier Construction

Predicts categorical class labels (discrete or nominal) use labels of the training data to classify new data.

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Logistic</th>
<th>Random Forest</th>
<th>SVM</th>
<th>Naïve Bayes</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.500</td>
<td>0.341</td>
<td>0.612</td>
<td>0.387</td>
<td>0.415</td>
</tr>
<tr>
<td>Conf. F-measur</td>
<td>0.540</td>
<td>0.335</td>
<td>0.636</td>
<td>0.359</td>
<td>0.540</td>
</tr>
<tr>
<td>Non. F-measur</td>
<td>0.545</td>
<td>0.323</td>
<td>0.640</td>
<td>0.344</td>
<td>0.545</td>
</tr>
</tbody>
</table>

Machine Learning Algorithm

Configuration bug report prediction framework: The whole framework includes two phases:

- Model building phase
- Prediction phase.

In the model building phase, our goal is to build a classifier (i.e., statistical model) by leveraging text mining techniques from historical bug reports with known labels (i.e., configuration or not). In the prediction phase, this classifier would be used to predict if an unknown bug report would be a configuration bug report or not (i.e., a non-configuration bug). Our framework first extracts features from a set of training bug reports (i.e., bug reports with known status).
Features are various quantifiable characteristics of bug reports that could potentially distinguish reports that are related to configuration bugs from those that are not. In this system, we use textual features from the natural-language description of bug reports. Our framework extracts the description and summary texts from bug reports. For each description and summary text, our framework tokenizes them, removes stop words (e.g., I, you, he, the), stems them (i.e., reduces them to their root forms, e.g., “configuration” and “configure” are reduced to “config”), and represents them in the form of a “bag of words”. Then, our framework applies feature selection techniques to select a subset of relevant textual features to further improve the prediction performance.

For each description and summary text, our framework tokenizes them, removes stop words (e.g., I, you, he, the), stems them (i.e., reduces them to their root forms, e.g., “configuration” and “configure” are reduced to “config”), and represents them in the form of a “bag of words”. Then, our framework applies feature selection techniques to select a subset of relevant textual features to further improve the prediction performance. Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data.

**Naive Bayes Multinomial (NBM)**

Naive Bayes multinomial (NBM) is similar to naive Bayes, but the label for a bug report is not simply determined by the presence or absence of terms in the bug report, rather by the number of times each of the terms appears in the bug report. In general, NBM performs better than naive Bayes when the total number of unique terms in the bug reports collection is large.

**K-Nearest Neighbor (kNN)**

K-nearest neighbor (kNN) is an instance-based algorithm for text mining. The general idea behind kNN is to predict the label of a bug report based on its k-nearest neighbors (i.e., bug reports). kNN has two steps: For an unlabeled bug report, kNN finds its k-nearest neighbors in the training bug reports according to a distance metric. kNN then assigns to this unlabeled bug report the most frequent label that its k-nearest neighbors have, i.e., if the number of neighboring bug reports that are configuration bug reports are more than those that are not, then we classify it as a configuration bug report; else otherwise.

**Support Vector Machine (SVM)**

Support vector machine (SVM) was developed from statistical learning theory, and it constructs a hyperplane or a set of hyperplanes in a high- or infinite-dimensional space, which are used for classification. Each training bug report is represented as a point in a multi-dimensional space where each term (i.e., feature) represents a dimension. SVM selects a small number of critical boundary instances (i.e., bug reports) as support vectors for each label (in our case, the labels are configuration and non-configuration), and builds a linear or non-linear discriminant function to form decision boundaries with the principle of maximizing the margins among training bug reports belonging to the different labels.

**Bayesian Network (BN)**

Bayesian network (BN) is a graphical model which uses probability theory to represent the relationships between terms and labels in bug reports. It is a directed acyclic graph (DAG) and each node in a BN represents either a term or the label. A directed edge between two nodes denotes that there is a causal (i.e., dependency) relationship.
between them. In the model building phase, BN would construct a Bayesian network from the training bug reports. In the prediction phase, this Bayesian network is then used to predict the label for a new unlabeled bug report.

III. EXISTING SYSTEM

In existing system, Configuration bugs are one of the dominant causes of software failures. Previous studies show that a configuration bug could cause huge financial losses in a software system. To make the project development team to utilize its resources efficiently. Previous bugs are good predictors of future bugs.

The source control repositories, bug reports, design and code artifacts etc. will be utilized as data sources. Open source projects like Eclipse, Mozilla and Android will be used for simulations. The importance of configuration bugs has attracted various research studies, e.g., to detect, diagnose, and fix configuration bugs. Given a bug report, an approach that can identify whether the bug is a configuration bug could help developers reduce debugging effort.

Disadvantages
- Configuration bugs are one of the dominant causes of software failures.
- Configuration bug could cause huge financial losses in a software system.
- The importance of configuration bugs has attracted various research studies, e.g detect, diagnose, and fix configuration.

IV. PROPOSED METHODOLOGY

In proposed system, develop a new automated framework that applies text mining technologies on the natural-language description of bug reports to train a statistical model on historical bug reports with known labels (i.e., configuration or non-configuration), and the statistical model is then used to predict a label for a new bug report. Developers could apply our model to automatically predict labels of bug reports to improve their productivity.

Our tool first applies feature selection techniques (e.g., information gain and Chi-square) to preprocess the textual information in bug reports, and then applies various text mining techniques (e.g., naïve Bayes, SVM, naïve Bayes multinomial) to build statistical models. We evaluate our solution on 5 bug report datasets including accumulo, activemq, camel, flume, and wicket. We show that naïve Bayes multinomial with information gain achieves the best performance. On average across the 5 projects, its accuracy, configuration F-measure and non-configuration F-measure are 0.811, 0.450, and 0.880, respectively. We also compare our solution with the method proposed by Arshad et al. The results show that our proposed approach that uses naïve Bayes multinomial with information gain on average improves accuracy, configuration F-measure and non-configuration F-measure scores of Arshad et al.'s method by 8.34%, 103.7%, and 4.24%, respectively.

Modern software systems allow users to customize the systems behaviors via configuration options. However, the flexibility of configuration options could potentially affect the reliability of the software systems. Previous studies show that configuration bugs (i.e., misconfiguration) are one of the major causes for the downtime of large-scale software systems.
V. LITERATURE SURVEY

In the year of 2013 the authors "S.Zhang, M. D. Ernst" proposed into their paper titled "Automated diagnosis of software configuration errors" such as The behavior of a software system often depends on how that system is configured. Small configuration errors can lead to hard-to-diagnose undesired behaviors. To present a technique and its tool implementation, called ConfDiagnoiser to identify the root cause of a configuration error a single configuration option that can be changed to produce desired behavior. Our technique uses static analysis, dynamic profiling, and statistical analysis to link the undesired behavior to specific configuration options.

It differs from existing approaches in two key aspects: it does not require users to provide a testing oracle to check whether the software functions correctly and thus is fully automated; and it can diagnose both crashing and non-crashing errors.

In the year of 2013 the authors "X. Xia, D. Lo, X. Wang, B. Zhou" proposed into their paper titled "Tag recommendation in software information sites" such as The propose Tag Combine, an automatic tag recommendation method which analyzes objects in software information sites.

Tag Combine has three different components. The scope of our system ranges from the design team until the training of the justice agents. A variety of online media to search and become informed of new and interesting technologies, and to learn from and help one another. It is common to see tags in software information sites and many sites allow users to tag various objects with their own words.

In the year of 2010 the authors "M. Gegick, P. Rotella, T. Xie" proposed into their paper titled "Identifying security bug reports via text mining: An industrial case study" such as A bug-tracking system such as Bugzilla contains bug reports (BRs) collected from various sources such as development teams, testing teams, and end users.

When bug reporters submit bug reports to a bug-tracking system, the bug reporters need to label the bug reports as security bug reports (SBRs) or not, to indicate whether the involved bugs are security problems. These SBRs generally deserve higher priority in bug fixing than not-security bug reports (NSBRs).

VI. EXPERIMENTAL RESULTS

![Fig.3. R-Studio Login Page](image-url)
Fig. 4. Loading CSV File

Fig. 5. Training Dataset
Fig. 6. Testing Dataset

Fig. 7. kNN Data Classification
In this approach, we propose an automated tool which applies feature selection and classification techniques to build a statistical model from the natural-language description of historical bug reports, to predict whether a newly submitted bug report is a configuration bug or not. Various feature selection techniques (e.g., information gain and Chi-square) and various classification techniques (e.g., naive Bayes, naive Bayes multinomial, kNN, SVM, and Bayesian network) have been investigated. We evaluate our proposed tool on 5 open source projects including a total of 3,203 bug reports. The experiment results show naive Bayes multinomial with information gain performs the best; on average across the 5 projects, its accuracy, configuration F-measure and non-configuration F-measure are 0.811, 0.450, and 0.880, respectively, which improve Arshad et al.’s method by 8.34%, 103.7%, and 4.24%, respectively. In the future, we plan to improve the effectiveness of our proposed tool further. We also plan to experiment with even more bug reports from more projects.

REFERENCES


BIOGRAPHY

Ms. Sulpriya. K. Studying M.E. Computer Science and Engineering in Mount Zion College of Engineering and Technology, which is located in Lena Vilakku, Padukkottai, Tamil Nadu, India.