ISSN:2320-0790

COMPUSOFT
An International Journal of Advanced Computer Technology

A Survey-Vulnerability Classification of Bug Reports using Multiple Machine Learning Approach

Krishna A Patel, Prof. Rohan C Prajapati
Ipcowala Institute of Engineering & Technology Dharmaj, Anand, Gujarat, India-388430

Abstract – As critical and sensitive systems increasingly rely on complex software systems, identifying software vulnerabilities is becoming increasingly important. It has been suggested in previous work that some bugs are only identified as vulnerabilities long after the bug has been made public. These bugs are known as Hidden Impact Bugs (HIBs). This paper presents a hidden impact bug identification methodology by means of text mining bug databases. The presented methodology utilizes the textual description of the bug report for extracting textual information. The text mining process extracts syntactical information of the bug reports and compresses the information for easier manipulation and divided into frequency base and context base bug then give bug ranking.

Keywords: Naïve Bayes, classification, bug database mining, text mining

I. INTRODUCTION

In data mining, high quality of data are a valuable asset. This also applies to empirical software engineering as well. Since now, mining data from changes and bug databases had become common. As bug database is built from bug reports, quality of bug reports are crucial to data quality. Correctly classified bug reports will greatly help in both research validity and modeling performance. More detail bug report will also contain more information which could help in understanding data.

In that paper presents a software vulnerability identification methodology using HIBs, that utilizes the textual description of the bugs that were reported to publically available bug databases. The presented methodology utilizes text mining techniques to 1) extract syntactical information of bug reports, 2) compares the information for easier manipulation, and 3) use this information to generate a feature vector which is used for classification. Thus, the presented system is intended to classify bugs as potential vulnerabilities as they are being reported to bug databases, thereby reducing the time software is exposed to attack through the vulnerability.

II. BECKGROUND

A. Bug Classification Methodology

Bug databases are used by software developers to identify and keep track of information about software bugs that were not identified at the time of software release. Developers will utilize these bug reports for different purposes such as improving reliability and improving future requirements [14][15]. Publically available bug databases enable users to report bugs as they encounter it and search the bug database for bugs they might encounter in the future [16]. Bug databases also keep track of the fixes being released for different bugs and what stage of the resolution process a bug is in. Because different entities with different levels of expertise and requirements report bugs to these databases, the information contained in bug reports is highly noisy and not in standard form [17][18]. However, this information has been successfully used for various classification purpose [17][18][19].

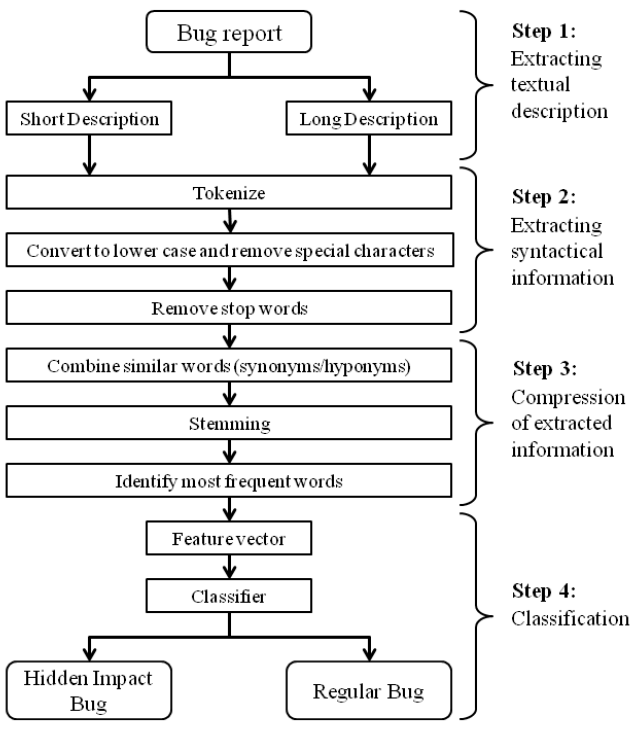
Bug report
Short Description
Long Description
Step 1: Extracting textual description
Tokenize
Convert to lower case and remove special characters
Remove stop words
Step 2: Extracting syntactical information
Combine similar words (synonyms/hyponyms)
Stemming
Identify most frequent words
Step 3: Compression of extracted information
Feature vector
Classifier
Step 4: Classification
Hidden Impact Bug
Regular Bug

Fig.1. Identification methodology of vulnerabilities using bug report[1]

In Step 1 the short and long descriptions of the bug report is extracted. The short description is a title provided by the reporter that is around 5-10 words in length. The long description is a more detailed description of the bug which may include how to recreate it, code snippets, memory dumps, etc.

In step 2, the most important and recurring syntactical information is extracted from the short and long description of the bugs. The syntactical information is extracted in the form of single unique words known as keywords. The extraction process removes words and symbols that might not carry a significant amount of information, and only extracts single words.

In step 3, compression of this extracted information is performed. Text mining techniques are used in this step to identify words that may carry similar information and combine them. This step reduces the feature space which decreases the resource utilization of the process. This step also counts the number of bugs each keyword has appeared in and identifies the most frequently used keywords in the bug descriptions.

In step 4, extracted set of keywords are used to create a feature space for the bug descriptions. Each dimension of the feature space consists of a set of words that carry similar information. This feature vector can be used by a classifier to perform the final classification. Classifier will classify a given bug as a potential HIB or a regular bug.

B. Flow of work

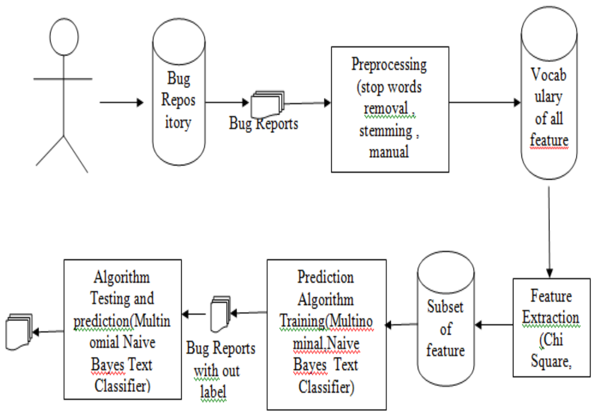
Bug Repository
Bug Reports
Preprocessing (stop words removal , stemming , manual
Vocabulary of all feature
Feature Extraction (Chi Square,
Subset of feature
Prediction Algorithm Training(Multinominal Naive Bayes Text Classifier)
Bug Reports with out label
Algorithm Testing and prediction(Multinomial Naive Bayes Text Classifier)

Fig. 2. Flow chart [8]

1 Input Data

Eclipse and Mozilla firefox data is obtained from bugzill – an open bug repository [20][21]. Dataset of almost 29,000 record set is obtained. This data is divided into training and testing groups and experiments are performed on different set of data from these groups.

1.2 Model for prediction

When the bug is first reported to repository, it is submitted to our proposed system as shown in Fig. System extracts all the terms in these reports using bag of words approach. The vocabulary is that of extremely high dimensionality and thus numbers of features are reduced by using chisquare algorithm. These features are used for training of classification algorithm which is then used for classification of bug reports. The classification algorithm used in proposed system is multinomial Naïve Bayes.

1.2.1 Pre-processing

Data pre-processing is the most important step of data mining. Data obtained from bug repositories in raw form and cannot be directly used for training the classification algorithm. The data is first pre-processed to make it useful for training purpose. Data pre-processing is the major time consuming step of data mining and most important as well. Stop-words dictionary and regular expression rules are used to filter useless words and filter the punctuations respectively. Porter stemming algorithm is used to stem the vocabulary.

1.2.2 Feature Selection

The vocabulary obtained after applying "bag of words" approach on data has very large dimensionality. Most of these dimensions are not related to text categorization and thus result in reducing the performance of the classifier. To decrease the dimensionality, the process of feature selection is used which takes the best k terms out of the whole vocabulary which contribute to accuracy and efficiency. There are a number of feature selection techniques such as Chi-Square Testing, Information Gain (IG), Term Frequency Inverse Document Frequency (TFIDF), and Document Frequency (DF). In this research, chi-square and TFIDF algorithms are used for feature selection.

1.2.3 Classifier Modeling

Text classification is an automated process of finding some metadata about a document. Text classification is used in various areas like document indexing by suggesting its categories in a content management system, spam filtering, automatically sorting help desk requests etc. Naïve Bayes text classifier is used in this research for bug classification. Naïve Bayes classifier is based on Bayes' theorem with independent assumption and is a probabilistic classifier. INDEPENDENCE means the classifier assumes that any feature of a class is unrelated to the presence or absence of any other feature.

III. CONCLUSION

In this paper, we proposes a method for automatically classify bug reports base on its textual information without the need to do a parameter tuning. We also experiment on how to optimize the bug report classification process that use parametric method to topic modeling bug reports. The experiment are done on vary topic numbers, pre-processing methods and classification technique. The result could serve as aguideline to efficiently employ this bug report classification process. For future work, we plan to tacle lack of data and imbalanced dataset, the problems found in multiclass bug report corpus. We also want to improve the nonparametric method classification method performance. We divided bug report into contaxt base and frequency base. We also added ranking in bug reports.

REFERENCES

[1] M. McQueen, "Software and human vulnerabilities," in *Proc. IEEE. Int.Conf. of the Industrial Electronics Society, (IECON), pp. 1-85, Nov. 2014.*

[2] K. Herzig, S. Just, and A. Zeller, "It's not a bug, it's a feature: how misclassification impacts bug prediction," in Proc. ICSE '13, 2013, pp. 392–401.

[3] Tao Xie and Suresh Thummalapenta, North Carolina State University, David Lo, Singapore Management University, Chao Liu, Microsoft Research ―Data Mining in Software Engineering‖, August, 2012, pp. 55-60

[4] E. Giger, M. Pinzger, and H. Gall, "Predicting the fix time of bugs," in *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering. ACM, 2010, pp. 52–56.*

[5] G. Boetticher, T. Menzies and T. Ostrand. PROMISE Repository of empirical software engineering data. http://promisedata.org/ repository, West Virginia University, Department of Computer Science, 2012.

[6] J. Arnold, T. Abbott, W. Daher, G. Price, N. Elhage, G. Thomas, A. Kaseorg, "Security Impact Ratings Considered Harmful," in *Proc. of the 12th Conf. on Hot Topics in Operating Systems , USENIX, May 2012.*

[7] Tao Xie and Suresh Thummalapenta, North Carolina State University, David Lo, Singapore Management University, Chao Liu, Microsoft Research ―Data Mining in Software Engineering‖, August, 2009, pp. 55-60

[8] ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6245635

[9] J. R. Quinlan, *C4.5: Programs for machine learning. Vol. 1. Morgan kaufmann, 1993.*

[10] The MITRE Corporation (1 Nov. 2011), *Common Vulnerabilities and Exposures (CVE) [Online]. Available: http://cve.mitre.org M.E. Computer Engineering*

[11] A. McCallum, K. Nigam. "A comparison of event models for naive bayes text classification," in *Proc. of AAAI-98 workshop on learning for text categorization, vol. 752, 1998.*

[12] Redhat, Inc. (1 May 2014), *Redhat Bugzilla Main Page* [Online]. Available: https://bugzilla.redhat.com/

[13] D. Wijayasekara, M. Manic, J. L. Wright, M. McQueen "Mining Bug Databases for Unidentified Software Vulnerabilities," in *Proc of the 5th Intl. IEEE Intl. Conf. on Human System Interaction, (HSI)*, June, 2012.

[14] A. J. Ko, B. A. Myers, D. H. Chau, "A Linguistic Analysis of How People Describe Software Problems," in *Proc. of the 2006 IEEE Symp. on Visual Languages and Human-Centric Computing (VL/HCC 2006)*, pp. 127–134, Sep. 2006.

[15] M. F. Ahmed, S. S. Gokhale, "Linux Bugs: Life Cycle and Resolution Analysis," in *Proc of The 8th Int. Conf. on Quality Software (QSIC '08)*, Aug. 2008, pp.396–401.

[16] J. Noll, S. Beecham, D. Seichter, "A Qualitative Study of Open Source Software Development: the OpenEMR Project," in *Proc of the Int. Symp. on Empirical Software Engineering and Measurement (ESEM'11)*, pp. 30–39, Sep. 2011.

[17] A. Lamkanfi, S. Demeyer, E. Giger, B. Goethals, "Predicting the severity of a reported bug," in *Proc. of the 7th IEEE Working Conf. on Mining Software Repositories (MSR 2010)*, pp. 1–10, May 2010.

[18] A. Lamkanfi, S. Demeyer, Q. D. Soetens, T. Verdonck, "Comparing Mining Algorithms for Predicting the Severity of a Reported Bug," in*Proc. of the 15th European Conf. on Software Maintenance and Reengineering (CSMR)*, pp.249–258, Mar. 2011.

[19] P. Bhattacharya, I. Neamtiu, C. R. Shelton, "Automated, highlyaccurate, bug assignment using machine learning and tossing graphs," in *The Journal of Systems and Software*, vol. 85, pp. 2275-2292, 2012.

[20] [online]. Available: https://bugzilla.mozilla.org/.

[21] [online]. Available: https://bugs.eclipse.org/bugs/.