International Journal of Soft Computing 3 (3): 195-200, 2008
ISSN: 1816-9503
© Medwell Journals, 2008

Premises Reduction of Rule Based Expert Systems Using Association Rules Technique

Ahmed T. Sadik
Department of Computer Science, University of Technology, Baghdad, Iraq

Abstract: The heart of expert system is the knowledge base that determines the power of expert system and search engine space. The one important form of this knowledge is the production rule. The premises of these rules are the heart of production rules, therefore, the reduction of these premises is very useful to reduce the time and space in search engine. In this study an approach will be presented to reduce the premises of production rules using association rules algorithm especially Apriori algorithm. Applying this approach gives a logical premise reduction which plays a good role in the search engine.

Key words: Expert systems, premises reduction, rule-based expert systems, production rules, association rules, apriori algorithm applications

INTRODUCTION

An expert system is a computer program that represents and reasons with knowledge of some specialist subject with a view to solving problems or giving advice (Jackson, 1990). An expert system may completely fulfill a function that normally requires human expertise, or it may play the role of an assistant to a human-maker. The symbolic reasoning of an expert system enables it not only to draw conclusions, through a process similar to the one used by human experts, but also to provide explanations concerning their estimations. Expert system technology is based on the domain knowledge of the problem being addressed. A problem domain defines the objects, properties, tasks and events within which a human expert works and also the heuristics that trained professionals have learned to use in order to perform better (Kelihn, 1995) user interface provides the mechanism (Al-Barky, 2003).

Figure 1 shows the most important modules that make up a rule-based expert system. The *user interacts* with the expert system through a user interface that simplifies communication and hides much of the system complexity (e.g., the internal structure of the rule base). An expert system employs a variety of interface styles, including question and answer, menu driven, natural language, or graphics interfaces, where the final decision on interface type is a compromise between user needs and the requirements of the knowledge base and inferencing system (George *et al*., 1998).

The heart of the expert system is the general knowledge base, which contains the problem solving knowledge of the particular application. The inference engine applies the knowledge to the solution of actual problems. It is essentially an interpreter for the knowledge base. The program must keep track of case-specific data: The facts, conclusions, and other information items relevant to the case under consideration. This includes the data given in a problem instance, partial conclusions, confidence measures of conclusions, and dead ends in the search process. This information is separate from the general knowledge base. The *explanation subsystem* allows the program to explain its reasoning to the user. These explanations include justifications for the system's conclusions (in response to how queries), explanations of why the system needs a particular piece of data (why queries) and where useful, tutorial explanations or deeper theoretical justifications of the program's actions (George *et al*., 1998).

Many systems also include a knowledge-base editor. Knowledge-base editors help the programmer locate and correct bugs in the program's performance, often accessing the information provided by the explanation subsystem. They also may assist in the addition of new knowledge, help maintain correct rule syntax, and perform consistency checks on the updated knowledge base (Georage *et al*., 1998).

PRODUCTION RULES

Knowledge in an expert system can be represented in various ways. Some of the well-known techniques for representation of knowledge include Production System, Logical Calculus and Structured Models (Konar, 2000).

The production system is the simplest and one of the oldest techniques for knowledge representation.

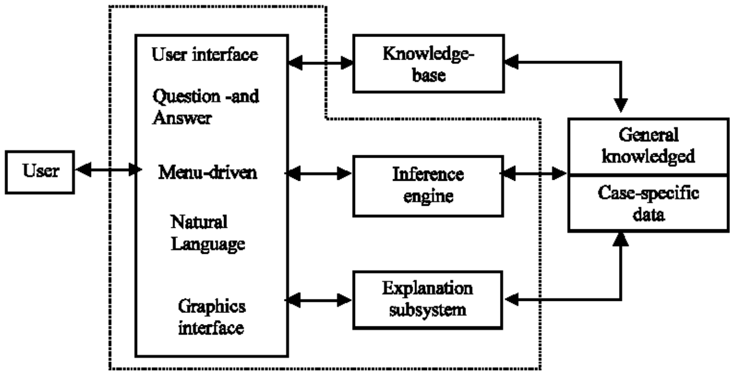

Fig. 1: Architecture of a Typical Expert System

A production system consists of 3 items: A set of Production Rules (PR), which together forms the knowledge base, one (or more) dynamic database(s), called the working memory and a control structure/ interpreter, which interpret the database using the set of production rules (Amit Konar, 2000; Jackson, 1990; Rakesh and Ramakrishnan, 1994). The production system, which has wide applications in automata theory, formal grammars and the design of programming languages, however, entered into knowledge engineering by Bauchanan and Feigenbarm (1978) only a few years ago.

The structure of production rules can be formally stated as follows:

PR1: $P_1(X) \wedge P_2(Y) \wedge \ldots\ldots P_n(X,Y) \rightarrow Q_1(Y) \vee Q_2(Z) \vee \ldots\ldots Q_m(Z,X)$

Where, P and Q are predicates; X,Y and Z are variables: $\wedge$, $\vee$ and $\rightarrow$denote the logical AND, OR and IF-THEN operators, respectively. The left-hand side of a production rules is called the premises/antecedent/conditional part and the right-hand side is called the consequent/ conclusion part (Konar, 2000; George *et al.*, 1998).

Many of the expert systems were production systems. This probably came about because production systems were invented for cognitive modeling and they seem to give quite an impressive account of some mental processes. The creators of expert systems believed that they have to model the mental apparatus of their experts, so they naturally turned to production systems (Daniel and Marcellus, 1989).

KNOWLEDGE BASE OPTIMIZATION IN A PRODUCTION SYSTEM

The performance of production system depends largely on the organization of its knowledge base. The inference derived by a production system per unit time, also called time efficiency, can be improved by reducing the matching time of the premises of production rules with the data in the working memory. Further, if the rules are constructed in a manner so that there is no conflict resolution to can be avoided. Another important issue of rule-base design is to select the rules so that the resulting state-space for rule firing does not contain any cycles. The last issue is to identify the concurrently friable rules that do not have conflict in their action parts (Konar, 2000). This, if realized for a rule-base system, will improve the performance to high extent. For optimization of rules in a rule-based system, Zupan *et al.* (1996) suggested the following points:

- Construct by backward reasoning a state-space graph from the desired goal nodes (states) up to the nodes, which cannot be expanded further in a backward manner. Each goal node, also called fixed point, is thus reachable (has connectivity) from all possible starting states. It may be noted that some of the connectivity from the starting nodes to the goal nodes may pass through the cycles. It should also be noted that the resulting state-space will not miss the shortest paths from the goal to any other state, as predecessor states of each state are found by an exhaustive breadth first search.
- The common states in the graph are replaced by a single vertex and the parallel paths are identified.
- Do not generate an existing state.

This study suggests an approach to reduce the premises of production rules as an optimization approach for production rules.

RULE-BASED EXPERT SYSTEMS

Rule-based expert systems represent problem-solving knowledge as *If... Then...* rules. This approach lends itself to the architecture of Fig. 1 and is one of the techniques for representing domain knowledge in an expert system.

It is also one of the most natural and remains widely used in practical and experimental expert systems (George *et al.*, 1998).

If we regard the expert system architecture in Fig. 1 as a production system, the knowledge base is the set of production rules. In a rule-based system, these condition-action pairs are represented as *If... Then...* rules, with the premises of the rules, the *if* portion, corresponding to the condition, and the conclusion, the *then* portion, corresponding to the action: when the condition is satisfied, the expert system takes the action of asserting the conclusion as true. Case-specific data are kept in the working memory. The inference engine implements the recognize-act cycle of the production system; this control may be either data-driven or goal driven (George *et al.*, 1998).

In a goal-driven expert system, the goal expression is initially placed in the working memory. The system matches rule conclusions with the goal, selecting one rule and placing its premises in the working memory. This corresponds to a decomposition of the problem's goal into simpler sub-goals. The process continues in the next iteration of the production system, with these premises becoming the new goals to match against rule conclusions. The system thus works back from the original goal until all the sub-goals in working memory are known to be true, indicating that the hypothesis has been verified. Thus, backward search in an expert system corresponds roughly to the process hypothesis testing in human problem solving .

In the data-driven reasoning, the algorithm for this is simple: Compare the contents of working memory with the conditions of each rule in the rule base using the ordering of the rule base. If the data in working memory supports a rule's firing the result is placed in working memory and control moves on to the next rule. Once all rules have been considered search starts again at the beginning of the rule set. However, the advantages of the rule-based approach include (Daniel and Marcellus, 1989; George *et al.*, 1998).

- The ability to use, in a very direct fashion, experimental knowledge acquired from human expert.
- The modularity of rules ease construction and maintenance.
- Good performance is possible in limited domain.
- Good explanation facilities.
- Rules map naturally into state space search.
- Rule chaining is fairly easy to trace and debug.
- Steps within the problem solution process are open to inspection.
- The separation of knowledge from control further simplifies development of expert systems by enabling an iterative development process where the engineer acquires, implements and tests individual rules.

On the other hand, the rule-based expert systems have disadvantages that are (Nilson, 1980):

- Often the rules obtained from human experts are highly heuristic in nature, and lack a deeper, functional knowledge of the domain.
- Missing information or unexpected data values can not be handled.
- Tendency to degrade rapidly near edges of the domain knowledge.
- Explanations function at the descriptive level only, omitting deeper and theoretical explanations.

The knowledge tends to be very dependent.

ASSOCIATION RULE ALGORITHMS DEFINITION

An association rule is a rule, which implies certain association relationships among a set of objects (such as those which occur together or one which implies the other), in a database. Given a set of transaction, where each transaction is a set of literal (called items), an association rule is an expression of the form XY, where X and Y are sets of items. The intuitive meaning of such a rule is that transactions of the database, which contain X, tend to contain Y (Rakesh and Ramakrishnan, 1994).

Association rules identify relationships between attributes and items in database such as the presence or absence of one pattern implies the presence or absence of another pattern. Mining of such rules is one of the most popular pattern discovery methods in KDD, an association rule is an expression $X \Rightarrow Y$ where $X = \{x_1, x_2 \dots x_n\}$ and $Y = \{y_1, y_2 \dots y_n\}$ are set of items with Left Hand Side (LHS) and Right Hand Side (RHS). The meaning of such rules is quite intuitive: given database D of transaction T where each transaction $T \in D$ is a set of items $X \Rightarrow Y$ which expresses that whenever a transaction T contains X, the T probably contains Y. Also the probability of rule strength is defined as the percentage of transactions containing Y in addition to X. The prevalence for rule is the percentage of transactions that hold all the items in the union. If prevalence is low, it implies that there is no overwhelming evidence that items in $X \cup Y$ occur together.

The rule $X \Rightarrow Y$ has support S in D if the fractions of the transactions in D contain $(X \cup Y)$ (Rakesh *et al.*, 1994).

The problem of mining association rules is to generate all association rules that have certain user-specified minimum support (called min-sup) and confidence (called min-conf) (Agrawal *et al*., 1994). The important measures for association rules, Support (S) and Confidence (C) can be defined as: The Support (S) of an association rule is the ratio (in percent) of the records that contain $(X \cup Y)$ to the total number of records in database. Therefore, if we say that the support of a rule is 5% then it means that 5% of the total records contains $(X \cup Y)$. Support is the statistical significance of a rule (Agrawal *et al*., 1994).

Support $(X \Rightarrow Y) = P(X \cup Y)$
Support $(X \Rightarrow Y)$ = frequent (X) / total number of records in database.

For a given number of records, Confidence (C) is the ratio (in percent) of the numbers of records that contain $(X \cup Y)$, to the number of records that contain X. thus, if we say that a rule has a confidence of 5% it means that 85% of the records containing X also contain Y. The confidence of a rule indicates the degree of correlation in the database between X and Y.

Confidence $(X \Rightarrow Y) = P(Y \cup X)$
Confidence $(X \Rightarrow Y)$ = frequent $(X \cup Y)$ / frequent (X)

Confidence is also a measure of rules strength. Mining of database consists of finding all rules that meet the user-specified threshold support and confidence.

HE SUGGESTED APPROACH FOR PREMISES REDUCTION USING ASSOCIATION RULES

The power of expert systems is concentrated in knowledge base, because the knowledge base represents the human experience, the time and space search in control of expert systems, therefore, the reduction of this knowledge will play a big role in the performance of expert systems. This study focuses on reduction of "*premises*" of production rules using association rule technique especially applying the *Apriori* algorithm. The standard *Apriori* algorithm is not reasonable because it checks k-item set with only (k-1)-item set. We need to check k-item set with all previous item sets. The algorithm of suggested approach is as follows:

```
Input:  Production rules of an expert system.
Output: Reduction production rules of an expert system.
Begin
        Convert the premises of production rules into
        transaction database.
        Apply an improved Apriori algorithm to find the
        association rules of resulting database.
        Sort the association rules descending (depending
        on transaction length of right-hand side then left-
        hand side).
Repeat
        Replace the symbols of first association rules
        with one symbol in transactions database;
        Delete the rest association rules that contain all
        symbols in the first one;
        Until no association rules;
        Recover the production rules from transaction
        database;
End.
```

ILLUSTRATED EXAMPLE

Assume the following rule-based expert system:

If A and B and C then X
If A and B and C and D then X
If A and B and C and E then Y
If B and C and D then Y
If B and C and D and E then Z
If C and D and E then X
If E and X then Q_1
If E and Y then Q_2
If X and Y then Q_1
If E and X and Y then Q_2

Step 1: Convert these rules into transaction tables as follows:

Transaction ID	Transaction
1	A, B, C
2	A, B, C, D
3	A, B, C, E
4	B, C, D
5	B, C, D, E
6	C, D, E
7	E, X
8	E, Y
9	X, Y
10	E, X, Y

Step 2: The association rules from applying improved *Apriori* algorithm are:

$A \rightarrow B$
$A \rightarrow C$
$A \rightarrow BC$
$B \rightarrow C$
$AC \rightarrow B$
$D \rightarrow C$
$AB \rightarrow C$
$BD \rightarrow C$

Step 3: Sort the above association rules descending:

$$\begin{gathered} A \to BC \\ AC \to B \\ AB \to C \\ BD \to C \\ A \to B \\ A \to C \\ B \to C \\ D \to C \end{gathered}$$

Step 4: Repeat replacing and deletion operations. According to the first association rule A→BC, replace each A, B and C with P1, then delete each association rule which contains A, B and C. The resulting transaction database:

Transaction ID	Transaction
1	P1
2	P1, D
3	P1, E
4	B, C, D
5	B, C, D, E
6	C, D, E
7	E, X
8	E, Y
9	X, Y
10	E, X, Y

The rest of association rules are:

$$\begin{gathered} BD \to C \\ D \to C \end{gathered}$$

Now, the first association rule BD → C, replace each D, B and C with P2, then delete each association rule which contains D, B and C. The resulting transaction database:

Transaction ID	Transaction
1	P1
2	P1, D
3	P1, E
4	P2
5	P2, E
6	C, D, E
7	E, X
8	E, Y
9	X, Y
10	E, X, Y

The rest of association rules are:
D → C

Now, the first association rule D→ C, replace each D and C with P3, then delete each association rule which contains D and C. The resulting transaction database:

Transaction ID	Transaction
1	P1
2	P1, D
3	P1, E
4	P2
5	P2, E
6	P3, E
7	E, X
8	E, Y
9	X, Y
10	E, X, Y

At this step there are no association rules.

Step 5: Recover the production rules from resulting database transaction:

If P1 then X
If P1 and D then X
If P1 and E then Y
If P2 then Y
If P2 and E then Z
If P3 and E then X
If E and X then Q1
If E and Y then Q2
If X and Y then Q1
If E and X and Y then Q2
Applying the suggested algorithm reduces the premises from 30 to 19 !!! .

CONCLUSION

The premises of rule-based expert system play a big role in the search engine complexity. In this study, a logical method is presented to reduce these premises using association rules mining in data mining technique. The presented method has proved a good logical tool for reducing the premises because it depends on the logical relations among these premises. Association rules mining using Apriori algorithm will produce logical association rules in the transactions, therefore, this study applies Apriori algorithm.

REFERENCES

Agrawal R. and R. Strikant, 1994. Fast Algorithms for Mining Association Rules. Proceeding of the 20th Int'l Conference on Very Large Databases, Santigo, Chile.

Agrawal, R., H. Mannila, R. Strikant, H. Toivenen and I. Verkamo, 1994. Fast Discovery of Association Rules. Proceeding of the 20th Int'l Conference on Very Large Databases, Santigo, Chile.

Agrawal, R., T. Imielinski and A. Swami, 1994. Mining Association Rules Between Sets of Items in Large Database. Washington, U.S.A.

Al-Bakry, A.A., 2003. Expert Systems Development Using Knowledge Agent. Ph. D. Thesis, University of Technology, Baghdad, Iraq.

Buchanan, B.G. and E.A. Feigenbaum, 1978. DENDRAL and Meta DENDERAL: Their Applications dimension. Artificial Intelligence, 11: 5-24.

Daniel H. Marcellus., 1989. Expert systems programming in Turbo prolog. Prentice-Hall Inc.

George, F. Luger and A.S. William, 1998. Artificial Intelligence Structures and Strategies for Complex Problem Solving. 3rd Edn. Addison Wesley Longmann Inc.

Jackson, P., 1990. Introduction to Expert Systems. 2nd Edn. Addison-Wisely.

Klein, M. and L.B. Methlie, 1995. Knowledge Based Decision Support Systems with Applications in Business. 2nd Edn. JohnWiley and Sons.

Konar, A., 2000. Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain. CRC Press.

Nilson, N.J., 1980. Principles of Artificial Intelligence, Morgan Kaufmann. San Mateom, CA, pp: 6-7.

Zupan, B. and A.M.K. Cheng, 1998. Optimization of Rule-Based Systems Using State-Space Graphs. IEEE Transaction on Knowledge and Data Engineering, Vol. 10.