A Rule Measure to Represent the Temporal Changes of Data Mining Patterns

Shonali Krishnaswamy, Arkady Zaslavsky
School of Computer Science & Software Engineering
Monash University, Melbourne, Australia
{shonali, azaslavs}@broncho.ct.monash.edu.au

Abstract

Data mining is the automated extraction of hitherto unknown patterns in large databases. Patterns or rules extracted by mining algorithms are generally true for the current state of the database. As the database state changes due to new transactions these rules may either become invalid or may have increased support. In this paper we present a new rule measure – Rule Trend Analysis (RTA) – which quantifies and represents the trend or behaviour of patterns extracted by data mining systems as these rules evolve along a continuous time interval. This measure is derived using the database log file by analysing transactions that are compliant with the rule and those that are not. We present both the theoretical aspects as well as the results obtained by implementing this technique of data mining rule analysis.

1. Introduction

Data mining or knowledge discovery in databases (KDD) is the automated extraction of hitherto unknown patterns in large databases. The prolific and widespread use of database technology in businesses, governments, educational and scientific organisations has led to the accumulation of vast amounts of data. However, the ability to analyse and interpret the data (of the order of several hundred gigabytes per organisation on average) has not kept pace with the technology for collecting and storing data [1]. This has been the underlying motivation for the emergence of knowledge discovery or data mining as a significant and active research area. The premise for knowledge discovery is that there is valuable knowledge or information hidden in the mass of data. [2] provides a formal definition of knowledge in this context as follows:

> "Given a set of facts (data) F, a language L, and some measure of certainty C, a pattern is defined as a statement S in L that describes the relationships among a subset F_S of F with a certainty c, such that S is simpler (in some sense) than the enumeration of all facts in F_S. A pattern that is interesting enough (according to a user imposed interest measure) and certain enough (again according to the user's criteria) is called *knowledge.* "

Data mining techniques aim to provide automated support for the extraction of such hidden information.

It has been recognised that one of the significant issues in knowledge discovery is the development of measures for the patterns extracted by the mining process. In this context, several measures have been proposed which provide statistical values that represent different characteristics of knowledge discovery rules such as *robustness* [3], *interestingness* [4], *confidence* and *support* [5] and *intensity* [6]. In this paper we present a new rule measure for knowledge discovery rules.

Two important characteristics of patterns extracted from knowledge discovery systems are as follows:

- They are representative of the current state of the database and are not true for all database states.
- Consequently, these rules can evolve over time and move from a stable state to a volatile state and vice versa

Thus, it is obvious that the characteristics of data mining rules do not remain constant or static and are subject to variations as the database state changes. Therefore, it becomes important to study the trend of a rule over a period of time. In fact, the significance of the extracted patterns are enhanced by extended validity over a continuous time interval and forms the basis for rules becoming part of an established belief system. The rule measure presented in this paper addressed this issue of rule trend analysis.

This paper is organised as follows. In section 2, we review related work that is specific to existing rule measures. In section 3, we discuss the theoretical basis for our rule

Proceedings of the Workshop on Computer Science and Information Technologies CSIT'99
Moscow, Russia, 1999

measure – *Rule Trend Analysis* (RTA). Section 4 presents the implementation and results obtained by testing the technique. Finally in section 5, we present the conclusions and future directions of this work..

2. Related Work

In this section we review previous work relating to rule measures for knowledge discovery. A major proportion of the work related to rule evaluation focuses on "interestingness". According to Silberschatz and Tuzhilin [4] measures of interestingness are grouped into two categories - *objective* and *subjective*. Objective measures of interestingness depend only on the underlying data used in the discovery process, whereas subjective measures also take into account the users who examine the pattern. In general, subjective measures are considered to be better than objective measures. A pattern is said to be "interesting" if it allows a user to take some action based on its discovery (i.e. it is "actionable") and if it is deviant from a general perception or belief (i.e. it is "unexpected"). "Belief" itself is quantified by assigning a confidence factor to it. Several methods for assigning degrees to beliefs exist. These include using Bayesian techniques (conditional probability) and statistical techniques (such as the t,F,z tests under certain confidence limits) [4].

Piatetsky-Shapiro and Matheus [7] propose heuristics which form the criteria for developing functions or measures that compute rule interestingness. These principles are summarised as follows:

> Let A→B be the rule that is being evaluated where A and B are the condition and decision parts of the rule respectively. Let N be the file size (i.e. the total number of tuples). Let |A|, |B| be the number of tuples satifying the condition A and B respectively. The rule interest measure (RI) should then satisfy the following requirements:

- RI = 0, if |A & B| = (|A| |B|) / N (i.e. A and B are independent).
- RI *increases* if |A & B| > (|A| |B|) / N (i.e. there is a positive correlation between A and B).
- RI *decreases* if |A & B| > (|A| |B|) / N (i.e. there is a negative correlation between A and B).

[8] extends the measure of interestingness by incorporating features such as "certainty factor" and "coverage" (the number of instances or tuples that satisfy a specific concept). They propose a fourth axiom or heuristic for rule-interest in addition to the three proposed by [7]. This essentially integrates coverage and specifies that given two rules, the one with the greater coverage is more interesting. The rule analysis involves three phases, namely,

- Identification of *potentially* interesting rules.
- Identification of *technically* interesting rules.
- Remove rules that are not genuinely interesting.

The first two phases do not require user intervention, but the final stage involves interaction with a domain expert and so is subjective. [9] proposes a measure for the interestingness of characteristic rules.

Fleury et al [6] introduce a measure called *rule intensity measurement* for evaluating the quality of a decision rule extracted from knowledge discovery systems. Their work is particular to rules extracted from sparse and noisy data. Further, they propose an algorithm for eliminating static rules or dependencies which they claim are "uninteresting" in a knowledge discovery context.

Hsu and Knoblock [3] introduce a new measure for estimating the "robustness" of rules extracted from knowledge discovery systems. Robustness is defined as a characteristic of a rule whereby the rule is unlikely to become inconsistent following changes to the database. That is, a rule becomes inconsistent when a database operation takes place that invalidates the constraints proposed by the rule. Intuitively, semantic integrity constraints (or any integrity constraint) would be highly robust rules since these rules would not be violated by database operations. They propose a measure for robustness based on Laplace Law of Succession and using (if available) transaction logs.

It can be seen that while the rule measures discussed above quantify different characteristics of rules, they do not represent the behaviour of a rule over a period of time. In many data mining applications including semantic query optimisation [10], intelligent query answering [11] and database schema refinement [12] it is important to be able to see how the representative character of a rule changes as a consequence of database transactions (and resulting changes to the state of the database). In this paper we present a new rule measure that addresses this issue.

3. Using the Database Log File to Determine the Dynamic Behaviour of KD Rules

Our rule measure - *Rule Trend Analysis* (RTA) - was developed as part of an automated rule translation process. The theoretical aspects of the rule translation involving the automatic generation of active database rules from decision rules extracted by knowledge discovery systems is discussed in [13]. In this section, we present our technique of rule analysis for data mining patterns.

For a given rule extracted by a data mining system, RTA is determined by examining the database transaction logs. RTA focuses on the number and frequency of database transactions that are compliant with the rule under consideration and presents a comparative analysis of this against the number and frequency of the transactions that are not compliant with the rule. The rule trend can either be positive (i.e. increasing) or negative (i.e. decreasing). Further any rule can have interspersed periods of positive and negative trends. Thus we establish the representative

character of a rule over the history of the database and indicate whether a rule should be incorporated into an established belief system or not.

Let a data mining rule have the following syntactic notation.

$(a_j,v_k) \Rightarrow (D,V)$; j = 1,2..n, k=1,2..m

The pair (a_j,v_k) is the relationship between an attribute *a* and a value *v*. Each pair (a_j,v_k) is separated by a conjunctive connector($\wedge$) or a disjunctive connector($\vee$). *D* is the decision attribute and *V* is its value or state. The predicate on the left of the implication operator (i.e. the antecedent) determines the decision predicate on the right (i.e. the consequent) of the rule. We define the following terms.

Rule Specific Database (R). This is a conceptual abstraction from the original database (*D*) of those specific objects (tuples that involve the antecedent and consequent attributes) that pertain to the rule *r* under consideration. This implies that $R \acute{I} D$.

Rule State. (S_r, E_r). S_r is a set of transactions $t_0, t_1, \ldots t_m$ from *R*. Further within S_r all transactions have identical effects (e.g. all database transactions involving the same attributes having the same values). E_r is the set of events $e_0, e_1, \ldots, e_m$ ordered along a temporal scale of occurrence (i.e. e_{j+1} occurs after e_j ; $0 \le j \le m$) and lead to $t_0, t_1, \ldots, t_m$ respectively. Note that there are no constraints for the events within a rule state to be the same (i.e. any database operation or external event is a valid event). Specifically a rule state can be represented as:

$(S_r,E_r) = \{ (t_0,e_0), (t_1,e_1),\ldots\ldots,(t_m,e_m) \}$

This implies that there is a homomorphic mapping from t to e.

$$\forall e_j: t_j = f(e_j) ; \; j=1,2..,m$$

Therefore the transactions within a rule state are also sequentially ordered based on the time of occurrence (i.e. t_{j+1} occurs after t_j ; $0 \le j \le m$).

We have primarily two classes of rule states: *rule compliant states* and *rule non-compliant states.* Rule compliant state (S_c,E_c) is a consecutive sequence of transactions which satisfy the rule pattern of *r*. Similarly, a rule non-compliant state (S_{nc},E_{nc}) is a consecutive sequence of transactions that do not satisfy the rule. Note that the need to specify rule non-compliant states is primarily due to probabilistic rules, as for an integrity constraint the number of non-compliant states would be zero.

Rule History Log. This is extracted from the database log file as a finite sequence of alternating compliant and non-compliant states. The structure of the rule history log is $(S_0,E_0),\ldots(S_m,E_m)$, where the following constraints hold true:

- For all (S_j,E_j), $0 \le j \le m$ representing a compliant state, if $j > 0$ then (S_{j-1},E_{j-1}) is a non-compliant rule state. This implies that (S_j,E_j) is not the first state in the rule history log. Thus, if a preceding state exists, then that state is non-compliant. The converse is also true that for every non-compliant state if a preceding state exists then that state is compliant.
- For all (S_j,E_j), $0 \le j \le m$ representing a compliant state, if $j < m$ (S_{j+1},E_{j+1}) is a non-compliant state. This implies that (S_j,E_j) is not the final state in the rule history log. Thus, if a following state exists for (S_j,E_j), then that state is non-compliant. The converse is also true that for every non-compliant state if a following state exists then that state is compliant.
- (S_{j+1},E_{j+1}) is the state immediately after (S_j,E_j). This can be derived from the definition of a rule state. Let $(S_j,E_j) = \{ (t_0,e_0), (t_1,e_1),\ldots\ldots,(t_n,e_n) \}$. Therefore the number of transactions in $(S_j,E_j) = n$. Let t_{n+1} be the $(n+1)^{th}$ transaction propagated by the event e_{n+1}. By definition e_{n+1} occurs after e_n and consequently t_{n+1} after t_n. If t_{n+1} has the same outcome as t_n the implication is that $(t_{n+1}, e_{n+1}) \hat{I} (S_j,E_j)$ and $n = n + 1$, which cannot hold true. Therefore e_{n+1} is an event that causes a state change (i.e. t_{n+1} has a different outcome from t_n and marks the beginning of a new rule state in the rule history log). We define a *break point* as a pair (t_j, e_j) which initiates a new rule state in the rule history log. In summary, we established that the rule history log is ordered in a temporal sequence of alternating rule states as illustrated in Figure 1.

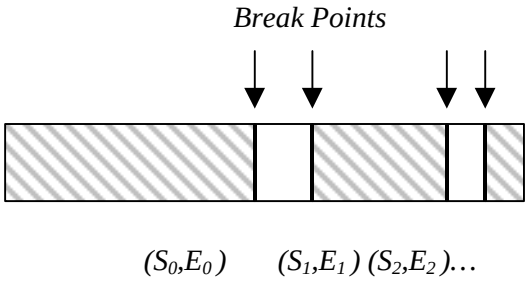

Figure 1 Rule History Log

We introduce two characteristics for rule states namely, $Size[S_r]$ and $Prop[S_r]$. Let the rule history log file have *m* rule states: $(S_0,E_0),\ldots,(S_m,E_m)$. Let $(S_j,E_j) = \{(t_0,e_0), (t_1,e_1),\ldots\ldots,(t_n,e_n) \}$, where $0 \le j \le m$. Then $Size[S_j]$ and $Prop[S_j]$ are:

$Size[S_j] =$ # transactions in $(S_j,E_j) = n_j$

$$Prop[S_j] = \frac{Size[S_j]}{\sum_{k=1}^{m} Size[S_k]}$$

We model rules behaviour or trend using the rule history log file and the *prop* measure of rule states. We obtain a series of ratios by dividing the *prop* values of compliant states by the *neighbour* state (i.e. non-compliant state). The *neighbour* can either be the immediate predecessor or the immediate successor and is determined as follows:

- If (S_0,E_0) is a compliant state then the neighbour is the immediate successor.
- If (S_0,E_0) is a non-compliant state then the neighbour is the immediate predecessor.

Under this scheme *increasing ratios* indicate a positive rule trend and *decreasing ratios* indicate a negative rule trend. This can be derived from the fact the *prop* values are proportional to *size*. Thus as the *prop* of the compliant states increases or the *prop* of the non-compliant states decreases the ratios would increase. Further, a natural corollary of the frequency of non-compliant states decreasing is that the *prop* of compliant states increases. The technique we presented the models the behaviour of a rule over the entire period of the rule history log. The algorithm in Figure 2 illustrates the procedure for obtaining the ratios from the rule history log.

4. Implementation and Experimental Evaluation

The rule trend analysis measure has been incorporated into our prototype CASE tool DBREFINE. In this section we present the object model of DBREFINE illustrated in Figure 3 and the results obtained by implementing this technique.

The significant classes are as follows:

- **RulePredicate.** This class provides the data structure for expressing the components of a decision rule.
- **DecisionRule.** This class specifies the structure and functionality of decision rules.
- **RuleAnalyser.** This is the most significant class in the model and the entire rule analysis (including the rule trend analysis) phase is encapsulated in it.
- **Connect.** This class provides the functionality for communicating with an Oracle database.
- **DBRefine.** This class is specifically designed to produce the final ECA equivalent form of the input decision rule. As is evident, the tool itself is a class from which DBRefine objects are instantiated at the interface level.

The remaining classes in the object model are primarily abstract base classes to extend genericity and facilitate reuse. In order to test the RTA technique we used the Victorian Food and Wine schema developed by [14], which is illustrated in Figure 4. Data mining rules were extracted from the schema (which has been implemented as an Oracle™ database).

Input. Rule History Log File (RHL),
No. of Rule States (n)
Output. Rule Trend (RT).
Method. Rule Stability Estimation.
Begin
Initialize count := 0
If (S_0,E_0) = 'Compliant' and n = 0 then
RT = 'Stable'
Else If (S_0,E_0) = 'Compliant' and n > 0 then
For j := 1 to n step 2 do
series[count] := Prop[S_j] / Prop[$S_{(j+1)}$]
count := count + 1
End for.
Else If (S_0,E_0) = 'Non-Compliant' then
For j := 2 to n step 2 do
series[count] := Prop[S_j] / Prop[$S_{(j-1)}$]
count := count + 1
End for.
End if.
End.

Figure 2 Rule Trend Analysis

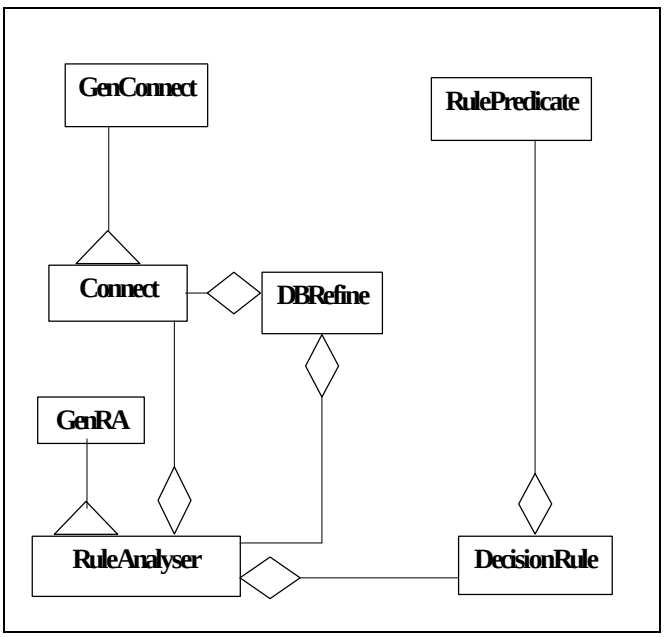

Figure 3 Object Model of DBREFINE

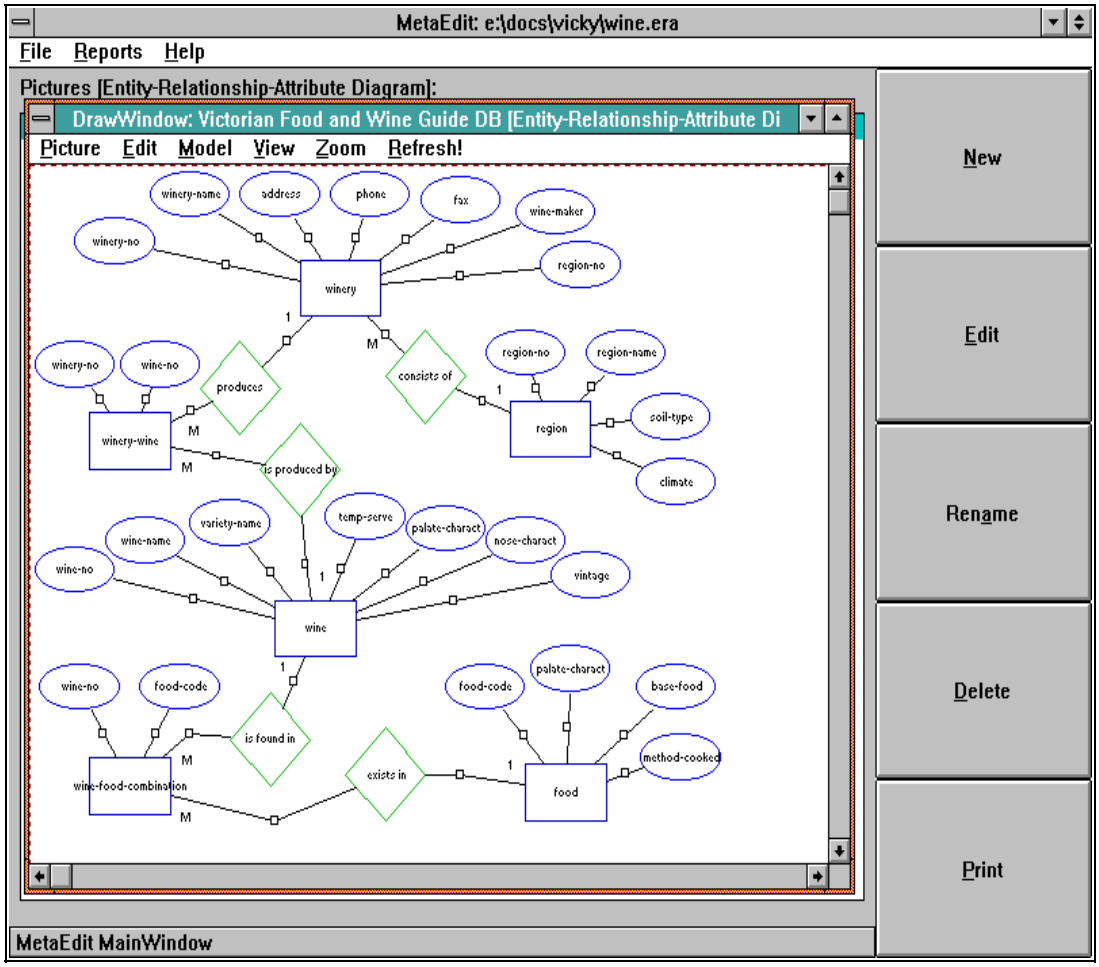

Figure 4 ER Model of the "Wine" Database

DBREFINE has been developed as a 32-bit application using C++ in a Windows 95/NT™ environment. It been developed as an object-oriented system and comprises approximately 4500 lines of code. The connections to the Oracle™ DBMS are made using the Oracle Call Interface (OCI) and this functionality is encapsulated in a separate class in order to make the system less vendor specific and more extensible.

DBREFINE takes as its primary input decision rules extracted from knowledge discovery systems. In order to perform the rule analysis discussed in section 3, it is necessary to be able to access the database schema (this is primarily because knowledge discovery rules do not specify the relations of the attributes involved in the rule). For example, a decision rule extracted from a database would have the following syntactic form:

(Soil_type = 'Deep Sands') $\wedge$ (Climate = 'Warm') => (Vareity_name= 'Dry_white')

Given such a rule, In order to perform the rule analysis, it is necessary to determine the relations to which the attributes (namely, 'Soil_type', 'Climate' and 'Variety_name') belong. it is necessary to access the schema for which the rule has been generated. DBREFINE requires the schema to be specified as an ASCII file from which the rule analyser determines the relations of the attributes or predicates in the rule. The third input required is the location of the database log file to perform the actual rule trend analysis.

The experimental evaluation was performed by simulating the database log file to produce random number of compliant and non-compliant transactions for the extracted rules. As discussed in section 3 of this paper as the log file changes in terms of the number and frequency of the compliant and non-compliant transactions, there is a corresponding change in the rule trend. We present some representative results of our simulation in figures 5, 6, 7.

Simulation #1:

Rule : (Soil_type = 'Red Ironstone') => (Variety_name = 'Dry_red')

Prop Values : 0.13, 0.06, 0.13, 0.09, 0.15, 0.04, 0.11, 0.02, 0.20, 0.02

Series Values: 2, 1.5, 3.5, 5, 9

Rule Stability: Increasing / Positive

Simulation #2:

Rule : (Soil_type = 'Red Ironstone') => (Variety_name = 'Dry_red')

Prop Values : 0.13, 0.06, 0.13, 0.08, 0.15, 0.04, 0.02, 0.04

Series Values: 2, 1.5, 3.5, 7, 3.5

Rule Trend : Decreasing / Negative

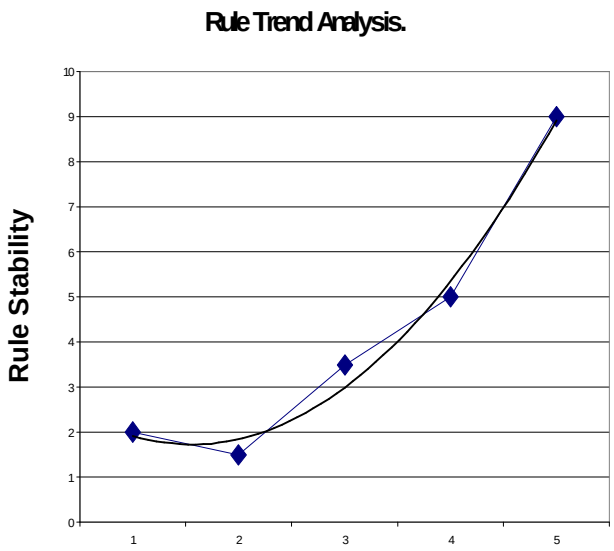

Figure 5 Chart of simulation 1

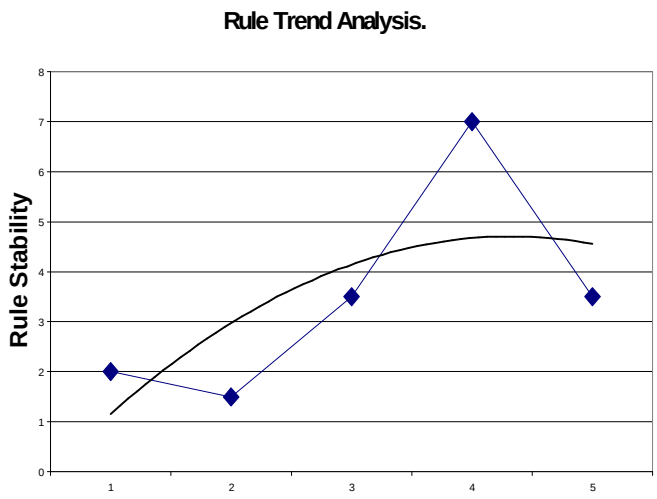

Figure 6 Chart of simulation 2

Simulation #3:

Rule: (Soil_type = 'Red Ironstone') => (Variety_name = 'Dry_red')

Prop Values : 0.13, 0.06, 0.13, 0.08, 0.15, 0.04, 0.02, 0.013

Series Values: 2, 1.5, 3.5, 7, 1.5

Rule Trend : Decreasing / Negative

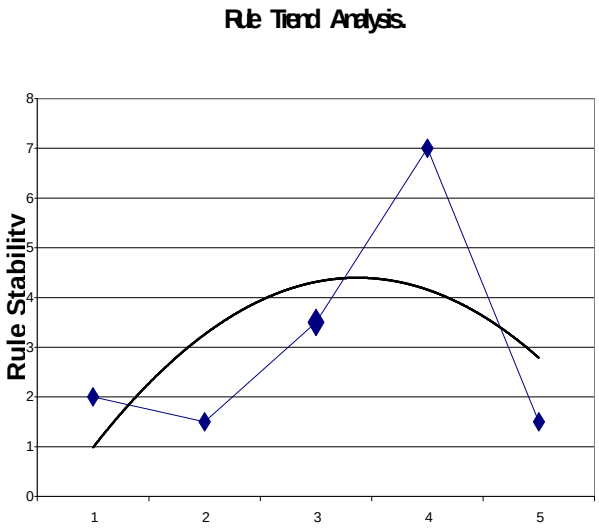

Figure 7 Chart of simulation 3

Thus it can be seen that using the database transaction logs it is possible to model the representative character of a data mining pattern over a temporal scale. Through the testing process it was found that the size of the log file was a major overhead to the system in terms of being memory intensive. The length and size of the log file considerably increased the processing of the rule trend analysis. Incremental scanning of the log file is an extension that we are considering as part of the future development of the tool.

5. Conclusions and Future Work

In this paper we presented a new rule measure for data mining patterns that represents the trend of a rule over a temporal scale. The rule trend analysis is performed using the database log file and presents rules as having positive trends or negative trends based on the number and frequency of the transactions that are compliant with the rule. We have discussed the implementation and presented the results obtained by testing this technique. As shown by the results, the measure provides an intuitive representation of the behaviour or evolution of data mining patterns over a

continuous time interval. The current focus of this project include the following:

- **Automated generation of the schema file.** Currently, DBRefine requires the database schema to be provided as an ASCII file, which is then used by the rule analyser. We are considering the automated generation of the schema file from the database dictionary.
- **Development of a graphical front end.** The current implementation of DBRefine does not have a graphical user interface. We are considering enhancing the functionality of DBRefine by incorporating a graphical front end to it.
- **Incremental scanning of the log file.** One of the weaknesses of the model is that scanning the log file for rule analysis is memory intensive. The incremental scanning of the log file based on a temporal aspect could significantly reduce the memory overhead. Therefore, we are examining techniques for incremental evaluation or analysis of the database log file to enhance the performance and reduce the memory overhead associated with the rule trend analysis.

References

1. Fayyad UM, Piatetsky-Shapiro G, Smyth P. "From Knowledge Discovery to Data Mining an Overview". In: Fayyad U, Piatetsky-Shapiro G, Smyth Pand Uthurusamy R. (ed) *Advances in Knowledge Discovery and Data Mining*, AAAI / MIT Press, 1996, pp. 1-37.
2. Frawley WJ, Piatetsky-Shapiro G. "*Knowledge Discovery in Databases,"*, AAAI / MIT Press, USA, 1991.
3. Hsu C, Knoblock CA., " Estimating the Robustness of Discovered Knowledge, In: Fayyad U and Uthurusamy R (ed) *The First International Conference on Knowledge Discovery and Data Mining*, AAAI Press, Canada, 1995, pp. 156-161.
4. Silberschatz A, Tuzhilin A. "What makes Patterns Interesting in Knowledge Discovery Systems", 1996, *IEEE Transactions on Knowledge and Data Engineering*, 8:6, pp. 970-974.
5. Chen M, Han J, Yu P. " Data Mining: An Overview from a Database Perspective", *IEEE Transactions on Knowledge and Data Engineering*, 1996, 8: 6, pp. 866-883.
6. Fleury L, Djeraba C, Briand H, Phillipe J. "Rule Evaluations in a KDD System", In: Revell Nand Tjoa AM (ed) *Proceedings of the Sixth International Conference on Database and Expert Systems*, Springer-Verlag, pp. 405-414, 1993.
7. Piatetsky-Shapiro G, Matheus CJ. "Measuring Data Dependencies in Large Databases", In: Piatetsky-Shapiro G (ed) *Proceedings of the Workshop on Knowledge Discovery in Databases*, AAAI/MIT Press, California, 1993, pp. 162-173.
8. Major JA, Mangano J. "Selecting Among Rules Induced from a Hurricane Database", In: Piatetsky-Shapiro G (ed) *Proceedings of the Workshop on Knowledge Discovery in Databases*, AAAI/MIT Press, California, 1993, pp. 28-44.
9. Kamber M, Shinghal R. "Evaluating the Interestingness of Characteristic Rules", In: Simoudis E, Han J and Fayyad U (ed) *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, AAAI Press, California, 1997, pp. 263-266.
10. Siegel M, Sciore E, Salveter S. "Rule Discovery for Query Optimisation", In: Piatetsky-Shapiro G and Frawley WJ (ed) *Knowledge Discovery in Databases*, AAAI/MIT Press, California, 1991, pp. 412-426.
11. Han J, Huang Y, Cercone N, Fu Y. "Intelligent Query Answering by Knowledge Discovery Techniques", *IEEE Transactions on Knowledge and Data Engineering*, 8:3, pp. 373-390.
12. Krishnaswamy S, Zaslavsky A. "Activating a Passive Database using Knowledge Discovery Techniques", In: Karlapalem K, Noamin A and Barker K (ed) *Proceedings of the 9th International Conference on Computing and Information* (ICCI'98), Canada, 1997, pp. 17-24.
13. Krishnaswamy S, and Zaslavsky A. "Using Knowledge Discovery Techniques for Database Schema Refinement", In: *Hawaiin International Conference on System Sciences (HICSS-32)*, IEEE Press, Hawaii, 1999, To appear.
14. Constantinidis V, Zaslavsky A., "Engineering an Ingres Active Database Using Conceptual Design Knowledge Elements", In: *Seventh International Conference on Software and Knowledge Engineering (SEKE '95)*, Maryland, USA, 1995.