VOL. 8, NO. 1, JANUARY 2013 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

AN EFFICIENT CLASSIFICATION OF FAULT DETECTION THROUGH COMPRESSED TREE (CT) APRIORI BASED APPROACH USING ARC-BC CLASSIFIER

R. Jeevarathinam[1] and T. Santhanam[2]
[1]Department of Computer Science, SNR Sons College, Coimbatore, India
[2]PG and Research Department of Computer Science, D G Vaishnav College, Chennai, India
E-Mail: mani_jeeva_2000@yahoo.com

ABSTRACT

Generally software systems are scantily documented and incomplete specifications in the documents results in high maintenance cost. To lower maintenance efforts, automated tools are necessary to aid software developers to understand their existing code base by extracting specifications from the software. Specification mining aids the document to intend software behaviour, software maintenance, refactor or add new features to software, and detecting software bugs. In this paper a new technique called CT Trace Miner is proposed to efficiently mine software specifications, which in turn mines software specifications from program execution traces. Then the mined specification using the CT Trace Miner approach is given as input to the ANOVA Two Way feature selection approach for selecting the best features. Conclusively, ARC-BC classifier is used to categorize the selected features. The experimental results exposed that the proposed approach provides better results than the existing approach.

Keywords: ARC-BC, Compressed tree (CT), ANOVA two way, software specification, rule mining.

INTRODUCTION

Computers as well as software running on them play an important role in each and every aspect. Software Reliability is necessary for the computer to maintain the software quality. Software reliability is considered as the possibility of bug-free software operates at a particular time interval in a specified atmosphere [1]. Because of Application Program Interfaces (API), software system interconnects with its environment such as network, memory, operating system, graphics card and other software etc. Certain programming rules (API specifications) are essential to use these APIs effectively [2].

Two main hindrances for the reliable operation of a software system are such as security and robustness [3]. API violations directs to these issues. Consistency of software system with specifications, which controls the usage of APIs, is the chief factor of software reliability. Software testing [4, 5], and static verification [6, 7] have been agreed by the industries to assure software reliability. Several problems occur with these approaches. Software testing techniques [8, 9] mainly focus on problems such as correctness of functionality and performance and it lacks to assure the absence of API violations.

Documented software specification is the main part of program artifacts. It describes about the software behavior. If the specification is well defined and clearly described, Software can be developed accurately and maintained easily. Specifications should be used as an input for formal program verification tools to identify bugs or convert it into runtime monitors to reveal the violations of specifications or properties of interests during program execution [10, 11, 12]. Hence, it is important if all programs and software projects are developed with clear, precise and well-documented specifications. But, normally documented specifications are insufficient or lacking in the industry. Specification mining is a computerize process to extract specifications from a program solves this issue.

Obtaining patterns of frequent repetitive software behaviors plays significant role in the current scenario. Monotonicity or apriori property is essential to aid in assuring scalability. A suitable apriori property is viewed as prune the search space having insignificant patterns. Novel iterative pattern mining approach for mining closed iterative patterns is established in this paper. The proposed rule mining algorithms mines a closed set of iterative patterns. A search space pruning approach is utilized by early reorganization for filtering and pruning of non-closed patterns to extract closed set of iterative patterns. Mined frequent patterns acquire global detailed behaviors of a system and software behaviors. Mined rules acquire constraints or implicit rules which are adhere by a piece of software during analysis process.

Many rules can be gathered from traces, but it should not be important. The information about support and confidence utilized in data mining [13] are used to identify major rules. Rules which satisfy user-specified thresholds with minimum support and confidence are considered as being statistically important. Software behaviours should be calculated based on historical data of software and known failures. It is necessary to construct a classifier to generalize the failures and to further detect other unknown failures. Besides, the classifier can aid other software engineering tasks.

RELATED WORK

Chao Liu *et al.*, [14] proposed a new technique to classify the structured traces of program executions through software activities graphs. He shows better advancement by analyzing the correct and incorrect processing in the code at the separation of program parts which results in defective executions. Hong Cheng *et al*., find a systematic examination of frequent pattern-based classification, and afford solid reasons which support pattern mining techniques. The Hong Cheng presented the classification framework with the usage of frequent pattern which attain better scalability and improved accuracy in classifying huge quantity datasets.

Ammons and Bodik [15] defined specification mining as a machine learning technique to find out the requirements of the protocols which should be followed by code while relating with an Application Program Interface (API) or Abstract Data Type (ADT). Feature selection is a major step in text categorization to minimize the feature space. Practical observations of text categorization shows that good text categorization performance is associated to certain feature selection criteria, and when a criterion is not satisfied, it often represents non-optimality of the technique.

Rehg *et al*., [16] presented a method which models program executions as Markov models and a clustering technique for Markov models that integrates multiple program executions into competent behavior classifiers. Bowring *et al*., [16] presented an active learning technique to build a classifier of program behaviors. Frequency profile of single events in the trace becomes the input for this approach. Bowring studied two groups of first-order Markov models, where each set characterizes correct and incorrect behaviors.

Gait feature subset selection [17 - 20] researches have mainly considered conventional dimensionality reduction or statistical tools such as Analysis Of Variance (ANOVA) and PCA. Still, some challenges remained for gait feature selection. The ANOVA technique helps to identify the features based on a null hypothesis test. However, the difference between classes highlighted by ANOVA is carried out by examining whether the population means vectors are the same, whether it lacks the explicit or definite relation with the recognition accuracy. Several classification techniques available in the literature such as decision tree induction [21], Bayesian networks [22] and association based classification [23]. Association-based classification has recently received much attention.

Buddeewong *et al*., [24] proposed a new association rule-based text classifier algorithm to enlarge the prediction accuracy of association rule-based classifier by categories (ARC-BC) algorithm. The proposed association rule generation technique generates two kinds of frequent item sets: Lk consist of all term that have no an overlap with other classes and OLk consists of all features that have an overlap with other classes. He also proposed a new join operation for the second frequent item sets OLk.

METHODOLOGY

A trace is a record of the execution of a computer program which reveals the sequence of operations executed. Dynamic traces are obtained by executing the program based on the input. This would result in abstracted traces. These abstracted traces are given as input to the Rule mining algorithm like Trace Miner and CT apriori Trace Miner. Then, the mined specifications are given as input to the feature selection approach ANOVA Two Way. Then the selected features are given to the ARC-BC classifier.

Classification of the software behavior comprises of three phases:

a) Rule mining
b) Feature selection
c) Classification

A. Rule mining

Association Rule Mining (ARM) [25] has been the focus in many research areas such as data mining, artificial intelligence, machine learning for a decade. Though, ARM has been employed in several application areas and it is also extended to data mining tasks of classification [26] and clustering. ARM approaches classify the issue into two segments. (i) To find the frequent patterns and then use them to form the rules. (ii) Generating the association rules in simple once frequent patterns are identified.

The problem of generating frequent item sets has been a wide research area and large number of algorithms has been proposed for it. Frequent item set mining is a major part of association rules mining. The general performance of association rules mining is determined as frequent item set mining is computationally more expensive. The frequent item set concept has been extended for several data mining techniques such as classification, clustering and sequential pattern discovery. The data structures play a key role in the performance of these techniques. A new data structure is represented here namely Compressed Tree (CT-Tree).

CT-Apriori Trace Miner

A specification database consists of information about often used patterns of potential program executions. The process of acquiring this information is called Frequent Pattern Mining and it can be recognized by several data mining techniques such as clustering, classification, prediction and association analysis. CT-Apriori (Compact Tree-Apriori) is based on association rules.

Let I = $\{i_1, i_2, \dots, i_m\}$ be a set of m execution traces. A subset X $\in$ I is called a trace set. A k-trace set is a trace set which contains k traces.

Definition: A specification database SPDB = $\{T_1, T_2, \dots, T_N\}$ is a set of N traces, where each trace is a set of traces T_n (n $\in$ {1, 2, …, N}) such that Tn $\in$ I. A specification T contains a trace set X if and only if X $\in$ T.

Algorithm description

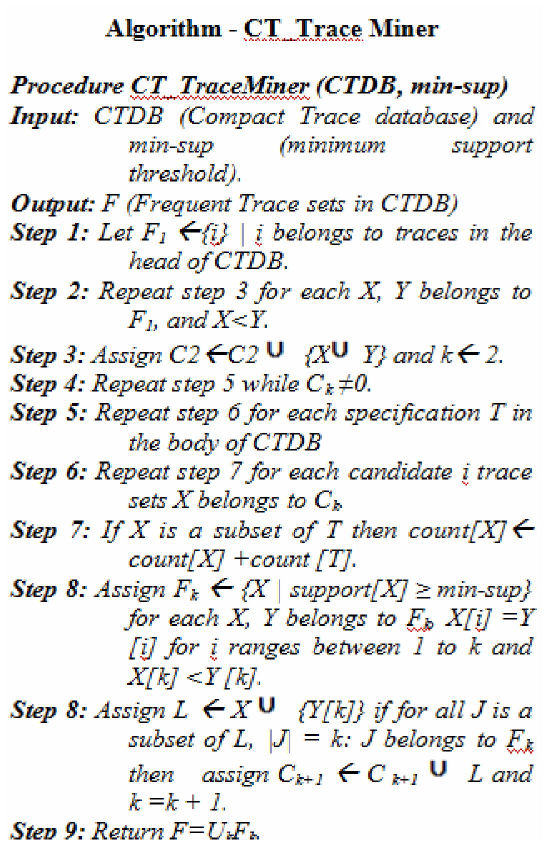
CT Trace miner algorithm

The Apriori algorithm is one of the most widely used algorithms for mining frequent patterns and association rules.

There are two vital differences between proposed CT Apriori algorithm and Apriori algorithm. They are as follows:

a) The CT-Apriori algorithm skips the initial search of database in the Apriori algorithm by reading the beginning part of the compact transaction database and inserting the frequent 1-itemsets into F1. Then candidate 2-itemset C2 is produced from F1 directly.
b) In CT-Apriori, as shown in the Figure-1 (step 10), counts are incremented by the occurrence count of that transaction stored in the body of the compact transaction database, which is always greater than 1 which differs from Apriori algorithm.

The proposed CT_Trace Miner algorithm is shown in the Figure-1.



Algorithm - CT_Trace Miner

```
Procedure CT_TraceMiner (CTDB, min-sup)
Input: CTDB (Compact Trace database) and
        min-sup (minimum support threshold).
Output: F (Frequent Trace sets in CTDB)
Step 1: Let $F_1 \leftarrow \{i\} \mid i$ belongs to traces in the
        head of CTDB.
Step 2: Repeat step 3 for each X, Y belongs to
        $F_1$, and $X<Y$.
Step 3: Assign $C2 \leftarrow C2 \cup \{X \cup Y\}$ and $k \leftarrow 2$.
Step 4: Repeat step 5 while $C_k \neq 0$.
Step 5: Repeat step 6 for each specification T in
        the body of CTDB
Step 6: Repeat step 7 for each candidate i trace
        sets X belongs to $C_k$
Step 7: If X is a subset of T then $count[X] \leftarrow$
        $count[X] + count[T]$.
Step 8: Assign $F_k \leftarrow \{X \mid support[X] \geq min\text{-}sup\}$
        for each X, Y belongs to $F_k$, $X[i] = Y[i]$ for i ranges between 1 to k and
        $X[k] < Y[k]$.
Step 8: Assign $L \leftarrow X \cup \{Y[k]\}$ if for all J is a
        subset of L, $|J| = k$: J belongs to $F_k$
        then assign $C_{k+1} \leftarrow C_{k+1} \cup L$ and
        $k = k + 1$.
Step 9: Return $F = \cup_k F_k$
```

Figure-1. CT_ Trace Miner algorithm.

B. Feature selection

Feature selection becomes an active research topic in statistics and pattern recognition. Feature selection is also known as variable selection, feature reduction, attribute selection or variable subset selection. It is the major technique commonly employed in machine learning by selecting a subset of relevant features to build robust learning models. It is important because it can simplify the data description and this in turn results in understanding the problems easily and solving the problems quickly.

Analysis of variance (ANOVA) is a method used to analyze data in which one or more response (or dependent or simply Y) variables are measured under various conditions identified by one or more classification variables. The combinations levels for these classification variables form the cells for the design of data.

Analysis of variance constructs several tests to determine the significance of the classification effects. A typical goal in such an analysis is to compare means of the response variable for several combinations of the classification variables. The least squares principle is a key object to compute the sum of squares in analysis of variance models. One-way analysis of variance (ANOVA) test measures significant effect of one factor, whereas two-way analysis of variance (ANOVA) tests (also called two-factor analysis of variance) measures the effect of two factors simultaneously. ANOVA two way analysis is the best feature selection method when compared with others. The two-way analysis of variance is an extension of one-way analysis of variance. There are two qualitative factors (A and B) on one dependent continuous variable Y. Three null hypotheses are tested in this procedure such as factor A does not influence variable Y, factor B does not influence variable Y, the effect of factor A on variable Y does not depend on factor B (i.e., there is no interaction of factors A and B).

Two-way analysis of variance requires data for each combination of the two qualitative factors A and B. A two-way ANOVA can be used when there are two independent variables (factors) influencing one dependent variable. The idea is that two variables which affects the dependent variable. Each factor will have two or more levels within it, and the degrees of freedom for each factor is one less than the number of levels.

The generally used statistical measure of ANOVA two-way ranking is defined through the following steps:

a) Determine whether the M rules have been drawn from the same rule database.
b) Cases are ranked and the mean rank is calculated for each sample.
c) The test statistic H is calculated as follows:

$$H = [12R/NM\ (M+1)] - 3N\ (M+1)$$

where N is the number of rows, M is the number of columns and R is the sum of squares of column rank totals.

A feature selection algorithm on Iterative patterns using ANOVA two way is used to filter indiscriminative patterns. The algorithm performs a sequential scan of the ranked iterative patterns. If a pattern covers some training instances, it will be selected. Any data instances covered by at least ± features will be removed from further consideration. The algorithm terminates if either all instances are covered by the selected features or the feature set becomes empty.

C. Classification

Association Rule-based Classifier by Categories (ARC-BC) algorithm is proposed to build an associative text classifier. ARC-BC (Associative Rule-based Classifier by Category) considers each set of rules belong to one category as a separate text collection to generate association rules. If rules belongs to more than one category it will be present in each set associated with the categories. The ARC-BC algorithm is described in below.

Algorithm: ARC-BC detects association rules on the training set of the rule database.

Input: A set of rules (D) of the form $D_i : \{c_i, t_1, t_2, \ldots, t_n\}$ where c_i is the category attached to the document and t_j are the selected terms of the document; a minimum support threshold; a minimum confidence threshold;

Output: A set of association rules of the form $t_1 \wedge t_2 \wedge \ldots \wedge t_n \Rightarrow c_i$ where c_i the category is and t_j is a term;

```
Algorithm - Association Rule based ARC-BC Classifier
Procedure: Rule Classification
Inputs: Di: Set of rules
ci : Category attached to the document
tj : Selected terms of the document
Output: A selected set of association rules
 $C_1 \leftarrow$ {Candidate 1 Normal trace and their support }
 $F_1 \leftarrow$ {Frequent 1 Error trace and their support }
for $(i \leftarrow 2; F_1(i-1) \neq \emptyset; i \leftarrow i+1)$ do {
  $C_i \leftarrow (F_{i-1} \bowtie F_{i-1})$
  $C_1 i \leftarrow C_1 i - \{c \mid (i-1)item - set\ of\ c \notin F_1(i-1)\}$
$D_i \leftarrow Filter\ Table(D_{i-1}, F_{i-1})$
foreach document $d$ in $D_i$ do {foreach $c$ in $C_i$ do
{$c.support \leftarrow c.support + Count(c, d)$}}
$F_1 i \leftarrow \{c \in C_1 i \mid c.support > \sigma\}$ }
Sets $\leftarrow \bigcup_i \{c \in F_i | i > 1\}$
$R = \emptyset$
foreach ruleset $I$ in Sets do { $\leftarrow R + \{I \Rightarrow Cat\}$ }
```

Figure-2. Constructing ARC-BC classifier.

In ARC-BC algorithm step (2) generates the frequent 1- item set. In steps (3-13), all the k-frequent item sets are generated and merged with the category in C_1. Steps (16-18) generate the association rules. The document space is reduced in each step of iteration by eliminating the transactions that do not contain any of the frequent item sets. This step is done by $FilterTable(D_{i-1}, F_{i-1})$ function. If the amount of rules generated is too large it is time consuming to read the set of rules for further tuning of the system.

EXPERIMENTAL RESULTS

Classification framework for software failure detection is evaluated using different programs is analyzed by Siemens Test Suite [18]. The test suite contains several programs. Each program contains different versions where each version has a bug. These bugs comprises of wide array of realistic bugs. Four largest programs are taken in the test suite. They are replace, schedule, print tokens and tot_ info. For this case study, replace and tot_ info datasets are used.

To simulate real life situation where there are many bugs occurring together, 3 bugs are injected to each program and add 3 additional simulated ordering bugs to the execution traces. Running the instrumented program with an input produces a trace. A set of traces is collected by running set of test cases provided by Siemens Test Suite. The test suite allows computing the actual correct output.

A. Replace dataset

Table-1 shows the performance evaluation of the mining algorithms for the replace dataset. The results of the CT apriori mining algorithm proposed in the present research work are evaluated. The results of the feature selection algorithms like fisher score and ANOVA Two way are obtained for CT Apriori Trace Miner. The results are obtained for the parametric standards like AUC, Accuracy and Time. It is observed from the Table-1 that the ANOVA Two Way feature selection approach provides better results when compared with the Fisher score feature selection approach. Time taken by the CT Trace Miner mining algorithm with the ANOVA Two Way feature selection approach is very less when compared to the fisher score feature selection approach. Table-1 shows the performance evaluation of the classification algorithms like ARC-BC with Fisher Score and ARC-BC with ANOVA. It is observed from the Table that the proposed ARC-BC classifier with ANOVA Two Way provides better results.

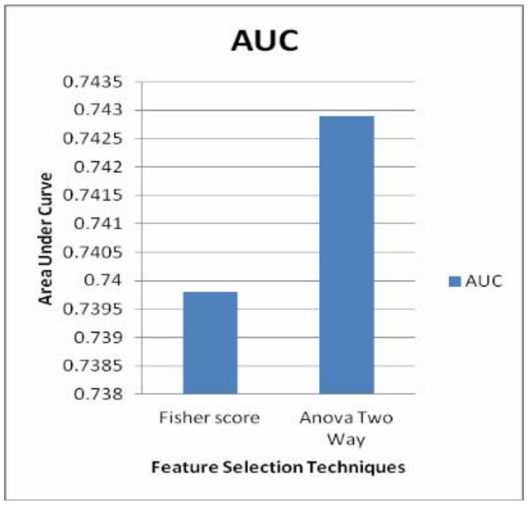


Figure-3. AUC value comparisons of ARC-BC classifier for replace dataset.

Table-1. Performance evaluation of ARC-BC classifier for replaces data set.

Rule mining algorithm	Classification result with feature selection approaches					
	ARC-BC with fisher score			ARC-BC with ANOVA Two Way		
	AUC	Accuracy (%)	Time (Sec)	AUC	Accuracy (%)	Time (Sec)
CT-TM	0.7398	99.34	0.25	0.7429	99.45	0.218

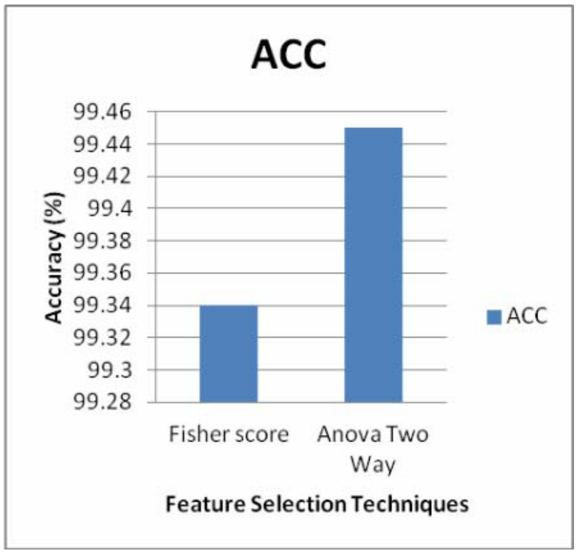


Figure-4. Comparison of classification accuracy for replace dataset.

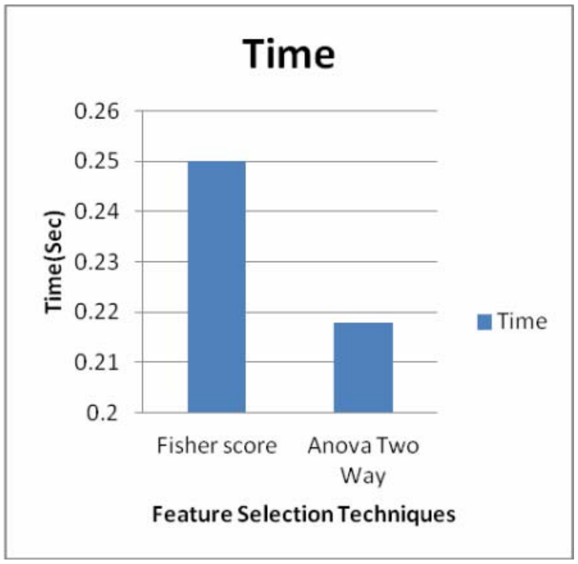


Figure-5. Execution time comparisons of ARC-BC classifier for replace dataset.

From the Figures 3, 4 and 5 it is clearly shown that the AUC value and classification accuracy of the ARC-BC classifier with ANOVA Two Way for the CT-Trace Miner is very high when compared with the ARC-BC classifier with fisher score approach using replace dataset. The execution time taken by the ARC-BC classifier with ANOVA Two Way approach is very less when compared with the ARC-BC classifier with fisher score approach.

B. Tot_info dataset

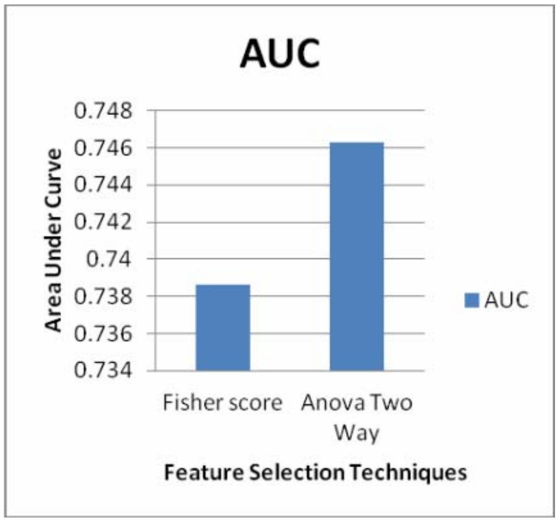


Figure-6. AUC value comparison of ARC-BC classifier for tot-info dataset.

The Figures 6, 7 and 8 clearly show that the AUC value and classification accuracy from Table-2 for the feature selection algorithms for CT-Apriori mining approach. ANOVA Two Way feature selection approach is observed to provide better results for CT Trace Miner. ANOVA Two Way technique is observed to take very less execution time when compared with the fisher score. Table-2 shows the performance evaluation of the ARC_BC classification algorithm with ANOVA Two Way and Fisher score for tot-info dataset. It is observed that the ARC_BC with ANOVA Two Way approach is observed to outperform the SVM with fisher score approach in terms of accuracy and execution time.

Table-2. Performance evaluation of ARC-B classifier for Tot_Info data set.

Rule mining algorithm	Classification result with feature selection approaches					
	ARC-BC with fisher score			ARC-BC with ANOVA Two Way		
	AUC	Accuracy (%)	Time (Sec)	AUC	Accuracy (%)	Time (Sec)
CT-TM	0.7386	88.1	0.375	0.7463	96.1	0.218

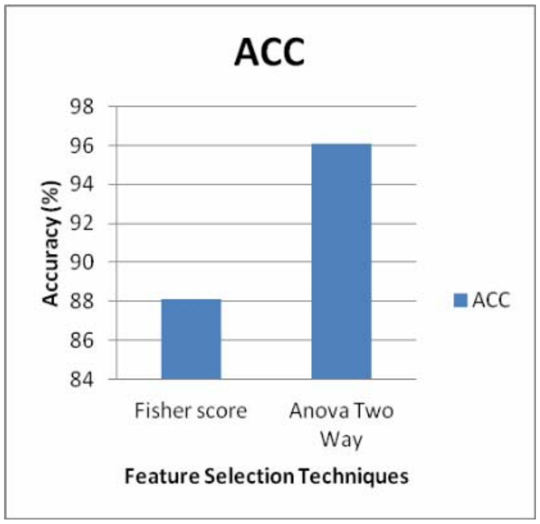


Figure-7. Comparison of classification accuracy for Tot-info dataset.

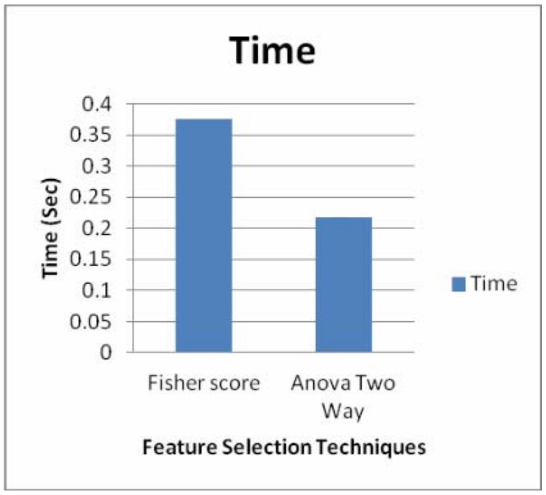


Figure-8. Execution time comparison of ARC-BC classifier for Tot-info dataset.

CONCLUSIONS

In this paper, an efficient approach for classification of fault detection through CT -Trace Miner is proposed. CT-tree is devised to generate compact specification database and store it into disk for effective frequent pattern mining and other mining process, in which compact database saves storage space and reduce mining time. It reduces the number of specifications in the original databases and save storage space. It also reduces the I/O time required by database scans and improves the efficiency of the mining process.

Mined specifications are then preceded by feature selection approaches. ANOVA Two way is used as the feature selection approach. The selected features are given to the ARC-BC classifier. ARC-BC provides the classified results of the software under several tests. It is examined from the experimental results that the proposed CT-Trace Miner approach with ANOVA Two Way feature selection provides very good results in detecting fault traces and normal traces when compared to the previous approaches.

The performance of the proposed approach is evaluated based on the parametric standards such as Accuracy, AUC and Execution Time. As a future work, the chance of direct mining can be focused using discriminative iterative patterns, applications of the classifier to other domains, and pipelining.

REFERENCES

[1] J. D. Musa, A. Iannino and K. Okumoto. 1987. Engineering and Managing Software with Reliability Measures. McGraw-Hill.

[2] M. Acharya, T. Xie, J. Pei and J. Xu. 2007. Mining API Patterns as Partial Orders from Source Code: From Usage Scenarios to Specifications. In: Proceedings of Joint Symposium on the Foundation of Software Engineering and European Software Engineering Conference (ESEC/SIGSOFT FSE) Symposium on the Foundations of Software Engineering. pp. 25-34.

[3] Benjamin Schwarz, Jeremy Lin and Wei Tu. 2005. Model Checking an Entire Linux Distribution for Security Violations. In: Proceedings of the Annual Computer Security Applications Conference (ACSAC), IEEE. pp. 13-22.

[4] Nathan Kropp, Philip J. Koopman and Daniel P. Siewiorek. 1998. Automated Robustness testing of off-the-shelf Software Components. In: Proceedings of the of IEEE International Symposium on Fault-Tolerant Computing (FTCS). pp. 230-239.

[5] Jennifer Haddox, Gregory Kapfhammer and Michael Schatz. 2001. Testing Commercial-off-the-shelf Software Components. In: Proceedings of the 18th International Conference and Exposition on Testing (ICET).

VOL. 8, NO. 1, JANUARY 2013 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

[6] GrammaTechCodeSurfer. http://www.grammatech.com/products/codesurfer/.

[7] Coverity's Static Analysis. http://www.coverity.com/products/static-analysis.html.

[8] Pattern Insight's Patch Miner. http://www.patterninsight.com/solutions/index.html.

[9] FindBugs. http://findbugs.sourceforge.net/.

[10] Thomas Ball and Sriram K. Rajamani. Automatically Validating Temporal Safety Properties of Interfaces. In: Proceedings of the SPIN 2001 Workshop on Model Checking of Software, LNCS 2057. pp. 103-122.

[11] Justin Forrester and Barton P. Miller. 2000. An empirical study of the robustness of Windows NT applications using random testing. In: Proceedings of the USENIX Windows Systems Symposium. pp. 69-78.

[12] Barton Miller, Ajitkumar Natarajan and Jeff Steidl. 1995. Fuzz revisited: A re-examination of the reliability of UNIX utilities and services. Computer Science Technical Report 1268, University of Wisconsin-Madison, United States.

[13] J. Han and M. Kamber. 2006. Data Mining Concepts and Techniques. 2nd Edition, Morgan Kaufmann.

[14] Chao Liu, Xifeng Yan, Hwanjo Yu, Jiawei Han and Philip S. Yu. Mining Behavior Graphs for Backtrace of Noncrashing Bugs.

[15] G. Ammons, R. Bodik and J. R. Larus. 2002. Mining specification. In: Proceedings of SIGPLAN-SIGACT Symposium on Principles of Programming Languages.

[16] J. F. Bowring, J. M. Rehg and M. J. Harrold. 2004. Active learning for automatic classification of software behavior. In: Proceedings of International Symposium on Software Testing and Analysis.

[17] B. Bhanu and J. Han. 2003. Human recognition on combining kinematic and stationary features. In: Proceeding of 4th Int. Conf. AVBPA. pp. 600-608.

[18] A. Veeraraghavan, R. Chellappa and A. Roy Chowdhury. 2004. Role of shape and kinematics in human movement analysis. In: proceedings of the IEEE Conference CVPR. 1: 730-737.

[19] Z. Liu, L. Malave, A. Osuntogun, P. Sudhakar and S. Sarkar. 2004. Toward understanding the limits of gait recognition. In: Proceedings of the SPIE. 5404: 195-205.

[20] M. S. Nixon, T. N. Tan and R. Chellappa. 2005. Human Identification Based on Gait. New York: Springer-Verlag. ch. 5, pp. 45-104.

[21] Mehmet Sabih Aksoy. 2005. Pruning Decision Trees Using R S3 Inductive Learning Algorithm. Mathematical and Computational Applications. 10(1): 113-120.

[22] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley and P. Smyth. 1997. Bayesian network classifiers. Machine Learning. 29: 131-163.

[23] B. Liu, W. Hsu and Y.Ma. 1998. Integrating classification and association rule mining. In: Proceeding of 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD'98), New York. pp. 80-86.

[24] S. Buddeewong and W. Kreesuradej. 2005. A New Association Rule-Based Text Classifier Algorithm. ICTAI. 17th IEEE International Conference on Tools with Artificial Intelligence. pp. 684-685.

[25] R. Agrawal, T. Amielinski and A. Swami. 1993. Mining association rule between sets of items in large databases. In: Proceeding of the ACM SIGMOD International Conference on Management of Data. May 26-28. pp. 207-216.

[26] X. Yin, J. Han, J. Yang and P. S. Yu. 2006. Efficient classification across multiple database relations: A crossmine approach. IEEE Trans. Knowledge and Data Engineering. 18: 770-783.