User-Assisted Knowledge Discovery:
How Much Should the User Be Involved

(Position Paper)

Avi Silberschatz
Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ

Alexander Tuzhilin *
Information Systems Department
Stern School of Business
New York University

Abstract
The premise of this position paper is that any successful “industrial-strength” knowledge discovery system should have a certain degree of user involvement in the discovery process. We outline a spectrum of degrees of user-involvement and present the Data-Monitoring and Discovery-Triggering approach that provides, in our opinion, a balanced “division of labor” between the KDD application developer and the discovery engine.

1 User-Assisted Discovery Process
There are two diametrically different approaches to pattern discovery in terms of the user-involvement in the discovery process. In the first approach, the users let the system do all the discovery work. They only specify the measures of interestingness for patterns and then “sit back” and watch how the discovery engine “automatically” generates interesting patterns in the data entirely on its own. In the second approach, the users do the discovery of interesting patterns in the data on their own without any help from the search engine. This can be achieved using repeated querying methods, running regressions, using data visualization techniques, as well as various other “manual” discovery methods. For example, the end-user can repeatedly ask SQL queries about the data, each subsequent query formulated based on the results returned by the previous query [KI91]. Using this approach, the user can discover interesting patterns in the data by analyzing answers to the queries. For example, the user can ask a query about annual sales in some company for the last quarter and another similar query for the current quarter and compare the two results. If the sales in that company changed significantly this quarter in comparison to the last quarter, then the user may have detected a potentially interesting pattern. Figure 1 summarizes these two approaches: the first “manual” approach is presented in the first column, while the second “automatic” approach is presented in the last column.

*This work was supported in part by the NSF under Grant IRI-93-18773.
approach to the delicate task of data archaeology. Also, in [Bra93], the IMACS system was presented, which discovers interesting patterns in the data in a semi-automatic way. Other systems besides IMACS take the “semi-automatic” approach to pattern discovery. For example, Knowledge Miner [SOMZ96, KRTS95] proposes an iterative process between high-level, user-specified patterns, their elaboration to database queries, and the analysis of these queries which results in the modified knowledge that will be used in the next cycle.

We agree with [Bra93] and [KRTS95] on the importance of user involvement in the knowledge discovery process and believe that any successful industrial-strength knowledge discovery system should follow the middle-ground approach and find a right degree of user involvement in the discovery process. Of course, an important issue for this approach is to find the right “division of labor” between the user and the search engine so that neither of them is “overwhelmed” with the required work.

In this paper, we address the issue of the level of user involvement in the discovery process and present the Data-Monitoring and Discovery-Triggering Paradigm that has a good balance between the roles of the user and the search engine in the discovery process.

2 Level of User Involvement in the Knowledge Discovery Process

It is impossible to discuss the role of the user in the discovery process without differentiating between various types of users since they play different roles in the discovery process. In this paper, we consider the two types of users: end-users who utilize the discovery system to extract interesting patterns from the data and KDD application developers who set up discovery systems and applications for the end-users to work with.1 These two types of users have different roles in the discovery process, as discussed below.

We believe, as well as other researchers (e.g., [Bra93, KRTS95]), that the process of knowledge discovery in large industrial applications is inherently an interactive process between the end-user and a KDD system. Then the role of the end-user is to analyze the patterns discovered by the system and provide the feedback on where the KDD system should focus the search for new patterns. For example, in the abstract-driven discovery framework [DT93], the end-user should interactively provide information about the regions of the search space on which the discovery system should concentrate. This concept of interactive communication between the end-user and the discovery search engine is graphically presented in Figure 2. The end-user provides inputs to the discovery engine about where to search for patterns in the search space. When the patterns discovered by the search engine are shown to the end-users, they analyze them and, based on this analysis, generate new inputs to the search engine on where to search in the next round of discovery iterations.

We believe that the role of the KDD application developer is to specify the conditions of when and how the discovery search engine should be triggered. It is also our belief that the “here-is-the-data-find-me-patterns” discovery paradigm is neither efficient, nor desirable. A more efficient and promising approach would be to monitor the data for certain significant changes in it and, based on these changes, fire discovery processes. In this approach, the role of a KDD developer would be to set up the monitoring and triggering mechanisms. We describe this Data-Monitoring and Discovery-Triggering paradigm in the next section.

3 Data-Monitoring and Discovery-Triggering Paradigm

The most interesting and challenging discovery problems arise when the data changes over time because the patterns also keep changing with the data. This is a typical situation in On-Line Transaction Processing (OLTP) systems, such as airline reservations, banking,
and insurance claim processing systems. Therefore, we will focus in this paper on an environment where new data is periodically added to the previously collected historical data.

Figure 3 illustrates graphically the Data-Monitoring and Discovery-Triggering (DMDT) paradigm for pattern discovery in changing data. Accordingly, the KDD developer sets up some monitors for spotting “significant” changes in the data. An important example of a significant change to the data might be an abnormal condition, such as sales in the third quarter dropped down by more than 10% in the Western Division of the company. Once the data monitor detects a significant change in the data, it triggers a discovery process by specifying the initial set of pattern templates that serve as inputs to the “Discovery Process” module. The Discovery module takes these pattern templates and searches for patterns corresponding to them. The discovered patterns are presented to the user and are also used for adjusting data monitors in a feedback loop.

Before describing the DMDT knowledge discovery scheme in greater detail, we first overview the concept of a pattern template introduced by other researchers.

### 3.1 Pattern Templates

Pattern templates are constructs for the specification of types of patterns to be discovered. Several researchers considered patterns templates in the context of knowledge discovery [Imi95, KMR+94, SOMZ96].

In [SOMZ96], pattern templates are called *metaqueries* and can be viewed as a two-part specification: the left-hand side (LHS) specifies a constraint on how data should be prepared, and the right-hand side (RHS) specifies an action to be applied on the prepared data. For example, consider metaquery

\[ P(X, Y) \land Q(Y, Z) \Rightarrow R(X, Z) \]

This metaquery is a template that instantiates various transitivity patterns depending on the predicates to be substituted for templates \( P, Q, \) and \( R \) and the attributes and values substituted for variables \( X, Y, \) and \( Z. \)

To illustrate, consider the transitivity pattern \( p(X, Y) \land q(Y, Z) \Rightarrow r(X, Z) \) [with probability \(Pr\)]. This pattern satisfies the above metaquery, where predicates \( p, q, \) and \( r \) are specific database relations, and \( Pr \) is the ratio of the \((X, Z)\) pairs satisfying the LHS and the RHS of the rule and those satisfying only the LHS. Then, the action in the RHS of the metaquery lies in computing the value of \(Pr\).

Imielinsky discussed a related concept of pattern templates in his Invited Talk at KDD-95 [Imi95], where he proposed a logical rule to be augmented with one or several operators that specify which patterns the KDD system should explore. For example, the template

\[ r(X, Y) \land q(Y, Z^+) \Rightarrow r(X, Z) \]

refers to the set of rules of the form

\[ r(X, Y) \land q(Y, Z) \land Z < c \Rightarrow r(X, Z) \]

where \(c\) is some constant.

The notion of a pattern template was also introduced in [KMR+94] for the purpose of identifying interesting patterns. A pattern template in [KMR+94] is a rule

\[ A_1 \land \ldots \land A_k \Rightarrow A_{k+1} \]

where each \(A_i\) is either an attribute name, a class name or an expression \(C+\) or \(C^*\) corresponding, respectively, to one or more elements or instances of the class \(C\). Such a template defines a class of rules that are instances of the template pattern. For example, consider a pattern template that says that students can take an advanced elective in some department if they have taken a core course offered by that department and one or several electives:

\[ \text{CORE(Dept,Student)} \land \text{ELECTIVE(Dept,Student)}+ \Rightarrow \text{ADV.ELECTIVE(Dept,Student)} \]

where \(\text{CORE}\), \(\text{ELECTIVE}\), and \(\text{ADV.ELECTIVE}\) are classes of certain courses offered at that university. Given this template and assuming that \(\gamma\) and \(\sigma\) are confidence and support thresholds [KMR+94], the rule

\[ \text{Intro_CS(CS,Student)} \land \text{Programming_Lang(CS,Student)} \land \text{Operating_Systems(CS,Student)} \Rightarrow \text{Special_Topic_in_CS(Students)} \ [\gamma, \sigma] \]

matches this template. Then, a pattern is interesting if it matches an inductive template [KMR+94].

Pattern templates are used for the discovery purposes in [Imi95] and [SOMZ96], where discovery processes based on pattern templates are discussed.

We will use pattern templates in our DMDT paradigm. However, in our case it does not matter which specific pattern templates and the associated discovery methods are used because the DMDT paradigm can accommodate any of these approaches. For example, metaqueries of [SOMZ96] can be used together with the discovery engine of Knowledge Miner [SOMZ96], or templates
of [Imi95] can be used in conjunction with the pattern discovery algorithms discussed in [Imi95]. Therefore, we will treat the pattern discovery component of the DMDT paradigm (see Figure 3) as a "black box" with the pattern templates as associated inputs.

3.2 DMDT Triggers

The processes occurring in the first two boxes in Figure 3 can be described with the set of extended (or DMDT) triggers that can be defined as follows. Let \( D \) be the old (historical) data stored in a database (e.g., student's grades over the past 10 semesters) and let \( \Delta D \) be new data to be added to the database (e.g., student’s grades for the current semester). Then, an extended trigger has the form

\[
\text{WHEN} \quad \text{new data } \Delta D \text{ becomes available} \\
\text{IF} \quad \text{"significant changes" in the data are observed when } \Delta D \text{ is added to the old data } D \\
\text{THEN} \quad \text{activate pattern templates}
\]

We call these triggers "extended" because they are extensions of classical triggers used in active databases [DHW94, WCD95] in the sense that the IF- and THEN-clauses cannot be always expressed in terms of the standard active database triggers [DHW94, WCD95].

The monitor for this rule is specified in its IF-clause and checks for "significant" changes in the data. The monitor is an expression in a first-order logic that also supports arbitrary user-defined functions.\(^2\) For example, the monitor may watch for changes to user-specified beliefs [ST95, TS] or look for the students who had an "outstanding performance" during the current semester, where "outstanding performance" is a complex statement that cannot be defined as a logical expression without user-defined functions (e.g., in SQL).

A monitor is activated periodically when the new data of the type specified in the WHEN-clause of the trigger becomes available. Some examples of these types of events are

- When grades for the current semester become available
- When daily banking transactions are recorded in the database
- When a new quarterly report of a company becomes available

When the monitor of a trigger gets activated by the events of the WHEN-clause and if the monitor detects the conditions specified in the IF-clause of the trigger, then the pattern templates specified in the THEN-clause of the trigger get activated and are added to the set of pattern templates used in the pattern discovery process.

Example 1: Consider the following DMDT trigger specifying that

When grades for the last semester become available and if women received better grades than men in more than 75% of the courses during that semester, then activate pattern templates R1 and R2

where R1 and R2 are defined as follows.

Let \( \text{GRADES}(\text{COURSE\_NO,SEMESTER,YEAR, AVG\_GRADE\_MEN,AVG\_GRADE\_WOMEN}) \) be the relation specifying average grades of men and women in a course having number COURSE\_NO and offered in semester SEMESTER and year YEAR and let the relation \( \text{GRADES1}(\text{COURSE\_NO,SEMESTER,YEAR, AVG\_GRADE,GENDER}) \) be similar to GRADES, but has additional attribute GENDER (males or females), and only one average grade attribute (for males or for females depending on the value of GENDER attribute). Then pattern templates R1 and R2 are

\(^2\)Typically, these functions are defined using one of the programming languages, such as C.
R1: In school X, women outperformed men in more than 75% of the courses, where X ranges over the set of schools at that university, such as School of Arts and Sciences, Business School, Schools of Engineering, Medicine, Law, and so on.

\[
\text{GRADE}(\text{course}_\text{no}, \text{semester}_\text{year}, \text{avg}_\text{grade}_\text{men}, \text{avg}_\text{grade}_\text{women}) \land \text{SCHOOL}(\text{course}_\text{no}) = X \\
\rightarrow \text{avg}_\text{grade}_\text{women} > \text{avg}_\text{grade}_\text{men} \text{ (with probability 75\%)}
\]

R2: There were large deviations (e.g., more than 10\%) in the grades of students of sex X (males or females) between this and the previous semesters.

\[
\text{GRADE}_{\text{S}}(\text{course}_\text{no}, \text{semester}_\text{year}, \text{avg}_\text{grade}_\text{X}) \text{ and } \text{GRADE}_{\text{L}}(\text{semester}_\text{year}_', \text{semester}_\text{year}_'', \text{avg}_\text{grade}_\text{X}') \text{ and } \text{NEXT}(\text{semester}_\text{year}_', \text{semester}_\text{year}_'', \text{year}') \\
\rightarrow \text{Large Deviation}(\text{avg}_\text{grade}_\text{X}')
\]

Note that R1 and R2 are pattern templates because they are rules parameterized by variables, such as X, and predicate templates, such as Large Deviation. For example, predicate template Large Deviation can be instantiated to \((\text{avg}_\text{grade}' - \text{avg}_\text{grade}) / \text{avg}_\text{grade} > 10\%\).

Given these definitions, the above trigger can be formally expressed as

\[
\text{WHEN} \quad \text{NEW GRADES ARRIVE}(\text{semester}, \text{year}) \\
\text{IF} \quad \text{No courses women did better}(\text{semester}, \text{year}) \\
/ \text{Total no courses}(\text{semester}, \text{year}) > 75\% \\
\text{THEN} \quad \text{activate pattern templates R1, R2}
\]

where \text{NEW GRADES ARRIVE}(\text{semester}, \text{year}) is an event specifying the arrival of grades for a given semester and

\[
\text{Total no courses}(\text{semester}, \text{year}) = \text{COUNT}\{ \text{course}_\text{no} | \text{GRADE}(\text{course}_\text{no}, \text{semester}_\text{year}, \text{avg}_\text{grade}_\text{men}, \text{avg}_\text{grade}_\text{women}) \}
\]

\[
\text{No courses women did better}(\text{semester}, \text{year}) = \text{COUNT}\{ \text{course}_\text{no} | \text{GRADE}(\text{course}_\text{no}, \text{semester}_\text{year}, \text{avg}_\text{grade}_\text{men}, \text{avg}_\text{grade}_\text{women}) \text{ AND } \text{avg}_\text{grade}_\text{men} < \text{avg}_\text{grade}_\text{women} \}
\]

When a trigger is fired and if its monitor detects “significant changes” specified in the IF-clause, then the set of pattern templates in the THEN-clause is being activated (i.e., these templates are added to the list of pattern templates that will be used in the discovery phase.) Once all the templates in all the DMDT triggers that fired in response to the addition of new data \(\Delta D\) are activated, the resulting set of templates, called the activated set, is served as the input to the discovery module in Figure 3.

Once new patterns are discovered based on the activated set of templates, these patterns should be used in a feedback loop for adjusting existing and introduction of new data monitors. For example, after running the pattern discovery process with templates R1 and R2, it may turn out that the threshold figure of 75\% in the monitor of the DMDT trigger from Example 1 is too high. Based on this information, we may want to adjust it to a lower value. Similarly, we may want to introduce new monitors if the discovered patterns suggest so. For example, if it is found that a competing product was eroding company’s sales in the Western Division, then the KDD developer may want to set an additional monitor that tracks the sales of the competing product.

3.3 The Role of the User in the DMDT Process

The DMDT paradigm is an example of a case of limited user involvement in the discovery process. The KDD developer has to set up the DMDT triggers that monitor significant changes to the data and these triggers launch discovery processes. As we can see, this paradigm still leaves room for the discovery since the activated pattern templates are used by the discovery engine for finding interesting patterns.

Note that the process of pattern discovery in the DMDT case is not a “blind” search since it uses activated pattern templates for focusing on the patterns related to these templates.

Therefore, the DMDT paradigm requires both the user involvement in setting triggers and also focuses the search for interesting patterns by using sets of activated pattern templates. Thus, the DMDT paradigm corresponds to the middle-ground (semi-automatic) approach shown in Figure 1.

4 Conclusion

We believe that in large commercial applications the user needs to be actively involved in the search for interesting patterns. For the end-user, this means interactive involvement in the search, and for the KDD application developer this means specification of conditions of when and how the discovery search engine should be triggered. We believe that the Data-Monitoring and Discovery-Triggering (DMDT) paradigm has a good balance between the roles of the KDD application developer and the discovery search engine.

We are currently developing a belief-driven discovery framework based on the DMDT paradigm [TS] that should further enhance the DMDT ideas.

References


