International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Towards The Roadmap of Defect Predictors & Its Diagnostics

Shubha Sharma[1], Anubhooti Papola[2]

[1]Master of Technology, Uttarakhand Technical University (FOT), Dehradun, India

[2]Assistant Professor, Uttarakhand Technical University (FOT), Dehradun, India

Abstract*: Data mining and efficient knowledge discovery in database have been attracting a significant amount of research, industry attention in mostly all the fields. The paper discusses about "Bug prediction": an essential phase in the process of bug triaging and fixing process. Bug prediction solutions have been proposed. The demonstration of their effectiveness however faces number of issues including bug triaging, inappropriate evaluation measures: likelihood, precision and recall values. The major drawback of bug prediction technique is unsatisfactory prediction resulting in least accuracy and conspicuous bug report. With the new advancement, machine learning classifiers have proved and emerged as a trusted way to predict the introduction of bugs in changes made to sources. The demerits of existing classifier based prediction techniques are insufficient data and slow performance. The objective is to reach optimal classification performance with naïve bayes accompanied with svm (support vector machine) classifiers and substantial feature extraction, further analyzing its performance on the basis of its evaluation measures-"likelihood, precision and recall value, f-measure". The evaluation result show that proposed approach can achieve better search results than existing search programs .the work will help in precise and relevant prediction of bug reports and will help in diagnostics of mislabeling of bug reports.*

Keywords: bug reports, prediction, search quality, maintenance.

1. Introduction

Identifying software faults in an effective manner is a crucial process since corrective maintenance costs increase exponentially if the faults are detected later in the software development life cycle [5]. Predictive models have been used to predict newly reported bugs by assigning priority of each bug. Bug tracking repositories like Bugzilla and others are used for monitoring and managing the software systems. In bug tracking system, record of different features about the bugs, such as when the bugs was reported and in which component are maintained. Work regarding software maintenance [1, 13] and evolution [12] often require information on both the bugs that are reported and the fixes that developers applied. Therefore, open source software systems receive abundant rate of bug reports daily. Also, bug triaging, the process of assigning bugs to a developer, is a labor-intensive, time-consuming and fault prone if done manually. Many a time's security reports are mislabeled and managing them consumes lots of time.

Databases with hundreds of fields and tables and millions of records and multi gigabyte size were commonplace and tera byte database came to use from last few years. Low noise data i.e. few data errors are another consideration and "accessing of statistical significance": problem occurred when the system is searching over many problems at a time. A key finding to the software testing is the fact that faults tends to cluster; i.e. to be contained in a limited number of software modules [11]. This strongly motivates the use of software fault prediction models which provide an sure and considerable indication whether the source is likely to contain faults i.e. to construct such prediction model that timely indicates fault segments or fault report, a number of solution have been proposed.

Previous work formulated the problem as a classification task that automatically suggested the files where a bug was almost likely to be fixed. Based on its bug report's content. They build predictive model that extracted features with actual fix locations and also trained a prediction model that later predicts the fix location for a new bug report or say small number of bug reports. Classification technique implemented is naïve bayes. This approach suffered from the demerit that only predictable reports were considered and deficient reports were avoided at the first step. And also this was the reason to lower its evaluation measures. In this paper the goal is to improve the prediction accuracy along with its evaluation measures. Before, the algorithm only concentrated on the best parameter for one particular model using a limited set of data. It is termed as "over fitting" .It used to model only the general patterns in the data and noise specific to the data set, resulting in poor performance of the model at the time of testing data. This called the need of cross validation, regularization and other sophisticated statistical strategies.

To summarize, we make the following key contribution in this paper:

1. We reach much higher prediction accuracy compared to other classification based bug prediction, fix and triaging methods by:
   - Using features of bug reports other than the textual data. Besides the normal bug description used in prior work, we incorporated more features like the component in which bug belongs to.
   - Most importantly, we have considered both security bug reports and non security bug reports considering statistical bug track and evaluated the result in first module using each reports probability separately. This helps in the wise evaluation of result.

We perform experimental evaluation using bug reports datasets obtained from real projects. We used larger datasets compared to previous work. The experimental results show

that building a classifier model using the different feature. Our evaluation shows the effectiveness of the two phase predictor over the other models and suggests tangible benefits when deploy.

Figure 1: shows the applications to manage database in various department.

2. The traditional method of turning data into data into knowledge relied on manual analysis & interpretation & faced many faults. The need of software maintenance and efficiency arises because it is important in every field such as:
   - Health care: common need for specialist to periodically analyze current trends and change in health care. In this case efficiency plays an important role.
   - Marketing: The primary application in database marketing system which analyze customer database to identify customer groups and forecasts their behavior. Thereby, fraud detection showed its importance in this field too to achieve customers satisfaction and security.

Likewise in every field where knowledge discovery and data access is used, data cleaning and fault prediction is also required in them. The rest of the paper is organized as follows: section2 describes the related work and section 3 describes proposed approach. The experimental evaluation and fault measure validation are presented in section 4.Section 5 discusses threats to validity and section 6 concludes the paper.

2. Related work

Many approaches adopted both machine learning and information retrieval techniques to improve the bug triaging process. Cubranic et al.[4] were the first to use a text classification approach to automatically assign bug reports to developers. Anvik et al. [6] [7] improved the approach proposed by Cubranic et al. by removing inactive developers. By inactive developers here, we mean that developers with a too low bug fixing frequency or developers those who are not working on the project anymore. They employed SVM, Naïve Bayes and Decision Trees classification techniques, and reported prediction accuracy of up to 64%.Several approaches were proposed in order to enhance bug assignment accuracy. Park et al.[8] proposed a bug triaging approach. Their approach incorporated collaborative filtering recommender and topic modeling to improve bug prediction and triaging, reduce the sparseness of training data. Zou et al. [13] proposed the training set reduction with both feature selection and instance selection technique for bug triage. Alenezi et al. employed five state-of-the art term selection methods on the textual description of bug reports to produce discriminating terms. After that they built a classification model on the discriminating terms using Naïve Bayes classifier.

Let us discuss here about assignment automation .assignment automation is to lighten the load for a triager by recommending, for a given bug report [7], who might be appropriate for it and address all the problems accurately. To date, two predominant types of approaches have emerged:
(1) Machine learning techniques and
(2) Statistical analysis techniques of bug tossing graphs.

In the first approach, features such as keywords and metadata are extracted from past bug reports and, together with data linking these bug reports to developers who then fixed them, used to train a machine learning model [7]. Several other studies refine the pure machine learning approach like Latent Dirichlet Allocation to categorize bug reports [9], leveraging fuzzy set-based modeling to automate developer tasks [3], or processing of source code authorship information to recommend developers [10]. In the second type of approach efficient statistical analysis is the major part if it is attained. This approach relies on statistical analysis of bug tossing graph. A bug tossing graph captures the history of bug report reassignment from developer to developer, and uses it as a source for a statistical analysis that aims to detect repeated pattern.

2.1 Classifier Used

Bayesian Network Classifier

Naïve Bayes Classifier: A naïve bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong (naïve) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model".

In simple terms, a naïve bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature [12]. For example, a fruit may be considered to be an apple if it is red, round and about 4 inches in diameter. Even if these features depend on each other or upon the existence of the other features, a naïve bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple. Depending on the precise nature of the model naïve bayes classifier, it can be trained efficiently in a supervised state. Also, in many practical applications, parameter estimation for naïve bayes models uses the method of likelihood; if we elaborate in simple words, one can work with the naïve bayes model without believing in Bayesian probability or using any Bayesian methods [4]. In spite of their naïve design and apparently over simplified assumptions, naïve bayes

classifiers have worked quite well in many complex real-world situations.

The Naive Bayes Probabilistic Model: The probability model for a classifier is a conditional model $P(c|f_{1....}f_n)$ over a dependent class variable with a small number of outcomes or classes, conditional on several feature variables f_1 through f_n.

Support Vector Machine: Support Vector Machines (SVM'S) are relatively new learning method used for binary classification. SVM's introduce the notion of a kernel induced feature space which cast the data into higher dimensional space where the data is separable [11]. Furthermore, the VC-dimension (a measure of a system's likelihood to perform well on unseen data) of SVM can be explicitly calculated. Overall, SVM'S are intuitive, theoretically well-founded and have shown to be practically successful. SVM's have also trained to solve regression tasks (where the system is trained to output a numerical value, rather than \yes/no'' classification) finding the optimal curve to the data is difficult and it would be not a intelligent decision to use the method of finding the optimal hyper plane. The process is to pre-process the data in such a way that the problem is transformed into one of finding a simple hyper plane.

Natural Language Processing: Natural language presents significant opportunities for mining in free form text, especially for automated annotation and indexing prior to classification of text corpora limited parsing capabilities can help substantially in the task of deciding what an article refers to. Hence, the spectrum from simple natural language processing all the way to language understanding can help substantially. Also natural language processing can contribute significantly as effective interface for stating hints to mining algorithms and visualization in explaining knowledge derived.

3. Proposed Approach

We propose machine learning approach to predict buggy files to fix from the given bug reports.

3.1 Feature Extraction

A bug report is the main source of information for developers to understand to understand a bug i.e. the bug summary briefly describes the bug while the initial description it in detail, metadata provides bug's basic information such as version and comments record discussions from bug reporters and developers. As the approach uses machine learning classification. The complete process contains the following major steps:

1) **Bug Representation**: The bug reports are collected and the collections of bug reports are represented and grouped together to form a master bug report. This master report would be useful ahead for the ongoing process.
2) **Query Formulation**: Now, for further processing of the model, the short description and the keywords of the master bug report are combined to one query. Note that every description should be valid. The short description of the report includes summary, initial description and metadata (OS, priority, status and reporter).
3) **Relevance Labeling:** we decided labels or grade according to the status in the training data that either it is resolved, new, unresolved, verified or unconfirmed. The top k reports that are verified are given the resolution fixed and also there are reports that carry a resolution grade: duplicate, invalid and others. Each report is also assigned a category of non security bug report (NSBR) and security bug report (SBR).
4) **Feature Vector Generation:** As our approach uses machine learning classification, we transformed a bug report into a feature vector.

The queries and textual bug reports require the following preprocessing steps:

1) Word regularization: All letters are lowercased. Words are segmented by standard whitespace characters.
2) Word stemming: Stemming is a commonly used technique to normalize words with the same root by removing their suffixes. For e.g.: compute is the stem of computes, computing and computed.
3) Word splitting: in bug reports, there is much source code mixed in the text.
4) Stop words removal: stop words are those words that do not contribute an actual semantic meaning, such as the prepositions. We remove the stop words, according to the stop word list.

After the whole process was executed, stop words from the summary will be removed .It also showed the frequency of an alphabet occurring maximum times.

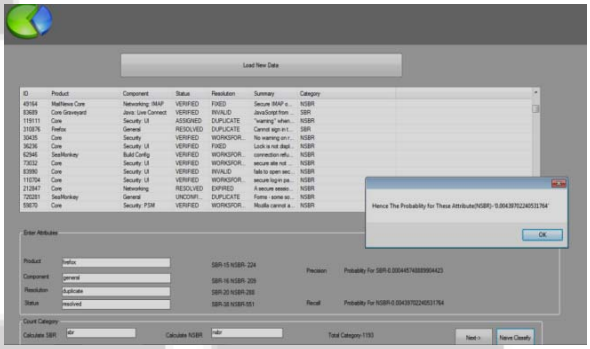

Figure 2: Showing the process of pre-pre-processed new loaded data with precision and recall value of bug reports with respect to values of SBR and NSBR.

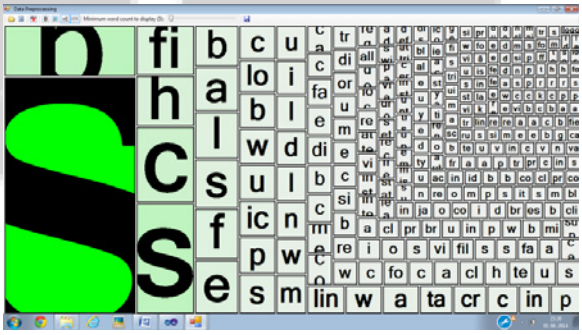

Figure 3: showing the frequency of the words occurrence as frequent item. Here S has the maximum frequency of occurring as a frequent item.

3.2 Data Preprocessing

A first important step in each data mining exercise is preprocessing the data. Bug reports are unstructured data which contain irrelevant terms .Therefore, it becomes necessary to apply the traditional text processing approach to transform the text data into a meaningful representation. After the stop words removal, the approach constructs a bug term matrix frequency. After that a filtering is performed to refine the training set further to remove reports that are assigned to inactive developers or reports that do not have sufficient words to describe themselves as meaningful information. Each observation (software module or file) in the data sets consists of unique ID, several static features and an error count with error description. Naive Bayes classifier is used to build the predictive model. Although it is known for its simplicity, it is well suited to our problem. The Naïve Bayes algorithm has been found to perform astonishingly well in information retrieval. In the existing work, precision values were not getting merge and likelihood was also not accurate. In the proposed work, after the stop words removal, we will get a frequency of the words i.e. occurrence of each word as a frequent item. Next we will calculate the recall and likelihood in the summary. It showed show that if we bug track words individually then the resulting precision and likelihood is high and iterations are low. So when we do it by implementing on summary through NLP (Natural Language Processing) taking it towards SVM (Support Vector Machine), likelihood can be reduced.

Taking this model to a statistical and probabilistic approach, it allows us to capture uncertainty about the model i.e. in a principle way by determining probabilities of the outcomes. We aim to solve diagnostic and predictive problem. After the mining process is been finished, the resultant file is been saved in an attribute relationship file format. This is been read as the test set file for each source file. The major problem is of mislabeling of SBR (secure bug reports) as NSBR (non secure bug reports) that causes serious damage to software systems stakeholders sometimes due to delay of identifying and fixing the involved security bug. Therefore we calculated the probability in terms of security bug reports and non security bug reports separately and their precision and recall value.

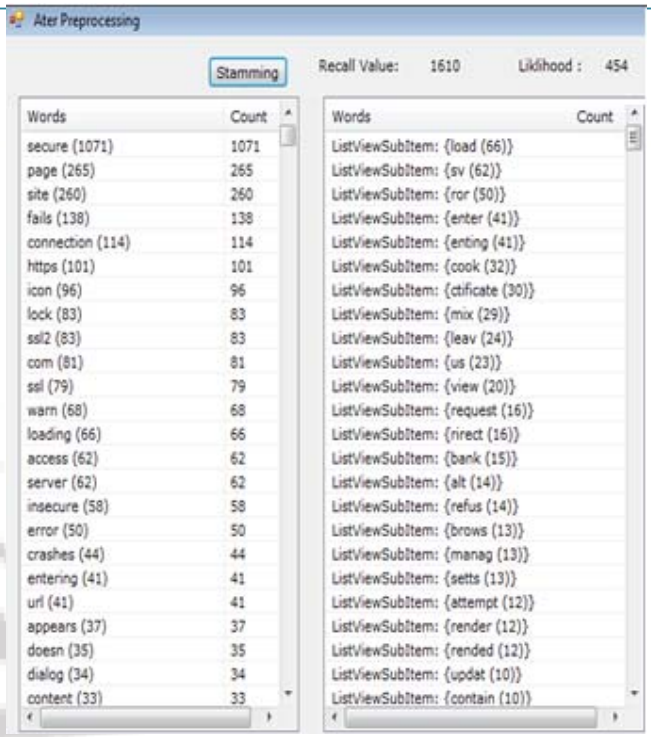

Figure 4: shows the count of each processed word after stemming each attribute individually with recall and likelihood.

4. Fault Measure Validation

To quantitatively evaluate a bug tracking approach, we propose to use standards metrics from the field of information retrieval, namely Likelihood, Precision, Recall, And F-measure metrics.

- Likelihood measures the accuracy of prediction results. This is an effective measure to evaluate recommendation techniques. We consider the prediction results to be correct if at least one of the recommended *k* files matches one of the actual source or patch files for a given bug report. If none of the recommended files matches, the prediction is correct. We denote the number of bug reports as N_C. If the addressing prediction is correct, N_{IC} if prediction is incorrect. The following formula computes the percentage of bug reports for which the prediction is correct:
  Likelihood = $N_C \div (N_{C+} N_{IC})$
- Precision characterizes the number of correctly predicted files over the number of files recommended by our approach. We denote the set of genuine files fixed for a bug report as F_B and the set of recommended files for diagnostics as $F_{R:}$
  Precision = $| F_{B \cap} F_R | \div | F_R |$
- Recall characterizes the number of correctly predicted files over the number of actual fixed files:
  Recall =$| F_{B \cap} F_R | \div | F_B$
- F_1 score is a measure of test's accuracy in statistics. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct results divided by the number of all returned results and r is the number of correct results divided by the number of results that should have been returned.

F_1=2.(Precision. Recall ÷ Precision + Recall)

The F_1 score can be interpreted as a weighted average of the precision and recall, where an F_1 score reaches its best score at 1 and worst score at 0.

4.1 Research Questions and Metrics

- Question 1: which variable lead to best bug prediction when using feature selection?
  The three main variables affect bug prediction performance that are discussed in this paper: 1) type of classifier (Naïve Bayes, Support Vector Machine), 2) process of feature extraction, and 3) whether multiple instances of a particular feature are significant i.e. in terms of count or whether only the existence of a feature is significant (binary).In this approach feature selection is performed, followed by computation per project accuracy .once all projects are complete, average values across all projects are computed. Every result in the table reports the standard deviation.

- Question 2: what is the sensitivity of the tool when training data is changed?
  Advanced techniques for diagnostic of data with the tool, use machine learning algorithms that rely on training data for computing the similarity thresholds for detection of bug links. Variations in real-world datasets may therefore impact the performance of such bug linking tools. Consequently, for a bug linking tool that relies on machine learning approaches, it is important to consider and investigate its sensitivity when data is trained and triaged.

- Question3: How is the proposed approach in terms of time efficiency?
  We, examined the efficiency of the proposed approach in terms of training time against different sizes of the training set, i.e. the numbers of pairs used for training the Naïve Bayes model and how much time effective is it to implement on a summary through SVM after being trained to reduce likelihood and get good results.

5. Threats to Validity

There are potential threats to the validity of our work. Like other data mining research work, our conclusions could be biased by the type of data that were used. To validate our approach, we perform evaluations on reports collected from popular, large – scale open source systems. Choosing reports from open source community was intentionally done because they are high quality reports and extraction of data from them could be the efficient one. Some may consider evaluation method as biased; we laid emphasis on measuring the prediction accuracy using likelihood, which considers the prediction to be correct if at least one of the recommended files matches the actual patch file. However the relative improvement using the classifier the extraction techniques cannot be ignored.

6. Conclusions and Future Work

Time and cost effective software development are decisive for today's developers, several approaches to tackle the problem of software bugs have been investigated. Software bug prediction can be regarded as just one piece of the solution to these issues. We computed prediction in a series of training and processing datasets. In our work we provide a clean open source datasets for bug diagnostics. The results showed that overall approach received very good precision, for maximum number of source files i.e. over 90%, for some programs, but delivers little lesser recall rates. The f-measure results proved to be improved but also showed there is room for improvement in the area of bug linking. Now, if we talk about misclassifying a faulty instance, then our findings indicate it to be the appropriate approach to solve this problem. Using Bayesian network learners, into these different information sources could be gained which is still a topic left for future research and efficient bug linking approach should be worked upon to solve the problem of software teams and save time.

References

[1] A. Mockus, R.T. Fielding, and J.D. Herbsleb, "Two case studies of open source software development: "Apache and Mozilla," ACM Trans. Softw. Eng, Methodol.vol. 11, no. 3, pp. 309-346, 2002.
[2] A. Zeller, J. Krinke, Essential Open Source Tool Set, John Wiley and Sons, 2005.
[3] A.Tamrawi, T.T. Nguyen, J.Al-Kofahi, and T.N Nguyen, "Fuzzy Set Based Automatic Bug Triaging (NIER track)", 33*rd ICSE, 2012, pp, 125*-130.
[4] Cheng, J., R. Greiner, J. Kelly, D. Bell and w.liu, 2002.Learning Bayesian networks from data: an information- theory based approach, Artificial Intelligence, 137:43-90.
[5] Based approach, D.Cubranic and G.C Murphy, "automatic bug triaging using text categorization," in *SEKE 2004: Proceedings of the Sixteenth International Conference on Software Engineering.* Cite seer 2004, pp. 92-97.
[6] Fisher, M., M. Pinzger and H.Gall, 2003.Populating a release history database from version control and bug tracking systems, proceeding on IEEE Conference on software maintenance.
[7] J.Anvik and G.C Murphy, "Reducing the effort of bug report triage: recommenders for development-oriented decisions," *ACM* Transactions on Software Engineering and Methodology (*TOSEM*), vol. 20, no 3, p. 10, 2011.
[8] J.Anvik, L. Hiew, and G.C Murphy, "who should fix this bug?'' in proceeding of the *28th international conference on software engineering*. *ACM*, 2006, pp.361-370.
[9] J-W. Park, M-W. Lee, J. Kim, S. Won Hwang, and S. Kim, "costriage: a cost-aware triage algorithm for bug reporting system." in, *AAAI,* W. Burgard and D. Roth, Eds. AAAI Press, 2011.
[10] K.Somasundaram and G.C.Murphy, "Automatic categorisation of bug reports using latent Dirichlet allocation," 5th *ISEC*, 2012, pp, 125-130.

[11] Kumaravel, A. and K. Rangarajan, 2013.Algorithm for automation specification for exploring dynamic labyrinths, Indian Journal of Science and Technology, 6(6).
[12] Kumar Giri, R. and M. Saikia, 2013. Multipath routing for admission control and load balancing in wireless mesh networks", International Review on Computer and Software, 8(3):779-785.
[13] M. Linares- Vasquez, K. Hossen, H. Dang, H. Kagdi, M. Gethers, and D. Poshyvank, "Triaging Incoming Change Requests: Bug or Commit History, or Code Authorship?" 28th, *ICSM,* 2012.
[14] Sherer, S., 1275. Software fault prediction, journal of systems and software, 29(2): 25-105.
[15] T.J. Ostrand, E.J. Weyuker, and R.M. Bell, "Predicting the location and number of faults in large software systems," IEEE Trans. Softw. Eng., vol. 31, no. 4, pp. 340-355, 2005.
[16] W. Zou, Y. Hu, J. Xuan, and H. Jiang, "Towards training set reduction for bug triage," in Proceeding of the 2011 IEEE 35th Annual Computer Software and Applications Conference, Ser. COMPSAC '11.
[17] Z.Yin, D.Yuan, Y.Zhou, S.Pasupathy, and L. Bairavasundaram, "How do fixes becomes bugs?" in *ESEC/FSE*, 2011.

Author Profile

Shubha Sharma has received her degree in bachelor Of technology in Information Technology from Uttarakhand Technical University, Dehradun in (2007-2011).At present she is pursuing Master Of Technology in Computer Science & Engineering from Uttarakhand Technical University in (2012-2014).

.