Locating Matching Rules by Mining Software Change Log

Jung-Te Weng and Ming-Shi Wang[1]

mswang@mail.ncku.edu.tw

[1]Department of Engineering Science, National Cheng Kung University

Abstract

A software system maintenance activity is typically performed under an environment of lacking knowledge about how to process it. This scarcity of knowledge may be caused by various factors, such as the large size and complexity of the systems, high staff turnover, poor documentation and long-term system maintenance. The study applies Apriori algorithm to extract information from software change logs. Unfortunately, the software change logs generate many rules. Because searches the suitable rule from many rules is difficult and important matter, especially. This study focuses on the software co-change dependency and proposes a classification model based on association mining, to deal with such kind of dependency. The model combines data mining technologies, the traditional decision-tree and neural learning capabilities, to handle the complicated and real cases, and then improve the rule searching efficiency and the matching accuracy.

Keywords: Software maintenance, Data mining, Decision Trees, Neural Network

1. Introduction

Software systems contain entities, such as functions and variables, which are related to each other. As software systems evolve to accommodate new features or repair bugs, change occur to these entities. Software maintainability is defined as “the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment”[1]. A considerable amount of system maintenance experiences can be found from the bug tracking and source code configuration management systems. Data mining technique was adapted to find the association rules with software change logs.

A change to one software component is likely to create inconsistencies with other components. Therefore all related components must modify possible inconsistencies. This process is called impact analysis [2]. Dependencies among components may include calls between functions, uses of variables and the relationship between a requirements and a design document [3]. Impact analysis generally begins by changing some components, then identifying which components are affected by the modification.

Dependency analysis, or understanding how components interact with each other to ensure all necessary changes have been made, has also been widely researched. For instance, Zhifeng and Rajlich [4] presented an approach that examines the effect of concealed dependencies on the process of change propagation, and an algorithm that warns about the possible presence of hidden dependencies. Arnold and Bohner [7-8] reviewed several formal models to arrange the propagation pattern of the changes. These methods based on code dependencies and algorithms, such as slicing and transitive closure, have been presented to assist code propagation analysis [7-8].

Shirabad et al.[5-6] utilized inductive learning to learn different relevant concepts in logically coupled files. The concept is a set of attributes, such as file name, extension and simple metrics like number of routines defined. If two files have these attributes, then they are relevant to each other. However, no relevant study provides a convincing example of such a concept. Hassan et al.[9] applied software measure method with heuristic model to predict the change propagation. Ying et al.[10] proposed applying market basket analysis techniques on the historical data of source code to help developers maintain the source code. These methods [9,10] obtained an average precision and recall of around 0.5-0.6 and 0.2-0.3, respectively. Neither Ying nor Hassan explained how suitable rules were obtained, but only the method of generating rules. Suitable rule is difficult to find from the large number of possible rules, but it is an important task.

Association rule analysis is a good choice for this study to concern with the problems of attribute dependencies. Generally, these dependencies are classified into two types: categorical-type dependency and numerical-type dependency. This study focuses on the categorical-type dependency, and applies the association rule mining technologies to extract past experiences. Then, the study uses traditional decision tree to achieve the dimension reduction work, and applies neural network techniques to classify those rules, with the following aims:

- Recommend likely co-changes.
- Prevent errors due to incomplete changes.
- Improve the rule searching efficiency and the matching accuracy.

2. Research Methodology

2.1. Data Mining

Data mining involves extracting valuable knowledge from a database, data warehouse or other large repository of information. Data mining integrates the gathering and cataloguing of information, and then generates rule-like knowledge from a large quantity of data. Significantly, data mining attempts to create association rules for a database or data warehouse. Association rules indicate the attribute-value conditions, typically using association analysis, that often occur together in a given set of data.

Attempts to obtain the association rule from a set of data were first made in 1993[11]. Association rule mining was initially focused on a supermarket "basket data", which holds items purchased on a transaction basis, and lists other products usually in the same purchase. This study applies data mining techniques to determine the change propagations for software maintenance. More precisely, the study uses the classical Apriori algorithm to discover interesting rules to maintain the software form change logs.

2.2. Dimension Reduction

A major problem in mining scientific data sets is that the data is often high dimensional, there are a large number of features representing the object. When the number of dimensions reaches hundreds or even thousands, the computational time for the pattern recognition algorithms can become prohibitive. This can be a problem, especially when some of the features are not discriminatory. The objective of study is to select some dimension of a data set in order to create a visualization from which relevant information can be extracted. We want to identify attributes that are significant in order to reduce dimensionality.

Dimension reduction can be used to improve the efficiency of visualization of large, multidimensional data sets and may be the accuracy of algorithms used for classification in data mining. The study applies decision tree to reduce dimension. The decision tree is interactively constructed by the user who uses his perception and data domain knowledge. This kind of interactive decision tree construction algorithm can only be used if the number of dimensions of the data is small enough.

2.3. Multilayer perceptron (MLP)

Multilayer perceptron (MLP) is a network of simple neurons called perceptrons. The perceptron computes a single output from multiple real-valued inputs by forming a linear combination according to its input weights and then possibly putting the output through some nonlinear activation function. A typical MLP network consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes. The input signal propagates through the network layer-by-layer. The signal-flow of such a network with one hidden layer is shown in Figure 1.

MLP networks are typically used in supervised learning problems. This means that there is a training set of input-output pairs and the network must learn to model the dependency between them. The supervised learning problem of the MLP can be solved with the back-propagation algorithm. The algorithm consists of two steps. In the forward pass, the predicted outputs corresponding to the given inputs. In the backward pass, partial derivatives of the cost function with respect to the different parameters are propagated back through the network. The chain rule of differentiation gives very similar computational rules for the backward pass as the ones in the forward pass. The network weights can then be adapted using any gradient-based optimization algorithm. The whole process is iterated until the weights have converged.

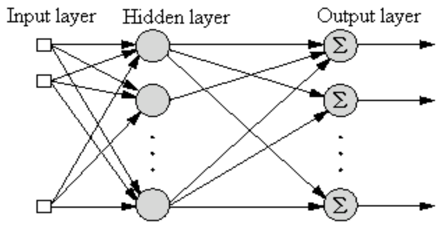

Figure 1: Signal-flow graph of an MLP

3. The Proposed Model and Results

The proposed model is applied to a legacy software system of banking. The two sub-systems, UCP (User Control Program) and Loan, of a legacy banking application were applied as the study case. A UCP system is used to arrange I/O controls, DBMS and to communicate with other IBM mainframe systems installed at different branches of the commercial bank.

This software system was written in the IBM360/370 assembly language and COBOL. The Loan sub-system was written in IBM JCL and COBOL. The two sub-systems contain more than 600 programs. The file revision log of these two sub-systems was recorded from 2000 to 2004. The banking system had 628 log transactions. The total number of changed items is 2842. The proposed model is depicted in Figure 2.

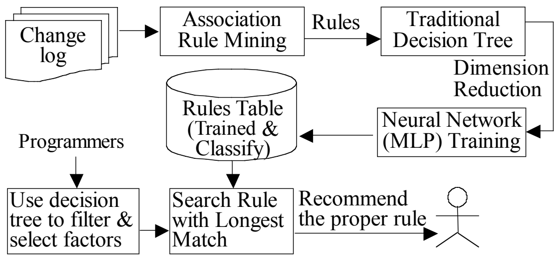
Change log
Association Rule Mining
Rules
Traditional Decision Tree
Dimension Reduction
Rules Table (Trained & Classify)
Neural Network (MLP) Training
Programmers
Use decision tree to filter & select factors
Search Rule with Longest Match
Recommend the proper rule

Figure 2: The architecture of the proposed model

The training data set is formed by randomly selecting 450 transactions from the log transactions for training. The remaining log transactions formed the testing data, which was applied to evaluate the model's prediction accuracy. The testing itemsets were formed by randomly selecting 30 log transactions from the testing data. These log transactions are divided into 3 group, for each group has 10 transactions. These groups keep approximately 10, 20 and 30 items, respectively. Finally, we test 3 groups in this study.

The study case uses Apriori algorithm to generate association rules. The minimum support (MS) and the minimum confidence (MC), considered were (1%, 25%) and (2%, 25%), respectively. If MS and MC are (1%, 25%), this case will generate more then 228,000 rules. The format of rule is represented as follows: $\{i_1,i_2, \ldots,i_m\}$ be a set of literals or called items. For each item represents a changed program. These rules had 262 different items for MS=1% and MC=25%. We translate these rules into a matrix. The matrix likes scattered and one item-attribute corresponds to one dimension. This matrix has 262 dimensions. If the data dimension is high, the human cognitive task for detecting correlations or discover hidden patterns are very hard. In order to deal with high dimensional data, we translate n-dimensional data to n(n-1)/2 matrices. A data point in two interval dimensions is represented by a 2-dimensions and corresponds to the class. The detail step is described in [13].

Finally, the study on average keeps 22 attributes to classify with MLP. The MLP is trained for classification those association rules. Finally, those rules are divided into 18 categories. The learning curve is shown in Figure 3. The objective of classification can improve the efficiency that the rule is found.

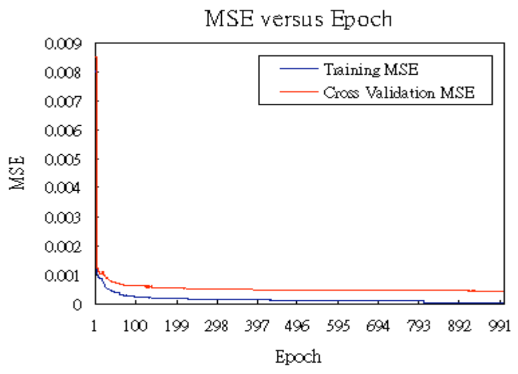

Figure 3: The learning curve.

In Figure 2, the block denoted as "Search Rule with Longest Match" is significant for the model. The longest-match scheme is adopted in this study. The rationale of longest-match is based on the conclusion that longer surfing path will contain more accurate and richer information about the user access pattern than the shorter ones [12]. The testing data depends on decision tree to determine the classification of rules, and locate the proper rule from this classification. Finally, the study uses the longest-match method to recommend the proper rule for programmers. For example, assume that the testing data is given by {CU_C01, CU_M01, GB_M01, GB_M41}.
Rule 1: {CU_C01, CU_M01, GB_M01} with support (S)=2% and confidence (C)=30%.
Rule 2: {CU_C01, CU_M01,GB_M01,GB_M41} with S=1.4% and C=29%.
Rule 3: {CU_A01,CU_C01,CU_M01,GB_M01,GB_ M41} with S=1.4% and C=28%.

The lengths of Rules 1, 2, and 3 are 3, 4 and 5, respectively. The numbers of matched items in Rules 2 and 3 are more than that of Rule 1. The number of matched items of Rules 2 and 3 are equivalent for the testing data, but the confidence of Rule 2 is more than that of Rule 3. In this case, the longest-match scheme specifies that Rule 2 is the best matched.

Precision and recall that measure the performance of a heuristic are defined as in standard information retrieval concepts [9]. Precision and recall must be considered together to measure the accuracy of the model. Considering the product of the factors can represent the join effect of the two factors.

Tables 1 and 2 summarize the results for banking application under association rule(AR) and association rule with classification(AR+C), respectively. Analytic results indicate that the average precisions of AR and AR+C were over 0.9. Since this investigation uses longest-match method to search, the precisions obtained from AR and AR+C two methods were also very close. Tables 1 and 2 are shown that the precision and recall of AR+C are better than AR on average.

Table 1: The results of precision (P) and recall (R) for AR

Number of test items	AR (2%,25%)			AR (1%,25%)		
	P	R	P*R	P	R	P*R
10	1.000	0.417	0.417	1.000	0.500	0.500
20	1.000	0.204	0.204	1.000	0.426	0.426
30	0.900	0.272	0.245	1.000	0.500	0.500
Average	0.967	0.298	0.289	1.000	0.475	0.475

Table 2: The results of precision (P) and recall (R) for AR+C

Number of test items	AR+C (2%,25%)			AR+C (1%,25%)		
	P	R	P*R	P	R	P*R
10	1.000	0.500	0.500	1.000	0.667	0.667
20	0.833	0.426	0.355	1.000	0.544	0.544
30	1.000	0.513	0.513	0.900	0.513	0.462
Average	0.944	0.480	0.456	0.967	0.575	0.558

The longest-match is a sequential search method. The study applies MLP to partition association rules. The first step, we find out the affiliated classification, then use longest-match method to find the proper rule from those rules of classification. The study that we perform in this validation took a few seconds on a Windows 2000 operation system with 512 MB RAM and P4-1.7 GHz Intel processors. From Tables 1 and 2 with (1%,25%), the average search time of AR and AR+C are 245.6 and 31.2 seconds, respectively. According to the study case the accuracy and efficiency of AR+C are better than AR.

4. Conclusions

In this paper we presented an approach for discovering matching rules in large software systems. It is much cost and time consuming for searching the proper rule. The study case produces more than 228,000 rules and with high dimensionally. The high dimensionality of scientific datasets can also be a problem in storage and retrieval. The proposed model can reduce dimension and MLP is used to classify rules. Our approach can be useful in suggesting further changes to be made and in warning about missing changes.

5. References

[1] Jane Huffman Hayes, Naresh Mohamed and Tina Hong Gao, Observe-mine-adopt (OMA): an agile way to enhance software maintainability, Journal of Software Maintenance and Evolution: Research and Practice, 2003, pp.297-323.

[2] R. J. Turver, and M. Munro, An Early Impact Analysis Technique for Software Maintenance, Software Maintenance: Research and Practice, Volume 6, 1994, pp. 35-52.

[3] Fyson M. J., Boldyreff C., Using Application Understanding to support Impact Analysis, Journal of Software Maintenance: Research and Practice, 10, Wiley, 1998, pp. 93-110

[4] Zhifeng Y, Rajlich V., Hidden dependencies in program comprehension and change propagation, Proceedings 9th (IWPC 2001). IEEE Computer Society Press, 2001; pp.293–299

[5] J. Sayyad Shirabd, T.C. Lethbridge, and S. Matwin, Mining the Maintenance History of a Legacy Software System, ICSM'03, September 22-26, 2003, pp.95-104.

[6] J. Sayyad Shirabd, T.C. Lethbridge, and S. Matwin, Supporting Software Maintenance by Mining Software Update Records, Proceedings of the 17th IEEE ICSM, 2001, pp. 22-31.

[7] R. Arnold and S. Bohner, Impact analysis - toward a framework for comparison, In IEEE ICSM 1997, Montral, Quebec, Canada, 1993, pp.292–301.

[8] S. Bohner and R. Arnold, Software Change Impact Analysis. IEEE Computer Soc. Press, 1996.

[9] Ahmed E. Hassan and Richard C. Holt, Predicting Change Propagation in Software Systems, Proceedings of ICSM 2004, September 11-17, 2004, pp.284-293

[10] A. T. Ying, G. C. Murphy, R. T. Ng, and M. C. Chu-Carroll, Predicting Source Code Changes by Mining Change History, IEEE Transactions on software engineering, VOL. 30, NO. 9, September 2004,pp. 574-586

[11] R. Agrawal, T. Imielinski, and A. N. Swami, Mining association rules between sets of items in large databases, In Proceedings of the International Conference on Management of Data, Washington, D.C., 1993, pp.207-216.

[12] Qiang Yang, Tianyi Li and Ke Wang, Building Association Rule Based Sequential Classifiers for Web Document Prediction, Journal of Data Mining and Knowledge Discovery, Volume 8, Issue 3, May 2004. pp.253-273.

[13] F.Poulet, Cooperation between automatic algorithms, interactive algorithms and visualization tools for visual data mining. In Proc, Visual Data Mining workshop, PKDD2002, 2002.