Dynamic Bug Triage System with Data Classification Techniques

Swapnil Pasarkar
Computer Engineering N.M.I.E.T
Savitribai Phule Pune University
Pune, India

Shilpa Nair
Computer Engineering N.M.I.E.T
Savitribai Phule Pune University
Pune, India

Neha Bagal
Computer Engineering N.M.I.E.T
Savitribai Phule Pune University
Pune, India

Kshitija Godse
Computer Engineering N.M.I.E.T
Savitribai Phule Pune University
Pune, India

***Abstract*— Bug triage system is the process of fixing bug, which aims to correctly assign a developer to a new bug. Programming organizations spend more than 45 percent of expense in managing programming bugs. We have proposed an approach to diminish the time cost in manual work, content classification procedures connected to lead programmed bug triage. In this paper, we address the issue of information diminishment for bug triage, i.e. How to lessen the scale and enhance the nature of bug information. In proposed approach we are performing data reduction on bug data set which will reduce the scale of the data as well as increase the quality of the data and then apply k-NN and Support Vector Machine classification or combinational approach of both on the reduced dataset. We are using instance selection and feature selection simultaneously with historical bug data. In this paper we have presented a new module which will describe the status of the bug like whether it assigned to any developer or not and whether the developer has rectified it or not. We consolidate case determination with highlight choice to at the same time lessen information scale on the bug measurement and the word measurement. The goal of bug triaging is to assign potentially experienced developers to new-coming bug reports. We balance the load between developers based on their experience and past history. We conduct experiments on four real data sets. The outcomes demonstrate that our data lessening can successfully decrease the information scale and enhance the exactness of bug triage. Our work gives a way to deal with utilizing systems on information preparing to shape lessened and superb bug information in programming advancement and maintenance.**

***Keywords*— Bug, Bug triage, Bug repository, Data reduction, Instance selection, Feature selection, Data Mining, K-NN Approach, SVM classifier, NLP*.**

I. INTRODUCTION

Programming organizations spend more than 45 percent of expense in managing programming bugs. An unavoidable stride of fixing bugs is bug triage, which plans to accurately allocate a designer to another bug. To diminish the time cost in manual work, content classification systems are connected to lead programmed bug triage. In this paper, we address the problem of information lessening for bug triage, i.e., how to decrease the scale and enhance the nature of bug information. We consolidate case choice with highlight determination to all the while lessen information scale on the bug measurement and the word measurement. To focus the request of applying occasion determination and highlight choice, we remove properties from authentic bug information sets and construct a prescient model for another bug information set. We observationally research the execution of information diminishment on absolutely 600,000 bug reports of two vast open source ventures, to be specific Eclipse and Mozilla. The outcomes demonstrate that our information diminishment can viably lessen the information scale and enhance the precision of bug triage. Our work gives a way to deal with utilizing strategies on information preparing to shape decreased and superb bug information in programming advancement and support. Bug triage is an extravagant stride of programming support in both work cost and time cost. In this paper, we consolidate highlight choice with occurrence choice to decrease the size of bug information sets and also enhance the information quality. To focus the request of applying case determination and highlight determination for another bug information set, we concentrate traits of every bug information set and train a prescient model taking into account authentic information sets. We observationally research the information lessening for bug triage in bug vaults of two expansive open source ventures, to be specific Eclipse and Mozilla. Our work gives a way to deal with utilizing systems on information preparing to shape lessened and brilliant bug information in programming advancement and support. In future work, we anticipate enhancing the consequences of information decrease in bug triage to investigate how to set up a high quality bug information set and tackle a space specific programming assignment. For foreseeing decrease orders, we plan to pay endeavors to find out the potential relationship between the characteristics of bug information sets and the decrease orders. In [1] instance selection and feature selection techniques are used to eliminate the stop words and create a reduced bug dataset and then K-NN approach is used to classify data. We are using combinational approach of both in our proposed paper and then we are applying Support Vector Machine classifier to triage the bug to appropriate developer for bug resolution.

II. BACKGROUND AND MOTIVATION

A. Background

Software bugs are common problems faced by IT industries over the world. Programming organizations spend more than 45 percent of expense in managing programming bugs. Software bugs, for example a popular and open source bug repository are maintained in bug repositories, Bugzilla is one such widely used repository. Once a software bug is found, it is tested by a reporter maybe the tester, developer or the deployer and then if it is identified as a real bug then the occurrence of bug is reported. A recorded bug is called a bug report, which has multiple attributes for reproducing the bug. A human triager will assign the occurred bug to a developer, once a bug report is formed, and this developer will try to fix this bug. An attribute assigned-to is attached to this developer. The assigned-to developer now starts to resolve the bug. He/she first checks whether the occurred bug is an existing bug or a new bug has been generated. If it is an existing bug then it is resolved immediately using existing solutions and the acknowledgement/report is provided to user. If in case a new bug has been generated, then triager tries to find the possible solutions considering the past history, test cases and the modular breakdown structure of the project. If still he is unable to resolve the bug then he redirects it to another developer. If this developer is able to resolve the bug then the repository is updated and now further the occurrence of this bug report will be redirected to this new developer. A status attribute is attached to each bug report this attributes changes dynamically according to the resolution steps taken to resolve the bugs.

B. Motivation

Manual bug triaging is a time consuming process. In a manual bug triage system a triager first manually analyses the bug and decides to whom this bug report should be given for resolution. The triager has to manually study the modular breakdown structure of the entire project to detect in which module the bug has actually occurred. He has to manually analyse all the test cases, functions to get the idea behind bug occurrence. He also has to study the testers, developers, deplorer's past history and experiences before redirecting to them for bug resolution. This is not an easy job and leads to more inaccuracy in results. As a result of this bug resolution may take more time than the deadlines maintained or bug may not be resolved at all. Therefore we have proposed a dynamic bug triage system in which on the occurrence of bug it will be dynamically assigned to the developer by dynamically considering the already formed bug repository. This dynamic triaging of the bug leads to more accuracy and the overall performance is improved.

III. RELATED WORK

Different modules have been considered in implementing the idea described in project. The four modules used are: Project Analyzer, Bug Repository Maintenance, Bug Report Analysis, Task Scheduling and Notification.

A. Project Analyser

Project analyzer enhances user to maintain all the information regarding project and The process of gathering requirements is usually more than simply asking users what they need and writing their answers down. Depending on the complexity of the application, the process for requirement gathering has a clearly defined process of its own. This process consists of a group of repeatable processes such as to capture, document, communicate, and manage requirements. Module is intended to gather all the information of software development, like Software breakdown structure, module specification, their development strategy, modular dependencies, database, team details, manager, developer, tester, deployment testing team, services details, external reference used for project or module development, third party tools, Module Input Output Details, etc. from company. This formal process, which will be developed in more detail, consists of four basic steps.

Elicitation, Validation, Specification, Verification, are different stages that are considered for gathering requirements to create an appropriate bug dataset.

B. Bug Repository Maintenance

The Bug Repository data set is a collection of models and metrics of software systems and their histories. The goal of such a data set is to allow people to compare different bug prediction approaches and to evaluate whether a new technique is an improvement over existing ones. In particular, the data set contains the data needed to.

- Prediction technique is run based on source code metrics and/or historical measures and/or process information (logs data).
- Compute the performance of the prediction by comparing its results with a set, i.e., the number post release defects reported in bug tracking system.

Bug prediction is performed at the class level using the designed data set. However class data is aggregated to derive the package or subsystem information, since for each class it specifies the package that contains it.

C. Bug Report Analysis

An automatic bug triaging system is presented by recommending one experienced developer for each new bug report. These following steps are performed.

1. *Representation Framework*

   We have a collection of bug reports, B = {b1, b|B|}. Each bug report has a collection of term, T = {t1… t|T|}, and a class label (developer), c C = {c1, …, c|C|}.

2. *Term selection methods*

   High dimensionality of term space is reduced using term selection by selecting the most discriminating terms for classification task. The methods give a weight for each term in which terms with higher weights are assumed to contribute more for classification task than terms with lower weights.

3. *Chi-Square (X2)*

   In statistics, the X2 test is used to examine independence of two events. The events, X and Y, are assumed to independent if P (XY) = P(X)P(Y) term selection, the two events are the occurrence of the term and the occurrence of the class.

4. *Term Frequency Relevance Frequency (TFRF)*

   The basic behind the TFRF method is that the more high frequency for a term in the positive category than in negative category, the more contributions it makes in selecting the positive instances from negative instances.

5. *Distinguishing Feature Selector (DFS)*

   DFS is new novel term selection method. It provides global discriminatory powers of the features over entire text collection rather than being class specific. DFS considers the following requirements:

- Distinct term is a term that occurs frequently in a single class and not in other classes,
- irrelevant term is a term that rarely occurs in a single class and not in other classes,
- distinctive term is a term that occurs frequently in all classes is irrelevant,
- A term that occurs in some of the classes. DFS assigns scores to features between 0.5 (least discriminative) and 1.0 (most discriminative).

D. *Task Scheduling and Notification*

The development group with has run into this issue so many times that we decided to write our own solution to this problem and make solve, this module reaches up to developer as well as customer from Bug Report Analysis phase . It's called Revalee—as in reveille: a signal to arise—and it is a Service that freezes your job requests in overall process until it's time to thaw them out.

IV. ALGORITHMS

Mainly two Methods: Instance Selection and Feature Selection and two Algorithms: SVM and k-NN have been considered in implementing the idea described in project. They are as follows:

A. *Instance selection*

Instance selection method is the one that address the need of computational loads and reducing storage requirements. Using instance selection methods we can achieve enhanced performance from the learning algorithm and make it work effectively. This method is used to scale down the data to select only the relevant data that can be used in data mining algorithm. Sampling is a procedure and an important part of instance selection method that can draw a sample by random process. This process contains samples and each sample has an appropriate probability distribution. Consider a situation in which the data size or contents is very large most of this data is not useful in the training phase of learning algorithm. Focusing, enabling and cleaning are the three main functions where instance As the harmful and superfluous instances are removed and only critical instances are retained , the problem of instance selection is viewed more in terms of instance deletion. The response time for classification decisions decreases as we remove a set of instances from the database. In classification competence the removal of instances may lead to either an increase or decrease. Therefore, we must be clear about the degree to which we are willing to let the original classification accuracy depreciate when applying an instance selection scheme to a database of instances. For example, the number of cases we are forced to remove might be too large if we have a fixed storage limit, and unavoidably result in a degradation of classification accuracy. Uninstructive storage reduction is the principle objective of an instance selection scheme. Here, classification accuracy is primary: the same (or higher) classification accuracy is desired but we require it taking up less space and faster. Accuracy should never suffer at the expense of increased performance.

A non-homogeneous class can be defined as one which is defined by a group of instances not sharing the same locality. Here, the notion of border instance does not make any sense. One might argue that all of the instances are critical to the definition of the class as they make up the borders; when working with problems of this form the instance selection is a bad proposition. In this type of situation the safest way to remove a number of instances by keeping only prototypical instances selection is mainly applied.

B. *Feature Selection*

The feature selection technique is defined as the process of selecting a subset of relevant features such as variables, predictors for use in model construction or text classification. It is also known as variable selection, attribute selection or variable subset selection. Feature selection techniques are used for three reasons:

- Simplification of models to make them easier to interpret by researchers/users,
- Shorter training times,
- Enhanced generalization by reducing over fitting.

A feature selection algorithm is represented as the combination of a search technique for proposing new feature subsets along with an evaluation measure which results in the formation of different feature subsets. To test each possible subset of features is the simplest algorithm in finding the one subset which minimizes the error rate. Subset selection is a method that evaluates a subset of features amount the available feasible subsets which is nearest to optimality. There are multiple objectives in a feature selection task so the choice of optimality criteria is a difficult task. Some of the common task incorporate a measure of accuracy, penalized by the number of features selected (e.g. the Naive Bayes classifier algorithm). Feature selection is considered to be a powerful tool for simplifying or speeding up computations, and when implemented appropriately it can lead to little loss in classification quality. In our paper we apply combinational approach of instance selection and then feature selection to process the query that is fired by the end-user. Application of these techniques results in stop word elimination and extracted query is obtained which is further analyzed for classification in the bug resolution process.

C. Support Vector Machine (SVM)

The Support Vector Machine algorithm is a classification algorithm used on specific database as defined. The Support Vector Machine are supervised learning models in machine learning. Were, machine learning is defined as the subfield of computational learning and computer science, which is evolved from these itself. It involves the prediction of the result data present on previous or historic data. The Support Vector Machine is used to analyze data and for recognition of data patterns in case of this paper. In machine learning, Support Vector Machine are supervised learning models. They are associated with learning algorithms that analyze data and recognize patterns, used for regression analysis and classification. A Support Vector Machine training algorithm builds a model that assigns new examples into one category or the other. That is why it is called as a non-probabilistic binary linear classifier. An Support Vector Machine model is a representation of the examples as points in space. In Support Vector Machine points are mapped so that the examples of the separate categories are divided as wide as possible or by a clear gap. New points are then mapped into that same space. Depending on which side of the gap they fall on they are predicted to which category they belong. In addition to performing linear classification, Support Vector Machines can perform a non-linear classification efficiently using the kernel trick which is implicitly mapping their inputs into high-dimensional feature spaces. For classification, to transform input data to a high-dimensional feature space in which the input data is more separable compared to the original input space nonlinear kernel functions are used. Then the Maximum-margin hyper planes are created. The produced model depends on only a subset of the training data near the class boundaries. Also, the model produced by Support Vector Regression ignores any training data that is sufficiently close to the prediction model.

D. k-NN

k-NN is k Nearest Neighbor algorithm. K-NN is algorithm is a pattern matching algorithm, used for regression and classification. This algorithm is non-parametric. In k-NN the function is only approximated locally and all computation is deferred until classification, as k-NN is instance based learning. In k-NN algorithm, the neighbors are taken from the object sets for the classification or the object property value, k-NN regression is known. There is a training set required for this k-NN algorithm, but no explicit training steps are required. The k-NN algorithm is used for pattern recognition. An example of instance-based learning is k-Nearest neighbor, in which the training data set is stored. Simply by comparing the record to the most similar records in the training set a new unclassified record may be found.

V. DISCUSSION

Earlier bug triage system was manual, the bugs were assigned manually to developer for generation of appropriate solution. In this paper, we see the dynamic approach to the bug triage system, which gives an advantage by assigning the bug to the developer automatically without going for manual assigning of the bugs. In our work, we propose the data reduction for bug triage. The paper which we present gives the approach for data reduction for Dynamic Bug Triage System to reduce the data sets scales and improve the quality of bug reports. Instance and feature selection are techniques that we have used for purpose to reduce the redundancy and noise in the bug data sets. The algorithms used in this paper are Support Vector Machine algorithm (SVM) and k nearest neighbor (k-NN) algorithm. These both Support Vector Machine and k-NN algorithms are used for classification of the bug data. The Support Vector Machine algorithm is used as classification algorithm for the given bug query by the user. The k-NN algorithm is used as comparison algorithm, to compare and match the query present in the bug repository. In our dynamic bug triage system, the use of four different modules is done. Those are project analyzer, bug repository maintenance, bug report analysis and task scheduling & notification.

This paper, constructs a predictive model that is used for determining the order of reduction of a new bug data set based on historical bug data sets. The statistic values of bug data sets, like the count of the words or the length of the bug reports, etc. are the attributes used in this model. We extract more accurate and detailed attributes of the dynamic bug triage system in future work. Our work is an ideal resolution to the prediction of reduction orders and can be viewed as a step towards the automatic prediction. In this, we do the extraction of all attributes of a bug data set and reporting of the bug data set is also considered in certain days. Therefore, process of the extraction of attributes from the bug datasets can be applied to the applications present in real world.

CONCLUSION AND FUTURE SCOPE

Bug triage in software maintenance is an expensive step in both labor cost and time cost. We combine feature selection with instance selection in this paper to reduce the scale of bug data sets as well as improve the data quality. The extraction of attributes of each bug data set is done and a predictive model is prepared based on historical Data sets to determine the order of applying instance selection and feature Selection for a new bug data set. The data reduction for bug triage in bug repositories is investigated of two large open source projects, namely Eclipse and Mozilla. We have presented dynamic approach to reduce bug data set to form reduced and high-quality bug data in software development and maintenance. We plan on improving the results of data Reduction in our future work in bug triage to explore how to prepare a high quality Bug data set and how to tackle a domain-specific software Task is presented. For predicting reduction orders, additional functions need to be added to find out the potential relationship between the attributes of bug data sets and the reduction orders.

ACKNOWLEDGMENT

We thank our internal guide Mr. S.B.Ingle and Nelsoft Technologies for their valuable guidance and support in simulation and implementation of project idea.

REFERENCES

[1] Towards Effective Bug Triage with Software Data Reduction Techniques., Jifeng Xuan, He Jiang, Member, IEEE, Yan Hu, Zhilei Ren, Weiqin Zou, Zhongxuan Luo, and Xindong Wu, Fellow, IEEE 2015.

[2] S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D. Ernst, "Finding bugs in web applications using dynamic test generationand explicit-state model checking," IEEE Softw., vol. 36, no. 4, pp. 474–494, Jul./Aug. 2010.

[3] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," ACM Trans. Soft. Eng. Methodol., vol. 20, no. 3, article 10, Aug. 2011.

[4] C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.

[5] K. Balog, L. Azzopardi, and M. de Rijke, "Formal models for expert finding in enterprise corpora," in Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval, Aug. 2006, pp. 43–50.

[6] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm," IEEE Trans. Knowl. Data Eng., vol. 24, no. 6, pp. 1146–1150, Jun. 2012.

[7] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," Data Mining Knowl. Discovery, vol. 6, no. 2, pp. 153–172, Apr. 2002.

[8] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: Improving cooperation between developers and users," in Proc. ACM Conf. Comput. Supported Cooperative Work, Feb. 2010, pp. 301–310.

[9] Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE.