A BELIEF-DRIVEN DISCOVERY FRAMEWORK BASED ON DATA MONITORING AND TRIGGERING

Avi Silberschatz
Bell Laboratories
700 Mountain Avenue
Murray Hill, NJ 07974

Alexander Tuzhilin
Department of Information Systems
New York University
Leonard N. Stern School of Business
44 West 4th Street, Suite 9-78
New York, NY 10012-1126
atuzhili@stern.nyu.edu

December 1996

Working Paper Series
Stern #IS-96-26

A Belief-Driven Discovery Framework Based on Data Monitoring and Triggering

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

Avi Silberschatz
Bell Laboratories
700 Mountain Avenue
Murray Hill, NJ

Abstract

A new knowledge-discovery framework, called *Data Monitoring and Discovery Triggering (DMDT)*, is defined, where the user specifies *monitors* that "watch" for significant changes to the data and changes to the user-defined system of *beliefs*. Once these changes are detected, knowledge discovery processes, in the form of *data mining queries*, are triggered. The proposed framework is the result of an observation, made in the previous work of the authors, that when changes to the user-defined beliefs occur, this means that there are interesting patterns in the data. In this paper, we present an approach for finding these interesting patterns using data monitoring and belief-driven discovery techniques. Our approach is especially useful in those applications where data changes rapidly with time, as in some of the On-Line Transaction Processing (OLTP) systems. The proposed approach integrates active databases, data mining queries and subjective measures of interestingness based on user-defined systems of beliefs in a novel and synergetic way to yield a new type of data mining systems.

1 Introduction

It has been argued in the KDD community that large "industrial strength" KDD systems should have the user be actively involved in the knowledge discovery process and that successful KDD systems should have a proper balance of pattern searching and user guidance capabilities [4, 7, 14, 17]. In this paper, we present a new knowledge-discovery framework, called *Data Monitoring and Discovery Triggering (DMDT)*, that combines the "hints" from the user as to what patterns to search for with various searching mechanisms. In particular, in the DMDT framework a user specifies *monitors* that "watch" for significant changes in the data, and, once these changes are detected, the discovery processes are triggered.

The DMDT framework is best motivated by the following typical business case example. Assume that you are the Chief Executive Officer (CEO) of a company and that you have a set of key indicators that are of crucial importance to your business (e.g., the total volume of sales, profit margins, and other critical success factors that you have to watch very closely). In addition, you have a certain set of beliefs about the way your company operates; for example, the belief that men

and women should have equal salaries on average, and the beliefs that there should be no other types of job discrimination (racial, religious, etc.) among your employees. Thus, you would like to monitor how the key indicators, including degrees of your set of beliefs, change with the updates to the old data and the addition of new data, and "raise flags," when unusual changes occur. However, the "devil is in the details," and it may not be sufficient for you to know that the sales of your leading product dropped by 15% this quarter. You may want to find other "deeper" patterns in the data related to the drop in sales, such as the observation that the key contributor to the drop in sales was a poor performance of the product in the North-Western region. Therefore, although you can watch for a few crucial aggregate pieces of information, you don't know what to look for "deeper" in the data.

This problem can be addressed by setting up *triggers* [6, 20] that *monitor* significant changes to the data and changes to the user-defined system of *beliefs* (e.g. indicators and beliefs important to the CEO). Once these changes are detected, knowledge discovery processes, in the form of *data mining queries*, are triggered. These discovery processes result in the generation of patterns satisfying the conditions of the data mining queries.

One of the important types of indicators are changes to the user-defined beliefs. For example, the CEO may discover that recently the average men's salaries started to exceed average women's salaries throughout the company, which violates the CEO's belief that the salaries should be equal. These changes can be detected through the use of *belief-driven discovery* in which the user defines a set of beliefs crucial to the application. Thus, the knowledge discovery system monitors changes to the degrees of these beliefs, and, once significant changes are detected, the discovery processes are launched. We use changes in degrees of user-defined beliefs to trigger discovery processes because we showed in [15] that these changes indicate existence of interesting patterns in the data. In this paper, we describe a technique to find these patterns. We also describe in the paper how to construct a system of beliefs in order to make the belief-driven discovery practical.

The DMDT framework is especially useful in those applications where data changes rapidly with time, as in some of the On-Line Transaction Processing (OLTP) systems, such as airline reservation, credit card processing, and ATM machine supporting systems. In this case, patterns keep changing over time, and new trends keep emerging in the data. Also, previously held beliefs change with time, and these changes can give the application developer hints of what to search for in the data.

The contribution of this paper lies in the introduction of the DMDT framework and the belief-driven discovery scheme. In particular, we introduce a new type of triggers suitable for knowledge discovery and combine the concepts from active databases [6, 20], data mining query languages

[10, 8, 14], beliefs, and belief revision systems [9, 18] in a coherent way when defining these triggers.

The remainder of the paper is organized as follows. In Section 2, we describe how our work builds on three areas of research: active databases, data mining query languages, and subjective measures of interestingness. In Section 3 we define the DMDT framework, and in Section 4 the belief-driven discovery scheme. Finally, we explain in Section 5 how to build a belief system in a practical way.

2 Related Work

The DMDT framework and the belief-driven discovery scheme lies at the confluence of the following three streams of research: (1) active databases; (2) data mining query languages, such as M-SQL [10], DMQL [8], and Metaqueries of [14]; and (3) our previous work on subjective measures of interestingness and on the use of beliefs and belief revision to define these measures of interestingness. In order to understand the DMDT framework and the belief-driven discovery scheme, we first have to explain the relationship of our work to these three areas.

2.1 Subjective Measures of Interestingness and Beliefs

We first define subjective measures of interestingness in terms of user-defined beliefs and then explain how they are related to the belief-driven discovery.

In [15, 16], we studied subjective measures of interestingness of patterns. In particular, we argued that one reason why a pattern is interesting is because it is unexpected (how often we are surprised with something and say "Gee, this is interesting..."). We also argued in [15] that unexpected patterns are interesting because they contradict our expectations which, in turn, depend on our system of *beliefs*. We defined beliefs in [15] as logical statements (expressions in first-order logic) and assigned a *degree* (or a *measure*, or a *confidence factor*) to each belief. For example, we may believe in the context of the university database that men and women should receive equal grades on average. Based on the past *evidence* ξ about students' grades, we can assign a certain degree to this belief b, i.e. $d(b|\xi)$, where d is a real number between 0 and 1 [15].

Beliefs have been extensively studied in the past, and there have been different approaches to defining degrees of beliefs proposed in the literature. Some examples of these approaches are [15]:

1. *Bayesian approach.* In Bayesian approach the degree of belief b is assigned as a conditional probability $P(b|\xi)$, where ξ is a past evidence. Then, given new evidence E, the degree of

belief is revised using the Bayes rule as

$$P(\alpha|E,\xi) = \frac{P(E|\alpha,\xi)P(\alpha|\xi)}{P(E|\alpha,\xi)P(\alpha|\xi)+P(E|\neg\alpha,\xi)P(\neg\alpha|\xi)} \quad (1)$$

2. *Dempster-Shafer Approach.* In Dempster-Shafer theory of evidential reasoning [18], one starts with a *frame of discernment* or Universe of Discourse (UoD) [19], denoted as Θ, that defines the domain of reference. Then a *basic probability assignment* is a mapping $m : 2^\Theta \rightarrow [0,1]$ such that $\Sigma_{A\subset\Theta} m(A) = 1$. Then the degree of belief given to $A \subset \Theta$ is specified by the *belief function bel* $: 2^\Theta \rightarrow [0,1]$ such that

$$bel(A) = \Sigma_{B\subset A, B\neq\emptyset} m(B)$$

In other words, the degree of a belief assigned to A is the sum of all the basic probability assignments $m(B)$ allocated to statements B that imply A.

3. *Frequency, Statistical, and Cyc's Approaches.* We will not describe these approaches here because of the space limitation. See [15] for their description.

Once new evidence E becomes available, the degree of belief $d(b|\xi)$ can change to absorb this new evidence. For example, when grades for the courses taken during the last semester become available, we can revise the degree of our belief that men and women receive equal grades on average. To update the degree of a belief, we need a *belief revision strategy* that computes $d(b|E,\xi)$, given $d(b|\xi)$. For example, in Bayesian approach, degrees of beliefs can be revised using the Bayes rule (1).[1]

We defined in [15] *interestingness* of a pattern p relative to a belief system B and prior evidence ξ, $I(p,B,\xi)$, in terms of how much degrees of beliefs change as a result of this new pattern p, i.e., as

$$I(p,B,\xi) = \sum_{\alpha_i\in B} w_i|d(\alpha_i|p,\xi) - d(\alpha_i|\xi)| \quad (2)$$

where w_i are weights of beliefs in the belief system (normalized to 1). Furthermore, we showed in [15] that, when the degree of at least one belief in the belief system changes with the new data ΔD, then there are interesting patterns in that data; that is, there exists p such that $I(p,B,\xi) > 0$.

This observation motivates the belief-driven discovery based on the measure of interestingness just described: *once degrees of some beliefs change with new data, then we know that there are interesting patterns in the data*, and we should discover them. In this paper, we describe a method

[1] It turns out that the Bayesian revision can be hard to compute for the reasons discussed in [15]. However, as explained in [15], the Bayesian revision can be approximated using some heuristics that make the revision process computationally feasible.

of discovering these patterns that is based on the paradigms of active databases and on data mining queries.

2.2 Active Databases

Our work is related to active databases [6, 20] because we monitor significant changes to the data, including the changes to the degrees of user-defined beliefs and trigger discovery processes when these significant changes occur. This monitoring process is done through the use of *DMDT triggers*, to be described below, which are extensions of classical ECA triggers used in active databases. Therefore, the DMDT framework relies on the extensive body of research in active databases and uses some of the techniques developed in that field.

Active databases in the data mining context have been explored before in [3]. In particular, Agrawal and Psaila [3] proposed to partition large temporal data sets according to the time periods, and the mining algorithm can be applied to each individual partition generating rules for that partition. Each rule has a set of statistical parameters associated with it, such as support and confidence. The set of values for each statistical parameter over time forms the history of the parameter for that rule. The authors in [3] defined triggers over the rulebase in which the triggering condition is a query on the shape of the history. An action part of a trigger consists of functions, such as *notify* or *show*, and of user-defined events.

Our work differs from the work of [3] in that we propose to use triggers during the pattern discovery process itself (since DMDT triggers operate on the data and trigger the discovery processes), whereas triggers of [3] are deployed in the analysis stage of the discovery process (after patterns are mined already) in order to detect certain changes in the parameter histories of already discovered rules. More specifically, triggers of [3] operate on the rulebase table (and not on the original data) and are defined in terms of the histories of the parameters stored in the rulebase.

2.3 Data Mining Query Languages

Our work is also related to data mining query languages [10, 8, 14] because the DMDT triggers activate user-defined data mining queries that search for types of patterns specified in these queries. *Data Mining queries*, also known as *pattern templates*, are constructs for the specification of types of patterns to be discovered[2]. Several researchers considered data mining queries (pattern templates) in the context of knowledge discovery [10, 8, 14, 12].

[2]In this paper we will assume that these two notions, data mining queries and pattern templates, are the same and will use them interchangeably throughout the paper.

Imielinski et al. [10] introduce the M-SQL query language as an extension of SQL by including a small set of primitive data mining operations. M-SQL queries operate on the underlying database *and* on a set of previously discovered rules and return a set of rules discovered in the data that satisfy the conditions of the query. For example, the query "Find all rules in Table T involving attributes Disease, Age, and ClaimAmt, which have a confidence of at least 50%" can be expressed in M-SQL as [10]

```
SELECT  *
FROM    Mine(T) R
WHERE   R.Body < {(Disease=*), (Age=*), (ClaimAmt=*)}
        and {} < R.Body and R.Consequent IN
        {(Disease=*), (Age=*), (ClaimAmt=*)} and R.Confidence > 0.5
```

The `Mine` operator returns all the association rules [1] that are true in T, and the WHERE-clause imposes conditions on them, such as that the body of a rule should not be empty and should contain attributes Disease, Age, and ClaimAmt. Also, the consequent of the rule must contain one of these attributes and that the confidence should be at least 50%. M-SQL queries are evaluated by launching discovery methods described in [10].

Han et al. [8] designed the data mining language DMQL. A query in DMQL consists of the specification of four data mining primitives: (1) the set of data relevant to the data mining process; (2) the kind of knowledge to be discovered, i.e. a characteristic, discriminant, classification, or an association rule; (3) the background knowledge consisting of a set of concept hierarchies or generalization operators which provide corresponding higher level concepts; (4) the interestingness of the knowledge to be discovered, which is specified with a set of different mining thresholds. For example, the query "Classify students according to their GPA's and find their classification rules for those majoring in computer science and born in Canada, with the attributes *birth_place* and *address* in consideration" can be expressed as [8]

```
Find classification rules for CS_Students
According to GPA
Related to birth_place, address
From Student
Where major = 'CS' AND birth_place = 'Canada'
```

Shen et al. [14] define *metaqueries* that can be viewed as a two-part specification. The left-hand side (LHS) of a metaquery 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

$$P(X,Y) \wedge Q(Y,Z) \Rightarrow R(X,Z)$$

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. For example, consider the transitivity pattern $p(X,Y) \wedge q(Y,Z) \Rightarrow r(X,Z)$ [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.

The notion of a pattern template was also introduced in [12] for the purpose of identifying interesting patterns. A pattern template in [12] is a rule

$$A_1 \wedge \ldots \wedge A_k \Rightarrow A_{k+1}$$

where each A_i is either an attribute name, a class name or an expression $C+$ or $C*$ corresponding, respectively, to one or more and zero or more 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:

CORE(Dept,Student) $\wedge$ ELECTIVE(Dept,Student)+ $\Rightarrow$ ADV_ELECTIVE(Dept,Student)

where CORE, ELECTIVE, and ADV_ELECTIVE are classes of certain courses offered at that university. Given this template and assuming that γ and σ are confidence and support thresholds [12], the rule

Intro_CS(CS,Student) $\wedge$ Programming_Lang(CS,Student) $\wedge$ Operating_Systems(CS,Student)
$\Rightarrow$ Special_Topics_in_OSes(CS,Student) $[\gamma, \sigma]$

matches this template. Then, a pattern is interesting if it matches an *inclusive* template [12].

In this work, we propose to tie user-defined data mining queries to significant changes in the data, including changes to user-defined beliefs and trigger these queries when significant changes in the data and beliefs occur. It does not actually matter in our case which specific data mining queries (pattern templates) and the associated discovery methods are used because the DMDT framework can accommodate *any* of these approaches.

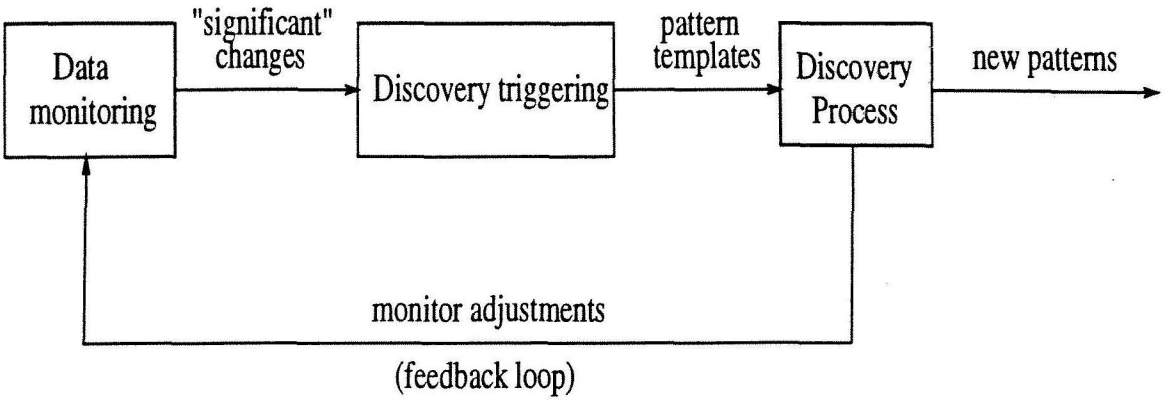

Figure 1: The Data Monitoring and Discovery Triggering Framework.

3 Data Monitoring and Discovery Triggering Framework

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 1 illustrates graphically the DMDT framework. 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 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 (data mining queries) 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.

The processes occurring in boxes "Data monitoring" and "Discovery Triggering" in Figure 1 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 ΔD 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

WHEN	new data ΔD becomes available
IF	"significant changes" in the data are observed when ΔD is added to the old data D
THEN	activate pattern templates (launch data mining queries)

We call these triggers "extended" because they are extensions of classical triggers used in active databases [6, 20] in the sense that the IF- and THEN-clauses cannot be always expressed in terms of the standard active database triggers [6, 20].

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 first-order logic that also supports arbitrary user-defined *functions*.[3] For example, the monitor may 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). Among all possible monitors, we distinguish the following two special types:

- *Simple monitors.* These can be expressed as logical expressions without user-defined functions (e.g., in standard SQL).
- *Belief monitors.* We can define a set of beliefs about the application at hand, such as "men and women studying at a university should receive equal grades," check how degrees of these beliefs change with new data, and monitor significant changes to these beliefs. We will describe this type of monitor in detail in Section 4.

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 all the banking transactions performed during a business day are recorded in the database at the end of the day in a batch mode
- When a new quarterly report of a company becomes available

The events of the WHEN-clause are standard "append" operations on a database and can easily be detected using standard event detection techniques of active databases [6, 20].

The THEN-clause of a DMDT trigger specifies a set of pattern templates (data mining queries). When the monitor of a trigger gets activated by the events of the WHEN-clause and if the monitor

[3]Typically, these functions are defined using one of the programming languages, such as C.

detects the conditions specified in the IF-clause of the trigger, then the data mining queries specified in the THEN-clause of the trigger get *activated* and are added to the set of queries used in the pattern discovery process. These queries (pattern templates) serve as an input to the "Discovery Process" module in Figure 1. Evaluation of these data mining queries constitutes the discovery process and results in the set of patterns (rules) satisfying the conditions of the WHERE-clauses of these queries.

Example 1 : Consider the relation GRADES that describes average grades that men and women received in various courses taken in different schools of the university over several semesters:

GRADES(Course_No,School,Semester,Year,Avg_Grade_Men,Avg_Grade_Women)

Also, consider the following specification pertaining to this relation:

> When grades for the last semester (FALL 1996) become available and if women received better grades than men in more than 75% of the courses during that semester, then find all the patterns (rules) pertaining to the years after 1993, involving attribute Schools and Semester, having condition Avg_Grade_Men < Avg_Grade_Women in the head of the rule, and having confidence [1] of more than 75%.

This specification can be formally expressed as the following DMDT trigger:

WHEN NEW_GRADES_ARRIVE(Semester, Year)
IF No_courses_women_did_better / Total_no_courses > 75%
THEN SELECT *
FROM Mine(GRADES) R
WHERE R.Body CONTAINS { Schools = *, Semester = *, Year > 1993 } AND
R.Consequent = { Avg_Grade_Men < Avg_Grade_Women } AND R.Confidence > 75%

where the WHEN-clause contains the event NEW_GRADES_ARRIVE(Semester, Year) specifying the arrival of new grades for the current semester, the IF-clause contains expressions defined as

Total_no_courses = COUNT{ Course_No |
GRADES(Course_No,School,Semester,Year,Avg_Grade_Men,Avg_Grade_Women) AND
Semester = "Fall" AND Year = 1996 }

No_courses_women_did_better = COUNT{ Course_no |
GRADES(Course_No,School,Semester,Year,Avg_Grade_Men,Avg_Grade_Women) AND
Semester = "Fall" AND Year = 1996 AND Avg_Grade_Men < Avg_Grade_Women }

and the THEN-clause contains the data mining query, expressed in an extended version of M-SQL [10][4]. Following the M-SQL conventions, the Mine(GRADES) operator returns the set of all the rules that hold on GRADES, and the WHEN-clause imposes restrictions on these rules (e.g. the consequent of a rule must be of the form Avg_Grade_Men < Avg_Grade_Women, the body of the rule must be non-empty and contain the clause Year > 1993, the confidence of the rule must be greater than 75%, and so on). Then the DataMine system [10] finds the rules satisfying the WHERE-clause of the M-SQL query without exhaustively processing all the rules returned by the Mine(GRADES) operator. □

We described the DMDT framework *intentionally in most general* terms, so that it could encompass various specific methods of data monitoring and discovery triggering. As a result of this, our approach is not limited to any specific DMDT scheme. In particular, it really does not matter which changes in the data are monitored in the IF-clause of a DMDT trigger. These changes can include such significantly different entities as changes in aggregate statistics (e.g., total volumes of sales) and changes in user-defined beliefs to be discussed in the next section. Similarly, it also does not matter which data mining query languages are used in the THEN-clause of a DMDT trigger. In fact, any of the query languages considered in Section 2.3, such as M-SQL, DMQL, Metaqueries, and pattern templates of [12], can be used in the THEN-clause of DMDT triggers.

Once new patterns are discovered, they are used for adjusting data monitors. For example, after executing the DMDT trigger from Example 1 and determining new patterns, it may turn out that the threshold value of 75% in the monitor of the trigger from Example 1 is too high. Based on this information, we may want to adjust it to a lower value. Another example of adjustment to the monitors will be described in the belief-driven discovery context in Section 5, when we may want to add new beliefs to the existing set of beliefs based on the discovered patterns.

We would also like to point out that data mining queries in the THEN-clause of a trigger are specified by the application developer. Therefore, the success of a data mining application depends, to a large extent, on how well the developer specified these data mining queries. If the application developer specifies a comprehensive set of well-chosen queries, then such an application will, certainly, discover more useful patterns.

The DMDT framework can be compared with the process of repeated issuing of data mining queries. The DMDT framework differs from it in the following two respects. First, data mining queries are issued *in response* to *significant* changes in the data and therefore these queries are

[4]We used an extension of M-SQL in this example because M-SQL deals with the association rules, and association rules are not suited well to express the DMDT trigger specified in this example (e.g., association rules, as considered in [10] do not support inequalities such as Avg_Grade_Men < Avg_Grade_Women). Therefore, we added this type of condition to M-SQL.

related to these changes and, hence, are more focused in this case. For instance, in Example 1, the data mining query in the THEN-clause of the trigger is formulated in response to the fact (detected by the monitor) that women received better grades than men in more than 75% of the courses during the last semester. Second, as was shown in [15] in case when we monitor beliefs, if degrees of some of the beliefs change, this means that there are interesting patterns in the data. Therefore, data mining queries in the THEN-clause of the DMDT trigger containing a belief monitor are issued *knowing* that there is some interesting information in the data. This is in contrast to the case of issuing "blind" data mining queries in hope to find something interesting in the data.

In this section, we described the Data Monitoring and Discovery Triggering framework. In the next section, we will describe a special case of this framework when monitors monitor changes to user-defined *beliefs*.

4 Belief-Driven Discovery

The belief-driven discovery scheme is a special case of the DMDT framework in which triggers monitor changes to *degrees of beliefs* when new data becomes available, and activate the discovery processes for those beliefs, degrees of which change significantly with new data. The importance of belief-driven discovery follows from the observation, made in [15], that the data contains interesting patterns when degrees of beliefs change with new data. Then, the DMDT triggers provide a mechanism for discovering these patterns.

DMDT triggers in the belief-driven discovery scheme have the form:

WHEN	new data ΔD becomes available
IF	$\lvert d(b\lvert D, \Delta D) - d(b\lvert D)\rvert / d(b\lvert D) > \epsilon$
THEN	activate pattern template(s) (launch data mining queries) for this rule

where $d(b|D)$ is the degree of belief in b given old data D, $d(b|D, \Delta D)$ is the degree of belief in b given D and new data ΔD, and ϵ is a threshold parameter specifying by how much the degree of belief should change in order to launch data mining queries specified in the THEN-clause.

The value of ϵ depends on the value of ΔD. For example, the degrees of beliefs about company's sales figures do not change as much in case they are recorded on a daily basis, as when they are recorded on a monthly, quarterly, or yearly basis. The parameter ϵ has to be selected *very carefully* because it determines sensitivity of rule triggering. In particular, if the value of ϵ is set too high, the rule may fire very seldom or even may not fire at all. If the value of ϵ is set too low, the rule may fire very often, and no interesting patterns be discovered as a result. We can let the application developer specify the values of ϵ. Alternatively, the process of selecting these values

can be automated, and we are working on techniques for selecting these ϵ's.

It is important to observe that the belief-driven discovery approach does not depend on the way in which we define the degree of a belief, and it can work with *any* degree of belief definitions presented in [15] (and reviewed in Section 2.1). Similarly, it is also general enough to work with *any* strategy for revising degrees of beliefs. This is the case because the IF-clause of the belief-driven DMDT trigger relies only on the values of $d(b|D)$ and $d(b|D, \Delta D)$.

As we pointed out already (at the end of Section 3), data mining queries in the THEN-clause are specified by the application developer. Therefore, it is up to him or her to select appropriate queries in the THEN-clause of a trigger. In the belief-driven discovery context however, there is a special group of queries that the application developer is strongly encouraged to include. These queries are based on the *negation* of the belief if its degree decreases with new data, and on the elaboration of the belief if the degree increases with new data. For example, if we believe that "men receive better grades than women," and the degree of this belief decreases based on the new data, then this special data mining query is "Find rules that show that during *the last* semester (Fall 1996) women received *better* grades than men." One way to express this query in the extended M-SQL would be as

```
SELECT   *
FROM     Mine(GRADES) R
WHERE    R.Body CONTAINS { School = *, Semester = "Fall", Year = 1996 } AND
         R.Consequent = { Avg_Grade_Men < Avg_Grade_Women }
```

Note that this query is based on the *negation* of the belief that men should receive better grades than women (because the new data decreases this belief). However, there are other queries that can be formed as negations of the belief, and we leave it up to the application developer to specify which special data mining queries should be formed in the THEN-clause of the trigger.

In addition to the special set of data mining queries directly related to the belief, the user, certainly, can form other queries unrelated to the belief. For example, if the degree of the belief that men receive better grades than women decreased with new data, then the following data mining query can also be added to the THEN-clause of the trigger[5]

> "Find rules that show that there are large improvements (e.g., more than 10%) in women's grades across various schools between this semester (Fall 1996) and the previous one"

[5]Similarly, we can add another query to the THEN-clause saying that the men's performance decreased during the last semester, or maybe consolidate the two queries into one. We do not specify it here because of the space limitations and because it is very similar to the first query.

```
SELECT   *
FROM     Mine(GRADES Q1, GRADES Q2) R
WHERE    R.Body CONTAINS { Q1.School = Q2.School = *,
         Q1.Course_No = Q2.Course_No, Q1.Semester = "Fall",
         Q2.Semester = "Spring", Q1.Year = Q2.Year = 1996 } AND
         R.Consequent = { (Q1.Avg_Grade_Women − Q2.Avg_Grade_Women) /
            Q2.Avg_Grade_Women > 10% }
```

Note that this query is only very marginally related to the belief that men should receive better grades than women. However, it is a good idea to issue this query in the connection to the decrease in the degree of the previous belief because it may discover the reasons why women outperformed men during the last semester.

It is crucial for the belief-driven discovery scheme to define a comprehensive set of beliefs because a poorly designed set of beliefs results in the discovery of only few interesting patterns. We will address the issues of defining a comprehensive set of beliefs in Section 5, where we present three complementary methods serving this purpose, i.e. belief elicitation from the domain expert, data-driven discovery of beliefs using standard machine learning methods, and pattern "recycling" techniques.

We described the belief-driven discovery process in terms of triggers. An alternative view of this process is presented in Figure 2 and works as follows. With each belief b in the system of user-defined beliefs B, we associate a set of data mining queries (pattern templates) that are activated once the degree of this belief changes with new data. When new data arrives, the degrees of beliefs in the belief system are revised according to the belief revision strategy adopted in that application. This process corresponds to the inner box in Stage 1 of Figure 2. Then, in Stage 2, the set of pattern templates (data mining queries) associated with the beliefs, whose degrees changed substantially, are *activated* in the sense that they will be used as inputs to the discovery module. Then, in Stage 3, data mining queries are launched, and new patterns are discovered using the discovery methods associated with data mining query languages [10, 8, 14]. Once the new patterns are discovered, they are not only presented to the user, but are also used to form *new beliefs*. This corresponds to the feedback loop from Stage 3 to the "New Belief Formation" module of Stage 1 in Figure 2. For example, assume that, as a result of the search described in the previous paragraph, we discovered two new patterns that women significantly outperformed men in sociology and in psychology. The newly discovered rules can be analyzed by the domain expert who can form new beliefs based on these rules. For example, by examining the previous two rules, the domain expert can form a new belief that women receive better grades than men in social sciences.

The process of new belief formation can be performed by the domain expert via "manual" inspection of discovered patterns or it can be automated. We will elaborate on the belief formation

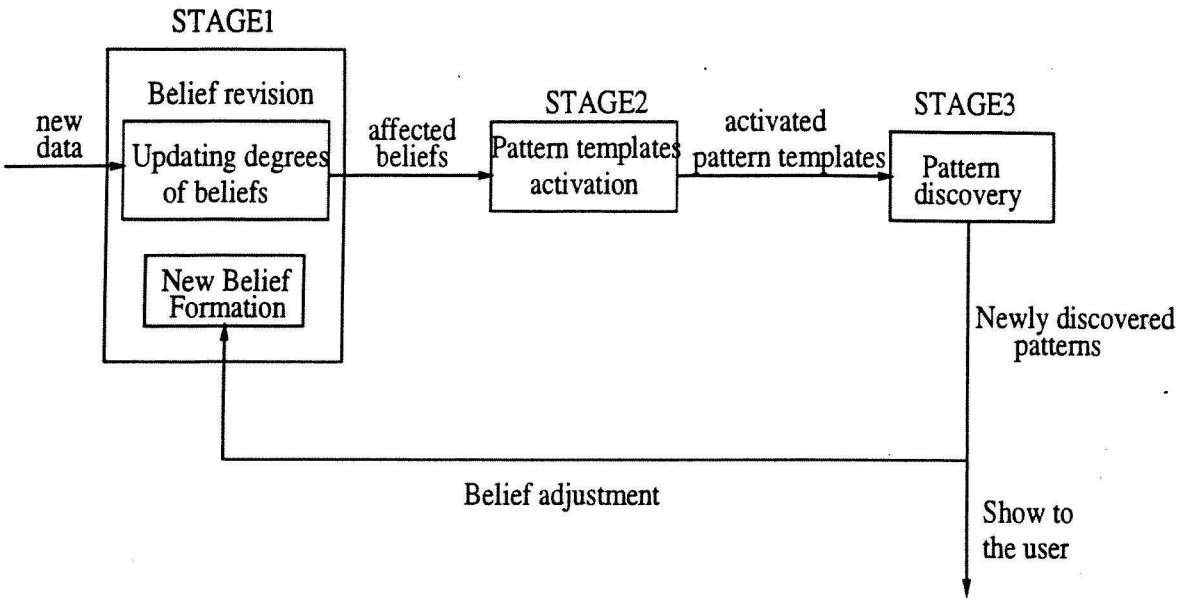

Figure 2: The Belief-Driven Discovery Framework.

process further in Section 5 when we discuss mechanisms for building belief systems.

In order to provide a flavor of how the belief-driven discovery works in practice, we present a small illustrative example.

Example 2 : Consider the GRADES relation defined in Example 1 and assume that we have only one belief about this application, i.e., that

B: men should receive grades that are similar to women's grades.

We can associate two triggers with this belief. The first trigger detects the situation when the degree of belief B is decreasing with the grades from the last semester:

WHEN NEW_GRADES_ARRIVE(Semester, Year)
IF $(d(B|\text{old\_grades}) - d(B|\text{old\_grades,new\_grades}))/d(B|\text{old\_grades}) > 5\%$
THEN launch queries Q11, Q12, Q13, Q14

where

Q11: find patterns confirming that women received better grades than men in the last semester

Q12: find patterns confirming that men received better grades than women in the last semester

Q13: find patterns confirming that men significantly improved their grades during the last semester

in comparison to the previous one (e.g. by more than by 10%)

Q14: find patterns confirming that women significantly improved their grades during the last semester in comparison to the previous one (e.g. by more than by 10%).

Queries Q11 and Q14 were expressed in an extension of M-SQL earlier in this section. Since the other two queries Q12 and Q13 are expressed in a way that is very similar to queries Q11 and Q14, we do not express them here because of the space limitation.

The second trigger detects the situation when the degree of belief is getting stronger with new data and can be expressed as

WHEN NEW_GRADES_ARRIVE(Semester, Year)
IF $(d(B|\text{old\_grades,new\_grades}) - d(B|\text{old\_grades}))/d(B|\text{old\_grades}) > 5\%$
THEN launch query Q21

where query Q21 is "find patterns confirming that women and men received very similar grades during the last semester (that differ by no more than 5%)." Formally,

```
SELECT   *
FROM     Mine(GRADES) R
WHERE    R.Body CONTAINS { School = *, Semester = "Fall", Year = 1996 } AND
         R.Consequent = { | Avg_Grade_Men − Avg_Grade_Women | / Avg_Grade_Women < 5% }
```

As was pointed out earlier in this section, the THEN-clause should contain one or several data mining queries that are directly related to the change in the degree of a belief. In our case, these queries are Q11 and Q12 for the first trigger and query Q21 for the second one.

Notice that not only queries Q11, Q12, and Q21 are natural to express in the association with the two triggers but so are also queries Q13 and Q14. However, the application developer specifying these two triggers does not have to stop at this point and might associate other types of data mining queries with these triggers. For example, he or she may want to check out trends about how much men or women increased their performance over a period of several semesters in addition to queries Q13 and Q14. These decisions are left up to the application developer and are based on the trade-off between the possibility of finding additional patterns and performance of the system due to running additional queries.[6] □

This example demonstrates that the ability to find interesting patterns in the data crucially depends on the ability of the application developer to formulate interesting data mining queries. For example, if the application developer did not include queries Q13 and Q14 in the first trigger, then he or she would run the risks of missing some interesting patterns.

[6]Needless to say that data mining queries can be computationally very expensive. See [10] for the discussion of how to improve performance of data mining queries.

In this section we described the belief-driven discovery process. For the remainder of the paper, we will concentrate on the question of how to build belief systems.

5 How to Build a Belief System

Belief systems in "industrial-strength" applications may require hundreds of beliefs, and it is important for the discovery process to construct "good" belief systems. In this section, we will address this issue and present some practically viable approaches to constructing comprehensive systems of beliefs.

5.1 Building a Belief System

To build a belief system, the application developer should come up with a set of beliefs and estimations of their degrees. First of all, we note that determining a "right" belief is more important than properly estimating its degree. In fact, it is not crucial to start with accurate degrees of beliefs because these degrees may be changed substantially after repeated updates. For example, assume that we initially strongly believe that men receive better grades than women at a university and that the data (grades over several semesters) demonstrates the opposite. Then after several revisions of our belief (one revision per semester), the degree of this belief will change in the "right" direction: initially, we will not be sure about our belief anymore and then, gradually, we will start believing in the opposite (that women receive better grades). In fact, Jaynes shows that this is the case for conditional probabilities [11].

Therefore, the main issue is to specify a comprehensive set of beliefs. This task can be achieved using any of the following methods that can also be combined together in order to produce better results:

- belief elicitation from the domain expert
- data-driven discovery of beliefs using machine-learning methods
- pattern "recycling"

We describe each of these approaches below.

Belief Elicitation. An immediate approach to building a belief system is to elicit beliefs from the domain expert through interviews. Although this can be a part of the belief system building strategy, there are two problems with this approach. First, the users may not really know what

they actually believe in; they can make vague and contradictory statements. Second, the users may have too many beliefs (e.g., thousands), and they don't know which of these beliefs are really important and worth including in the belief system. Therefore, the belief elicitation method can only be an initial step in the process of constructing a belief system. It has limited applicability, and should be complemented by other methods.

Data-driven discovery of beliefs. An alternative to the belief elicitation from the domain expert is *"learning"* these beliefs from the data, which can be achieved as follows. Since we are focusing on the discovery paradigm in which new data is periodically appended to the old data, we can use standard machine learning methods, such as C4.5 [13], CART [5], or Quest [2], applied to the *old* data in order to discover interesting patterns in that data. Then *some* of the discovered patterns can be converted into user-defined beliefs, and the "strengths" of the patterns (such as confidence and support [1]) can be used to define initial degrees of beliefs.

An important practical consideration is how to convert discovered patterns into beliefs that are really meaningful to the user. For example, assume that we applied C4.5 to the GRADES relation containing the old data and generated 1000 patterns from it. One way would be to let the user (e.g. college administrator) to go over these 1000 patterns and generate a set of beliefs that is really meaningful to him or her. Since the process of generating an *initial* set of beliefs happens very infrequently (e.g. once in a couple of years in the university example), this "manual" approach provides a viable solution in many applications. However, in the applications where machine learning methods generate a large volume of patterns on the old data, there is clearly a need for the methods that automate the conversion of patterns into user beliefs, and development of such automation procedures constitutes a topic of future research.

Pattern "Recycling." In the process of belief learning described in the previous paragraph, we proposed to use *external* discovery systems, such as C4.5 and Quest, to generate an *initial* set of beliefs. Since we could have missed some of the beliefs using the data-driven discovery methods described above and since some of the beliefs *change* over time,[7] it is important to supplement the methods described in the previous paragraph with the techniques that continuously generate new beliefs and revise the old ones.

One such approach would be to use the belief-driven discovery system, described in Section 4, for that purpose. In other words, the patterns discovered by the discovery module in Stage 3 in

[7] As an example, assume that it was believed at some point that the stock of company XYZ would never fall below 50. But today this may no longer believed to be true.

Figure 2 are shown to the user *and* some of them are "recycled" and used in forming new beliefs. This feedback process is diagrammatically shown in Figure 2 with the feedback loop from Stage 3 to the "New Belief Formation" module in Stage 1. In addition, these patterns are also used in adjusting parameters of some of the old beliefs.

As in the case of the data-driven discovery of beliefs, the processes of new belief formation and belief adjustments can be done either "manually" by the domain expert or "automatically" by a software system. Learning beliefs from the data manually requires extensive user involvement. However, the role of the user in this method is *passive* in comparison to a more *active* role in the belief elicitation method. All the user does is being shown some patterns and asked if these patterns are important and should be converted into beliefs, and, if yes, then how this conversion should be done.

5.2 Tuning the Belief-Driven Discovery System

Assume that we specified a set of DMDT triggers, turned the system on, new data started arriving and nothing happens (i.e. triggers are not firing). This is a problem, and we address it now. We identify three reasons why this can happen:

- There are no interesting patterns in the new data;
- The threshold parameters ϵ's in the IF-clauses of DMDT triggers were set too crudely and need to be adjusted;
- The belief system is too "coarse" and needs to be "refined" in the sense to be explained below.

The first item (no interesting patterns in the new data) is beyond our control, and there is nothing we can do about it. To address the second issue, we have to adjust parameters ϵ. This can be done either "manually" by the user or in a more "scientific" manner by some software. The issue of adjusting ϵ's automatically constitutes a topic of current research.

The third reason why triggers may not fire is because the beliefs can be too "coarse." For example, assume that we believe that men and women should receive equal grades on average in a university. Since there are many students involved, this belief may change very insignificantly with new data, and the corresponding trigger may not fire. One solution to this problem would be to replace this belief with several "refined" beliefs, one for each school in that university. One example of such refined belief would be "men and women in the *business school* should receive equal grades." Then, we can monitor several more refined beliefs instead of one coarse belief in hope that at least one trigger corresponding to one of the more refined smaller beliefs would fire. Unfortunately, it is

not always the case that, when a coarse belief changes, at least one smaller belief changes as well. The following example illustrates this point.

Example 3 : Assume that we believe that "men and women should receive equal grades on average (average taken over all the courses)" (call it believe Bel), and we want to replace it with a "finer" belief Bel' saying that "for every course, men and women should receive equal grades on average." Assume that we believe in Bel' with certainty (e.g. $d(Bel'|\xi) = 1$). To see if a change in the degree of belief Bel should cause changes in the degree of belief Bel', consider the following situation.

Assume that in the last semester there were only two courses offered, Course1 and Course2. In Course1, there were 50 men and 5 women, and all students received grade B. In Course2, there were 50 women and 5 men, and all students received grade A. Clearly, this data supports the claim that in both courses men and women received equal grades. But since we believe in Bel' with certainty, its degree should not change with new "positive" evidence [15]. In contrast to this, the degree of Bel should change because most women received grades A and most men received grades B.

Therefore, while the degree of a "coarser" belief (Bel) changed, the degree of "finer" belief (Bel') did not change. □

We are currently working on the problem of identifying under which condition a belief can be replaced with a set of more "refined" beliefs so that, when the degree of the main belief changes, the degree of at least one of the "refined" beliefs should also change.

6 Summary

We described a new framework of Data Monitoring and Discovery Triggering (DMDT) based on monitoring significant changes in the data and letting these changes trigger discovery processes. We also described a special case of the DMDT framework when the system monitors changes to the degrees of user-defined beliefs and triggers the discovery processes based on changes to these beliefs. This approach was motivated by the observation made in [15] that when degrees of beliefs change, this means that there are some interesting patterns in the data. Technically, the monitoring and discovery triggering processes can be implemented using DMDT triggers that combine features from the fields of active databases, data mining queries, and beliefs and belief revisions. We described the DMDT framework and the belief-driven discovery scheme *intentionally* in general terms so that it can encompass a variety of approaches. In particular, DMDT triggers can monitor different

types of changes in the data (including changes to user-defined beliefs), can utilize any system of beliefs and belief-revision strategies, and can launch data mining queries expressed in different query languages. This framework is especially well-suited for the applications where data changes rapidly over time, such as in the case of On-Line Transaction Processing Systems.

The practicality of the proposed framework depends on two issues that were addressed in the paper. The first issue is now to build a comprehensive set of beliefs. We considered three approaches to solving this problem: solicitation of beliefs from the domain expert, data-driven discovery of beliefs through machine learning methods, and recycling of patterns discovered using the DMDT framework. The second issue is how to set parameters so that the DMDT triggers would fire on a regular basis. In the belief-driven discovery scheme, this includes setting ϵ values in the IF-clauses and defining beliefs at proper granularity levels. We proposed that ϵ parameters be set by the users, but are also looking into the ways to automate this process. Splitting of beliefs into smaller beliefs to make triggers fire more regularly constitutes the topic of current research. Another interesting topic that we want to address is how to maintain the system of beliefs in an efficient manner.

References

[1] R. Agrawal, T. Imielinsky, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference*, pages 207–216, 1993.

[2] R. Agrawal, M. Mehta, J. Shafer, R. Srikant, A. Arning, and T. Bollinger. The Quest data mining system. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, August 1996.

[3] R. Agrawal and G. Psaila. Active data mining. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Montreal, Canada, August 1995.

[4] R. J. Brachman and T. Anand. The process of knowledge discovery in databases: A human-centered approach. In *Advances in Knowledge Discovery and Data Mining*, chapter 2. AAAI Press, 1996.

[5] L. Breiman, J. H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth Publishers, 1984.

[6] U. Dayal, E. Hanson, and J. Widom. Active database systems. In W. Kim, editor, *Modern Database Systems: The Object Model, Interoperability, and Beyond.* ACM Press, 1994.

[7] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In *Advances in Knowledge Discovery and Data Mining*, chapter 1. AAAI Press, 1996.

[8] J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. DMQL: A data mining query language for relational databases. In *Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, June 1996.

[9] E. J. Horvitz, J. S. Breese, and M. Henrion. Decision theory in expert systems and artificial intelligence. *Journal of Approximate Reasoning*, 2, July 1988.

[10] T. Imielinski, A. Virmani, and A. Abdulghani. DataMine: Application Programming Interface and Query Language for Database Mining. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, August 1996.

[11] E.T. Jaynes. *Probability Theory: The Logic of Science.* Cambridge University Press. To appear.

[12] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *Proceedings of the Third International Conference on Information and Knowledge Management*, December 1994.

[13] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[14] W.-M. Shen, K.-L. Ong, B. Mitbander, and C. Zaniolo. Metaqueries for data mining. In *Advances in Knowledge Discovery and Data Mining*, chapter 15. AAAI Press, 1996.

[15] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering.* Special Issue on Data Mining. To appear.

[16] A. Silberschatz and A. Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Montreal, Canada, August 1995.

[17] A. Silberschatz and A. Tuzhilin. User-assisted knowledge discovery: How much should the user be involved. In *Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, June 1996.

[18] P. Smets. Belief functions. In P. Smets, A. Mamdani, D. Dubois, and Prade H., editors, *Non-Standard Logics for Automated Reasoning.* Academic Press, 1988.

[19] J. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, 1988.

[20] J. Widom, S. Ceri, and U. Dayal. *Active Database Systems: Triggers and Rules for Advanced Database Processing.* Morgan Kaufmann, 1995.