Finding Actionable Knowledge via Automated Comparison

Lei Zhang[*1], Bing Liu[*2], Jeffrey Benkler[♦3], Chi Zhou[#4]

[*]*Department of Computer Science, University of Illinois at Chicago*
851 S. Morgan St, Chicago, IL, USA 60607
[1]lzhang3@cs.uic.edu
[2]liub@cs.uic.edu

[♦]*Motorola, Inc*
600 N.U.S. Highway 45 MD: AS220, Libertyville, IL, USA 60048
[3]jeffbenkler@motorola.com

[#]*Motorola Labs*
1301 E. Algonquin Rd, Schaumburg, IL, USA 60196
[4]Chi.Zhou@motorola.com

Abstract— **The problem of finding interesting and actionable patterns is a major challenge in data mining. It has been studied by many data mining researchers. The issue is that data mining algorithms often generate too many patterns, which make it very hard for the user to find those truly useful ones. Over the years many techniques have been proposed. However, few have made it to real-life applications. At the end of 2005, we built a data mining system for Motorola (called Opportunity Map) to enable the user to explore the space of a large number of rules in order to find actionable knowledge. The approach is based on the concept of rule cubes and operations on rule cubes. A rule cube is similar to a data cube, but stores rules. Since its deployment, some issues have also been identified during the regular use of the system in Motorola. One of the key issues is that although the operations on rule cubes are flexible, each operation is primitive and has to be initiated by the user. Finding a piece of actionable knowledge typically involves many operations and intense visual inspections, which are labor-intensive and time-consuming. From interactions with our users, we identified a generic problem that is crucial for finding actionable knowledge. The problem involves extensive comparison of sub-populations and identification of the cause of their differences. This paper first defines the problem and then proposes an effective method to solve the problem automatically. To the best of our knowledge, there is no reported study of this problem. The new method has been added to the Opportunity Map system and is now in daily use in Motorola.**

I. Introduction

In many engineering applications of data mining, the task is to find causes of different problems so that improvements can be made to product designs and/or manufacturing processes in order to produce better products in the future. Our work reported in this paper belongs to this type and is done in collaboration with Motorola Inc. In one of the main applications, Motorola's goal is to discover possible reasons of cellular phone call failures (i.e., to identify situations in which all phones or even a particular model of phones are more likely to fail). Engineers can use the discovered information to focus their efforts to determine what may be the root causes of the failures in the phone designs through further laboratory simulation and investigation. Throughout this paper, we will use this application as an example to motivate and illustrate our proposed ideas and the system. This application was also the first application that Motorola was interested in. However, our deployed system (called Opportunity Map) has been used in a large number of other data mining applications involving more than 30 data sets in Motorola since its deployment. Our work is thus general and is not specific to a particular application.

This project started in August 2004 when we were asked to build a data mining system to solve the above problem. The data set (cellular call logs) that we obtained is like any classification data set, except that it is very large. It has more than 600 attributes and more than 200GB of data every month. Some of the attributes are continuous and some are categorical. One attribute indicates the final disposition of the call such as *failed during setup*, *dropped while in progress*, and *ended successfully*. This attribute is the usual class (target) attribute in classification (or supervised learning), and it takes categorical/discrete values. The classes are highly skewed in the data because successfully ended calls represent a very large proportion of the data and the failure cases are rare. However, it is the rate of incidence of the failure classes that is interesting to our users. Unbalanced sampling is used before mining, which has been shown to work quite well.

Two types of data mining are usually performed on such data.

Predictive data mining: The objective of predictive data mining is to build a predictive or classification model from the training data set that can be used to classify future cases or to predict the classes of future cases. This has been the focus of research of the machine learning community. However, our application is not aimed at predicting failure during a call. Instead, we need to find interesting patterns or knowledge that can be used by engineers to improve phone designs, which belongs to diagnostic data mining discussed below.

Diagnostic data mining: The objective of diagnostic data mining is to find causes of problems and/or actionable

1084-4627/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDE.2009.135

knowledge in order to solve the problems. No prediction or classification is needed. In the above example, the problems are *failed during setup* and *dropped while in progress*. A large number of data mining applications in engineering domains are of this type because product improvement is the key issue.

From a research point of view, diagnostic data mining falls in the area of interestingness in data mining, which aims to find interesting and actionable knowledge from a large number of patterns discovered from the data [e.g., 2, 15, 17, 19, 26,, 28]. The interestingness problem can be stated as follows: Many data mining techniques often produce a large number of rules or patterns (e.g., association rules), which make it very difficult for manual inspection of the rules to identify those interesting or actionable ones. Over the years, many techniques have been proposed to deal with this problem in order to help the user find actionable knowledge. However, despite these efforts, interestingness remained to be a challenging problem. Few existing techniques have made it to real life applications [20]. This is in contrast with predictive data mining or classification, which has been highly successful. A large number of classification techniques are widely used in practical applications in almost every domain.

The difficulty of the interestingness problem is often attributed to the fact that interestingness is highly subjective. It depends on the user's current needs and his/her existing domain knowledge [2, 15, 17, 28, 33, 32]. While this is true, in [20] we pointed out several misconceptions of the previous research on interestingness and also discovered a major shortcoming of the existing rule mining paradigm itself. This paradigm tends to fragment the knowledge space and creates many holes in the space, which make it very difficult for the user to find interesting knowledge. A new approach was then proposed, which is based on the idea of rule cubes and operations on rule cubes. A rule cube is like a data cube but stores rules. Operations on rule cubes are OLAP operations [4, 8] with some enhancements and visualization. Thus, the idea is to make it easy for the user to systematically explore the rule space and in the process to find actionable knowledge. In Section III, we will introduce this approach. At the end of 2005, we built a system, called Opportunity Map, based on the approach, which was also deployed in Motorola at that time. The system has been in regular use in Motorola since then. In the past 2.5 years, several improved versions were also built and deployed.

In the application process, some important issues also manifested. One of the key issues is that although the operations on rule cubes are flexible, each operation is primitive and has to be initiated by the user. Finding a piece of actionable knowledge typically involves a large number of operations and extensive visual inspection by the user, which are labor-intensive and time-consuming on the user's part. Thus, the question is whether we can automate some operations to make the process of finding useful knowledge more convenient.

From our observations and monthly interactions with our users, we also discovered that *comparison* is the key to finding interesting and actionable knowledge. Note that all the analysis were done by Motorola engineers using our system as their applications need extensive domain knowledge. Over the period of more than two years, our users showed us many interesting findings. It has always been the case that each finding was presented in contrast with some other piece of information or knowledge. We thus conjecture that a piece of knowledge is only interesting in a *meaningful context* or in *comparison* with some other piece(s) of knowledge (the context), which our users agree.

This conjecture is by no means surprising. It is probably true everywhere in general because in life it is always the differences of things that make them interesting.

Although comparison is an important problem, limited work has been done in the data mining research community. In this paper, we propose a particular comparison function which has been shown crucial in all our applications so far. We thus believe that the proposed function is generic and is likely to be applicable to engineering applications in other domains.

The problem that the function aims to solve is to compare sub-populations of two attribute values with respect to their differences on some target class. For example, our classes are "dropped call" (abnormal) and "successful call" (normal). We have an attribute called Phone-Model. After the user sees that the drop rate (proportion of dropped calls) of phone model 1 (*bad phone*) is significantly higher than the drop rate of phone model 2 (*good phone*). Note that the two phone models are two values of the attribute Phone-Model. The user wants to know which attributes best distinguish the two phones. This involves the comparison of behaviours of the two phones from every attribute dimension. Although our early versions of the system allow the user to compare two phones with regard to an attribute, comparing more than 200 attributes is a daunting task. Note that although the original data set has more than 600 attributes, only more than 200 attributes are related to phone performances. These attributes were identified by our domain experts. The user not only needs to perform a large number of slicing and dicing operations on the rule cubes. He/she also has to manually compare the visualization results of our system. Both these tasks are very time-consuming. In the end, the user may find that Time-of-Call is a significant attribute that distinguishes these two phone models. For example, in the evening, both phones behave similarly, but in the morning and in the afternoon phone 1 performs much worse than phone 2. This piece of knowledge is very useful because based on this the design engineers can investigate what may cause the poor drop rate of the day time from the design point of view so that future designs can avoid the problem.

This work makes the above tedious and mentally demanding comparison task completely automatic. The user only needs to select two phones or any two attribute values to be compared. The system then ranks all the other attributes based on their levels of interestingness and highlights where the user should focus his/her attention on. He/she thus only needs to view the top few attribute to find the highly

distinguishing attributes. In Section III, we will formulate the problem. In Section IV, we propose and justify the formula used to compute the interestingness of each attribute based on the comparison of behaviours of two sub-populations defined using the two attribute values.

The new function has been implemented in the current version of the Opportunity Map system. Our users are very happy about this function. In the past, they have to perform a large number of operations in order to find something interesting. Due to the large number of attributes, they typically only selectively compare a few attributes that they believed to be most relevant. This results in some important and unexpected attributes being overlooked. The new function solves the problem. It not only makes their work easier but also ensures that no important attribute is missed.

II. Related work

This work is related to three main areas of research, rule interestingness analysis, rule visualization and exception mining from data cubes in the OLAP framework. Below we discuss them in turn.

There are several existing interestingness methods that can help the user find interesting knowledge.

Unexpectedness: In this method, the user is asked to give some existing knowledge and the system then finds those unexpected rules [5, 9, 10, 12, 17, 19, 25, 31, 32, 37]. These approaches are based on the comparison of the user's existing knowledge with the discovered rules. However, this approach did not work well in practice because our users were not sure what to expect. They wanted our system to find interesting knowledge for them.

Rule ranking: This method ranks rules according to some interestingness measures [2, 10, 32]. Our experiences show that almost all top ranked rules represent some artifacts of the data rather than any useful patterns. Moreover, giving only individual rules without contexts to compare with is not appropriate or interesting as discussed in our previous work [20]. Although our method presented in this paper also performs ranking, we do not rank rules, but attributes based on comparison. Our approach is also entirely different because we want to find the most distinguishing attributes of two attribute values together with a class, rather than measuring each rule in some way and then ranking all the rules.

Querying and filtering: In [7, 15, 24], some data mining query languages are proposed that can be used to select the right data to mine different types of rules. These systems are not for querying the mined rules. [7, 22, 34, 35] report several rule query languages to enable the user to specify what rules that he/she needs and the system then retrieves the relevant rules. We tried this approach, but our users did not know what to ask as discussed in [20]. In [33], a set of rule post-processing operators are defined to allow the user to filter unwanted rules, select rules of interest to him/her and group rules. These techniques are useful but not sufficient.

In the OLAP context, [29, 30] reports a "discovery-driven" exploration method for mining exceptions from data cubes. Exceptions are cube cells with dramatically larger or smaller values than other cells. The technique in [29, 30] also summarizes the exceptions at appropriate levels of detail or explain them by finding lower level cells which contribute to the exceptions. More specifically, an information theoretic formulation and a dynamic programming algorithm are used to explain differences in multidimensional aggregates. It lets the analyst get summarized reasons for drops or increases as observed at an aggregated level. Our work is quite different. First of all, our cubes contain rules, which are different from the traditional data cubes. Second, our cubes have no hierarchy and thus no multiple levels of aggregations as all attributes are treated at the same and one level. Third, our method does not find exceptions (our previous work in [20] does). Instead, the proposed algorithm finds attributes which best distinguish the user-selected attribute values with respect to a class. The idea will be clear later.

Regarding data mining results visualization, our work is related to rule visualization [13]. [11] proposes interactive mosaic plots to visualize the contingency tables of association rules. In [6], classification rules are visualized using rule polygons. [21] visualizes association rules found in text documents. [38] visualizes the temporal behaviour of rules. [24] uses parallel coordinates to visualize rules. Important rules in terms of support and confidence values are also highlighted with a grid view. [11] uses 3D graphs to visualize rules by emphasizing their supports and confidences. In [13], a post-processing environment is proposed to browse and visualize association rules so that the user can divide a large rule set into smaller ones. In [21], ordering of categorical data is studied to improve visualization, reducing clutter. It is mainly useful for parallel coordinates such as [24] and other general spreadsheet types of visualization.

The above approaches mainly help to visualize individual rules. They do not actively help the user find useful knowledge. They visualize rules without sufficient contextual information because rule mining techniques eliminate much of such information. They thus differ from our approach in terms of both the goal and the visualization. Our visualization is based on rule cubes, OLAP operations and comparisons.

III. Problem Statement

In this section, we first describe the type of rules used in our applications, and then give some background of our previous work on using rule cubes to support the process of discovering actionable knowledge. The problem solved in this paper is defined in the framework.

A. Class Association Rules

As our data sets are typical classification data sets, rules are of the following form

$$X \rightarrow y,$$

where X is a set of conditions and y is a class, e.g., for our example data set $y \in \{$*failed-during-setup*, *dropped-while-in-progress*, *ended-successfully*$\}$. This paper focuses on helping the user identify useful knowledge based on such rules. These rules basically give the conditional probabilities of $Pr(y|X)$,

which are exactly what a diagnostic data mining application is looking for. Moreover, such rules are easily understood by the user, which is a necessary requirement of our applications.

It is easy to see that such rules are *classification rules*, which can be produced by classification algorithms such as decision trees [27], rule induction [16], and class association rule mining [18]. However, traditional classification techniques such as decision trees and rule induction are not suitable for the task due to the following reasons:

1. A typical classification algorithm only finds a very small subset of the rules that exist in data [27]. Most of the rules are not discovered because their objective is to find only enough rules for classification. However, the subset of discovered rules may not be useful in the application. Those useful rules are left undiscovered. We call this the *completeness* problem.
2. Due to the completeness problem, the context information of rules is lost, which makes rule analysis later difficult because the user does not see the complete information.

Class association rule mining [18] is found to be more suitable as it generates all rules in data that satisfy the user specified minimum support and minimum confidence thresholds. Class association rules are a special type of association rules [1] with only a class on the right-hand-side of each rule. Although class association rule mining requires every attribute in the data to be discrete, this is not a problem as there are many existing discretization algorithms that can be used to discretize each continuous attribute into intervals. Class association rules are, however, still not sufficient as we will see below.

B. Rule Cubes

In [20], we proposed to use cubes and OLAP operations on cubes to enable the user to explore the rule space in systematic and flexible ways. Enhanced with several methods to automatically find exceptions, trends and influential attributes (called general impressions [17, 20]), this framework has been shown highly successful in helping the user find actionable knowledge. What is also important is that rule cubes and operations on rule cubes can be visualized easily. Good visualization is a must for real-life applications. Below, we introduce rule cubes. The operations on rule cubes are basically the same as those in OLAP, but without multiple levels of aggregations. In a nutshell, rule cubes are cubes that store rules.

Let the set of attributes in the data D be $A = \{A_1, A_2, \ldots, A_n\}$. Let the class attribute be C. Let the domain or the set of possible values of an attribute A_i be $dom(A_i)$.

Class association rules (CAR) can be converted into cubes as follows: A class association rule miner [18] first mine rules of the form: $X \rightarrow y$, where X is a set of conditions, and $y \in dom(C)$ is a class. A condition is an attribute value pair: $A_i=a_{ij}$ ($a_{ij} \in dom(A_i)$). Every condition uses a distinctive attribute.

Given any subset of attributes, $\{A_{i1}, \ldots, A_{ip}\} \subseteq A$, we use S to denote the set of all possible rules using the set of attributes:

$$S = \{(A_{i1}=v_1, \ldots, A_{ip}=v_p \rightarrow C = c_k) \mid v_1 \in dom(A_{i1}), \ldots, v_p \in dom(A_{ip}), c_k \in dom(C)\}$$

The supports and confidences of the rules are omitted here. It is easy to see that all the rules using the attributes can be represented as a cube with p+1 dimensions (1 being the class attribute). We call this a *rule cube*. The measurement attribute, which we do not have explicitly, is the support/frequency count of data records that satisfy each rule. Thus, each cell of the rule cube is represented with

$$A_{i1}=v_1, \ldots, A_{ip}=v_p, C = c_k$$

Its cell value is the number of data points that contains ($A_{i1}=v_1, \ldots, A_{ip}=v_p, C = c_k$), which is simply the support count of the rule:

$$A_{i1}=v_1, \ldots, A_{ip}=v_p \rightarrow C = c_k$$

The confidence of the rule can be computed with

$$conf(A_{i1}=v_1,...,A_{ip}=v_p \rightarrow C=c_k) = \frac{\sup(A_{i1}=v_1,...,A_{ip}=v_p, C=c_k)}{\sum_{j=1}^{|dom(C)|}\sup(A_{i1}=v_1,...,A_{ip}=v_p, C=c_j)}, \quad (1)$$

where the function *sup* gives the support count of a cube cell. Let us see an example. We have a data set with three attributes. One of them is the class attribute C, which has two values, *yes* and *no*. The other two attributes are A_1 and A_2. A_1 has four possible values a, b, c, d, and A_2 has three possible values e, f, g. Assume that the data set has 1158 data points. The rule cube is shown in Fig. 1, which represents 24 rules (3×4×2).

As an example, the rule, $A_1 = a$, $A_2 = e \rightarrow C = yes$, has the support of 100/1158, and the confidence of 100/(100+50). The rule, $A_1 = a$, $A_2 = f \rightarrow C = yes$, has the support of 0 and the confidence of 0. We see that the support and the confidence of each rule can be easily computed if we have the rule cube. In visualizing rules, we will have both values computed and visualized.

One important note is that to store rules in a rule cube, we need to set both the minimum support and minimum confidence in rule mining to 0. Otherwise, many cells in a cube will be empty as we do not know their counts. Setting the two thresholds to 0 is also very important from a context point of view because it removes holes in the knowledge space and which make discovering actionable knowledge much easier for the user. See [20] for detailed discussions. Clearly, this will result in a huge number of rules due to combinatorial explosion. However, our experiences show that practical applications seldom needs long rules (with three or more conditions). Thus, we only store two-condition rules. When longer rules for some attributes or values are needed, a restricted mining can be carried out, which is done in our system. In our current implementation, we store all 3-dimensional rule cubes. For each cube, one of the dimensions is always the class attribute. The other two dimensions are two other attributes.

OLAP operations, such as roll-up, drill-down, slice and dice, are used to explore these cubes. Mining of general

expressions are also carried out in the process. In the actual visualization, supports and confidences of each rule are also displayed.

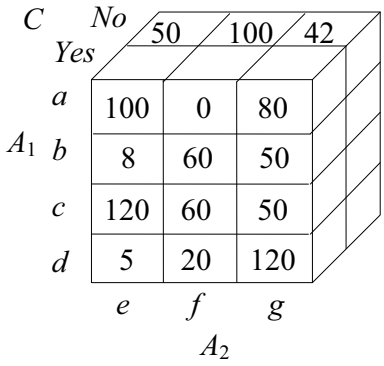


Fig. 1: A rule cube example representing 24 rules.

C. Problem Definition

We now define the specific problem that we want to solve in this work.

Input: Two one-conditional rules composed of two values v_{ij} and v_{ik} of the same attribute A_i and a class of interest c_a:

Rule 1: $A_i = v_{ij} \rightarrow c_a \quad sup_1 \quad cf_1$

Rule 2: $A_i = v_{ik} \rightarrow c_a \quad sup_2 \quad cf_2$

where sup_1 and sup_2 are the supports of the two rules respectively, cf_1 and cf_2 are the confidences of the two rules respectively. These terminologies conform to those in association rules mining [1, 8]. Without loss of generality, we assume that $cf_1 < cf_2$ and both supports (sup_1 and sup_2) are large enough for meaningful analysis (which is decided by the user). Note that in the rule cube framework, the two rules are simply two cells in a rule cube.

It is possible that cf_2 is much large than cf_1, and the user wants to find out what is the cause of the large difference. For example, in our application, drop rates of the two phones are very different. Drop rates are the confidences here.

The task: Compare the two sub-populations of data records with $A_i = v_{ij}$ and $A_i = v_{ik}$ respectively. Let the original data set be D. The two sub-populations, denoted by D_j and D_k, are defined as:

$$D_j = \{d \in D \mid A_i(d) = v_{ij}\} \text{ and}$$
$$D_k = \{d \in D \mid A_i(d) = v_{ik}\},$$

where $A_i(d)$ means the attribute A_i of the data record d.

The objective of the task is to produce a ranked list of attributes:

$$A_{r1} \geq A_{r2} \geq \ldots \geq A_{r(n-1)}$$

where $A_{rf} \in A$ and $A_{rf} \neq A_i$ ($A_i \in A$). The ranking is done based on an *interestingness measure M*, i.e.,

$$\text{if } M_i \geq M_j \text{ then } A_{ri} \geq A_{rj}$$

We will present the interestingness measure in Section IV. Let us use our example application to give an intuition about the interestingness measure.

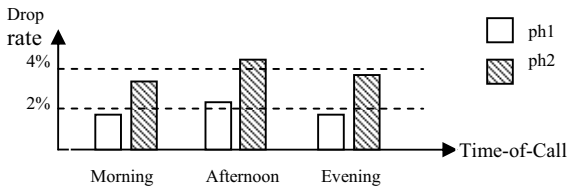
For example, the user sees that two phone models have very different drop rates, which are the confidences of the following two rules:

$$\text{PhoneModel} = \text{ph1} \rightarrow \text{drop} \quad cf_1 = 2\%$$
$$\text{PhoneModel} = \text{ph2} \rightarrow \text{drop} \quad cf_2 = 4\%$$

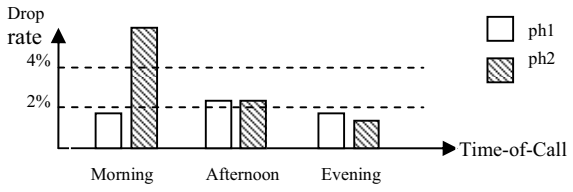
Clearly, phone 1 (ph1) is much better than phone 2 (ph2) on the drop rate (confidence). The idea is that the user wants to know which attributes best distinguish the two phones with regard to the huge difference in the drop rates. In other words, he/she would like to see certain values of the attributes that contribute significantly to the additional drops of ph2 as compared to ph1.

To make this concrete, let us use an example attribute to illustrate what is interesting and what is not. This illustration also serves as a motivation for the proposed interestingness measure presented in the next section. We use the attribute Time-of-Call, which shows the time when each call is made. Assume that the attribute takes only three values, morning, afternoon and evening. Fig. 2 shows two situations: one is interesting and the other is not.

In Fig. 2(A), the drop rates of the two phones are shown side-by-side for morning, afternoon and evening (of the attribute Time-of-Call). As we know that ph2's drop rate is twice as large as that of ph1, then Situation 1 in Fig. 2(A) is not interesting because ph2's drop rate is about twice as bad as that of ph1 for every attribute value (morning, afternoon or evening). Thus, the attribute Time-of-Call in this case does not give us any extra information because we already know that ph2's drop rate is twice as high as that of ph1. This situation is thus expected.



(A): Situation 1



(B): Situation 2

Fig. 2: Illustration of the interestingness measure

However, the situation in Fig. 2(B) is very interesting because we can see that the drop rates of ph1 and ph2 in the

afternoon and in the evening are about the same, but the drop rate of ph2 is much higher than 4% in the morning. Thus, it is quite clear that it is the morning calls that make ph2 bad. This piece of information is highly actionable because it isolates the problem of ph2 (phone 2). The design engineers can then investigate what phone parameters or components cause this high rate of call drops in the morning (all the other conditions are the same for the calls on both phones).

In contrast, in the first situation (Fig. 2(A)), the user still does not know what is wrong with ph2 because the problem with ph2 cannot be isolated based on Time-of-Call.

With this example, we are ready to present the interestingness measure, which is used to rank the attributes. Before we go to that, let us first see how the user can find such important attributes based on OLAP operations on rule cubes. Essentially, for a rule cube that contains the Phone-Model attribute, the user needs to do a slice operation by selecting two values, i.e., ph1 and ph2. The system also needs to compute the confidence values (drop rates) of the drop call class and shows the drop rates along the other dimension (or attribute). The visualization is like any of those in Fig. 2. The user then needs to manually inspect and compare all the bars in order to find whether there is anything interesting. To find the best attribute, the user potentially has to perform these actions for every attribute. If the data set has many attributes (hundreds of them), this is a daunting task. In fact, one of our users (the third author of the paper) literally went through all the attributes one-by-one for one data set because this comparison is extremely important. Imagine in the application, many pairs of phones need to be compared; this becomes an even harder, if not impossible, task. The proposed auto-comparison comes handy and has been shown very helpful.

One question that one may ask is whether the comparison capability is only applicable to different products or it is generally applicable to any attribute values. Clearly, comparing behaviors or performances of different products is useful in any engineering or manufacturing domain because it enables the engineers to pinpoint the specific weaknesses (or strengths) of a product in comparison with its competitors. However, this comparison is also very useful for other attributes. For example, we may find that in general calls in the morning tend to drop much more frequently than in the afternoon. Then, it is interesting to know what cause this poor performance in the morning. It may be discovered that the network equipment is not stable in the morning due to high call volumes.

IV. THE INTERESTINGNESS MEASURE

We now present the interestingness measure based on the comparison of two sub-populations defined in Section III(C). Some other related issues will also be dealt with.

A. The Interestingness Measure

Let the original data set be D and the set of attributes of D be $A = \{A_1, A_2, \ldots, A_n\}$. To simplify the presentation, without loss of generality, we assume that the attribute used in the two rules given by the user for analysis is the first attribute. Then the rules are:

Rule 1: $A_1 = v_i \rightarrow c_a \quad sup_1 \quad cf_1$

Rule 2: $A_1 = v_j \rightarrow c_a \quad sup_2 \quad cf_2$

Again without loss of generality, we assume that $cf_1 < cf_2$. We will not consider the supports (sup_1 and sup_2) as we assume that they are sufficiently large for useful analysis as determined by the user. The sub-populations D_1 and D_2 are:

$$D_1 = \{d \in D \mid A_1(d) = v_i\} \text{ and}$$
$$D_2 = \{d \in D \mid A_1(d) = v_j\}.$$

where $A_1(d)$ is the first attribute of the data record d. Recall our objective is to discover what causes cf_2 to be high as compared to cf_1, i.e., to discover attributes that best distinguishes $A_1 = v_i$ and $A_1 = v_j$ with respect to class c_a.

The two sub-populations or sub-datasets D_1 and D_2 are compared with respect to each attribute from A_2 to A_n. Given an interestingness measure M for comparison, the algorithm for computing the interestingness value (M_i) of each attribute A_i is as follows:

for each $A_i \in \{A_2, A_3, \ldots, A_n\}$ **do**
  $M_i \leftarrow M(D_1, D_2, A_i)$
end-for
rank $A_2, A_3, \ldots, A_n$ based on their interestingness values $M_2, M_3, \ldots, M_n$

Fig. 3: The algorithm

The interestingness computation is based on the attribute values of $A_i \in \{A_2, A_3, \ldots, A_n\}$. Let all possible values (or the domain) of A_i be $v_1, v_2, \ldots, v_m$. We want to aggregate the contribution from each value based on D_1 and D_2. The contribution from each value v_k, denoted by W_k, is computed as follows:

Let
$$F_k = cf_{2k} - cf_{1k} \times \frac{cf_2}{cf_1} \quad (1)$$

$$W_k = \begin{cases} F_k \times N_{2k}, & F_k > 0 \\ 0, & otherwise \end{cases} \quad (2)$$

where cf_{1k} and cf_{2k} are confidence values of the following two rules in the sub-populations (sub-datasets) D_1 and D_2 respectively:

Rule 1: $A_1 = v_k \rightarrow c_a \quad cf_{1k}$ (computed from D_1)

Rule 2: $A_1 = v_k \rightarrow c_a \quad cf_{2k}$ (computed from D_2)

N_{2k} is the data count of the attribute value v_k in D_2.

The meaning of this formula is the following: F_k represents the additional amount of confidence beyond the expected confidence. The expected confidence of cf_{2k} is the second term, $cf_{1k} \times (cf_2/cf_1)$. $F_k \times N_{2k}$ is thus the contribution in terms of the actual number of data records.

If $F_k \leq 0$, then $W_k = 0$ because cf_{2k} is less than or equal to the expected confidence. This is appropriate also because if for one value the confidence is low, the confidences for some other values must be high which will be considered in the computation of F_k because the overall confidence is fixed, which is cf_2.

Finally, the overall interestingness is computed with:

$$M_i = \sum_1^m W_k \qquad (3)$$

Justification of formula (3): Although we are unable to show equation (3) is the best interestingness measure, which is probably impossible, we can show that it is a reasonable measure by justifying it with the two boundary situations, which are given in (A) and (B) of Fig. 4 respectively. They are also quite intuitive.

We again use our running example to illustrate. We are interested in two phones, ph1 and ph2 which have very different drop rates, reproduced below for convenience of discussion. The confidence of each rule (cf_i) is the drop rate.

PhoneModel = ph1 → drop $cf_1 = 2\%$

PhoneModel = ph2 → drop $cf_2 = 4\%$

Let us compare the two phones based on the Time-of-Call attribute (which is A_i) in the sub-datasets D_1 and D_2 (which can be easily obtained and thus omitted).

In Fig. 4(A) (which is similar to Fig. 3(A)), we can see the drop rate comparison of the two phones. According to the drop rate of 2% for ph1 and 4% for ph2, the expected drop rate of ph2 is twice as large as that of ph1 for every attribute value. Then, the situation in Fig. 4(A) is completely uninteresting because the confidence for each attribute value is expected. Thus, its interestingness value M_i should be 0. Indeed, this is the case according to equation (3). This is one extreme (or boundary) case. It is easy to see that 0 is the minimum value of equation (3).

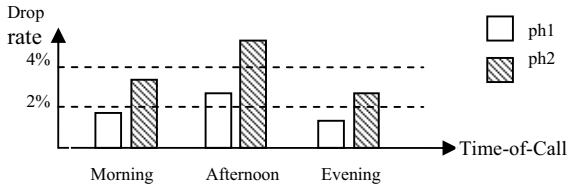
Fig. 4(B) gives the other extreme (or boundary) case, the maximum interestingness. In this situation, there is no drop for ph2 in the morning or in the afternoon. All the dropped calls occurred in the evening, and every evening call results in a drop, i.e., 100% drop rate. Note also that in the evening the drop rate for ph1 is the lowest. It is easy to show that this situation gives the maximum interestingness value for M_i according to equation (3).

Although we use an example here to explain, we can prove mathematically that these are true extremes in general for the interestingness measure given in equation (3).

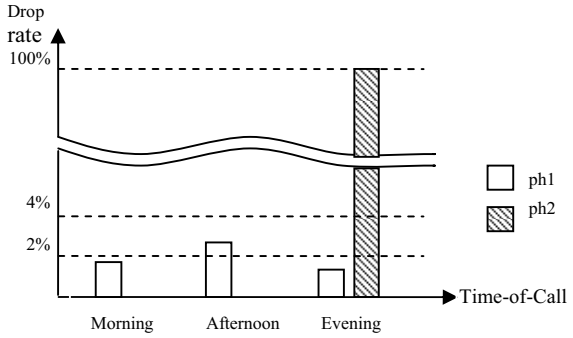
Proof (sketch):

Minimum: From equation (3) and (1), it is clear that the minimum value of M_i is 0. From equation (1), we observe that if every cf_{2k}/cf_{1k} is the same as cf_2/cf_1, then $F_k = 0$ and $M_i = 0$. It is also easy to show that there is no other situation where every F_k can be 0 because for a value v_k if cf_{2k}/cf_{1k} is less than cf_2/cf_1, then for another value v_j, cf_{2j}/cf_{1j} must be greater than cf_2/cf_1 because both D_1 and D_2 are fixed and cf_2 and cf_1 are also fixed.

Maximum: It is clear that there is not a fixed maximum value for M_i in general. However, for each specific problem, there is a maximum value, which is illustrated in Fig. 4(B). That is, all the data records of class c_a in D_2 have only one particular value for the attribute and no data records of any other class have this value, i.e., 100% confidence for c_a, and this attribute value also has the lowest confidence for class c_a for data records in D_1. This is easy to see because in this case $N_{2k} = cf_2 \times |D_2|$ (the total number of data records of class c_a in D_2). The detailed proof requires some algebraic manipulations. We omit it here as it is fairly straightforward. It is also easy to show that there is no other case that can reach this maximum value.



(A): Situation 1



(B): Situation 2

Fig. 4: Two boundary or extremely situations

B. Confidence Interval

An important question that we have not addressed so far is the statistical significance of differences of individual confidence values of rules. For example, if we have two rules with the confidences, $cf_{1k} = 10\%$ and $cf_{2k} = 12\%$, the question is whether the two confidence values are really different statistically. If we cannot show that, our interestingness results are of little use.

In statistics, the common approach to deal with this problem is to compute the confidence interval for each population proportion [23] (which is our confidence value). Note that *confidence value* from data mining [1, 8, 16] and *confidence interval* from statistics [23] are different concepts and should not be confused.

In our interestingness computation, we consider the confidence interval of each confidence value. The details are as follows: We compute the confidence interval margin based on the statistical confidence level of 0.95 using the following formula [23],

$$e_{jk} = Z\sqrt{\frac{cf_{jk}(1-cf_{jk})}{N_{jk}}}$$

where cf_{jk} is the confidence value of the rule with attribute A_i ($\in \{A_2, A_3, \ldots, A_n\}$) and value v_k, and N_{ik} is the number of data records with the value v_k in dataset D_j ($j = 1, 2$). z is a constant

and depends on the required statistical confidence level. Table 2 gives three z values. These values are available in any standard statistics text [e.g., 23]. Thus, the confidence interval for cf_{jk} is simply, $cf_{jk} \pm e_{jk}$

TABLE I
Z VALUE TABLE

Confidence level	**0.90**	**0.95**	**0.99**
z	1.64	1.96	2.58

In the computation of interestingness M_i, we use the following revised confidences:

$rcf_{1k} = cf_{1k} + e_{1k}$

$rcf_{2k} = cf_{2k} - e_{2k}$.

The final formula for F_k is:

$$F_k = rcf_{2k} - rcf_{1k} \times \frac{cf_2}{cf_1}$$

C. Pruning Property Attributes

In some cases, high-rank attributes may not provide us any interesting information. This happens when certain attribute value v_k never occurs in D_1 but occurs frequently in D_2. It is easy to see that in such a case the attribute can be ranked very high because $cf_{1k} = 0$. Two reasons can cause this to happen:

1. Due to some inherent property of the data, it does not use this attribute value. For example, our data set has an attribute called Phone-Hardware-Version, which has two values: "version 1" and "version 2". It so happens that phone 1 (ph1) uses only version 1 and phone 2 (ph2) uses only version 2. Based on our formula, the Phone-Hardware-Version attribute can be ranked quite high, but it is not interesting, and thus should be pruned.
2. Due to some unexpected reason, the value never occurred. For example, it may so happen that ph1 was never used in the morning. This is very interesting as it is unexpected.

We call such attributes *property attributes*. In our system, they are automatically detected and put in a separate list.

Although case (2) is possible, it rarely occurs in practice. Based on our applications, almost all property attributes are not interesting. However, we cannot prune an attribute simply because one such value is detected as other values may still be comparable. Thus, more stringent conditions are used in our system. The following procedure is used to detect property attributes.

Recall that the possible values of attribute A_i are $v_1, v_2, \ldots, v_m$. Let p_{1k} be the number of data records with value v_k in D_1 and p_{2k} be the number of data records with value v_k in D_2. Let P be the number of attribute values that satisfy the condition, (($p_{1k} = 0$ and $p_{2k} > 0$) OR ($p_{1k} > 0$ and $p_{2k} = 0$)).

Formally,

$$P = |\{k \in \{1, 2, \ldots, m\} \mid (p_{1k} = 0 \wedge p_{2k} > 0) \vee (p_{1k} > 0 \wedge p_{2k} = 0)|$$

Let T be the number of attribute values of A_i that satisfy the condition: ($p_{1k} > 0 \vee p_{2k} > 0$), i.e.,

$$T = |\{k \in \{1, 2, \ldots, m\} \mid (p_{2k} > 0) \vee (p_{1k} > 0)|$$

A_i is a property attribute if the following quantity is greater than a threshold τ:

$$\frac{P}{T} > \tau$$

In our system, τ is set to 90%. This parameter is not crucial as property attributes are not physically removed. They are simply stored in another list, which can still be viewed by the user if he/she wishes.

V. Evaluation

Since the proposed system helps the user find subjectively interesting/actionable knowledge, it is difficult to have an objective measure of its effectiveness. As a strong evidence of that, the system has been deployed in Motorola Inc, and it is in regular use for analyzing many types of data sets. Here, we first briefly introduce the Opportunity Map system, and then describe a case study to show how the user interacts with the system to find actionable knowledge. Finally, we also present an experimental evaluation on the computation time.

A. Opportunity Map

The Opportunity Map system consists of six main components: a discretizer, a class association rule (CAR) generator, a general impression (GI) miner, a comparator and a visualizer. Given a data set, all continuous attributes are first discretized using the discretizer (a manual discretization option is also available). The discretized data is fed into the CAR rule generator. The resulting rules form 3-dimensional virtual rule cubes. If the user wishes to have rules with more conditions, the CAR miner can perform a restrict mining to mine rules with some fixed conditions. Otherwise, it will cause combinatorial explosion. The user uses the visualizer to explore the rule space based on OLAP operations. GI miner is called when requested based on the sub-cube shown on screen. GI miner is from our previous work [20], which identifies trends, exceptions and important attributes. The comparator is proposed in this paper. Below, we give more details on rule cube visualization.

Due to the use of rule cubes and OLAP operations, the visualization is simple. In our system, every visualization screen is a 2-dimensional matrix. Each grid in the matrix visualization visualizes one or more cells in a rule cube.

B. A Case Study

We now give a case study using a call log data set to show how the user interacts with the system to find actionable knowledge. The goal is to discover possible causes of call failures (or dropped calls). This version of the data contains 41 attributes, in which one attribute is the class attribute. The majority class covers a very large proportion of the data. Due to confidentiality, we could not disclose the exact percentage. All the attribute names and values are also replaced with

Fig. 5: Initial visualization showing general knowledge: all 2-dimensional rule cubes.

generic names and values.

The visualization has two main modes, overall visualization mode, and detailed visualization mode. In the overall visualization mode (Fig. 5), the X axis is associated with all attributes in the data. The Y axis is associated with all the classes. For each attribute (a column), each grid shows all one conditional rules of the corresponding class value. Each rule is visualized as a thumbnail bar. The height of the bar is the rule confidence value. Thus, this screen simply shows all the 2-dimensional rule cubes. Each rule cube is formed by the class attribute and one other attribute. Each column shows one cube.

The system supports automatic scaling among classes to address the class imbalance issue. Scaling increases relative proportions. In Fig. 5, it has already been applied. Otherwise, we will not see anything for the minority classes, which are the classes that our users are interested in. The proportion of data for each class is shown with the bar on the left.

Blue color is used by default. Some attributes may have so many possible values that the grid size may be inadequate to draw them all. Light blue is used to indicate this. To see all values, the user can either increase the grid size or use a detailed visualization (see below). Various support counts and proportions are written on the screen or in the Information Panel on the right when the user moves the mouse over the screen.

This overall visualization mode is able to summarize a number of important properties of the data immediately:

1. The data distribution of each attribute is illustrated by the distribution bars at the top of each column above the X axis. For the class attribute, they are on the left of the Y axis.
2. One-conditional class association rules are visualized for all attributes. Each rule is represented as a small bar in the visualization, with its context information: rules of all other values of the attribute (X direction in each grid) and all other classes (Y direction across rows) are visualized side by side.
3. Trends are detectable from the shape in each grid. Strong unit trends are indicated using color arrows: red for decreasing, green for increasing and gray for stable trends.

Visualizing all the 2-dimensional rule cubes in the overall visualization is useful to get the user started and to pick the right attributes for further study using the *detailed visualization*.

A detailed visualization shows either a larger version of a 2-dimensional rule cube, or a 3-dimensional rule cube. The details and examples of these visualizations can be found in [20]. Below, we only focus on the new comparison capability.

A comparison starts with the user viewing the detailed visualization of one attribute, e.g., the phone model attribute.

Fig. 6 visualizes the phone model attribute (on the X axis) with all classes (the Y axis). This is simply a 2-dimensional rule cube. It reveals the following detailed pieces of knowledge:

1. The exact drop rates of individual phones.
2. The exact counts and percentages (which are not shown in the overall visualization).

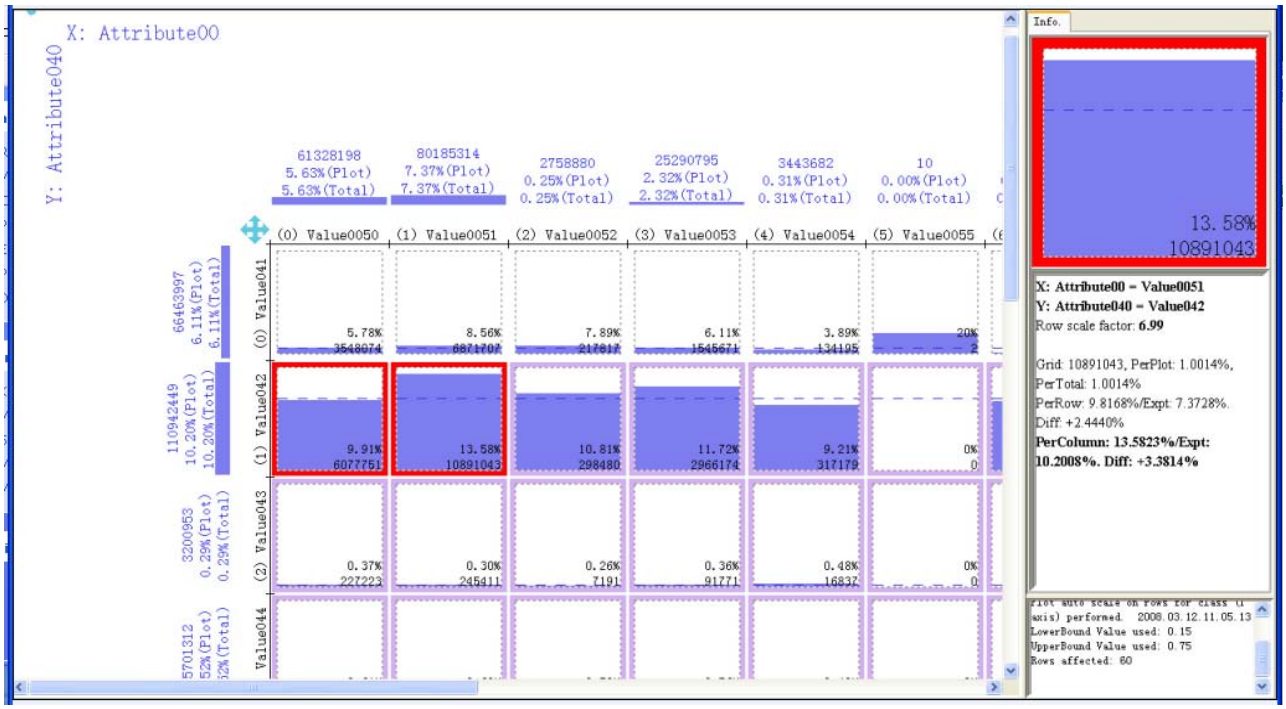


Fig. 6**:** Comparing two phones regarding a particular type of dropped calls in the detailed visualization

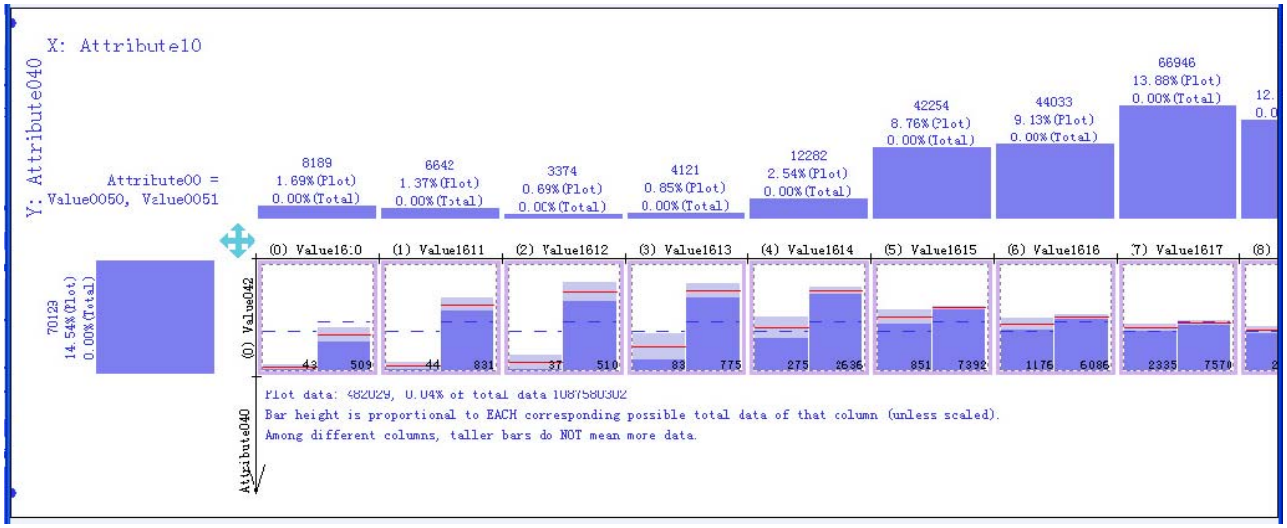


Fig. 7: The top-ranked attribute.

Now the user is interested in finding out why the first phone and the second phone (in the two red boxes) have a big difference in terms of a particular type of dropped calls (the class). Then the user simply chooses these two phones and performs a comparison.

The system uses the method presented in Section IV to rank all the attributes. The top ranked attribute is shown in Fig. 7. Here each grid visualizes the drop rates of the two selected phones (for easy comparison), the first one (on the left) is the good phone (lower drop rate) and the second one (on the right) is the bad phone (higher drop rate).

The red lines are the actual drop rates computed based on the data. The grey region at the top of each bar is the confidence interval computed based on the formula in Section IV (B). It is clear that the bad phone (on the right) is particular bad for the first few values of the attribute. Its drop rates are dramatically higher than the second phone considering the confidence intervals. For the later values, the two phones perform similarly. This piece of information is valuable as it pinpoints the issue of the phone, which enable the engineers to investigate which parts of the phone design cause the problem. Without the new capability, finding the attribute by going

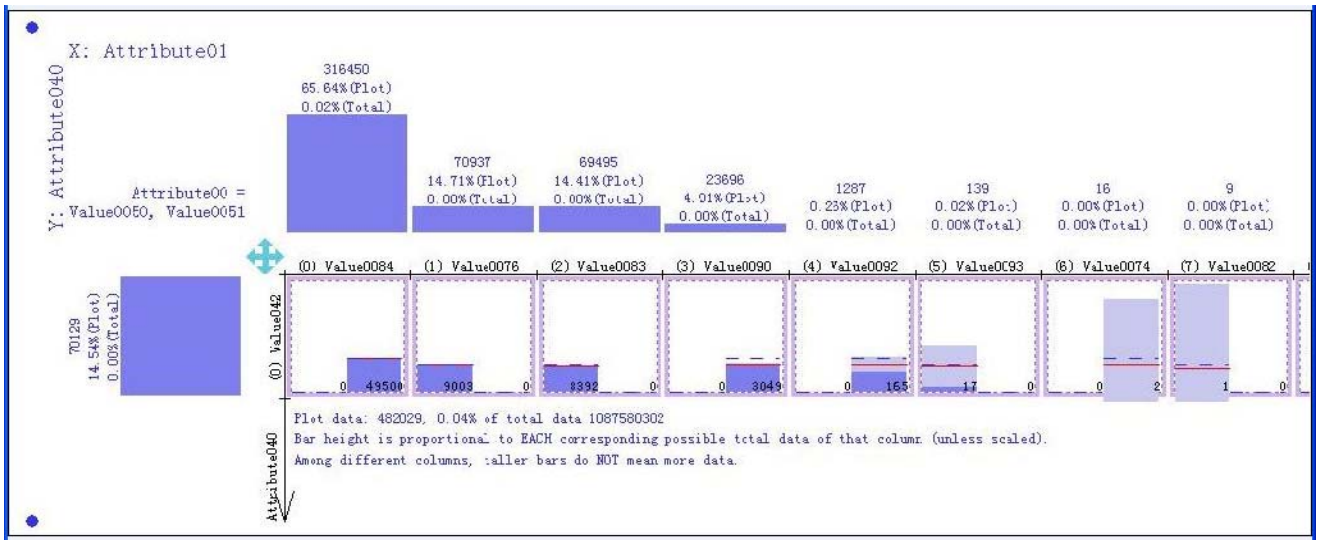


Fig. 8: Property Attribute

through all attributes one-by-one manually is a daunting task.

Fig. 8 shows a property attribute. It can be seen in the first grid on the left that the first phone does not use that attribute value at all (0 count). For the other grids this is true as well, i.e., either the first or the second phone in a grid does not use the attribute value in question. Such attributes are usually not interesting as they are artefacts of the data, rather than true patterns.

C. *Performance Evaluation*

The subsection presents performance and scalability results of the system. All the experiments were conducted on a Dell PC with Intel Core2 Quad CPU 2.40GHZ, 1G memory and Window XP system.

As described above, the comparison computation is based on rule cubes generated by Opportunity Map. We first reports the time required for comparison, which is the focus of this work. Since ranking is involved, more attributes clearly need more time. The system also needs to be interactive when the user is using it. We thus experimented with different number of attributes, i.e., 40, 80, 120 and 160. Note that since the comparison uses only rule cubes, the computation time is not affected by the original data set size (which will be discussed next). Fig. 9 plots the computation time in seconds used in the comparison for each number of attributes.

From the figure, we observe that as the number of attributes increases from 40 to 160, the processing time goes up linearly. What is more important is that even with 160 attributes the system is still highly interactive as it only takes 0.8 second to do the computation. The user will hardly notice it.

Although rule cube generation is not part of this work, for completeness we also include its evaluation result. The data set used here (the same as the one used for comparison) has about 2 million data records and 160 attributes. This data set was being analysed in Motorola at the time when this paper was written. We report two sets of scale-up experiments. The first set shows the execution time as the number of attributes increases from 40 to 160 (all 2 million data records are used) (Fig.10). The second set shows how the system performs as the number of data records increases from 2 to 8 million (all 160 attributes are used) (Fig. 11). To increase the number of data records, we simply duplicate the data set.

Fig. 10 shows a nonlinear growth, which is expected as the number of attributes increases. Fig. 11 is linear as the number of records increases. The long rule cube generation time is acceptable in practice because the generation is done off-line, e.g., in the evening. For huge data sets, sampling is applied.

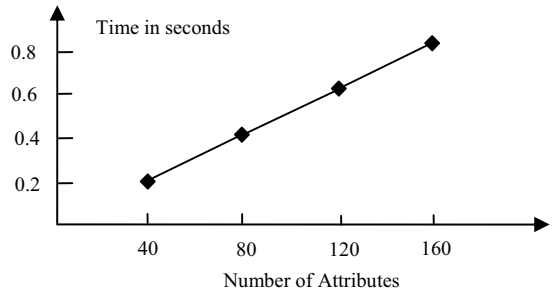


Fig. 9 Comparison computation time on different attribute number

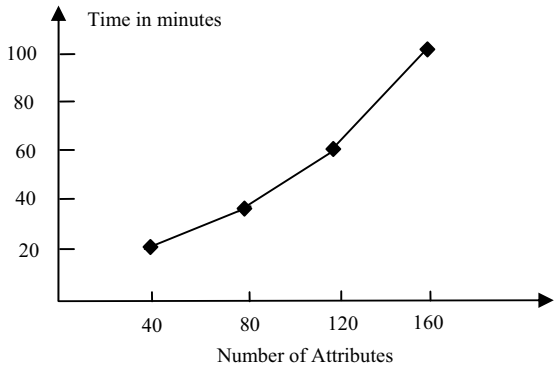


Fig. 10: System computation time on different attribute number

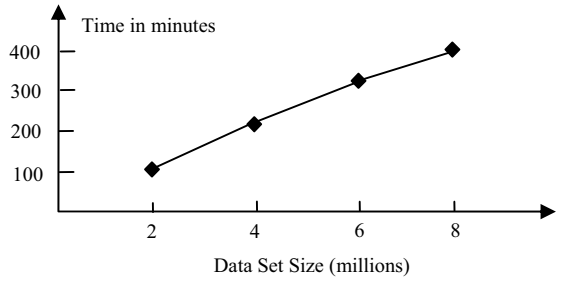


Fig. 11: System computation time on different data set size

VI. CONCLUSIONS

This paper introduces a novel comparison capability, which has been implemented in a deployed data mining system Opportunity Map. The system is in regular use at Motorola. To our knowledge, the proposed comparison has not been studied in the research literature. However, our applications show that it is very important in practice. It not only saves the user a tremendous amount of time and effort, but also enables him/her to examine every possible piece of actionable knowledge systematically and conveniently. Due to the success of the system, it has been used to analyze many different types of data (more than 20 datasets). Most data sets are analyzed continuously for product improvements. Our future work will improve the system further to incorporate more comparison techniques to help the user find more sophisticated and actionable knowledge automatically.

ACKNOWLEDGMENT

We thank Kaidi Zhao for many technical helps. Kaidi implemented the first version of Opportunity Map, which was deployed in Motorola at the end of 2005. We also thank Weimin Xiao from Motorola Labs for many useful discussions. This project is funded by Motorola Inc.

REFERENCES

[1] Agrawal, R. and Srikant, R. “Fast algorithms for mining association rules.” *VLDB-94,* 1994.

[2] Bayardo, R. and Agrawal, R. “Mining the most interesting rules.” *KDD-99*, 1999.

[3] Bendat J., Persol A. *Random data: analysis and measurement procedures*. Wiley-Inter science. 2005.

[4] Chaudhuri, S. and Dayal, U. An overview of data warehouse and OLAP technology. *ACM SIGMOD Record*, March 1997.

[5] Dong G., Li J. Interestingness of discovered association rules in terms of neighborhood-based unexpectedness. *PAKDD-98*.

[6] Han J, Cercone N. "RuleViz: A model for visualizing knowledge discovery process". *KDD-00*, 2000.

[7] Han J., Fu, Y., Wang W., Koperski, K. and Zaiane, O. DMQL: a data mining query language for relational databases. *SIGMOD Workshop on DMKD*, 1996.

[8] Han, J and Kamber, M. *Data mining: concepts and techniques*. Morgan Kaufmann, 2001.

[9] Hussain, F, Liu, H, Suzuki, E., Lu, H. Exception rule mining with a relative interestingness measure. *PAKDD-00*, 2000.

[10] Hilderman, R., Hamilton, H. “Evaluation of interestingness measures for ranking discovered knowledge.” *PAKDD-2001*.

[11] Hofmann H, Siebes A., Wilhelm, A. "Visualizing association rules with interactive mosaic plots". *KDD-00.*

[12] Jaroszewicz, S., and Simovici, D. “Interestingness of frequent itemsets using bayesian networks as background knowledge.” *KDD-04*, 2004.

[13] Jorge A., Pocas J., Azevedo P. "Post-processing environment for browsing large sets of association rules". *PKDD-02 VDM Workshop*, 2002.

[14] Keim D. "Information visualization and visual data mining". *IEEE Trans. Vis. Comput. Graph*, 2002.

[15] Klemetinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A.I. “Finding interesting rules from large sets of discovered association rules.” *CIKM-1994*, 1994.

[16] Liu, B. *Web Data Mining: exploring hyperlinks, contents, and usage data.* Springer. 2006

[17] Liu B., Hsu W. and Chen S., “Using general impressions to analyze discovered classification rules.” *KDD-97*, 1997.

[18] Liu B., Hsu W., and Ma Y. “Integrating classification and association rule mining.” *KDD-98*, 1998.

[19] Liu, B., Hsu, W., Mun, L., & Lee, H. “Finding interesting patterns using user expectations.” *IEEE TKDE*, 11(6), 1999.

[20] Liu B., Zhao K., Benkler, J. and Xiao W. Rule Interestingness Analysis Using OLAP Operations. *KDD-2006*.

[21] Ma, S., Hellerstein J. "Ordering categorical data to improve visualization. " *INFOVIS-99*, 1999.

[22] Meo, R. Psaila, G., and Ceri, S. “A new SQL-like operator for mining association rules.” *VLDB-96*, 1996.

[23] Neter, J, Wasserman, W., and Whitemore, G. A. *Applied Statistics*. Allyn and Bacon, 1993.

[24] Ong K-H, Ong K-L, Ng W-K, Lim E-P. "CrystalClear: active visualization of association rules". *ICDM-02 Workshop on Active Mining (AM-02)*, 2002.

[25] Padmanabhan, B. and Tuzhilin, “A. knowledge refinement based on the discovery of unexpected patterns in data mining.” *Decision Support Systems*, 33(3), July 2002.

[26] Piatesky-Shapiro, G., and Matheus, C. “The interestingness of deviations.” *KDD-94*, 1994.

[27] Quinlan J.R. *C4.5: Programs for Machine Learning*. 1993.

[28] Silberschatz, A, and Tuzhilin, A. What makes patterns interesting in knowledge discovery systems. *IEEE TKDE* 8(6), 1996.

[29] Sarawagi, S. “Explaining differences in multidimensional aggregates.” *VLDB-1999*.

[30] Sarawagi, S. “User-adaptive exploration of multidimensional data.” *VLDB-2000*.

[31] Suzuki, E. Autonomous discovery of reliable exception rules. *KDD-97*, 1997.

[32] Tan, P-N. & Kumar, V. “Interestingness measures for association patterns: a perspective.” *KDD-2000 Workshop on Post-processing in ML and DM*, 2000.

[33] Tuzhilin, A. and Adomavicius, G. “Handling very large numbers of association rules in the analysis of microarray data.” *KDD-02*, 2002.

[34] Tuzhilin, A., and Liu, B. “Querying multiple sets of discovered rules. *KDD-02*, 2002.

[35] Virmani A., Imielinski, T. “M-SQL: A query language for database mining.” *Journal of DMKD*, 1999.

[36] Vapnik, V. *The nature of statistical learning theory*. 1995.

[37] Wang K., Jiang Y., Lakshmanan L. V.S. “Mining unexpected rules by pushing user dynamics.” *KDD-03*, 2003.

[38] Zhao K., Liu B., Tirpak T. and Xiao W. “A visual data mining framework for convenient identification of useful knowledge.” *ICDM-05*. 2005.