International Journal of Applied Sciences, Engineering and Management
ISSN 2320 – 3439, Vol. 04, No. 01, January 2015, pp. 41 – 45

Towards Effective Bug Triage with Software Data Reduction Techniques

G. S. Sankara Rao[1], Ch. Srinivasa Rao[2]

[1]M. Tech Student, Dept of CSE, Amrita Sai Institute of Science and Technology, Paritala, Krishna-521180.
[2]Assistant Professor, Dept of CSE, Amrita Sai Institute of Science and Technology, Paritala, Krishna-521180.

ABSTRACT: Software companies use over 45 percent of cost in dealing with software bugs. An inevitable step of fixing bugs is bug triage, which objective is to correctly assign a developer to a new bug. To shrink the time cost in manual work, text classification techniques are applied to conduct automatic bug triage. In this paper, we tackle the problem of data reduction for bug triage, i.e., how to diminish the scale and improve the quality of bug data. We combine instance selection with feature selection to simultaneously diminish data scale on the bug dimension and the word dimension. To decide the order of applying instance selection and feature selection, we take out attributes from historical bug data sets and build a predictive model for a new bug data set. We empirically investigate the performance of data reduction on totally 600,000 bug reports of two great open source projects, namely Eclipse and Mozilla. The results explain that our data reduction can effectively diminish the data scale and improve the accuracy of bug triage. Our work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance.

Index Terms*: Mining software repositories, application of data preprocessing, data management in bug repositories, bug data reduction, feature selection, instance selection, bug triage, prediction for reduction orders.*

INTRODUCTION

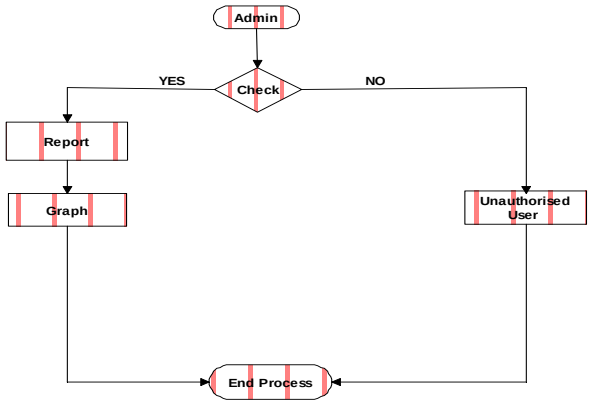
Mining software repositories is an interdisciplinary domain, which Objective is to employ data mining to deal with software engineering problems. In modern software development, software repositories are large-scale databases for storing the output of software development, e.g., source code, bugs, emails, and specifications. Traditional software analysis is not completely suitable for the large-scale and complex data in software repositories. Data mining has emerged as a promising means to handle software data. By leveraging data mining techniques, mining software repositories can find out interesting information in software repositories and resolve real world software problems. A bug repository (a typical software repository, for storing details of bugs), plays an important role in managing software bugs. Software bugs are inevitable and fixing bugs is costly in software development. Software companies use over 45 percent of cost in fixing bugs. Large software projects deploy bug repositories (also called bug tracking systems) to sustain information collection and to aid developers to handle bugs. In a bug repository, a bug is maintained as a bug report, which records the textual description of reproducing the bug and updates according to the status of bug fixing. A bug repository provides a data platform to support many types of tasks on bugs, e.g., fault prediction, bug localization, and reopened bug analysis. In this paper, bug reports in a bug repository are called bug data.

The primary contributions of this paper are as follows

1) We present the problem of data reduction for bug triage. This problem Objective is to augment the data set of bug triage in two aspects, namely a) to simultaneously reduce the scales of the bug dimension and the word dimension and b) to improve the accuracy of bug triage.2) We propose a combination approach to addressing the problem of data reduction. This can be viewed as an application of instance selection and feature selection in bug repositories. 3) We make a binary classifier to foretell the order of applying instance selection and feature selection. To our knowledge, the order of applying instance selection and feature selection has not been investigated in related domains.

SYSTEM ARCHITECTURE

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



SYSTEM ANALYSIS

Existing system: A time-consuming step of handling software bugs is bug triage, which Objective is to assign a correct developer to fix a new bug. In conventional software development, new bugs are manually triaged by an expert developer, i.e., a human triage. Due to the large number of daily bugs and the lack of expertise of all the bugs, manual bug triage is costly in time cost and low in accuracy. In manual bug triage in Eclipse, percent of bugs are assigned by mistake while the time cost between opening one bug and its first triaging is 19.3 days on average. To evade the expensive cost of manual bug triage, existing work has anticipated an automatic bug triage approach, which applies text classification techniques to foretell developers for bug reports. In this approach, a bug report is mapped to a document and a related developer is mapped to the label of the document. Then, bug triage is changed into a problem of text classification and is automatically resolved with mature text classification techniques, e.g., Naive Bayes. Based on the results of text classification, a human triage assigns new bugs by incorporating his/her expertise. To improve the accuracy of text classification techniques for bug triage, some further techniques are investigated, e.g., a tossing graph approach and a collaborative filtering approach. though large-scale and low-quality bug data in bug repositories block the techniques of automatic bug triage. .Since software bug data are a kind of free-form text data (generated by developers), it is required to generate well-processed bug data to make easy the application In this paper, we address the problem of data reduction for bug triage, i.e., how to diminish the bug data to save the labor cost of developers and improve the quality to facilitate the process of bug triage. Data reduction for bug triage aims to construct a small-scale and high-quality set of bug data by removing bug reports and words, which are redundant or non-informative. In our work, we join existing techniques of instance selection and feature selection to simultaneously reduce the bug dimension and the word dimension. The reduced bug data contain fewer bug reports and fewer words than the original bug data and provide similar information over the original bug data. We evaluate the reduced bug data according to two criteria: the scale of a data set and the accuracy of bug triage. To avoid the bias of a single algorithm, we empirically examine the results of four instance selection algorithms and four feature selection algorithm.

Disadvantages: We present the problem of data reduction for bug triage. This problem aims to augment the data set of bug triage in two aspects, namely. To simultaneously decrease the scales of the bug dimension and the word dimension. and to progress the accuracy of bug triage. We intend a combination approach to addressing the problem of data reduction. This can be viewed as an application of instance selection and feature selection in bug repositories. We build a binary classifier to predict the order of applying instance selection and feature selection. To our knowledge, the order of applying instance selection and feature selection has not been investigated in related domains.

Proposed System: In this part, we present the data preparation for applying the bug data reduction. We assess the bug data reduction on bug repositories of two large open source projects, namely Eclipse and Mozilla. Eclipse is a multi-language software development environment, including an Integrated Development Environment (IDE) and an extensible plug-in system; Mozilla is an Internet application suite, including some classic products, such as the Firefox browser and the Thunderbird email client. Up to December 31, 2011, 366,443 bug reports over 10 years have been recorded to Eclipse while 643,615 bug reports over 12 years have been recorded to Mozilla. In our work, we gather continuous 300,000 bug reports for each project of Eclipse and Mozilla, i.e., bugs 1-300000 in Eclipse and bugs 300001600000 in Mozilla. Actually, 298,785 bug reports in Eclipse and 281,180 bug reports in Mozilla are unruffled since some of bug reports are removed from bug repositories (e.g., bug 5315 in Eclipse) or not allowed anonymous access (e.g., bug 40020 in Mozilla). For each bug report, we download web pages from bug repositories and extract the details of bug reports for experiments. Since bug triage aims to forecast the developers who can fix the bugs, we follow the existing work to eliminate unfixed bug reports, e.g., the new bug reports or will-not-fix bug reports. Thus, we only wish bug reports, which are fixed and duplicate (based on the items status of bug reports). in addition in bug repositories, several developers have only fixed very few bugs. Such inactive developers may not provide sufficient information for fire casting correct developers. In our work, we remove the developers, who have fixed less than 10 bugs.

Word Dimension: We use feature selection to eliminate noisy duplicate words in a data set

Bug dimension: Instance selection can remove uninformative bug reports; meanwhile, we can observe that the accuracy may be decreased by removing bug reports. **Word dimension:** By eliminating the uninformative words, feature selection improves the accuracy of bug triage. This can recover the accuracy loss by instance selection.

RELATED WORK: Our work is related to works on Towards Effective Bug Triage with Software Data Reduction.

BACKGROUND AND MOTIVATION

Background:

Bug repositories are widely used for maintaining software bugs, e.g., a popular and open source bug repository, Bugzilla. Once a software bug is found, a reporter (typically a developer, a tester, or an end user) records this bug to the bug repository. A recorded bug is called a bug report, which has multiple items for detailing the information producing The bug. In Fig. 1, we show a part of bug report for bug 284541 in Eclipse.2 In a bug report, the summary and the description are two key items about the information of the bug, which are recorded in natural languages. As their names suggest, the summary denotes a general statement for identifying a bug

while the description gives the details for reproducing the bug. Some other items are recorded in a bug report for facilitating the identification of the bug, such

Product Web Tools Platform (WTP)

DATA REDUCTION FOR BUG TRIAGE:

Applying Instance Selection and Feature

Selection

In bug triage, a bug data set is converted into a text matrix with two dimensions, namely the bug dimension and the word dimension. In our work, we leverage the combination of instance selection and feature selection to generate a reduced bug data set. We replace the original data set with the reduced data set for bug triage. Instance selection and feature selection are widely used techniques in data processing. For a given data set in a certain application, instance selection is to obtain a subset of relevant instances (i.e., bug reports in bug data) while feature selection aims to obtain a subset of relevant features (i.e., words in bug data) . In our work, we employ the combination of instance selection and feature selection. To distinguish the orders of applying instance selection and feature selection, we give the following denotation. Given an instance selection algorithm IS and a feature selection algorithm FS, we use FS!IS to denote the bug data reduction, which first applies FS and then IS; on the other hand, IS!FS denotes first applying IS and then FS.

Benefit of Data Reduction

In our work, to save the labor cost of developers, the data reduction for bug triage has two goals, 1) reducing the data scale and 2) improving the accuracy of bug triage. In contras to modeling the textual content of bug reports in existing work , we aim to augment the data set to build a preprocessing approach, which can be applied before an existing bug triage approach. We explain the two goals of data reduction as follows.

Reducing the Data Scale

We reduce scales of data sets to save the labor cost of developers. Bug dimension. the aim of bug triage is to assign developers for bug fixing. Once a developer is assigned to a new bug report, the developer can examine historically fixed bugs to form a solution to the current bug report . For example, historical bugs are checked to detect whether the new bug is the duplicate of an existing one moreover, existing solutions to bugs can be searched and applied to the new bug . Thus, we consider reducing duplicate and noisy bug reports to decrease the number of historical bugs. In practice, the labor cost of developers (i.e., the cost of examining historical bugs) can be saved by decreasing the number of bugs based on instance selection. Word dimension. We use feature selection to remove noisy or duplicate words in a data set. Based on feature selection, the reduced data set can be handled more easily by automatic techniques (e.g., bug triage approaches) than the original data set. Besides bug triage, the reduced data set can be further used for other software tasks after bug triage(e.g., severity identification, time prediction, and reopened bug analysis).

PREDICTION FOR REDUCTION ORDERS:

an instance selection algorithm IS and a feature selection algorithm FS, FS ! IS and IS !FS are viewed as two orders for applying reducing techniques.

Hence, a challenge is how to determine the order of reduction techniques, i.e., how to choose one between FS! IS and IS ! FS. We refer to this problem as the prediction for reduction orders.

Reduction Orders:

To apply the data reduction to each new bug data set, we need to check the accuracy of both two orders (FS ! IS andIS!FS) and choose a better one. To avoid the time cost of manually checking both reduction orders, we consider predictingthe reduction order for a new bug data set based on historical data sets. As shown in Fig. 2c, we convert the problem of prediction for reduction orders into a binary classification problem. A bug data set is mapped to an instance and the associated reduction order (either FS ! IS or IS ! FS) is mapped to the label of a class of instances. Fig. 3 summarizes the steps of predicting reduction orders for bug triage. Note that a classifier can be trained only once when facing many new bug data sets. That is, training such a classifier once can predict the reduction orders for all the new data sets without checking both reduction orders. To date, the problem of predicting reduction orders of applying feature selection and instance selection has not been investigated in other application scenarios.

Attributes for a Bug Data Set: To build a binary classifier to predict reduction orders, we extract 18 attributes to describe each bug data set. Such attributes can be extracted

before new bugs are triaged. We divide these 18 attributes into two categories, namely the bug report category (B1 to B10) and the developer category (D1 to D8).In Table 2, we present an overview of all the attributes of a bug data set. Given a bug data set, all these attributes are extracted to measure the characteristics of the bug data set. Among the attributes in Table 2, four attributes are directly counted from a bug data set, i.e., B1, B2, D1, and D4; six attributes are calculated based on the words in the bug data set, i.e., B3, B4, D2, D3, D5, and D6; five attributes are calculated as the entropy of an enumeration value to indicate the distributions of items in bug reports, i.e., B6, B7, B8, B9, and B10; three attributes are calculated according to the further statistics, i.e., B5, D7, and D8. All the 18 in Table 2 can be obtained by direct extraction or automatic calculation. Details of calculating these attributes.

Experiments on Bug Data Reduction

Data Sets and Evaluation: We examine the results of bug data reduction on bug repositories of two projects, Eclipse and Mozilla. For each project, we evaluate results on five data sets and each data set is over 10,000 bug reports, which are fixed or duplicate bug reports. We check bug reports in the two projects and find out that 45.44 percent of bug reports in Eclipse and 28.23 percent of bug reports in Mozilla are fixed or duplicate. Thus, to obtain over 10,000 fixed or duplicate bug reports, each data set in Eclipse is collected from continuous 20,000 bug reports while each bug set in Mozilla is collected from continuous 40,000 bug reports. Table 3 lists the details of ten data sets after data preparation. To examine the results of data reduction, we employ four instance selection algorithms (ICF, LVQ, DROP, and POP), four feature selection algorithms (IG, CH, SU, RF), and three bug triage algorithms (Support Vector Machine, SVM; K-Nearest Neighbor, KNN; and Naïve Bayes).

IMPLEMENTATION

Instance Selection: Instance selection and feature selection are widely used techniques in data processing. For a given data set in a certain application, instance selection is to obtain a subset of relevant instances (i.e., bug reports in bug data) while feature selection aims to obtain a subset of relevant features (i.e., words in bug data). In our work, we employ the combination of instance selection and feature selection. To

Data Reduction: In our work, to save the labor cost of developers, the data reduction for bug triage has two goals. reducing the data scale and improving the accuracy of bug triage. In contrast to modeling the textual content of bug reports in existing work, we aim to augment the data set to build a preprocessing approach, which can be applied before an existing bug triage approach. We explain the two goals of data reduction as follows.

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple.

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.
3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.

- ❖ Signal important events, opportunities, problems, or warnings.
- ❖ Trigger an action.
- ❖ Confirm an action.

CONCLUSION

Bug triage is an costly step of software maintenance in both labor cost and time cost. In this paper, we merge feature selection with instance selection to decrease the scale of bug data sets as well as progress the data quality. To find out the order of applying instance selection and feature selection for a new bug data set, we mine attributes of each bug data set and train a predictive model based on historical data sets. We empirically investigate the data reduction for bug triage in bug repositories of two large open source projects, namely Eclipse and Mozilla. Our work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance. In future work, we plan on improving the results of data reduction in bug triage to discover how to prepare a high quality bug data set and tackle a domain specific software task. For predicting reduction orders, we plan to pay efforts to determine the potential relationship between the attributes of bug data sets and the reduction orders.

REFERENCE

[1] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in Proc. 28th Int. Conf. Softw. Eng., May 2006, pp. 361–370.

[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 generation and 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] Bugzilla, (2014). [Online]. Avaialble: http://bugzilla.org/

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] V. Bolon-Canedo, N. S anchez-Marono, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," Knowl. Inform. Syst., vol. 34, no. 3, pp. 483–519, 2013.