ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 2, February 2017

Efficient and Dynamic Bug Report Prediction Using Mining Techniques

K. Sulopriya[1], P. Rajkumar[2], N. Mohan Prabhu[3]

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.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 2, February 2017

TABLE I. STATISTICS OF COLLECTED DATASETS.

Project	# Bugs	Time	# Confs	# Terms
accumulo	181	2011. 10 - 2013. 06	33	227
activemq	175	2005. 12 - 2007. 12	29	327
camel	1,189	2007. 07 - 2013. 09	333	1,261
flume	279	2010. 07 - 2013. 05	83	341
wicket	1,379	2006. 11 - 2013. 09	46	1,340

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.

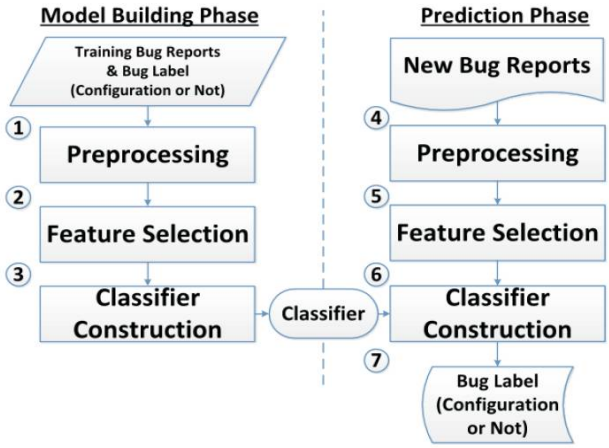

Fig.1 System Architecture Design

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 II. CONFUSION MATRIX.

Classified as	True Class	
	Configuration	Non-configuration
Configuration	TP	FP
Non-configuration	FN	TN

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 2, February 2017

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 p 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 III. ACCURACY, CONFIGURATION F-MEASURE (CONF. F-MEASURE), AND NON-CONFIGURATION F-MEASURE (NON. F-MEASURE) FOR THE 5 DATASETS WITH INFORMATION GAIN AS THE FEATURE SELECTION TECHNIQUE. THE LAST COLUMN SHOWS THE AVERAGE ACCURACY AND F-MEASURE SCORES ACROSS THE 5 DATASETS.

Evaluation	Techniques	accumulo	activemq	camel	flume	wicker	Average
Accuracy	NB	0.823	0.823	0.740	0.681	0.931	0.800
	NBM	0.840	0.823	0.734	0.703	0.954	0.811
	kNN	0.823	0.829	0.728	0.717	0.968	0.813
	SVM	0.829	0.840	0.733	0.728	0.964	**0.819**
	BN	0.779	0.846	0.750	0.692	0.964	0.806
Conf. F-measure	NB	0.500	0.340	0.418	0.341	0.307	0.381
	NBM	0.525	0.392	0.463	0.450	0.422	**0.450**
	kNN	0.238	0.167	0.331	0.288	0.120	0.229
	SVM	0.340	0.125	0.236	0.333	0.074	0.222
	BN	0.167	0.308	0.410	0.000	0.039	0.185
Non. F-measure	NB	0.893	0.898	0.833	0.790	0.964	0.875
	NBM	0.904	0.896	0.823	0.797	0.976	0.880
	kNN	0.900	0.904	0.829	0.823	0.984	0.888
	SVM	0.902	0.912	0.839	0.829	0.982	**0.892**
	BN	0.873	0.913	0.842	0.818	0.982	0.885

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).

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 2, February 2017

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.

TABLE IV. ACCURACY, CONFIGURATION F-MEASURE (CONF. F-MEASURE), AND NON-CONFIGURATION F-MEASURE (NON. F-MEASURE) FOR THE 5 DATASETS WITH CHI-SQUARE AS THE FEATURE SELECTION TECHNIQUE.

Evaluation	Techniques	accumulo	activemq	camel	flume	wicker	Average
Accuracy	NB	0.823	0.823	0.738	0.681	0.930	0.799
	NBM	0.845	0.817	0.738	0.703	0.953	0.811
	kNN	0.823	0.823	0.737	0.717	0.968	0.814
	SVM	0.834	0.834	0.733	0.728	0.964	**0.819**
	BN	0.779	0.846	0.750	0.692	0.964	0.806
Conf. F-measure	NB	0.500	0.340	0.421	0.341	0.304	0.381
	NBM	0.533	0.385	0.467	0.450	0.414	**0.450**
	kNN	0.238	0.114	0.360	0.288	0.120	0.224
	SVM	0.348	0.121	0.240	0.333	0.074	0.223
	BN	0.167	0.308	0.410	0.000	0.039	0.185
Non. F-measure	NB	0.893	0.898	0.831	0.790	0.963	0.875
	NBM	0.907	0.893	0.827	0.796	0.975	0.880
	kNN	0.900	0.902	0.834	0.823	0.984	0.889
	SVM	0.905	0.909	0.838	0.829	0.982	**0.892**
	BN	0.873	0.913	0.842	0.818	0.982	0.806

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

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 2, February 2017

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 predicators of future bugs.

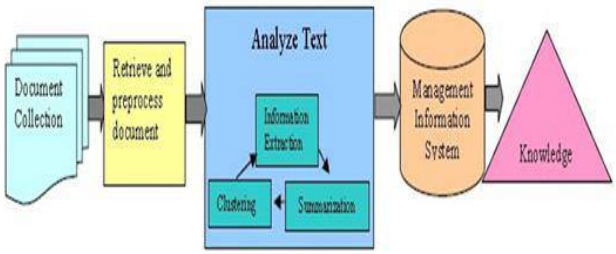

Fig.2. Existing System

The source control repositories, bug reports, design and code artifacts etc. will be utilized as data sources. Open sources 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., naive Bayes, SVM, naive 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 naive 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 naive 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.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 2, February 2017

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 ConfDiagnoser 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 in- formation 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

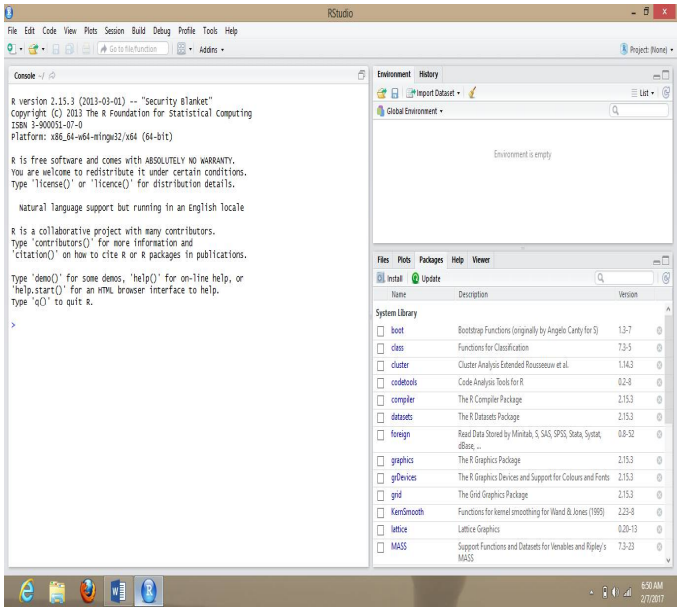

Fig.3. R-Studio Login Page

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 2, February 2017

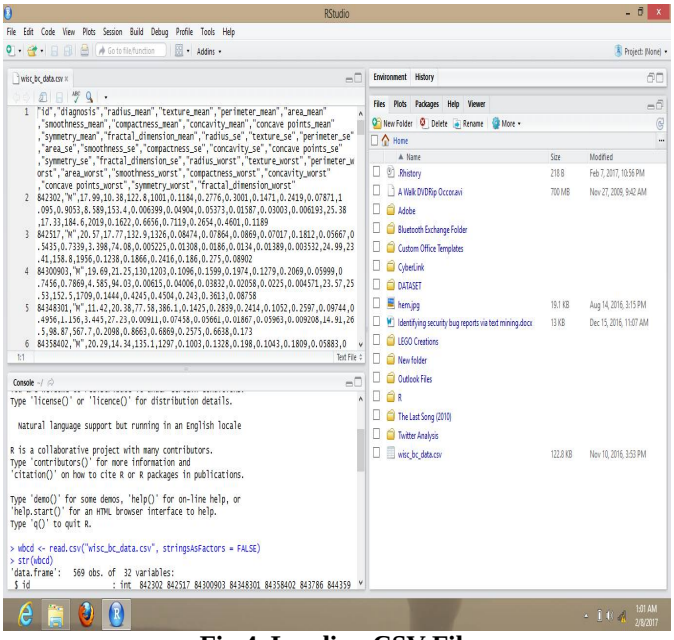

Fig.4. Loading CSV File

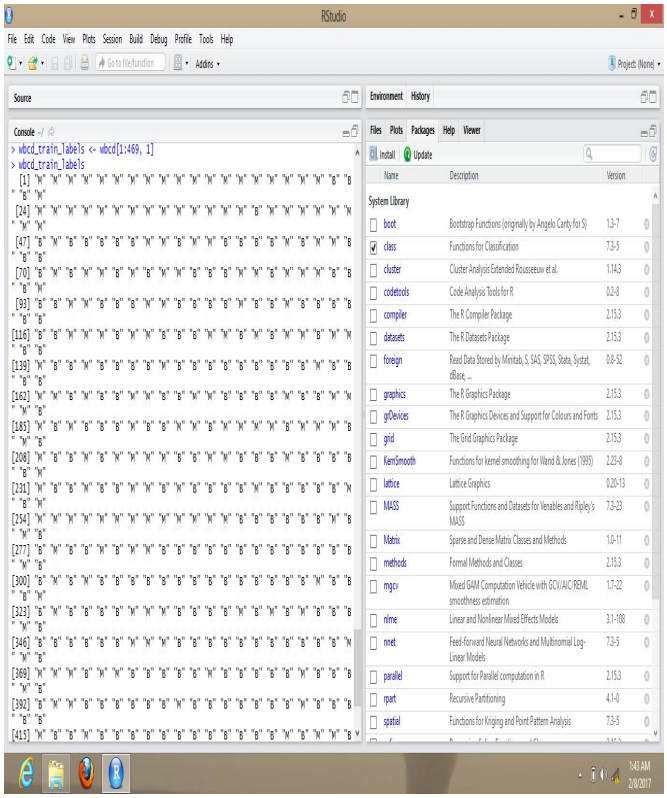

Fig.5. Training Dataset

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 2, February 2017

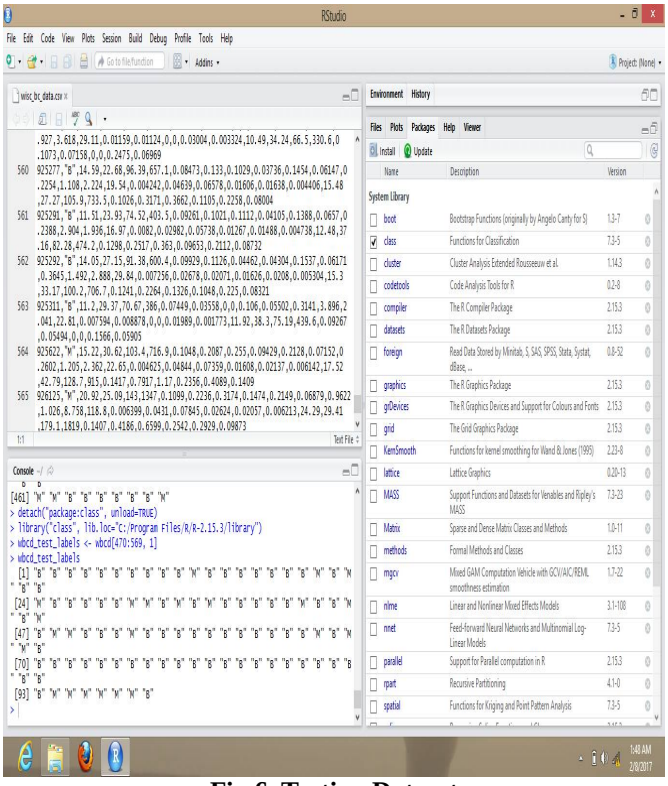

Fig.6. Testing Dataset

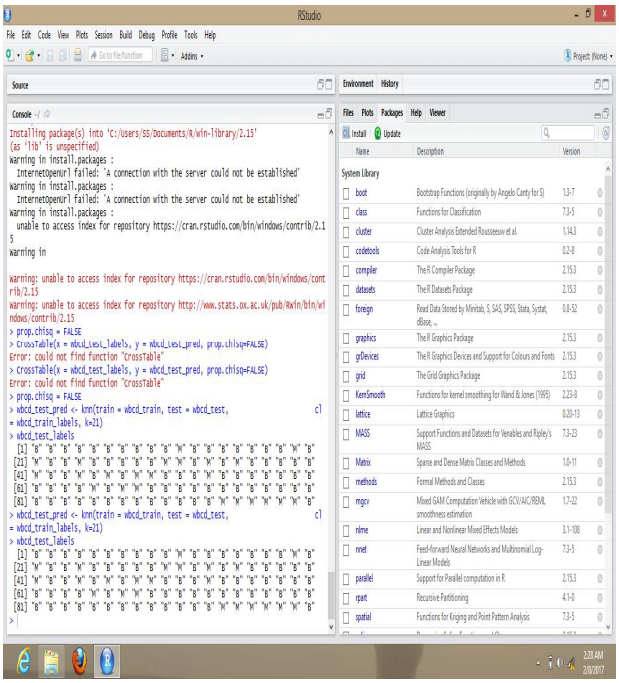

Fig.7. kNN Data Classification

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 2, February 2017

VIII. CONCLUSION AND FUTURE SCOPE

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

[1] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do internet services fail, and what can be done about it?" in USENIX Symposium on Internet Technologies and Systems, vol. 67. Seattle, WA, 2003.
[2] L. A. Barroso and U. H¨olzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," Synthesis Lectures on Computer Architecture, vol. 4, no. 1, pp. 1–108, 2009.
[3] R. Johnson, "More details on todays outage," http://www.facebook.com/notes/facebook-engineering/more-details-on-todays-outage/431441338919, 2010.
[4] Y. Sverdlik., "Microsoft: Misconfigured network device led to azure outage," http://www.datacenterdynamics.com/focus/archive/2012/07/microsoft-misconfigured-network-device-led-azure-outage, 2012.
[5] A. Team, "Summary of the amazon ec2 and amazon rds service disruption in the us east region," http://aws.amazon.com/cn/message/65648/, 2011.
[6] N. Eddy, "Human error caused google glitch," http://www.eweek.com/c/ a/ Enterprise-Applications/Human-Error-Caused-Google-Glitch/ ,2009.
[7] B. Hale, "Why every it practitioner should care about network change and configuration management," http://web.swcdn.net/ creative/pdf/Whitepapers/Why Every IT Practitioner Should Care About NCCM.pdf , 2012.
[8] A. Kapoor, "Web-to-host: Reducing total cost of ownership," Technical Report 200503, The Tolly Group, Tech. Rep., 2000.
[9] Z. Yin, X. Ma, J. Zheng, Y. Zhou, L. N. Bairavasundaram, and S. Pasupathy, "An empirical study on configuration errors in commercial and open source systems," in Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. ACM, 2011, pp. 159–172.
[10] H. J. Wang, J. C. Platt, Y. Chen, R. Zhang, and Y.-M. Wang, "Automatic misconfiguration troubleshooting with peerpressure." in OSDI, vol. 4, 2004, pp. 245–257.
[11] S. Zhang and M. D. Ernst, "Automated diagnosis of software configuration errors," in Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013, pp. 312–321.
[12] T. Xu, J. Zhang, P. Huang, J. Zheng, T. Sheng, D. Yuan, Y. Zhou, and S. Pasupathy, "Do not blame users for misconfigurations," in Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. ACM, 2013, pp. 244–259.
[13] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in ICML, vol. 97, 1997, pp. 412–420.
[14] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip et al., "Top 10 algorithms in data mining," Knowledge and Information Systems, vol. 14, no. 1, pp. 1–37, 2008.
[15] C. C. Aggarwal and C. Zhai, Mining text data. Springer, 2012. \[16] S. Kim, E. J. Whitehead, and Y. Zhang, "Classifying software changes: Clean or buggy?" Software Engineering, IEEE Transactions on, vol. 34, no. 2, pp. 181–196, 2008.
[17] T.-D. B. Le and D. Lo, "Will fault localization work for these failures? an automated approach to predict effectiveness of fault localization tools," in Software Maintenance (ICSM), 2013 29th IEEE International Conference on. IEEE, 2013, pp. 310–319.
[18] X. Xia, D. Lo, X. Wang, X. Yang, S. Li, and J. Sun, "A comparative study of supervised learning algorithms for re-opened bug prediction," in Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on. IEEE, 2013, pp. 331–334.

BIOGRAPHY

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