A Web-Based Knowledge Engineering and Management System

Chien-Chung Chan[1]
Department of Information Science and Applications
Asia University
500 Liufeng Rd.
WuFeng, Taichung, 41354
Taiwan
chan@asia.edu.tw
Department of Computer Science
University of Akron
Akron, OH 44325-4003
chan@uakron.edu

Abstract

This paper presents a web-based system for knowledge engineering and management. The system consists of three components: data preprocessing, machine learning, and rule-based classifier generator. The only requirement from end users is data sets stored in CSV (Comma Separated Value) format. The system integrates data base systems, rule-based systems, and machine learning systems with a web-based interface to provide an easily accessible and managable knowledge engineering tool that can release end users from any programming burdens.

Keywords: *Knowledge Engineering, Database Systems, Expert Systems, Machine Learning*

1. Introduction

Knowledge Discovery in Databases (KDD) is mainly concerned with how to develop new approaches and techniques to enable efficient extracting of useful information from very large databases. In general, KDD (also called data mining by many people) refers to the overall process of finding and interpreting useful patterns from data. A typical KDD process may consist of six steps: (1) select an application domain and create a target data set, (2) preprocess and clean the data set, (3) transform the data set into a proper form, (4) choose the functions and algorithms of data mining, (5) validate and verify discovered patterns, and (6) apply the discovered patterns [1, 2]. In summary, we observe two important features of KDD: it is a process and it deals with very large data sets.

Recently, major vendors of database systems such as Oracle, IBM, and Microsoft have extended their systems to support some functionalities of KDD. These systems may be used to create data mining models based on supported data mining algorithms. Data mining results may be exported through files in the format of PMML (Predictive Model Markup Lanaguage) [3], which is an xml-based language that provides a way for applications to define statistical and data mining models and to share models between PMML compliant applications. Since data base management systems are well-developed and matured technology, these systems can provide efficient and reliable tools to deal with very large data sets. However, they do not provide tools for making inference.

Research in AI has led to the development of expert systems since late 70's [4]. Most successful expert systems have used if-then rules to represent knowledge of experts, therefore, they are also called rule-based systems. Basic structure of a rule-based expert system includes a rule base, which is a set of if-then rules, and an inference engine, which can be used to infer answers to given inputs by using rules in the rule base. Most expert system development environments such as CLIPS [5] and JESS [6] provide tools for end users to program or enter rules into rule base. However, they do not provide machine learning tools to generate rules from data sets. The strength of rule-based environments is to do pattern matching and reasoning. The weakness of

[1] The work was partially supported by the University of Akron, Faculty Summer Research Fellowship 2007.

them is I/O facilities for developing user interfaces. In addition, inference engines may not support reasoning with uncertain rules.

Most data mining tools are based on results of machine learning research, which is a well-established area in AI [7, 8], and there have been many successful applications. One of the well-known programs is Quinlan's C4.5 [9], a revised commercial version is called C5.0, which can be used to generate classifier programs from collections of training examples. The generated classifier can be represented as a decision tree or a set of production rules. The C4.5 system can also generate an inference engine that will apply the generated decision trees or rules to produce answers for queries entered interactively. In C5.0, the system can generate C souce codes of classifiers for developing embedded applications. Another well-known open-source data mining package is WEKA [8], which is implemented in Java. The WEKA package is quite comprehensive, and it provides tools for pre-processing and typical data mining tasks such as classifying, clustering, and association rule mining. However, no inference engines for using the generated rules are provided. Because it is implemented in Java, it is easy to embed WEKA's tools in Java applications. Data mining tools based on rough sets approach such as ROSE2 [10, 11] and ROSETTA [12, 13] are also quite powerful. All these tools provide programming libraries to support embedded applications. However, to deliver stand-alone applications, they all require some programming to provide end-user interfaces for specific data sets. In addition, there is one issue that has not been addressed by most data mining packages, namely, how to manage the data sets and classifiers generated by these systems.

Table 1: Feature matrix of some knowledge engineering tools.

	MS SQL	C4.5 CART	Weka	ROSE-2	CLIPS JESS
PreProcessing	No	No	Yes	Yes	No
Learning	Yes	Yes	Yes	Yes	No
Embedded Inference Engine	No	No	No	No	Yes
End-User Interface	API	API	API	API	API
Web-Based Access	API	API	API	API	API
Reasoning with Uncertainty	No	Yes	Yes	Yes	No
Knowledge Management	No	No	No	No	No

Table 1 summarizes our observation of features available in some popular tools for data mining, machine learning, and expert system shells. It is clear from the feature matrix table that some programming skills are required by end users in order to deliver their data mining results, and knowledge management is an area that needs more attention.

In our proposed system, the focus is from the perspectives of end users who do not have programming skills. The requirements from the end users are to provide data sets and the knowledge of pre-processing the data. The system provides tools for preprocessing the data set. Then, from a pre-processed data set, it will generate a meta-data file containing attribute information of the data and a set of if-then rules with uncertainties. The rules are generated using the BLEM2 learning program [14]. The resulting meta-data file and rule file are used by a web-based classifier generating system to generate a web-based user interface for running the target classifier. For each data set and target classifier, the corresponding meta-file and rule-file are stored in a database system to provide manageability. It is an integrated rule-based data mining system that is capable of creating rule-based systems with web-based user interface from data sets selected from SQL tables or provided by end users. It provides a streamlined integration of three technologies, namely, database systems, machine learning, and expert systems. Rules generated by the system are based on rough set theory. Therefore, it is capable of dealing with uncertain rules.

The paper is organized as follows. In Section 2, we present an overview of the system and its components in detail. Section 3 gives information of its implementation. Section 4 is the conclusions and future work, followed by references.

2. System Overview

2.1 Rule-Based Classifier

A rule-based classifier system is a set of if-then rules that implements a classifier. In general, the inference step in a rule-based classifier is quite simple: if the input data match the conditions on the Left Hand Side (LHS) of a rule, then return the decision on the Right Hand Side (RHS) of the rule. Because conditions of different rules are not necessarily mutually exclusive, it is possible that multiple rules may be applicable for a given input. When the RHS returned by the rules are different, then the answer returned by the system is not unique. We call this condition *rule conflict*. One way for resolving rule conflict is to use the majority voting method, i.e., to return the value produced by majority of the rules, and break ties arbitrarily. It is also possible to use methods based on theories such as fuzzy sets [15, 16] and rough sets [17, 18]. In our system, a maximum

sum and a partial match approaches based on rough set theory have been implemented.

2.2 Rule-Based Classifier Generator

Our focus is on rule-based systems, because if-then rules are simple and popular form of representing knowledge. The inception of a web-based multi-tier rule-based system was introduced in [19]. Early prototypes of the system were implemented in Java servlets and SQL server [20] and in .NET framework [21, 22]. The main functionality was to automate the generating of user interface for rule-based classifiers (RBC) using SQL databases. The rule base of an RBC is stored as a relational table in the database. The inference engine of an RBC was implemented using SQL scripts by taking advantage of the pattern matcher functionalities of a SQL processor.

The architecture of RBC generator is a multi-tier client-server system shown in Figure 1. Clients interact with the backend SQL server through services provided by the application server in the middle tier using a web-browser. The application server is responsible for dispatching requests to the intended backend server, receiving responses from backend server, and presenting the final results back to the clients.

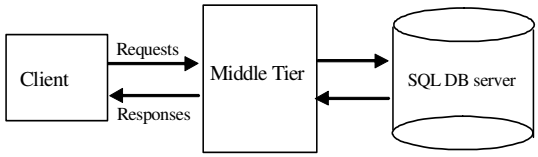

Figure 1. Architecture of RBC generator.

The rule base is generated by a Table Translator as shown in Figure 2. The Table Translator takes a rule set file and a metadata file as inputs, and it generates a SQL script for creating tables in the database. This requires the rule set file to be in certain format. The Table Translator is implemented as Java servlet in the Middle Tier. It is used to upload the two input files (metadata and rule set) and to store them temporarily on the server. After verifying that these tables match the format, it starts creating the tables in the database.

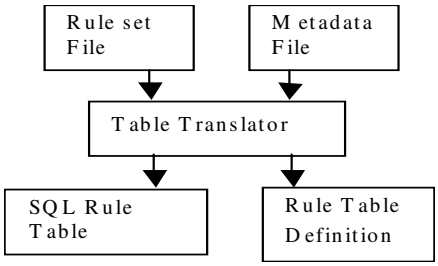

Figure 2. Table translator**.**

Rules of different target classifiers are stored in dynamically generated relational tables separately. The inference engine that is shared by all target classifiers is implemented as dynamic SQL statements generated from user inputs. It uses the pattern matching ability of a SQL processor to determine the rules that are firable. Rule conflict resolution strategies are implemented using stored procedures. For instance, to implement the Max-Sum approach, we can use a SQL statement that returns the decision-attribute which has the maximum sum of certainty*strength value whenever the selected condition-attributes meet the user selected values as shown in the following.

```
SELECT TOP 1 <decision-attribute> AS decision,
       sum(certainty*strength) AS certainty
FROM  <domain name>
WHERE  <selected condition-attributes> = <selected value>
GROUP BY  <decision-attribute>
ORDER BY  certainty DESC
```

The majority voting method can be implemented as:

```
SELECT TOP 1 <decision-attribute>, COUNT(*) AS
    MAX_NUM
FROM <meta-data table>
WHERE <selected condition-attributes> = <condition-attributes
    values in meta table> OR < not selected condition-attributes> is
    NULL
GROUP BY <decision-attribute>
ORDER BY MAX_NUM
```

The above SQL statement returns one of the decision-attribute that has maximum number of matches whenever the selected condition-attributes meet the condition-attributes values stored in the database or the non selected condition-attributes has NULL values in the database.

Conflict resolution approaches are stored in a relation table and retrieved dynamically based on user's preference during the execution of a RBC.

In summary, a rule-based classifier is implemented by two servlets: a Table Translator and a SQL query generator that connect to a backend SQL database server. The detailed design of backend database was reported in [21, 22]. By integrating SQL server with rule-based classifiers, we have added management capabilities to a knowledge engineering tool. Metadata and rule bases of applications are stored in sql tables, which are used to automate the generation of user interfaces for different applications and to control access previledges of different types of users. The delievery of final results is facilitated by web-based technology, i.e., a hyperlink is what is needed to deliver and access an application.

2.3 System Components and Workflow

In addition to the rule-based classifier generator, our system has included two more components: data preprocessing and machine learning programs. All components are integrated into the SQL database server. The complete workflow of our system is shown in the following.

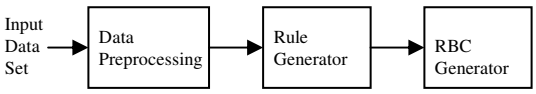

Figure 3. System components and workflow.

Input Data Set

Required format for input data set is a text file with comma separated values (CSV), which can be created using MS Excel program. It is assumed that there are N columns of values corresponding to N attributes or variables, which may be real or symbolic values. The first N – 1 attributes are considered as condition attributes and the last one is the decision attribute.

Data Preprocessing Component

This component can be used to process missing values in records, generate data statistics, and discretize domains of numeric attributes into a finite number of intervals. The preprocessed data files can be used to generate metadata attribute information files and training data files for learning programs. All raw data and preprocessed data are stored into databases.

Rule Generator Component

Any rule learning programs can be used as a rule generator. Currently, we use the symbolic learning program BLEM2 [14] to generate rules with uncertainty from discretized training data files and corresponding attribute files. The component has web-based interface and can be used to upload training data files in Weka, C4.5, and BLEM2 formats, view and download learned rules, generate testing data files, and testing rules using exact or partial matches with different weighting functions.

RBC Generator Component

This component is introduced in Section 2.1. It is used to generate a web-based Rule-Based Classifier (RBC) from a rule file and a metadata file. Early prototypes were implemented in [21, 22]. For each pair of metadata and rule files, the user interface for running the classifier is generated dynamically. It includes an inference engine shared by all target classifiers. In current system, we have added manual rule editor and backward chaining inference.

3. System Implementation and Demo

The system was implemented using Microsoft .NET framework 2.0 and MS SQL server 2005 Express. Scripting is done by C# and JavaScript. The software requiremenths of the system are Windows XP Professional (service pack 2), Microsoft Internet Information Services IIS 4.0 or better, MS SQL 2005 Express edition, Microsoft .NET framework 2.0 redistributable package, Windows installer 3.0 except for Windows 98/ME, which require Windows Installer 2.0 or later, and Microsoft Internet Explorer 5.01 or later. It has been tested on Personal computers and laptops with minimum requirements of Intel or Pentium III 600MHz or faster processor (1 GHz is recommended), minimum of 192 MB RAM (512 MB is recommended), 350 MB of hard disk space for SQL Server 2005, .NET framework 280 MB(x86), and CD-ROM or DVD ROM.

Demonstration will be presented at the conference.

4. Conclusions and Future Work

We have presented a framework for developing a knowledge engineering and management system by integrating technologies from database systems, machine learning systems, expert sytems, and multi-tier systems. Database systems provide the ability to deal with large data sets efficiently, the capability of managing user interfaces and rule bases, security, and pattern matching for implementing inference engines. Machine learning programs are used to generate rules from application data sets, and expert systems provide mechanisms of inference. The web-based technologies make it simple to deliver applications and provide global accessability. Most importantly, the integrated system can release end users from programming burdens.

Our system has touched the area of knowledge management with simple rule base management. There are rooms to identify and develop more sophicated management-related features, which may be further complicated by different knowledge representation schemes. Ideally, we would like to develop polymorphic programming interface for integrating different machine learning and data preprocessing algorithms.

5. References

[1] Fayyad, U., Editorial, Int. J. of Data Mining and Knowledge Discovery, Vol.1, Issue 1, 1997.

[2] Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery: an overview," in Advances in Knowledge Discovery and Data Mining, Fayyad et al (Eds.), MIT Press, 1996.

[3] Data Mining Group: http://www.dmg.org/pmml-v3-0.html

[4] Buchanan, B.G. and E.H. Shortliffe, eds. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading, MA, Addison-Wesley, 1984.

[5] Giarratano, J. and G. Riley, *Expert Systems Principles and Programming*, 3rd ed., PWS Publishing Company, 1998.

[6] Friedman-Hill, E., Java Expert System Shell, Sandia National Laboratories, Livermore, CA., *http://herzberg.ca.sandia.gov/jess*

[7] Mitchell, T.M., *Machine Learning*, The McGraw-Hill Companies, Inc., 1997.

[8] Witten I.H. and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, San Francisco, Morgan Kaufmann, 2000.

[9] Quinlan, J.R., *C4.5: Programs for machine learning*. San Francisco, Morgan Kaufmann, 1993.

[10] Predki, B., R. Slowinski, J. Stefanowski, R. Susmaga, and Sz. Wilk: ROSE - Software Implementation of the Rough Set Theory. In: L.Polkowski, A.Skowron, eds. *Rough Sets and Current Trends in Computing, Lecture Notes in Artificial Intelligence, vol. 1424.* Springer-Verlag, Berlin (1998), 605-608.

[11] Predki, B., Sz.Wilk: Rough Set Based Data Exploration Using ROSE System. In: Z.W.Ras, A.Skowron, eds. *Foundations of Intelligent Systems, Lecture Notes in Artificial Intelligence, vol. 1609.* Springer-Verlag, Berlin (1999), 172-180.

[12] Øhrn, A., J. Komorowski, ROSETTA: a rough set toolkit for analysis of data, P.P. Wang editor, *Proc. Third International Joint Conference on Information Sciences*, Volume 3, pp. 403 - 407, Durham, NC, March 1997.

[13] Øhrn, A., J. Komorowski, A. Skowron, P. Synak, The design and implementation of a knowledge discovery toolkit based on rough sets: The ROSETTA System. In *Rough Sets in Knowledge Discovery 1: Methodology and Applications*, Volume 18 of *Studies in Fuzziness and Soft Computing*, L. Polkowski and A. Skowron editors. Physica-Verlag, Heidelberg, 1998.

[14] Chan, C.-C. and S. Santhosh, "BLEM2: Learning Bayes' rules from examples using rough sets," *Proc. NAFIPS 2003, 22nd Int. Conf. of the North American Fuzzy Information Processing Society*, July 24 – 26, 2003, Chicago, Illinois, pp. 187-190.

[15] Zadeh, L.A., "Fuzzy sets," *Information and Control*, 8:338-353, 1965.

[16] Gupta, M.M., R.K. Ragade, and R.R. Yager, *Advances in Fuzzy Set Theory and Applications*, editors, North-Holland, Amsterdam, 1979.

[17] Pawlak, Z., "Rough sets: basic notion," *Int. J. of Computer and Information Science* 11, pp. 344-56, 1982.

[18] Pawlak, Z., J. Grzymala-Busse, R. Slowinski, and W. Ziarko, "Rough sets," *Communication of ACM*, Vol. 38, No. 11, November, 1995, pp. 89-95.

[19] Chan, C.-C., L.K. Verma, N. Khasawneh, and S. Rahman, "Generation of executable rule-based classifiers using Java Technology," *Proc. 12th MidWest Artificial Intelligence and Cognitive Science Conference MAICS 2001*, March 31 – April 1, 2001, Miami University, Oxford, Ohio, pp. 45-48.

[20] Khasawneh, N. and C.-C. Chan, "Servlet-based implementation for rule-based classifiers," *Proc. 12th MidWest Artificial Intelligence and Cognitive Science Conference MAICS 2001*, March 31 – April 1, 2001, Miami University, Oxford, Ohio, pp. 70-74.

[21] Su, Zhicheng, "Dot Net implementation for rule-based classifiers," Master's research report, Department of Computer Science, University of Akron, June, 2003.

[22] Chan, C.-C. and Zhicheng Su, "From Data to Knowledge: an Integrated Rule-Based Data Mining System," *Proceedings of the 17th International Conference of Software Engineering and Knowledge Engineering*, July 14 – 16, Taipei, 2005, pp. 508-513.