Autonomous Integration of Induced Knowledge into the Expert System Inference Engine

Nittaya Kerdprasop, *Member, IAENG*, and Kittisak Kerdprasop

***Abstract*—The iterative process of data mining comprises of three major steps: pre-data mining, mining, and post-data mining. Pre-data mining is the preparation of data to be in a suitable format and contain a minimal but sufficient subset of relevant features. The mining step concerns the application of appropriate learning method to the well prepared data. Post-data mining step evaluates and employs the learning results. This research studies the post-data mining processing. Most data mining systems finish their processing at the knowledge presentation of the mining step. Our work, on the contrary, extends further the post-data mining processing to the level of knowledge deployment. This paper illustrates the knowledge deployment step in which its input is the induced knowledge, in the formalism of traditional classification rules. These rules are then evaluated and filtered on the basis of coverage measurement. High coverage rules are transformed into decision rules to be used by the inference engine of the expert system. The coupling of induced decision rules into the expert system shell is designed to be an automatic process. The accuracy of recommendation given by the expert system is evaluated and compared to other classification systems and the real diagnosis given by medical doctors. The experimental results confirm the high accuracy of our expert system and the induced knowledge base.**

***Index Terms*—Decision rules, Automatic knowledge base creation, Expert system inference engine, Logic-based system**

I. Introduction

DATA mining is a novel intelligent technology of the late decades aiming at automatic discovery of novel and useful knowledge from large repositories of data. Most data mining systems fulfill this main purpose by discovering a lot of potential knowledge. Unfortunately, the discovered knowledge is also abundant, especially in a specific task of association rule mining [2], [4], [6], [13]. Actionable and useful knowledge is hardly to pinpoint and extract from a large stack of redundant, irrelevant, and sometimes obvious and uninteresting information.

Manuscript received December 13, 2010; revised January 20, 2011. This work was supported in part by grants from the National Research Council of Thailand (NRCT) and the Thailand Research Fund (TRF). The Data Engineering and Knowledge Discovery (DEKD) Research Unit has been continually supported by Suranaree University of Technology.

Nittaya Kerdprasop is the co-founder and principal researcher of the DEKD research unit. She is also an associate professor at the school of computer engineering, Suranaree University of Technology, 111 University Avenue, Nakhon Ratchasima 30000, Thailand (phone: +66-(0)44-224432; fax: +66-(0)44-224602; e-mail: nittaya@sut.ac.th).

Kittisak Kerdprasop is the director of DEKD research unit and the associated professor of the school of computer engineering, Suranaree University of Technology, 111 University Avenue, Nakhon Ratchasima 30000, Thailand (e-mail: kittisakThailand@gmail.com).

Interestingness is one important research issue since the beginning of the data mining as a new discipline [6], [10], [12], [14]. Piatetsky-Shapiro and Matheus [10] developed the KEFIR system to be used with the health insurance system. Interestingness of this system focuses on the deviation of the induced knowledge from its norm. Silberschatz and Tuzhilin [12] proposed a different criterion of evaluating interestingness. They considered the probabilistic belief as a main measurement. Other metrics that can be employed as interestingness measurement of the induced knowledge include coverage, confidence, strength, significance, simplicity, unexpectedness, and actionability [7], [10], [11]. Among these metrics, unexpectedness and actionability are the most difficult criteria to be evaluated systematically due to their subjective nature.

Most researchers deal with the interestingness problem during the mining step. One practical technique is to use the pruning method [3], [11] to reduce the number of induced knowledge. Another technique is the application of prior or domain knowledge during the mining step [5], [8], [9] in order to select only useful and relevant knowledge.

Although the techniques of pruning and applying domain knowledge can reduce the number of discovered knowledge, their main purpose is for the performance improvement of learning method rather than to evaluate and filter the discovered knowledge. The recent work of Adomavicius and Tuzhilin [1] proposed the validation technique after the mining step to select relevant knowledge.

Our work is also in the category of filtering knowledge after the mining step. Therefore, the proposed techniques can be considered as a post-data mining processing. We employ the coverage criterion as a basis for transforming the induced knowledge into the probably useful information for the recommendation system. We also extend the data mining process towards a tight coupling of the knowledge base system. The practicality aspect of our system is demonstrated through the expert system that can provide some useful recommendation to the general users.

II. Knowledge Evaluation and Integration Method

The proposed post-data mining processing technique is a final part of our SUT-Miner research project. The project aims at designing and developing a complete data mining system that can convey the induced knowledge to the systems that employ such knowledge. The framework of the SUT-Miner system is given in Fig.1. The system is composed of three main parts: Pre-DM, DM, and Post-DM. Pre-DM is the first part responsible for data preparation, whereas DM is a subsequent step of mining for knowledge.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

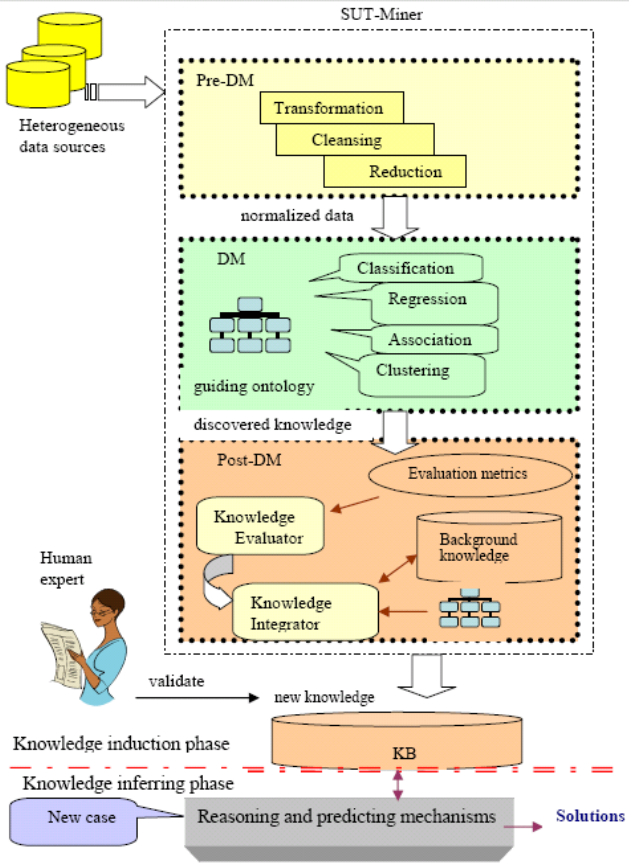


Fig. 1 Framework of the SUT-Miner system

Evaluating and selecting relevant and useful knowledge is the last part of the SUT-Miner system and also the main focus of this paper. We design the Post-DM part to consist of two main modules: knowledge evaluation, and knowledge integration and manipulation. The relationships of these two modules and other interfaces are given in Fig. 2.

The design of knowledge evaluation module (Fig. 3) is based on the decision-tree induction method [11] because the structure of induced tree is appropriate for generating reasoning and explanation in the expert system shell. The induced knowledge is in a formalism of decision rules incorporated with probabilistic values. This value is intended to be used as the degree of potential applicability of each decision rule. The probabilistic values are indeed the coverage values of decision rules.

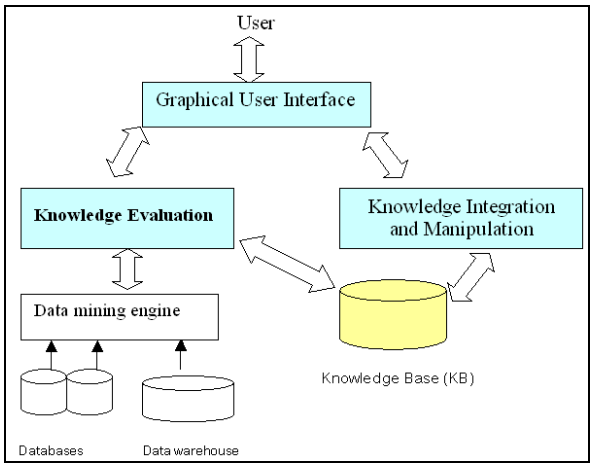


Fig.2 Conceptual design of post-DM modules

Algorithm 1 Knowledge evaluation
Input: a data model as decision tree with node and edge structures
Output: a set of probabilistic decision rules ranking in descending order
(1) Display GUI to get a dataset name and a minimum probability value
(2) Traverse tree from a root node to each leaf node
(2.1) Collect edge information and count number of data instances
(2.2) Compute probability as a proportion (number of instances at leaf node) / (total data instances in a data set)
(2.3) Assert a rule containing a triplet <attribute-value pair, class, probability value> into temporary KB
(3) Sort rules in the KB in descending order according to the rules' probability
(4) Remove rules that have probability less than the specified threshold
(5) Assert selected rules into the KB and return KB as an output

Fig.3 Knowledge evaluation algorithm

The output of algorithm in Fig.3 is the induced knowledge represented as decision rules. These rules are then transformed by algorithm 2 (Fig. 4) to be expert rules and consulting rules of the expert system. The expert rules are to be used by the inference engine for giving recommendation to general users. Consulting rules are for reasoning and giving explanation when requested by the users. Both expert and consulting rules are stored in a knowledge base file.

We implement the integrated data mining engine and the expert system shell with the logic-based system, therefore the expert and consulting rules are encoded as Horn clauses.

Algorithm 2 Knowledge integration and manipulation
Input: a set of probabilistic decision rules stored in KB
Output: a set of rules to be used by an expert system shell
(1) For each probabilistic decision rule
(1.1) Scan information in the If-part and the Then-part
(1.2) Generate head of expert rule from the Then-part
type(Then-part, probability value) :-
(1.3) Generate body of expert rule from the If-part
:- atrribute_name1(value), …, attribute_nameN(value).
(1.4) Write expert rule in a file, knb-file
(2) For each data attribute
(2.1) Generate head of consulting rule
attribute_name(X) :-
(2.2) Generate body of consulting rule
:- menuask(attribute_name, X, [list of attribute values]).
(3) Assert consulting rules into the knb-file and return KB as an output

Fig.4 Knowledge integration and manipulation algorithm

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

III. Implementation

A. Knowledge Induction

The first part of our implementation is the knowledge induction, which is the decision tree learning. The implementation is based on the logic-based system using the SWI Prolog (www.swi-prolog.org). The advantage of such logic-based system is that the input data, program coding, and output are all taking the same form, that is, Horn clauses.

For the purpose of demonstration, we use a small dataset named lens data from the UCI repository (available at http://archive.ics.uci.edu/ml/). The dataset contains 24 instances of patients under examination for proprietary of wearing contact lenses. The dataset in a format of Horn clauses are shown in Fig.5.

```
%% Data lens
%     attributes: names and their possible values
attribute(age, [young, pre_presbyopic, presbyopic]).
attribute(spectacle, [myope, hypermetrope]).
attribute(astigmatism, [no, yes]).
attribute(tear, [reduced, normal]).
attribute(class, [ yes, no]).
%     data
instance(1, class=no, [age=young, spectacle=myope,
                       astigmatism=no, tear=reduced]).
instance(2, class=yes, [age=young, spectacle=myope,
                       astigmatism=no, tear=normal]).
instance(3, class=no, [age=young, spectacle=myope,
                       astigmatism=yes, tear=reduced]).
instance(4, class=yes, [age=young, spectacle=myope,
                      astigmatism=no, tear=normal]).
```

Fig. 5 Examples of data instances represented as Horn clauses

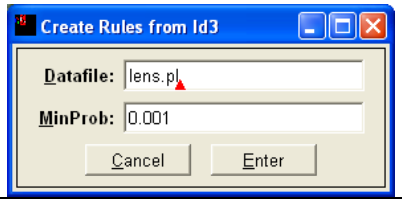
The steps in our main module of knowledge induction are given in Fig.6. The module 'mainDT' is to be called with the specified parameter 'Min'. This parameter is the probability threshold that programmers or users have to identify. It is the metric for pruning the final results of knowledge induction. Any decision rules with probability values less than this threshold will not be included in the knowledge base. The running result of this module displayed as a tree structure is given in Fig. 7, whereas the final result displayed as decision rules is in Fig. 8.

```
mainDT(Min) :-
      init(AllAttr, EdgeList),    % initialize node and edge info
      getnode(N),                 % get node ID
      create_edge(N, AllAttr, EdgeList),    % create tree
      addAllKnowledge,            % generate decision rules
      selectRule(Min, Res),       % select top rules
      tell('1.knb'),              % write selected decision rules to file
      writeHeadF,                 % transform to expert rules (head)
      maplist(createRule1, Res),
      nl, writeTailF,             % generate body of expert rules
      told,                       % write expert rules to file and close it
      writeln(endProcess).
```

Fig.6 Main module in a knowledge induction program

```
File  Edit  Settings  Run  Debug  Help
1 ?- listing(node).
:- dynamic node/2.

node(1, [2, 4, 6, 8, 10, 12, 14, 20, 22]-[1, 3, 5, 7
, 9, 11, 13, 15, 16, 17, 18, 19, 21, 23, 24]).
node(2, []-[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 2
3]).
node(3, [2, 4, 6, 8, 10, 12, 14, 20, 22]-[16, 18, 24
]).
node(4, [2, 4, 6, 8]-[]).
node(5, [10, 12, 14]-[16]).
node(6, [10, 12]-[]).
node(7, [14]-[16]).
node(8, [20, 22]-[18, 24]).
node(9, [20]-[18]).
node(10, [22]-[24]).

true.

2 ?- listing(edge).
:- dynamic edge/3.

edge(0, root=nil, 1).
edge(1, tear=reduced, 2).
edge(1, tear=normal, 3).
edge(3, age=young, 4).
edge(3, age=pre_presbyopic, 5).
edge(5, spectacle=myope, 6).
edge(5, spectacle=hypermetrope, 7).
edge(3, age=presbyopic, 8).
edge(8, spectacle=myope, 9).
edge(8, spectacle=hypermetrope, 10).

true.
```

Fig.7 Discovered knowledge displayed as tree structures



0.5 >> [tear=reduced] >> no,

0.166667 >> [tear=normal, age=young] >> yes,

0.0833333 >> [tear=normal, age=pre_presbyopic, spectacle=myope] >> yes

Fig.8 Running result of lens dataset with the probability threshold of 0.001

B. Knowledge Integration to the Expert System Engine

The second part of our implementation is the automatic generation of knowledge base and the inference engine of the expert system. Structure of our simple expert system is graphically shown in Fig. 9. The module to be used by the users is the 'menuask' module. Inference engine performs load and solve actions for loading the knowledge base into the working memory and solving the questions ask by the users, respectively.

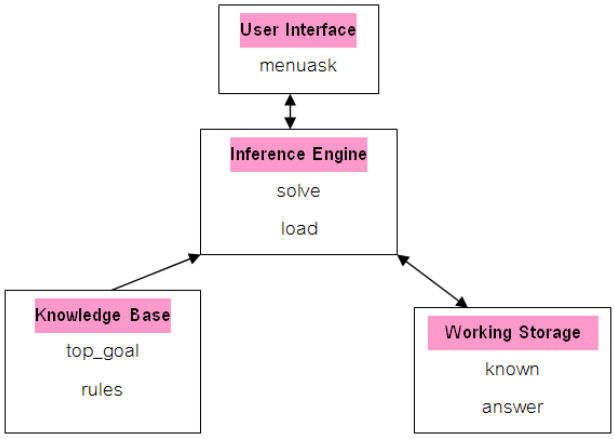


Fig.9 Structure of the expert system shell

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

The working storage performs action related to reasoning and giving explanation to the users. The knowledge base contains two kinds of rules: the expert rules and consulting rules. These rules are automatically transformed from the induced knowledge. The implementation of modules to generate expert rules is given in Fig. 10, whereas the modules for generating consulting rules are in Fig. 11.

```
writeHeadF :-
        format('% 1.knb ~n% for expert shell. --- written
                by Postprocess'),
        format('~n% top_goal where the inference
                starts.~n'),
        format('~ntop_goal(X,V) :- type(X,V).~n').

writeTailF:-
        findall(_,(attribute(S,L),
        format('~n~w(X):-menuask(~w,X,~w).
                %generated menu',[S,S,L]))  ,_),
        format('~n~n%end of automatic post process').
```

Fig.10 Modules for transforming induced knowledge into the expert rules

```
transform1([X=V], [Res]) :-
    atomic_list_concat([X,'(',V,')'], Res1),
    term_to_atom(Res, Res1),!.

transform1([X=V|T], [Res|T1]) :-
    atomic_list_concat([X,'(',V,')'], Res1),
    term_to_atom(Res, Res1),
    transform1(T, T1).

createRule1(I) :- I=Z>>X>>Y,
    transform1(X, BodyL),
    format('~ntype(~w,~w):-', [Y,Z]),
    myformat(BodyL) ,
    write('  % generated rule'),!.

myformat([X]) :- write(X), write('.'),!.
myformat([H|T]) :- write(H), write(','), myformat(T).
```

Fig.11 Modules to transform decision rules to be consult rules

From the discovered knowledge:

0.5 >> [tear=reduced] >> no,

0.166667 >> [tear=normal, age=young] >> yes,

0.0833333 >> [tear=normal, age=pre_presbyopic, spectacle=myope] >> yes

The first rule states that if the patients have reduced tear production rate, then they should not wear the contact lenses. This rule has the probability of 0.5 to be applicable to the future cases. The second rule advises the patients at young age with normal tear production to be suitable for wearing contact lenses with probabilistic applicability of 0.166667 to the future cases. The last rule also recommends contact lens to the patients at the pre-presbyopic age with normal tear production and myope prescription. The probabilistic applicability of the last rule is 0.0833333. The transformation results of these rules into the expert and consulting rules of the knowledge base are given in Fig.12.

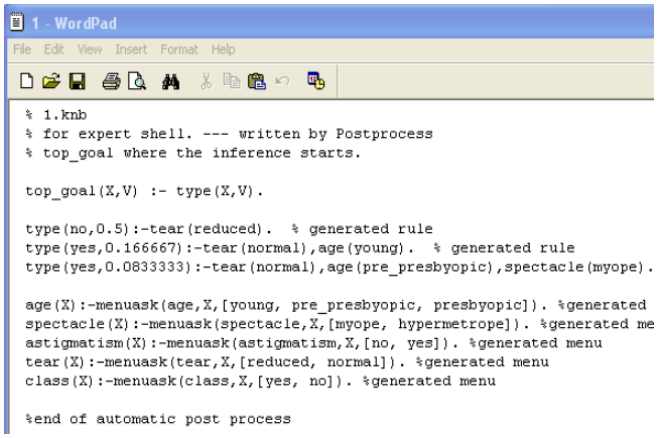


Fig.12 Knowledge base of the expert system

The module or predicate ‘top_goal’ in Fig.12 is the top module of the knowledge base that the inference engine will use in the process of searching for an appropriate recommendation. The predicate ‘type’ is an expert rule. The predicates ‘age’, ‘spectacle’, ‘astigmatism’, ‘tear’, and ‘class’ are consulting rules.

To demonstrate the use of our expert system, we give an example in Fig.13. Once the knowledge had been induced and the knowledge base of the system was generated, the expert system can be invoked through the interface of the SWI Prolog by calling the ‘expertshell’ command. The prompt sign of the system is now changed to ‘expert-shell>’ indicating that the system is ready for solving the users’ questions.

The first step of using the expert shell is to load the knowledge base into working memory. This is simply done through the command ‘load’. The knowledge base file, which is 1.knb in this example, is then compiled. To start asking question, the users have to invoke the system with the ‘solve’ command.

The expert system then processes the recommendation steps through a series of interactive statements. In this example, the system asks for the patient’s tear production rate and age. Then it recommends that this patient can wear contact lenses with probability value of 0.166667. If the users want to see the explanation, they can issue the command ‘why’.

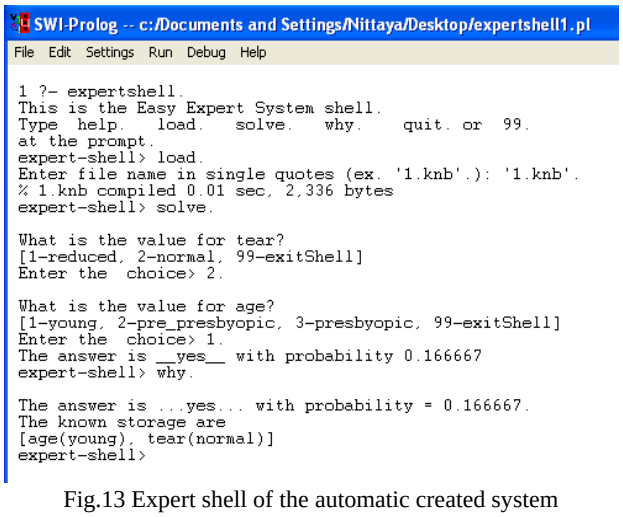


Fig.13 Expert shell of the automatic created system

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

IV. Knowledge Base Evaluation

In this paper, we devise an automatic method to generate knowledge base of the expert system with the purpose that the discovered knowledge can be easily accessible and interpreted by novice users. Since the knowledge base is systematically created, rather than encoded by the domain experts, we have to test the accuracy of the recommendation results against the advice given by human experts.

We had tested our system with the two different datasets: the post operative patients and the breast cancer recurrences. Both datasets are available through the UCI repository (http://archives.ics.uci.edu/ml/). For the post operative patient dataset, we use 70 instances as a training data for the knowledge induction module. The remaining 16 instances are to be used as a test set. The learning objective of post operative dataset is to give recommendation to medical professionals based on the patient's conditions after operation whether the patient should be sent to general ward for further observation or the patient is in good condition and be prepared to go home. The breast cancer recurrence training dataset contains 175 instances, whereas the test dataset contains 16 instances. The learning objective is to predict if the cured breast cancer could recur.

The accuracy of recommendation given by our expert system is also compared with the classification results obtained from three different learning methods, that is, ID3 (decision-tree induction algorithm), PRISM (rule induction algorithm), and neural network. Both recommendation and classification results are evaluated against the real diagnosis given by the medical doctors. The results of accuracy comparison on post operative patients and breast cancer recurrences datasets are given in Figs. 14 and 15, respectively. The detailed results on test datasets are also provided in Tables 1 and 2.

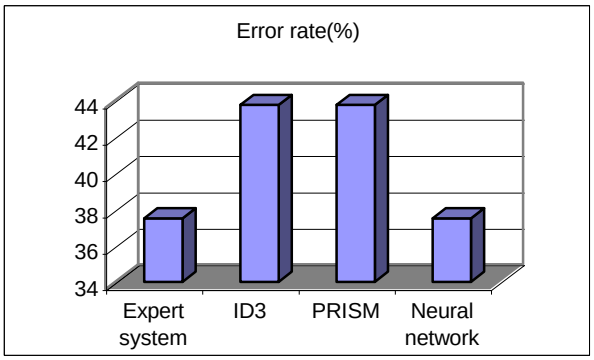


Fig. 14 Recommendation results on post-operative patients

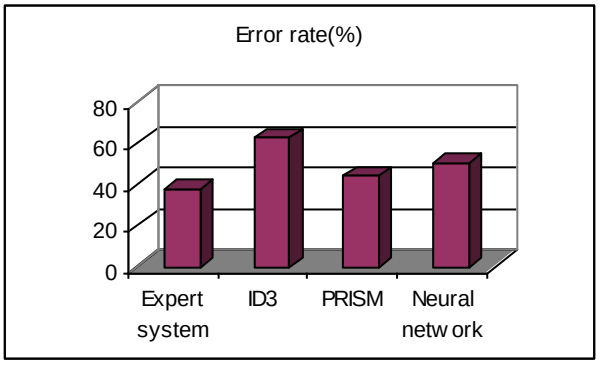


Fig. 15 Recommendation results on breast cancer recurrences

TABLE I
Recommendation and Classification Results for Post Operative Patients

Patient	Doctor's diagnosis	Recommendation & Classification Results			
		Expert System	ID3	PRISM	Neural network
1	Ward	Ward	Ward	Ward	Ward
2	Ward	Ward	Ward	Ward	Ward
3	Ward	Ward	Un-classified	Home	Ward
4	Ward	Home	Home	Home	Home
5	Ward	Ward	Ward	Un-classified	Ward
6	Home	Ward	Ward	Home	Home
7	Ward	Home	Home	Home	Home
8	Ward	Ward	Ward	Ward	Ward
9	Home	No answer	Ward	Ward	Ward
10	Ward	Home	Home	Home	Home
11	Ward	Ward	Ward	Un-classified	Ward
12	Ward	Ward	Ward	Ward	Ward
13	Home	Ward	Ward	Ward	Ward
14	Ward	Ward	Ward	Ward	Ward
15	Ward	Ward	Ward	Ward	Ward
16	Home	Ward	Ward	Ward	Ward

TABLE II
Recommendation and Prediction Results of Breast Cancer Recurrences

Patient	Doctor's diagnosis	Recommendation & Classification Results			
		Expert System	ID3	PRISM	Neural network
1	No	No	Yes	No	Yes
2	No	Yes	Yes	Yes	Yes
3	Yes	Yes	No	No	No
4	Yes	No answer	No	No	Yes
5	No	Yes	Un-classified	No	No
6	No	No answer	Yes	No	No
7	Yes	No	No	No	No
8	No	No	Yes	Yes	No
9	Yes	No	No	Yes	No
10	No	Yes	No	No	Yes
11	No	No	No	No	No
12	No	Yes	Yes	Yes	Yes
13	Yes	Yes	No	No	Yes
14	No	No answer	No	No	No
15	No	No	Un-classified	No	Yes
16	Yes	Yes	Yes	Yes	Yes

It can be seen from the results that the expert system can produce a quite high accurate recommendation. Its error rate from both datasets is around 37.5%. The accuracy is even lower than the neural network method in the breast cancer recurrences dataset. This low error is due to the probabilistic method that we apply to the knowledge induction step. Only knowledge with high probabilistic value is selected and subsequently transformed to be expert and consult rules in the knowledge base.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

It can be noticed from the detailed classification and recommendation results that the false negative rate of our expert system is much lower than the other classification system. For these two specific domain datasets, false negative error is more critical than the false positive error. Therefore, our inductive expert system provides a safer recommendation to the medical practitioners.

The models obtained from the decision tree induction (ID3, as well as our expert system) and the rule induction (PRISM) methods are incomplete. Thus, for some cases the models could not find appropriate recommendation or prediction. This situation does not happen to the model obtained from the neural network learning method. Model completeness is therefore an important issue for the improvement of our expert system.

V. Conclusion

Data mining is an intelligent data analysis method currently adopted by many enterprises and organizations. Most data mining systems, however, perform the mining steps up to the final stage of inducing and generating all the discovered knowledge. The subsequent steps of knowledge selection and turning the discovered knowledge into profitable actions are the responsibility of users that normally are the data mining experts. That means general and novice users should use these systems with difficulty.

We thus propose the method to make data mining system be friendly to general users by integrating the system into the inference engine of the expert system. The discovered knowledge is then presented through the recommendation of the system. The amount of knowledge has been customized to be minimal via the probabilistic threshold during the knowledge induction phase. Our expert system integrated with the induced knowledge module is thus the approximate system. From the evaluation results, the accuracy of recommendation given by the system is acceptable. Therefore, the proposed method is promising for the implementation as a large system to be applicable to the real world data.

We thus plan to extend our current research to the direction of implementing a real system, rather than the prototype as proposed in this paper. The user interface is to be adjusted to suit general users. The system is also to be evaluated by the human experts. The expert rules obtained from the induced knowledge are also to be modified to deal with exceptional cases.

References

[1] G. Adomavicius and A. Tuzhilin, "Expert-driven validation of rule-based user models in personalization applications," *Journal of Data Mining and Knowledge Discovery*, vol.5, no.1/2, pp.33-58, 2001.

[2] R.J. Bayardo, R. Agrawal, and D. Gunopulos, "Constraint-based rule mining in large, dense databases," in *Proc. 15th Int. Conf. on Data Engineering*, March 1999.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Clasification and Regression Tree*. Belmont: Wadsworth, 1984.

[4] S. Brin, R. Motwani, J.D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," in *Proc. ACM SIGMOD Conf.*, 1997.

[5] P. Clark and S. Motwani, "Using qualitative models to guide induction learning," in *Proc. Int. Conf. Machine Learning*, 1993.

[6] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo, "Finding interesting rules from large sets of discovered association rules," in *Proc. 3rd Int. Conf. Information and Knowledge Management*, 1994.

[7] J. Major and J. Mangano, "Selecting among rules induced from a hurricane database," in *Proc. AAAI Workshop on Knowledge Discovery in Databases*, 1993.

[8] J. Ortega and D. Fisher, "Flexibly exploiting prior knowledge in empirical learning,"

[9] M. Pazzani and D. Kibler, "The utility of knowledge in inductive learning," *Machine Learning*, vol.9, 1992.

[10] G. Piatetsky-Shapiro and C.J. Matheus, "The interestingness of deviations," in *Proc. AAAI Workshop on Knowledge Discovery in Databases*, 1994.

[11] J.R. Quinlan, *C4.5: Program for Machine Learning*. Morgan Kaufmann, 1992.

[12] A. Silberschatz and A. Tuzhilin, "What makes pattern interesting in knowledge discovery systems," *IEEE Transactions on Knowledge and Data Engineering*, vol.8, no.6, December 1996.

[13] E. Suzuki, "Autonomous discovery of reliable exception rules," in *Proc. 3rd Int. Conf. KDD*, 1997.

[14] K. Wang, S.H.W. Tay, and B. Liu, "Interestingness-based interval merger for numeric association rules," in *Proc. 4th Int. Conf. KDD*, 1998.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)