ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Defect Prediction by Pruning Redundancy in Association Rule Mining

Amarpreet Kaur
Research Scholar, Computer Science & Technology
Central University of Punjab
Bathinda, India

Dr. Satwinder Singh
Assistant Professor, Computer Science & Technology
Central University of Punjab
Bathinda, India

Abstract: Defect prediction is a major problem during software maintenance and evolution. It is important for the software developers to identify defective software modules to improve the software quality. Many organizations want to predict the defects in software systems, before they are deployed, to improve and measure the quality of software. Different researchers proposed various approaches to extract the defect prone modules in the specific software system. This paper focuses on an effective model, called Apriori, which uses the approach of association rule mining. Association rule mining remains a very popular and effective method to extract meaningful information from a large data set. Apriori algorithm is based on the discovery of association rules for predicting whether a software module is defective or not. Different algorithms perform in different manner on distinct datasets. This paper analyzes the shortcomings of Apriori algorithm and studies the improvement strategies to improve the performance of Apriori algorithm by removing the redundancy of rules generated on the basis of different parameters. In this paper, we use a new method to find the best 'n' association rules out of pool of 'k' association rules based on heuristic analysis. This study will help improve the existing software defect prediction models in terms of precision, performance and other aspects.

Keywords: Defect Prediction, Data Mining, Association Rule Mining, Apriori

I. INTRODUCTION

Software quality plays a vital role in the software industry. However, it is very expensive to build software of optimum quality. Thus, to enhance the efficiency and usefulness of quality assurance and testing, there should be an efficient model for tracking the defects. As there are many techniques to tackle the defects, but it will be more efficient to predict them in advance so as to reduce the testing effort and improve the quality of the software. To upgrade the quality of software system, defect-prone modules are identified using Software defect prediction. [1] In addition to this, defect prediction is an important issue during software maintenance and evolution. A number of organizations desire to reveal the defects in advance in software systems to improve and measure the quality and maintenance effort, before they are deployed.

Most of the models utilized for defect prediction consider size and complexity metrics. But, some are based on testing data or take an approach based on more than one variable. These models are not reliable as they are unable to process with the relationships between defects and failures which are unknown. Most of the prediction models tend to model only part of the underlying problem. But adjoining approach builds the association rules on the attributes of the software which are taken as item sets.

ASSOCIATION RULE MINING

There are numerous mining techniques for instance clustering, classification, and association rule mining to extract meaningful information from large datasets. This study focuses on association rule mining, which produces interesting relations between variables in the dataset in the form of association rules. To create such association rules, frequent patterns must be generated first. Furthermore, these frequent pattern mining forms the crux of any association rule mining process.

Association rule mining means searching the attribute-value conditions that frequently occur together in a dataset [2]. It tries to find possible associations between item sets in large datasets. An approach to apply association rule mining is proposed by using Apriori algorithm to predict the defects in the specific software system. We applied the Apriori on open source datasets to perform the experimental evaluation of the proposed model. [3]Ordinal association rules specify ordinal relationships between record attributes for a given set of records described by a set of attributes that hold for a specific sample of the records. But attributes with different domains and relationships between them, exist in real world data sets[4]. In such situations, sequential association rules do not have enough strength to describe data regularities. Consequently, association rules were introduced in order to be able to capture various kinds of relationships between record attributes[5]. Relational association rule mining has not been applied so far for predicting if a software entity is defective or not. We therefore aim in this paper at methods based on relation association rules, whose effectiveness is studied through the experimental results. Association rule mining exposed multiple applications in distinct areas such as Crime Detection/Prevention, Cyber Security and crowd mining (i.e. Market basket analysis to improve the sales).

II. LITERATURE REVIEW

To manage a large scale software development, quantitative models are required. To assess the complexity software engineers provided us with few techniques. To form the multivariate models, the main approaches used so far are multivariate regression analysis and classification trees. L. C. Briand, Basili, & Thomas[6] studied these techniques and their flaws and proposed a new better approach called pattern recognition. The main problem with classification trees was that they select most relevant variable assuming that variable

© 2015-19, IJARCS All Rights Reserved

to be equally relevant regardless of its value (i.e. high/low), whereas Regression techniques use the dummy variables to deal with discrete variables. Although regression techniques can be very useful and effective but interpretation of the models based on regression equations is complex task [6]. They proposed a model called OSR (Optimized set Reduction) which uses pattern recognition technique to combine the expressiveness of classification tress with a statistical approach. To assess the effectiveness of OSR, it was applied to estimate productivity for each project based upon COCOMO cost drivers. OSR is useful for extracting meaningful patterns from the datasets for prediction, risk management, and quality evaluation. [7] A tool supporting this approach was developed during TAME project in University of Maryland, and it was investigated to solve several software engineering problems such as project cost estimation. The software can be very complex, so applying testing efforts on all the software components is not an efficient especially when resources available are limited. In such situation, one needs to identify the components with high/low fault frequency. Hence to improve the reliability of the system, researchers enhanced optimized set reduction approach which they proposed earlier, logistic regression and classification trees [8]. These methods were implemented on 146 components of an ADA system which consists of 260 KLOC (kilo lines of code). OSR implemented with TAME project chose correctness and completeness performance metrics to evaluate their prediction models. The best performance among was achieved with OSR technique in this study with correctness and completeness of above 90%. OSR patterns seem to be more stable and interpretable.

Data reduction property of OSR is the main strength but sometimes the important information is lost. It is found that highly structures [1]Pattern recognition tasks require architectures with many parameters to make loading of weights effective. Moreover, it takes more execution time on large datasets. In order to solve the problem of working on large databases [9] proposed two new association rule mining algorithms, Apriori and AprioriTid, to generate candidate itemsets. AprioriTid has the property that database is not used for counting support after the first pass. In this research paper the relative performance of the algorithms was assessed. To assess the relative performance, experiments were performed on IBM RS/6000 530H workstation. The data was stored in 2GB SCSI 3.5" drive. Algorithms like AIS and SETM were used to compare the performance of different rule mining algorithms. Apriori proposed earlier takes a lot of space and response time due to its exponential complexity. In order to reduce its complexity [10] proposed an improved Apriori Algorithm based on the matrix. This paper presents mathematical formula for selecting the cluster and an improvement by using parallel algorithm. It compares the time consumed by original Apriori with the improvised Apriori. Results show that proposed approach reduced the memory space and time consumption by 63.17%. As the massive amount of data increasing day by day is far beyond the approach of a single machine. This paper proposed a new faster and efficient algorithm based on Apriori, called R-Apriori which is a parallel Apriori implementation which reduces the computational complexity of distributed Apriori and hence improves the performance [11]. It was an enhanced version of YAFIM (yet another Frequent Item Mining), which is an implementation on spark, implemented on five large datasets: T1014D100 K (artificial datasets generated by IBM's data generator), Retail dataset (for market basket analysis), Kosarak dataset (donated by Ferenc Bodon), BMS Web view (used for KDD cup 2000), T2510D10K (synthetic dataset generated by random transaction database)

R-Apriori outperforms classical Apriori on spark platform for various datasets. There are few many more models for frequent data mining .These can be used for defect prediction. However, performance of different techniques varies. To compare the performance [12] implemented four models for static defect prediction as Naive Bayes, Neural Networks, Association rules and Decision Tree for extracting software defect models. The performance of all four approaches was compared by accuracy, precision, F-measure and classification metrics. The comparative studies show that the correctness of all four algorithms is nearly equal but Naive Bayes gives the best performance. However, Association rules approach has the highest precision among all four approaches. But the authors have applied these techniques on specific data not generalized it for all the software datasets[13]. So it can be a good future work to test the validation of results on various data sets.

Although many models have been proposed in the software defect prediction literature, researchers are still focusing on developing more accurate defect predictors using association rules.[14] Recent results show that researchers should concentrate on improving the quality of the data to overcome the limits of the existing software prediction models. For this purpose [15] introduced relational association rules which are capable of discovering various kinds of associations and correlation between data in large datasets. Relational association rules are an extension to the association rules. A software module can be characterized by set of software metrics which may decide whether or not a module is defective. [16] Proposed a new algorithm called DOAR (Discovery of Ordinal Association Rules) that efficiently finds all ordinal association rules of any length that hold over a dataset. This algorithm identifies the ordinal association rules using an iterative process that consists in-length-level generation of candidate rules, then the verification of candidates for minimum support and confidence compliance is done. DOAR performs multiple passes over the dataset. It provides two functionalities:

- All interesting association rules of any length
- Finds all maximal interesting relational association rules of any length.

Four software metrics were considered to predict the default by implementing DPRAR; a supervised method for detecting software entities with defects, based on relational association rule mining. Experiments were performed on six different datasets including MW1 dataset which contains data about a zero gravity experiment related to combustion, The JM1 dataset which contains data about a real-time predictive ground system, The PC1 dataset built for functions from a flight software for earth orbiting satellite. Considering accuracy, [17]DPRAR classifier outperforms the other classifiers on six of the datasets. It can be extended to identify and consider different types of relations between the software metrics. Also, to investigate how the length of the rules and the confidence of the relational association rules discovered in the training data may influence the accuracy of the classification task. For future use, it can be combined with other machine learning based predictive models to hybridize classification model. DPRAR performed better for 45 evaluation measures and worse in 23 out of 69 measures.

© 2015-19, IJARCS All Rights Reserved

Different association rule mining algorithms can be applied in software engineering domain, but the question here is which of them will work efficiently. The most commonly used algorithms are Apriori and Frequent Pattern Tree Algorithm. To analyze the effectiveness of these algorithms [18] conducted a research to compare the efficiency on different types of datasets. It considered Software risk factors and software risk mitigation factors to analyze the effectiveness. The research used the methodology of analyzing and comparing the two algorithms in terms of tracing the software risk and software risk mitigation factors and introduced the new adaptive structure. [19] The comparative study concludes that the FP tree algorithm works efficiently with large data sets whereas Apriori works efficiently on small databases. [20]in single machine environment, Apriori and FP- Growth algorithms are not much efficient for large-scale data association rule mining. They suffer from the problems of low consumption performance and poor reliability. Also, they predict the fault prone modules but not the defects which occur in those modules. To implement it in more efficient manner [21] focused on prediction of fault-prone and modules as well as identifying types of defects that occur in fault-prone modules. Karthik et Al. Used clustering rules to classify the defaults based on their level of complexity into Simple, Moderate and Complex categories. This proposed model helped to identify a maximum number of defects that is associated to a given defect. It improved the accuracy by using this methodology and results in lower rate of false negatives.

III. RESEARCH METHODOLOGY

Apriori, Éclat, and FP-Growth are some of the widely used algorithms for association rule mining proposed in recent researches. Most of these algorithms scan the dataset sequentially to generate frequent patterns, which is one of the most popular data mining techniques. These frequent patterns can later be used for rule generation. Apriori uses this approach as explained below.

The Apriori algorithm was proposed by R. Agrawal et al. [8]. It works on the basic observation that an itemset is frequent only if all its non-empty subsets are also frequent. It works on an iterative approach in which frequent item sets are found in the current iteration using the results of the previous iterations. First, it finds singleton frequent items where an item occurring more times than a specified minimum threshold,[22] which is termed as support count, is termed frequent. Following this approach, *K*-frequent itemsets are found using *K-1* frequent itemsets. After finding singleton frequent items, a candidate set is generated. The candidate set for the *kth* iteration has all combinations of items having size K whose all possible subsets are present in *K-1* frequent itemsets[23]. These frequent itemsets generated in *kth* iteration are used for the generation of candidates in next iteration. It keeps iterating until all frequent patterns are found. Many researchers have applied neural networks approach for defect predictions to improve quality. Three-layer neural network with sigmoid function was used by training the neural network based on historical data[24]. As discussed earlier, classic Apriori lacks in utilizing time and memory[25]. Moreover, the rules generated by existing algorithms are not efficient because these approaches consider all the strong rules to make the predictions.[26] But some of those rules are redundant. A new approach to weed out the weak rules was suggested by Kumari, D. et. Al [27]on the basis of three parameters: support, confidence and correlation. Further, to analyze the quality of association rules, eight interestingness measures were used. These interesting measures can be calculated statistically within resulting rule set generated[28]. To remove the redundancy of those rules, a new approach will be suggested by pruning them on the basis of specific measures. After generating the rules validation of results is required to evaluate the proposed method. Hence, this approach will work as shown in figure 1.

A. Apriori Algorithm

The process of the algorithm is as follows.

Step1. Set the minimum values for support and confidence.

Step2. Construct the candidate 1-itemsets; generate the frequent 1-itemsets by pruning some candidate 1-itemsets if they do not meet the requirement of minimum support.

Step3. Join the frequent 1-itemsets with each other to construct the candidate itemsets constructed in step2 and prune some infrequent itemsets from the candidate 2-itemsets to create the frequent 2- itemsets.

Step4. Repeat step3 until no more candidate itemsets can be created.

Pseudo-Code

C_k: Candidate item set of size k
L_k: frequent itemsets of size k
L1= {frequent items};
for (k= 1; Lk !=∅; k++)
Do
Begin
C_{k+1}= candidates generated from L_k;
for each transaction t in database do
Increment the count of all candidates in
C_{k+1} that are contained in t
L_{k+1}= candidates in C_{k+1} with min_support
end
return U_k L_k;

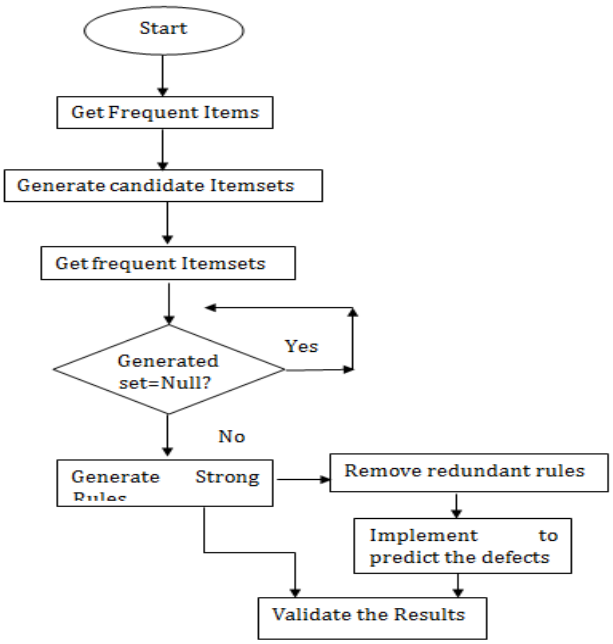

Figure 1 Flowchart of Proposed Model

IV. RESULTS AND DISCUSSION

The objective of this paper is to discover the interesting rules by considering the different parameters- Support, Confidence

© 2015-19, IJARCS All Rights Reserved

and Lift. These parameters are used to prune weak association rules which tend to creep into the top n association rules. Apriori algorithm is used to generate association on the basis of rules support and confidence framework. Support=30%, Confidence=0.95 is used, hence all the association rules which qualify these two threshold will be considered.

Table 1 Top 10 Association rules selected

Lhs	support	confidence	lift
{LCOM=0,DIT=0,IFANIN=2}	0.31	0.9982262	1.048
{LCOM=0,IFANIN=2}	0.33	0.9979227	1.047
{LCOM=0,IFANIN=2,NOC=0}	0.33	0.9978947	1.047
{LCOM=0,DIT=0,NOC=0}	0.35	0.9944312	1.044
{NOC=0,RFC=13}	0.5	0.9902125	1.039
{DIT=0,IFANIN=2,NOC=0}	0.36	0.9893617	1.038
{DIT=0,IFANIN=2}	0.37	0.9885397	1.037
{IFANIN=2,NOC=0}	0.42	0.9815772	1.030
{DIT=0,NOC=0}	0.48	0.9756305	1.024
{IFANIN=1,NOC=0}	0.39	0.9378134	0.984

We are taking Eclipse 3.1 dataset for finding the association rules between OO-metrics for getting the best predictor of fault using these rules. First of all, convert the dataset column value from numerical to nominal value and apply the algorithm. There are four possible outcomes which are formulated in the form of table called confusion matrix, as follows:

	Actual Value	
Predicted Outcome	True Positive(TP)	False Positive(FP)
	False Negative(FN)	True Negative(TN)

After getting the interesting rules, we implement them on different datasets to predict the faults or defects. Maintaining the Integrity of the Specifications

Based on top 10 association rules, we received predicted outcomes, which are compared with actual values. Results show that the association rules generated performs better to predict the non-defective modules than defective modules. Hence we got the values of performance measures of 10 different rules for predicting defective and non-defective modules. To summarize, average of measures for 10 rules is calculated, which is as follows:

Table 2 Performance measures for all four versions

	Measures	3.1	3.2	3.3	3.6
Defective	**Precision**	**0.23**	**0.35**	**0.38**	**0.48**
	Recall	**0.65**	**0.65**	**0.64**	**0.6**
Non-Defective	**Precision**	**0.98**	**0.97**	**0.97**	**0.96**
	Recall	**0.89**	**0.85**	**0.92**	**0.91**

The pictorial representation of the results is shown in figure 2 for comparison of the precision and recall values for the different four versions. Results show that precision for predicting defective modules increases as we go for higher versions.

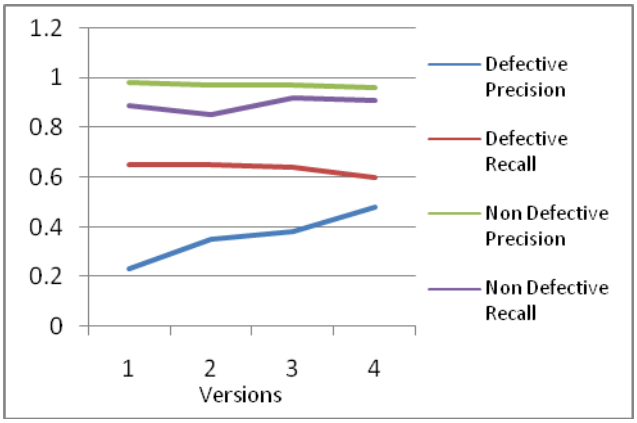

Figure 2 Comparison of performance for different versions

Precision for predicting defective modules is low, because the approach used generates rules based on frequent itemsets. In our data sets, majority of modules were non-defective. Hence it generates better rules for predicting non-defective modules.

V. CONCLUSION

In this paper, we conducted association rule mining and statistical analysis to predict the software defects. This paper also presented the heuristics to rank the association rules by considering three parameters: support, confidence and lift. This method will generate best associations as it can weed out weak associations and actual best association rules will be easily recognized. Therefore, for those datasets which contain large number of transactions, it can give efficient association rules. We found that the predictor proposed give better results in predicting the presence and absence of faults in Eclipse software.

The existing algorithms can be parallelized using MapReduce platform which is a familiar fault tolerant framework[29]. A recently proposed in-memory distributed dataflow platform called Spark which overcomes the drawbacks of MapReduce. The knowledge of available software repositories can be integrated with current software development and tools and fault detection systems to notify software developers with potential software defective modules.

VI. REFERENCES

[1] X. Amatriain, A. Jaimes, N. Oliver, and J. M. Pujol, *Data Mining Methods for Recommender Systems*. 2011.

[2] R. Agrawal, "Mining Association Rules between Sets of Items in Large Databases," no. May, pp. 1–10, 1993.

[3] P. He, "The Research of improved Association Rules Mining Apriori," *Proc. - 3rd Int. Conf. Converg. Hybrid Inf. Technol. ICCIT 2008*, no. August, pp. 0–2, 2004.

[4] R. B. Diwate and A. Sahu, "Data Mining Techniques in Association Rule: A Review," vol. 5, no. 1, pp. 227–229, 2014.

[5] V. Mangla, C. Sarda, and T. Nadu, "Improving the efficiency of Apriori Algorithm in Data Mining," *Int. J. Eng. Innov. Technol.*, vol. 3, no. 3, pp. 393–396, 2013.

[6] L. C. Briand, V. R. Basili, and W. M. Thomas, "A pattern recognition approach for software engineering data analysis," *IEEE Transactions*, vol. 18, no. 11. pp. 931–942,

© 2015-19, IJARCS All Rights Reserved

1992.
[7] X. Deng and X. Wang, "Mining Rank-Correlated Associations for Recommendation Systems," *IEEE*, no. 062112065, pp. 625–629, 2009.
[8] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proceedings of the 20th International Conference on Very Large Databases*. pp. 487–499, 1994.
[9] J. Tian, "An empirical comparison and characterization of high defect and high complexity modules," vol. 67, pp. 153–163, 2003.
[10] A. Bhandari, A. Gupta, and D. Das, "Improvised Apriori Algorithm Using Frequent Pattern Tree for Real Time Applications in Data Mining," *Procedia Comput. Sci.*, vol. 46, no. Icict 2014, pp. 644–651, 2015.
[11] S. Rathee, M. Kaul, and A. Kashyap, "R-Apriori: An Efficient Apriori based Algorithm on Spark," *ACM*, pp. 27–34, 2015.
[12] A. H. Yousef, "Extracting software static defect models using data mining," *Ain Shams Eng. J.*, vol. 6, no. 1, pp. 133–144, 2014.
[13] R. Mishra, "Comparative Analysis of Apriori Algorithm and Frequent Pattern Algorithm for Frequent Pattern Mining in Web Log Data .," *Int. J. Comput. Sci. Inf. Technol.*, vol. 3, no. 4, pp. 4662–4665, 2012.
[14] T. M. Khoshgoftaar and N. Seliya, "Software Quality Classification Modeling Using The SPRINT Decision Tree Algorithm Taghi," pp. 365–374, 2002.
[15] T. M. Khoshgoftaar, B. Raton, and R. M. Szabo, "An Application of Zero-Inflated Poisson Regression for Software Fault Prediction," pp. 66–73, 2001.
[16] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," *Inf. Sci. (Ny).*, vol. 264, pp. 260–278, 2014.
[17] J. Leskovec, *Mining of Massive Datasets*. 2014.
[18] M. Zhang and C. He, "Survey on Association Rules Mining Algorithms 2 Basic Principles of Association Rules," pp. 111–118, 2010.
[19] B. Goethals, "Survey on Frequent Pattern Mining," pp. 1–43, 2003.
[20] S. Veeramalai, N. Jaisankar, and A. Kannan, "Efficient Web Log Mining Using Enhanced Apriori Algorithm with Hash Tree and Fuzzy," vol. 2, no. 4, pp. 60–74, 2010.
[21] R. Karthik and N. Manikandan, "Defect association and complexity prediction by mining association and clustering rules," *ICCET 2010 - 2010 Int. Conf. Comput. Eng. Technol. Proc.*, vol. 7, pp. 569–573, 2010.
[22] S. Deepa and M. Kalimuthu, "An Optimization of Association Rule Mining Algorithm using Weighted Quantum behaved PSO," vol. 3, pp. 80–85, 2012.
[23] S. Agarwal, "Prediction of Software Defects using Twin Support Vector Machine," pp. 128–132, 2014.
[24] Q. Wang and B. Yu, "Extract Rules from Software Quality Prediction Model Based on Neural Network," no. Ictai, pp. 0–2, 2004.
[25] I. Qureshi, J. Ashok, and V. Anchuri, "A Survey on Association Rule Mining Algorithm and Architecture for Distributed Processing," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 3, pp. 4674–4678, 2014.
[26] T. M. Khoshgoftaar, "Tree-Based Software Quality Estimation Models For Fault Prediction," 2002.
[27] D. Kumari and K. Rajnish, "A new approach to find predictor of software fault using association rule mining," *Int. J. Eng. Technol.*, vol. 7, no. 5, pp. 1671–1684, 2015.
[28] J. Manimaran and T. Velmurugan, "Analysing the quality of Association Rules by Computing an Interestingness Measures," vol. 8, no. July, 2015.
[29] Z. Rong, D. Xia, and Z. Zhang, "Complex statistical analysis of big data: Implementation and application of apriori and FP-growth algorithm based on MapReduce," *Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS*, no. 2012, pp. 968–972, 2013.

© 2015-19, IJARCS All Rights Reserved