Improving Naïve Bayes Classifier for Software Architecture Reconstruction

Zahra Sadri Moshkenani
Faculty of Computer Engineering
Najafabad Branch, Islamic Azad University
Isfahan, Iran
zahra_sadri_m@sco.iaun.ac.ir

Sayed Mehran Sharafi
Faculty of Computer Engineering
Najafabad Branch, Islamic Azad University
Isfahan, Iran
mehran_sharafi@iaun.ac.ir

Bahman Zamani
Department of Computer Engineering
University of Isfahan
Isfahan, Iran
zamani@eng.ui.ac.ir

***Abstract*—Documentation of software architecture is a good approach to understand the architecture of a software system and to match it with the changes needed during the software maintenance phase. In several systems like legacy or older systems these documents are not available or if available; they are not up-to-date and usable. So, reconstruction of architecture in order to maintain these systems is a necessary activity. When we talk about software architecture, usually we are looking for a modular view of architecture with low coupling and high cohesion. In this paper, we try to improve the algorithm of machine-learning which is presented before for architecture recovery, and we propose using it for architecture reconstruction in order to obtain optimum modularity in the architecture with low coupling and high cohesion. The proposed algorithm is evaluated in a case study, and its results are presented.**

Keywords-software architecture reconstruction, reverse engineering, machine learning, data mining, naïve Bayes classifier

I. INTRODUCTION

When implementing software, software architecture is always considered as the base of software implementation process and has the key role in developing software and its evolution. Software architecture results in better response to the needs of people interested in the software being developed, ease of software design and implementation, easier support and maintenance of the software, and also simpler and faster software evolution. In architecture design, designing modules with low coupling and high cohesion increases the reusability of the modules and helps develop the next similar software more easily and faster, and also helps maintain currently-developed software. Documentation of architecture is a good approach to understand the architecture of a software system and to match it with the changes needed during software maintenance process. In several systems, like legacy systems, such documents are not available and if so, they are out-of-date. Reconstruction of architecture due to maintaining these systems is a necessary activity.

Among the terms referring to software architecture reconstruction process, two terms: recovery and discovery, are more frequently used. The process of reconstructing architecture which is performed bottom-up is called recovery, and the process of reconstructing architecture performed top-down is called discovery [1]. Reconstruction has 2 phases [2]: Reverse engineering and forward engineering. The former, sometimes called architecture recovery, concentrates on understanding the architecture of the existing system and concludes dynamic and static views of software. The latter, sometimes called architecture discovery is concerned with rebuilding the architecture of the system using modern technologies.

Software architecture reconstruction methods usually use machine learning, data mining or a combination of the two techniques. These techniques act intelligently, i.e. they classify the elements in the architecture of a system with respect to their specific properties, e.g. access to global variables. So, the architecture will be recovered exactly only if the primary architecture, that the software is designed based on it, is created regarding all principles of architecture design. For this reason, in this paper we plan to improve the algorithm of machine learning which is presented before in [3] for architecture recovery, and use it for rebuilding architecture in order to obtain an optimum modularity in the architecture with low coupling and high cohesion.

This paper continues in the following sections: section 2 presents a brief introduction to machine learning and data mining. Section 3 explains the approach suggested in [3] for architecture recovery using naïve Bayes classifier. Section 4 describes evaluation method and a case study for evaluating the performance of improved naïve Bayes. In section 5, difficulties identified with the approach suggested in [3], are introduced along with the proposed solutions to solve them with the aim of improving naïve Bayes classifier. Section 6 presents the results of evaluation of improved naïve Bayes classifier. Section 7 suggests how to use machine learning, and particularly naïve Bayes classifier in data mining context, in order to rebuild the architecture with low coupled and high cohesive modules. Section 8 concludes the paper.

II. MACHINE LEARNING AND DATA MINING

Machine learning involves techniques that are subset of artificial intelligence and enables the computer to gain knowledge from data sets and from the behaviors that are faced (i.e. the environment the computer acts in); so the computer changes its behavior according to the environment and its changes. Using these techniques and having the

978-1-4673-2467-0/12/$31.00 ©2012 IEEE

source code or the non-complete or out-of-date architecture, one can rebuild the architecture of legacy software systems.

The goal and main task of machine learning techniques is classification, also known as pattern recognition. By pattern we mean the behavior seen amongst a set of data in the system. As an example, the behavior that 90% of customers who buy item A buy item B or E too but they do not buy item C, is a pattern.

In machine learning techniques, machine learns to classify various samples in the system based on their patterns. This means that by making intelligent decisions based on extracted patterns, the machine puts each sample in the class related to its pattern.

Data mining is the process of discovering new patterns from large data sets and uses methods that are in the intersection of artificial intelligence, machine learning, statistics and probabilities, and database systems. Data mining techniques are very useful in reverse engineering and (those) tasks related to system maintenance [2]. These techniques help discover relationships and dependencies among components of the system. Also, they have the ability to work on large data sets without having any pre-knowledge about the system. So, using these techniques in reconstructing the architecture of legacy systems or systems with non-complete or out-of-date documents is useful. The goal of data mining is to extract knowledge out of a data set. This is achieved in 4 ways [2]:

a) Classification: To decide which pre-determined class to put a component in. The process of making decision is performed using extracted patterns.

b) Clustering: The process of using similarity function to decide about putting elements in which pre-determined classes based on their properties.

c) Association: Patterns extraction

d) Sequence: To recognize the sequence of events, e.g. if a customer bought item A, then in 30% of cases, he will also buy item B within next 2 weeks.

The task of data mining is to discover patterns which are not pre-known, among a lot of data. These patterns are used in machine learning to teach the learner so that it can classify elements correctly in pre-determined classes considering their patterns. Data mining techniques discover relations between data and then perform classifying, clustering, pattern extracting, recognizing sequence of events by using data sets, and extract unknown information about the data without having any pre-knowledge about them. While machine learning algorithms teach the learner to predict and make decisions about new data by using pre-knowledge. Some data mining applications use machine learning algorithms and vice versa.

The main method for reconstructing architecture is a 3-phase process:

a) Recovery: Storing the information related to objects and components of a system such as access to the variables.

b) Discovery: Extracting needed information by applying one of data mining algorithms on the obtained information in a, and clustering components within the system using similarity function (in fact, using machine learning algorithms).

c) Integrating obtained information in a and b, and demonstrating reconstructed architecture.

III. BAYESIAN LEARNING AND NAÏVE BAYES CLASSIFIER

Naïve Bayes classifier is a classification algorithm based on Bayes theory, and it is shared in both data mining and machine learning techniques. Assuming that n attributes of a random variable are conditionally independent, this classifier predicts which class is the best for the random variable. Bayesian learning is a machine learning algorithm which uses naïve Bayes classifier. In learning problems, the important issue is to find an assumption which has the most occurring probability value in the case when value vector for random variable's properties vector exists. Such an assumption is called Maximum a Posterior (MAP) [3]. Equation (1) is the Bayes formula which is used in naïve Bayes classifier as similarity function.

$$Y_{MAP} = \text{argmax}_{y_k} P(Y = y_k) \prod_i P(X_i|Y = y_k) \quad (1)$$

This means that we select the k'th value of Y that causes most probability value to be obtained, as Y_{MAP}. In (1), Y is a random variable and X is a vector of properties related to Y. In other words, $X=<X_1,X_2, \dots ,X_m>$ where X_l shows 1st property of Y. y_k is k'th value for random variable Y and X_i is i'th vector for considered properties. Part $\prod_i P(X_i|Y = y_k)$ shows the weight or relative frequency of appearing (occurring) properties of y_k in the sample space. Example 1 explains how naïve Bayes classifier operates in architecture recovery problems and is derived from [3]. Note the following parameters:

f_i: i'th function that we are determining the best subsystem to include the function.

s_j: j'th subsystem that we are checking.

g_k: k'th global variable.

$P(f_i | s_j=T)$: The probability of f_i belongs to subsystem s_j considering access to specific variables.

scount[j]: The number of functions already included in subsystem s_j.

gcount[k]: The number of functions belonging to s_j and having access to g_k.

fcount: the number of functions whose subsystems have been determined.

TABLE I. EXAMPLE 1

Function	g_1	g_2	g_3	Subsystem
f_1	T	T	F	s_1
f_2	F	T	T	s_2
f_3	T	F	T	s_1
f_4	T	F	F	s_2

Example 1: Table I. shows each function's access to global variables and the subsystem that includes the function, e.g. the first row of the table shows that f_1 has access to g_1 and g_2 and is included in s_1.

Suppose that f_5 is recently added to this system and has access to g_1, g_2, and g_3, (the value of vector X for f_5 is <T,T,T>). To recognize which subsystem is better to include f_5, we do as follows (It is necessary to mention that in [3] Bayesian learning is used to recover architecture and not to rebuild it. That is to say the objective of presenting this example in [3] and using Bayes learning to solve it, has been to recognize which subsystem has included f_5, but not to answer which subsystem is better to include the f_5):

First we compute the probability of f_5 belonging to s_1 considering that f_5 has access to g_1, g_2 and g_3. Then, we compute the probability of f_5 belonging to s_2 considering that f_5 has access to the same variables. Then, we select the subsystem that causes more probability to include f_5. Regarding [4] and the solved example in [3], the following equations have been used to compute probabilities:

$$P(s_j) = scount[j]/fcount \quad (2)$$

$$P(g_k = T|s_j) = gcount[k]/scount[j] \quad (3)$$

The solution of the example is as follows:

$$P(f_5|s_1 = T) = P(s_1 = T|g_1 = T, g_2 = T, g_3 = T)$$
$$= P(s_1)P(g_1 = T|s_1)P(g_2 = T|s_1)P(g_3 = T|s_1)$$
$$= 2/4 \times 2/2 \times 1/2 \times 1/2 = 1/8$$
$$P(f_5|s_2 = T) = P(s_2 = T|g_1 = T, g_2 = T, g_3 = T)$$
$$= P(s_2)p(g_1 = T|s_2)P(g_2 = T|s_2)P(g_3 = T|s_2)$$
$$= 2/4 \times 1/2 \times 1/2 \times 1/2 = 1/16$$

$P(f_5|s_1=T)$ is the probability of f_1 to be included in s_1. $P(s_1=T|g_1=T,g_2=T,g_3=T)$ is the probability of s_1 to have access to g_1, g_2, g_3. According to the solved example, noting that $P(f_5|s_1=T) > P(f_5|s_2=T)$ (i.e. existing functions in s_1 have more access to the 3 variables than functions in s_2), then s_1 is selected for f_5.

IV. Method and Case Study System to Evaluate Proposed Classifier

Because provided algorithm in [3] is evaluated using k-fold validation method, on Mosaic 2.6, which is an open source web browser, we will evaluate improved naïve Bayes classifier on the same system using the same evaluating method.

Mosaic 2.6 includes 11 subsystems, 818 routines and 348 global variables. Table II presents subsystems within Mosaic 2.6 [3]. Regarding [3], since mail-processor and mosaic-comments subsystems include relatively few functions, we suppose that these two subsystems are not in Mosaic 2.6 and omit their functions from functions set.

To evaluate the proposed (improved) naïve Bayes classifier, we use 2 set types, training set and test set. Training set is a one in which classification of samples is specified and is used to train the classifier. To evaluate naïve Bayes classifier performance, test set should be used. Test set means a set of functions that are not classified yet and the trained classifier should classify them. Members of training set and test set are specified using k-fold validation.

K-fold validation is a technique for assessing how the results of a statistical analysis will be generalized to an independent data set. In this technique, the main set is divided into k equally sized sets. One of this k sets is test set and the k-1 remaining sets are training sets. K-fold validation runs k times and each of k sets is selected as test set exactly once.

In this paper, evaluation has been performed by 3-fold validation. There are several errors in functions divide which have been performed in [3] so we present the modified sets as table III, table IV and table V.

V. Difficulties of Bayesian Learning

A. Zero probabilities

If the value of an operand in calculating the probability using (1) becomes 0, the result will be 0. It means that under the check function cannot be placed in any of existing subsystems and must be included in a new subsystem. Please pay attention to example 2.

Example 2: The newly added function to this system (Table VI.) is f4 and has access to g1 and g2. Now, using naïve Bayes classifier and considering the low coupling and high cohesion, we want to determine the best subsystem that f4 can be placed within. According to (1) YMAP is:

$$\max\begin{cases} P(s_1)P(s_1|g_1 = T)P(s_1|g_2 = T) = 0 \\ P(s_2)P(s_2|g_1 = T)P(s_2|g_2 = T) = 0 \end{cases}$$

So YMAP=0 and it means that f4 belongs to neither s1 nor s2 and a new subsystem like s3 is needed. Perhaps one concludes that YMAP= 0 means that f4 can be placed in every subsystem; but if the effective probabilities in calculating YMAP, are the same and are not zero, this conclusion will be true. Considering that f4 has access to g1 and g2, adding a new subsystem s3 for f4 will cause low data cohesion within the subsystems and

TABLE II. Mosaic 2.6

subsystem name	Number of functions
User-interface-manager	219
Cci-manager	144
Mosaic-manager	113
Helpers	85
Image-processors	64
Annotations	52
Hotlists-manager	49
History-manager	45
Newsgroup-manager	30
Mail-processor	11
Mosaic-comments	6

TABLE III. Set 1

Subsystems	Functions
User-interface-manager	72
Cci-manager	45
Mosaic-manager	38
Helpers	27
Image-processors	22
Annotations	18
Hotlists-manager	18
History-manager	15
Newsgroup-manager	12
Total	267

TABLE IV. Set 2

Subsystems	Functions
User-interface-manager	72
Cci-manager	48
Mosaic-manager	39
Helpers	28
Image-processors	21
Annotations	19
Hotlists-manager	16
History-manager	15
Newsgroup-manager	9
Total	267

TABLE V. Set 3

Subsystems	Functions
User-interface-manager	75
Cci-manager	51
Mosaic-manager	36
Helpers	30
Image-processors	21
Annotations	15
Hotlists-manager	15
History-manager	15
Newsgroup-manager	9
Total	267

high data coupling between them, but as mentioned before, we are going to reconstruct architecture with high data cohesive and low data coupled subsystems. Solving the zero probabilities becomes more important when sparse data are under the check sets, i.e. they include many zero value data.

As a solution to this difficulty, we can use Laplacian correction or Laplacian estimator. If under the check set is large enough so that adding one unit to each denominator in probability fraction makes negligible changes, then Laplacian correction is suitable to solve zero probabilities problem. The first difficulty with Laplacian correction is the afore-mentioned assumption. It is not clear what large enough means, and sometimes it is possible that under the check set is not large enough, and changes like this will greatly affect all probabilities. Another difficulty in Laplacian correction is recalculating overhead time for recently calculated probabilities. In calculating probabilities when we face zero probability, we must apply the correction in all probabilities, calculated or not calculated so far. For calculated probabilities we

TABLE VI. EXAMPLE 2

Function	g_1	g_2	g_3	Subsystem
f_1	T	F	T	S_1
f_2	T	F	T	S_1
f_3	F	T	T	S_2

must recalculate the probabilities applying Laplacian correction which enforces the calculator system to tolerate recalculation overhead of time. We have proposed an approach to be used instead of Laplacian correction to solve zero probabilities problem which applies improvement in naïve Bayes classifier performance.

When, in calculating probabilities, at least one of the operands equals 0, we put 1 instead of that operand(s).

Now a new problem will be raised which is described by the following example.

$$P(f_1|s_1 = T) = 1/4 \times 0 \rightarrow \text{correction} \rightarrow 1/4 \times 1 = 1/4$$
$$P(f_1|s_2 = T) = 1/16$$

These calculated probabilities resulted that f1 must be a member of s1, while we know that before applying the correction, s2 used to be the appropriate subsystem for f1. So, when applying correction to an operand, set the value of a boolean variable like isZero to true. Having a probability calculated, check the flag isZero. If it is true, then multiply calculated probability by -1. If in computing probability more than one operand has 0 values, we set the flag isZero equal to true only once.

$$P(f_1|s_1 = T) = 1/4 \rightarrow \text{isZero} = \text{true?} \rightarrow P(f_1|s_1 = T) = -1/4$$
$$P(f_1|s_2 = T) = 1/16$$

And it illustrates f_1 must be considered as a member of s_2. Now if we have:

$$P(f_1|s_1 = T) = 1/4 \times 0 \rightarrow \text{correction} \rightarrow 1/4 \times 1 = 1/4, \text{isZero} = \text{true} \rightarrow P(f_1|s_1 = T) = -1/4$$
$$P(f_1|s_2 = T) = 1/8 \times 0 \rightarrow \text{correction} \rightarrow 1/8 \times 1 = 1/8\,, \text{isZero} = \text{true} \rightarrow P(f_1|s_2 = T) = -1/8$$

Then, regarding that minus sign in a probability only shows that the correction has applied to this probability, the minimum probability between the two probabilities ought to be selected, because the operands which participated in calculating P(f1|s1=T) have greater values than those that took part in calculating P(f1|s2=T) and because we need data cohesive subsystem, we select s1 for f1. Another problem with this approach is separating these two states:

$$P(f_1|s_1 = T) = 1 \times 0 \times 0 \rightarrow \text{correction} \rightarrow -1$$
$$P(f_1|s_2 = T) = 1 \times 1 \times 0 \rightarrow \text{correction} \rightarrow -1$$

Considering these two calculated probabilities, s2 will be selected for f1, because functions in s2 have access to global variables more than functions in s1. Using a counter which counts the number of real 1s, this problem will be solved. Assign a counter to each subsystem and before applying the correction on probabilities, check the operands' values. For each value which equals 1, increase the counter by 1. Now if the calculated probabilities for two subsystems are equal, compare the subsystems' counter value. Each subsystem that has greater counter value will be selected for f1.

Having the presented approach applied, if we come to two subsystems that although the maximum probability is obtained for them, still have equal value of probability, then the one with more functions should be selected.

B. Equation for Computing $P(s_j)$

Of course, this difficulty appears only in problems related to discovery or evolution of architecture because we are not going to find a pattern. As said before, P(sj) is computed using (2). In (2) the more functions exist in subsystem j, the more value is obtained for P(sj). Regarding statistics and probabilities, computing probability of subsystem using (2) is not correct because when selecting a subsystem for the function that is being checked, our sample space members are all existing subsystems in the system, while (2) considers number of functions whose subsystems have been determined as sample space members. Another wrong result that computing probability of subsystem causes to take place, using (2), is wrong classifying, i.e. to locate a function in a wrong subsystem because the number of

repetitions of a subsystem is more/less than the others. Modified equation for (2) is (4).

$$P(s_j) = 1/\text{subsyscount} \quad (4)$$

Where, subsyscount is the number of subsystems that exist in mosaic system.

Note the case that was seen in the Mosaic system. In the Mosaic system, when implemented, naïve Bayes classifier using (2) (i.e. according to [3]), 15 cases of functions are classified wrong. However, if we use (4) instead of (2) in implementation of the classifier, these 15 functions are also classified in the right subsystem.

There are 9 subsystems in mosaic system. Therefore, the probability of selecting each subsystem to include the function is 1/9. Now, as an example, we mention a sample function that exists in mosaic and was wrongly classified before improving the equation but after improving it, the function is concluded in a correct subsystem. The function f_{423} has access to g_{86} and g_{116}. Using (2), the computed probabilities and suggested subsystem for this function has been as below(of course, we compute probabilities only for the two subsystems, out of 9 existing subsystems that finally compete with each other, user-interface-manager or UIM and cci-manager or CCI).

$$p(f_{423}|\text{UIM} = T) = p(\text{UIM}) \times p(g_{86} = T|\text{UIM}) \times p(g_{116} = T|\text{UIM}) = 0.275 \times 2.916 \times 10^{-3} = 8.019 \times 10^{-4}$$

$$p(f_{423}|\text{CCI} = T) = p(\text{CCI}) \times p(g_{86} = T|\text{CCI}) \times p(g_{116} = T|\text{CCI}) = 0.179 \times 3.795 \times 10^{-3} = 6.793 \times 10^{-4}$$

Considering the computed probabilities, the appropriate subsystem for f423 is UIM. While calculated value for part $\prod_i P(X_i|Y = y_k)$ in (1), which shows the weight or relative frequency of accessing to the global variables by subsystem sj when CCI is under the check, is 3.7952×10^{-3} and when the UIM is being checked is 2.9160×10^{-3}. It means that the probabilities of accessing to the g86 and g116 from the CCI are more than those of the UIM. Yet, since the number of repetitions of the UIM has been more than that of CCI among the functions whose subsystem has been determined, i.e. there has been a greater number of functions in this subsystem, the obtained probability for the UIM has become greater and has removed the effect of less access probability to the global variables from this subsystem and results more probability to select UIM. This, i.e. to locate the f423 in UIM, causes low data cohesive and high data coupled subsystems. Noting the existing architectural documents of mosaic, the function f_{423} is included in CCI. Using (4) in mosaic system following results is obtained for the function f_{423}.

$$p(f_{423}|\text{UIM} = T) = p(\text{UIM}) \times p(g_{86} = T|\text{UIM}) \times p(g_{116} = T|\text{UIM}) = 0.111 \times 2.09 \times 10^{-3} = 3.23 \times 10^{-4}$$

$$p(f_{423}|\text{CCI} = T) = p(\text{CCI}) \times p(g_{86} = T|\text{CCI}) \times p(g_{116} = T|\text{CCI}) = 0.111 \times 3.79 \times 10^{-3} = 4.20 \times 10^{-4}$$

Regarding the obtained probabilities, the CCI will be selected for locating f423. Noting high cohesion and low coupling and existing documents of mosaic system, this selection is a right selection.

Computing access probability for each global data is to compute the probability of a pattern to take pace following naïve Bayes classifier. But considering the mentioned difficulty in this subsection, the basic problem is that in patterns that occur, the subsystem that has most functions in itself gains more probability of being selected. So, we consider the selecting probability of subsystems to be equal. In fact, computed probabilities for accessing every global data determines the weight or relative frequency of accessing global data by existing functions in a specific subsystem; e.g. access to g12 via the functions in s1 is 0.3 meaning that the existing functions in s1 have access to g12 with relative frequency of 0.3.

VI. Evaluation Results for Proposed Naïve Bayes Classifier

The results obtained from applying evaluation to the improved algorithm of Bayes learning on the mosaic system, to recover (not reconstruct) architecture, have been shown in table VII.

Table VII shows the proposed (improved) naïve Bayes classifier evaluation results. For example, in this table, data exist in first row, and the second column means that after the zero probabilities is solved, the classifier correctly classifies 114 out of 267 functions that exist in the first data set. Since different data sets result in different outcomes, we could not attain the numbers of the evaluation in [3] and we compared the proposed classifier evaluation results with those of evaluating the classifier which we implemented according to [3]. The important issue is: in each evolution with different data sets, always the proposed classifier leads to better outcomes.

TABLE VII. Evaluation results

data set	correctly classified (naïve Bayes classifier)	correctly classified (naïve Bayes classifier, only first improvement is applied)	correctly classified (improved naïve Bayes classifier, both improvements are applied)
1	114 out of 267=42.7 %	137 out of 267=51.3 %	142 out of 267=53.1 %
2	96 out of 267=36%	118 out of 267=44.2 %	124 out of 267=46.4 %
3	97 out of 267=36.3%	132 out of 267=49.4 %	136 out of 267=50.9%
Average	38.3%	48.3%	50.1%

In [3], Bayesian learning and particularly naïve Bayes classifier is proposed as a method for architecture recovery. Since the goal of classifier which is proposed in [3] is subsystems' data cohesion, it intelligently chooses functions which have access to the same global variables to place them into the same subsystem. Because as the improved classifier evaluation's outcome illustrates, the maximum average percentage of correctly classified functions is 50.1% and as Bayesian learning algorithm is a common algorithm in both machine learning and data mining we propose using this classifier in forward engineering and software evolution phase instead of recovery (Reverse engineering) phase. As

proposed classifier concentrates on data cohesive subsystems, using the classifier in the mentioned phases, results in high data cohesive and low data coupled subsystems.

In the forward engineering phase, Bayesian learning algorithm will be used to diagnose classes and to build new architecture by calculating the probabilities; i.e. the use of data mining. In this phase, machine learning (in fact Bayesian learning) algorithm will be used to cluster functions that exist in source code in order to obtain new optimum architecture. On the other hand, Bayesian learning algorithm can be used in software evolution phase to determine that under the check function, using predetermined classes(by data mining) and information about accesses to global variables, and regarding high cohesion and low coupling, should be included in which subsystem; means the use of machine learning.

VII. CONCLUSIONS AND FUTURE WORK

This paper explains the difficulties with naïve Bayes classifier in software architecture context and presents an approach to solve them. Since the proposed classifier acts intelligently and constructs data cohesive subsystems, the concluded architecture from this classifier includes high data cohesive and low data coupled modules. This architecture will correspond to the architecture that the software is based on, if in designing base module view of architecture, low coupling and high cohesion be regarded and for each function only one appropriate subsystem exist. So, it is suggested that we use the proposed classifier for architecture reconstruction and obtain the optimum architecture using source code and also for architecture evolution instead of using the classifier for architecture recovery.

As a future work, we plan to develop an automated tool to use the proposed classifier in both mentioned processes (architecture reconstruction process and architecture evolution process).

ACKNOWLEDGEMENT

We want to acknowledge H.A Babri and O. Maqbul, writers of [3], for providing us with the needed data for the evaluation of the proposed classifier.

REFERENCES

[1]Ducasse Stephane, and Pollet Damien, "*Software Architecture Reconstruction: A Process-oriented Taxonomy"*, IEEE Transaction on Software Engineering, 2009 ,35(4): 573-591

[2]Montes de Oca, C. Carver, and D.L, "*Identification of Data Cohesive Subsystems Using Data Mining Techniques"*, International Conference on Software Maintenance, 1998,16-23

[3]Maqbool O., and Babri H.A, "*Bayesian Learning for Software Architecture Recovery"*, International Conference on Electrical Enginnering(ICEE 07), 2007 ,1-6

[4]Mitchell Tom, "*Machine learning"*, 2nd Edition, Mc Graw Hill, 1997

[5]Bishop Christopher M., "*Pattern Recognition and Machine Learning"*, 1st Edition, New York, Springer, 2006

[6]Witten Ian, Frank Eibe, and Hall Mark, "*Data Mining: Practical Machine Learning Tools and Techniques"*, 3rd Edition, Morgan Kaufman, 2011

[7]Sartipi K, Kontogiannis K, and Mavaddat F, "*Architectural Design Recovery using Data mining Techniques"*, Proceedings of the Fourth European Software Maintenance and Reengineering, 2000 ,129-139

[8]Yanbing Guo George, Atlee Joanne M., and Kazman Rick, "*A Software Architecture Reconstruction Method"*, Proceedings of the TC2 First Working IFIP Conference on Software Architecture, 1999,15-3

[9]Bass Len, Clements Paul, Kazman Rick, *Software Architecture In Practice*, 2nd Edition, Addison Wesley, 2003