To Appear in DSS Journal

Unexpectedness as a Measure of Interestingness in Knowledge Discovery

Balaji Padmanabhan
Operations and Information Management Department
The Wharton School, University of Pennsylvania
balaji@opim.wharton.upenn.edu

Alexander Tuzhilin
Information Systems Department
Stern School of Business, New York University
atuzhili@stern.nyu.edu

Unexpectedness as a Measure of Interestingness in Knowledge Discovery

Abstract

Organizations are taking advantage of "data-mining" techniques to leverage the vast amounts of data captured as they process routine transactions. Data-mining is the process of discovering hidden structure or patterns in data. However several of the pattern discovery methods in data-mining systems have the drawbacks that they discover too many obvious or irrelevant patterns and that they do not leverage to a full extent valuable prior domain knowledge that managers have. This research addresses these drawbacks by developing ways to generate *interesting* patterns by incorporating managers' prior knowledge in the process of searching for patterns in data. Specifically we focus on providing methods that generate *unexpected* patterns with respect to managerial intuition by eliciting managers' beliefs about the domain and using these beliefs to seed the search for unexpected patterns in data. Our approach should lead to the development of decision support systems that provide managers with more relevant patterns from data and aid in effective decision making.

Keywords: Interestingness of Patterns, Unexpectedness, Beliefs, Belief-driven Rule Discovery

"If you do not expect it, you will not find the unexpected, for it is hard to find and difficult."

- Heraclitus of Ephesus, 544 - 484 B.C.

1. Background and Research Motives

Technological and organizational trends are increasingly leading to knowledge intensive work environments. *The Work of Nations* [14] identifies a fundamental stream of work that involves

the production of *information goods*, rather than physical goods. The manager-analyst in these environments is more of a *knowledge worker* who routinely deals with information to produce value added information products.

Technological trends have resulted in organizations accumulating enormous amounts of data on several facets of their operations. For example, many organizations such as credit-card companies or retailing outlets record every single transaction a customer performs. It has been estimated [12] that businesses generate gigabytes of data every year and that the total quantity of data tracked doubles approximately every two years.

Organizations may proactively build a technological infrastructure to capture and store such data. Firms build data warehouses to support various kinds of decision making tasks such as planning marketing promotions using scanner data on sales. In other cases the legal environment in many economies may *require* that organizations collect and maintain operational data (such as telephone conversations with clients or credit card transactions) that could grow to enormous proportions. In either case, the data *exists* and organizations should leverage the data for competitive advantage by distilling potentially valuable "nuggets" of information. A role of intelligent systems in such an environment is to provide an infrastructure that identifies hidden patterns in the gathered data and thereby empower managers to make more effective decisions.

Data-mining is the process of discovering hidden structure or patterns in data. There have been many successful data-mining applications [7] in areas such as customer profiling, fraud detection, telecommunications network monitoring and market-basket analysis. These applications are driven by methods that *discover* patterns in data. Several of these methods such as association rule algorithms [1] are rule-discovery methods that discover patterns in the form of IF-THEN rules[1]. In a supermarket transactions dataset, for example, rules may indicate patterns such as "shoppers who buy diapers on Friday tend to buy beer too" (*IF diaper, friday THEN beer*) . The discovered rule in this case could be used to plan shelving arrangements - placing beer near diaper shelves may increase sales of beer. In a credit-card transactions dataset finding rules of the form *IF <condition> THEN <fraudulent_transaction>* is valuable since these rules

[1] In this paper we use the terms "patterns" and "rules" interchangeably

indicate conditions that result in a fraudulent transaction. There are several commercial data-mining tools [10] that incorporate methods for rule-discovery and these tools are used in as diverse areas such as fraud detection and medical research [10]. In this research we focus on improving the data-mining task of discovering rules in business datasets.

Data mining and data warehousing tools provide an infrastructure in knowledge-intensive work environments that could potentially *informate* skilled manager analysts. However several rule discovery approaches proposed in the literature have the following drawbacks:

1. These methods often generate a very large number of rules, most of which obvious or irrelevant, that result in a data-mining problem of the second-order - the interpretation and evaluation of the discovered rules could be a highly resource consuming exercise. Recently researchers from Stanford University [6] applied an association rule generating algorithm to a subset of census data containing about 30,000 records. Their algorithm generated over 20,000 rules from the census data. In their conclusions, they remark:

   "*Looking over the implication rules generated on census data was educational. First, it was educational because most of the rules themselves were not. The rules that came out at the top, were things that were obvious.*"- [6]

2. An important objective of data-mining is to discover *interesting* patterns in data. Most of the existing approaches in the literature on knowledge discovery and data mining use *objective* measures of interestingness, such as confidence and support [1], for the evaluation of the discovered patterns. These objective measures capture the *statistical strength* of a pattern. It has been argued in [8, 12, 16, 17] that besides objective measures of interestingness, *subjective* measures are equally important. These subjective measures, such as *unexpectedness* [8, 16, 17] and *actionability* [3, 12, 16, 17], assume that the interestingness of a pattern depends on the decision-maker and does not solely depend on the statistical strength of the pattern. Consider the following two patterns in a supermarket transactions data set:
   - The shopping volume on Saturday is greater than the volume on any other day of the week (*True* for 98% of the weeks in the data)

- When store coupons are available on Friday, these coupons are *not* used (*True* for 60% of the weeks in the data).

Using the strength (objective criteria) as the selection criterion for the interestingness of the rules will result in the first pattern being chosen as "more interesting" since the pattern is true for 98% of the weeks. However, this pattern may be obvious to any domain expert and hence represents little added value. In contrast, the second pattern is true for only 60% of the weeks, but is unexpected, since it challenges conventional wisdom that a majority of shoppers tend to use coupons at a store if they are available. A subjective criterion such as *unexpectedness* would, therefore, rate the second pattern as "more interesting" than the first.

3. Most of the existing algorithms such as CART [5], Apriori [2], C4.5 [13] are primarily data-driven and do not fully exploit domain knowledge and intuition that managers in a business environment have. Managerial intuition develops over several years of experience and could be an invaluable input to any knowledge discovery process.

These drawbacks are serious concerns given that the users of these systems need to understand and act on the data in ever shorter amounts of time. In this paper we propose new methods of discovery that address these drawbacks by discovering *unexpected* patterns that take into consideration prior background knowledge of managers. This prior knowledge constitutes a set of expectations or beliefs that managers have about the problem domain. We use these beliefs to seed the search for patterns in data that *contradict* the beliefs. Patterns contradictory to prior knowledge are by definition *unexpected.*

Methods that discover unexpected patterns with respect to prior knowledge are consistent with the general nature of scientific inquiry. The philosopher Karl Popper stresses the importance of *falsification* [4] for scientific inquiry. It is important to develop strong theories about a domain, but the rules of science demand that we also formulate the exact circumstances under which these theories can be falsified [4]. Moreover, the discovery of unexpected patterns addresses the drawbacks mentioned above:

- Since the search is *focused* on finding potentially interesting patterns, the problem of generating too many obvious or irrelevant patterns is avoided.

- The search uses subjective criteria for interestingness by leveraging prior domain knowledge (which constitute a manager's *subjective* input into the discovery process). Managers are potentially invaluable sources of intuition that may be specific to the task at hand. When such intuition forms the basis for an initial set of expectations, finding unexpected patterns in data attains significance since they potentially test the robustness of managerial intuition.

An Approach to Pattern Discovery

Our approach to pattern discovery begins with a set of beliefs that represent a decision maker's prior domain knowledge. These beliefs can either be elicited from the decision maker initially or "learned" from the data using machine learning methods and shown to the decision maker for his or her approval. Our approach has two complementary facets:

1. Discovery of unexpected patterns in data
2. Knowledge refinement based on the discovery of unexpected patterns

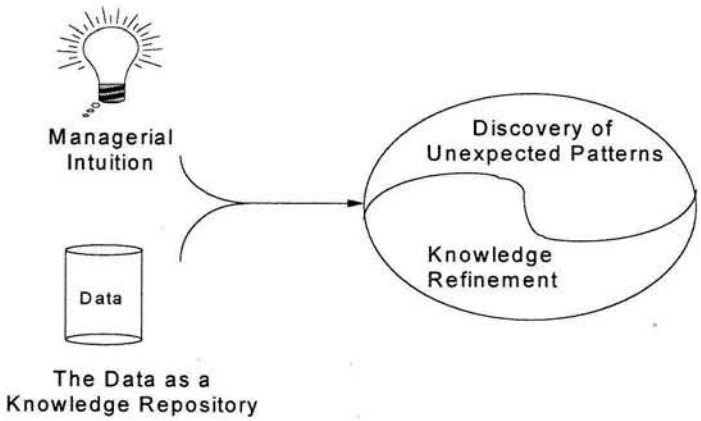

Figure 1. Complementary Nature of the Discovery of Unexpected Patterns and Knowledge Refinement

The discovery of patterns in data that contradict prior knowledge can be used in building theories about the domain. There could be several reasons why prior intuition and patterns from data may

conflict. For example, managerial intuition may have developed over years of prior experience and the data might reflect patterns on current environmental conditions distinctly different from previous conditions. Whatever the reasons, resolving such contradictions is important and could lead to a deeper understanding of the domain. Deming refers to knowledge developed through iterative theory building and refinement as "profound knowledge". The discovery of unexpected patterns with respect to specific intuitions could therefore lead to learning and the refinement of prior knowledge. In this sense, *the discovery of unexpected patterns and refinement of prior knowledge form two sides of the same coin* (as illustrated in Figure 1).

In this paper, we focus only on the discovery of unexpected patterns given an initial set of beliefs. We do not address the issue of how to build a "good" set of beliefs. We assume that it can be generated using methods described in [17], such as elicitation of beliefs from the domain expert, learning them from data, and refinement of existing beliefs using newly discovered patterns. A similar issue of how to specify an initial set of beliefs has also been addressed in [9]. The rest of the paper is organized as follows. In Section 2 we discuss unexpectedness and present a definition for the unexpectedness of a rule. After briefly presenting some preliminaries of association rules in Section 3, we describe an algorithm for the discovery of unexpected rules in Sections 4 and 5. In Section 6 we present results from applying our method and a standard association rule generating algorithm on consumer purchase data and present conclusions in Section 7.

2. Unexpectedness of a Rule

Unexpectedness of a rule has been considered before in [16, 17], [8] and [15]. However, [16, 17], [8] and [15] present different approaches to defining this concept.

The approach presented in [8] captures a measure of rule "distance" but not "unexpectedness" for the following reason. The approach is based on a *syntactic* comparison between a rule and a belief. In [8], a rule and a belief are "different" if either the consequents of the rule and the belief are "similar" but the antecedents are "far apart" or the consequents are "far apart' but the

antecedents are "similar", where "similarity" and "difference" are defined syntactically based exclusively on the structure of the rules. For example, consider a belief *man(X)* $\rightarrow$ *human(X)* and a rule is *woman(X)* $\rightarrow$ *human(X)*. According to [8], this rule has an "unanticipated condition" (*woman(X)*) and thus the rule is "different" from the belief. However, [8] stops short from declaring this rule to be "unexpected" relative to the belief. Indeed, this rule, though different from the belief, is *not* unexpected since one does not *contradict* the other in any way.

In [16, 17], a rule is considered to be unexpected if it, intuitively, "shakes" the system of beliefs, including changes to the degrees of these beliefs. Our approach differs from [16, 17], in that we consider *logical contradiction* of a rule with a belief, whereas [16, 17] define unexpectedness in *probabilistic* terms of how much the degree of belief is affected by a rule. We believe that our definition of unexpectedness is simpler and more operational.

An alternative approach is presented in [15] that discovering "exception rules" in the form of rule-pairs but does not begin with prior background knowledge. The approach in [15] discovers pairs of association rules A $\rightarrow$ B and their corresponding exceptions A, C $\rightarrow$ B', where A and C are conjunctions of <attribute, value> pairs and B and B' are <attribute, value> pairs corresponding to the same attribute but with different values. Further the unexpectedness of the exception rule in [15] is defined by an additional constraint that the "reference rule" C $\rightarrow$ B' has low confidence. [15] argues that if the reference rule has high confidence then the exception rule A, C $\rightarrow$ B' will not be unexpected.

The approach presented in [15] is based on an interesting probabilistic approach and has the advantage that it does not depend on prior domain knowledge. However, it has been argued [16, 17] that unexpectedness is inherently subjective and that prior beliefs of the user are, therefore, an important component of unexpectedness. Further, unexpectedness as defined in [15] can be restrictive since it does not capture some exceptions that are unexpected in the sense defined below.

In the rest of this section we present a new definition of unexpectedness of a rule. In order to define the concept of unexpectedness, we first present some preliminaries including definitions

of rules and beliefs. We consider rules of the form X → A, where X and A are conjunctions of literals (i.e., either atomic formulas of first-order logic or negations of atomic formulas). We also associate with the rule some measure of its statistical "strength" [11], such as "confidence" and "support" [1]. We say that a rule *holds* on a dataset *D* if the confidence of the rule is greater than 50% (the threshold confidence can also be chosen to be any value greater than 0.5).

We define a belief as a statement of the form Y → B, where Y and B are defined as for the rule. Associated with a belief is its *degree* [16, 17]. Degrees of beliefs are subjective in the sense that they are defined by the user and are revised according to some belief revision procedure [16, 17].

The approach in [8] considers beliefs that incorporate fuzzy linguistic modifiers (such as "low", "high", "small" etc.). An example of such a belief is "if temperature is *high* then heart_rate is *low*". An advantage of this approach is that it permits the user to specify beliefs without drawing hard artificial boundaries around *continuous* variables. In this paper, however, we do not consider beliefs that incorporate fuzzy modifiers since we focus our studies on pattern discovery in *discrete* data. We plan, however, to incorporate fuzziness into the representation of user beliefs in our subsequent work when we consider continuous variables as well.

We also make an assumption of *monotonicity of beliefs*. In particular, if we have a belief Y → B which we expect to hold on a dataset *D* with degree *d*, then the belief will also be expected to hold on any "statistically large"[2] subset of *D* with degree d_1 that is greater than 0.5. We believe that this is a reasonable assumption for the following reason. Assume that we have two non-monotonic beliefs "*bird(X)* → *flies(X)*" and "*bird(X), penguin(X)* → ¬*flies(X)*". Hence in the current form it appears that we expect the belief that birds fly to hold in general for the entire class of birds, but we do not expect it to hold for a subset that consists of penguins. However these can be transformed into the following pair of monotonic beliefs "*bird(X),* ¬ *penguin(X)* → *flies(X)*" and "*bird(X), penguin(X)* → ¬*flies(X)*". In general, if we have a non-monotonic belief (that we expect *not* to hold for some subset of the data), we incorporate our knowledge of why

[2] In this paper, we use a user-specified *support* threshold value to determine if the subset is large enough.

we do not expect the belief to hold on the subset into the belief, thereby making the belief more specific. We can do this iteratively until we have a set of monotonic beliefs[3].

Given these preliminary concepts, we are ready to define unexpectedness of a rule.

Definition. The rule $A \rightarrow B$ is *unexpected* with respect to the belief $X \rightarrow Y$ on the dataset D if the following conditions hold:

(a) B *AND* Y |= FALSE. This condition imposes the constraint that B and Y logically contradict each other.

(b) A *AND* X holds on a statistically large[3] subset of tuples in D. We use the term "*intersection of a rule with respect to a belief*" to refer to this subset. This intersection defines the subset of tuples in D in which the belief and the rule are both "applicable" in the sense that the antecedents of the belief and the rule are both true on all the tuples in this subset.

(c) The rule $A, X \rightarrow B$ holds. Since condition (a) constrains B and Y to logically contradict each other, it logically follows that the rule $A, X \rightarrow \neg Y$ holds. □

We believe that this definition captures the spirit of "unexpectedness" for the following reasons:
(1) The heads of the rule and the belief are such that they logically contradict each other. Therefore in *any* tuple where the belief and the rule are both "applicable," if the rule holds on this tuple, the belief cannot hold and vice-versa.
(2) Since both a rule and a belief hold *statistically*, it is inappropriate to label a rule "unexpected" if the intersection of the contradicting rule and the belief is very small. Hence we impose the condition that the intersection of the belief and the rule should be statistically large. Within this statistically large intersection, we would expect our belief to hold because of the *monotonicity* assumption. However if the rule holds in this intersection, the belief cannot hold because the heads of the rule and belief logically contradict each other. Hence the expectation that the belief should hold on this statistically large subset is contradicted.

[3] This process of converting non-monotonic beliefs to monotonic beliefs can be automated by letting the user specify non-monotonic beliefs with *exceptions*. Then the system automatically converts these to a set of monotonic beliefs.

Our method of representation of beliefs can also be used to represent beliefs in which the expected confidence is less than 50% by converting the beliefs into those that have expected confidence greater than 50%. For example, consider the belief that Y is true 1% of the cases in which X is true. This is equivalent to the belief that NOT(Y) should be true 99% of the cases in which X is true.

The approach presented in this paper differs from that in [15] in the following aspects:

- The approach presented in [15] does not depend on prior beliefs but discovers pairs of rules (that can be considered as beliefs) and their exceptions simultaneously. The approach presented in this paper begins with a system of beliefs.
- The approaches consider different types of unexpectedness. The approach presented in this paper is based on the monotonicity of beliefs, while exceptions in [15] are based on the structure of the rule-pair discovered and additional probabilistic constraints.
- The approach in [15] discovers only certain refinements to rules as exceptions, while the approach presented in this paper discovers all refinements that are unexpected and also unexpected generalizations as well.

We presented a general definition of unexpectedness in this section. We next present an algorithm for finding unexpected rules. Since association rules is a very popular method for defining patterns with many efficient discovery algorithms developed for them, we focus in the rest of the paper on the discovery of unexpected association rules.

One way to generate unexpected association rules would be to follow the approach proposed in [8]: run standard association rule discovery algorithms [2] and then select unexpected rules using the definition of unexpectedness introduced in this section. The main problem with this approach is that of efficiency. It may turn out that there are few unexpected patterns and it would, therefore, not be efficient to generate a large number of patterns before selecting the unexpected ones [4].

---

[4] As [6] show, in a census data set of 30,000 records, the number of rules generated was more than 20,000.

3. Association Rule Preliminaries

In this section we provide an overview of association rules and sketch the algorithms for discovering association rules proposed in [2]. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of discrete attributes (also called "items" [1]). Let an *atomic condition* be defined as a proposition of the form "attribute = value", where the attribute can take on a discrete set of mutually exclusive values. An *itemset* is a conjunction of atomic conditions. Let $D = \{T_1, T_2, ..., T_N\}$ be a relation consisting on N transactions [2] $T_1,...,T_N$ over the relation schema $\{i_1, i_2, \ldots, i_m\}$. A transaction T_i is said to "contain" an itemset if the itemset holds on T_i.

An association rule is an implication of the form *body* $\rightarrow$ *head* where "body" is an itemset and "head" is an itemset that contains only a single atomic condition. The rule holds in *D* with confidence *c* if *c*% of the transactions that contain *body* also contain *head.* The rule has support *s* in *D* if *s*% of the transactions in *D* contain both *body* and *head.* The search for association rules is usually constrained to rules that satisfy minimum specified support and confidence requirements. An itemset is said to be *large* if the percentage of transactions that contain it exceed the minimum specified support level.

Various efficient algorithms for finding all association rules in transactions databases have been proposed in [2]. These algorithms operate in two phases. In the first phase, all large itemsets are generated. This phase utilizes the observation that all subsets of a large itemset are large. Candidate itemsets of length k are generated from the set of large itemsets of length (k-1) by imposing the constraint that all subsets of length (k-1) of any candidate itemset must be present in the set of large itemsets of length (k-1). The second phase of the algorithm generates rules from the set of all large itemsets. For example, let I_1 = {age = high, income = high} and I_2 = {age = high}. From the supports of these two itemsets the confidence, c, of the rule " *if (age = high) then (income = high)* " can be calculated as *c = support({age = high, income = high}) / support({age = high}).* Hence in this phase, given the set of all large itemsets, significant rules involving these itemsets are generated.

4. Discovery of Unexpected Association Rules

In this section we present an extension to the algorithm of [2] that takes a set of beliefs, B, and discovers *unexpected* association rules. In this paper we restrict our attention to beliefs that have the same syntax as association rules and where the head of the belief involves a binary attribute[5].

4.1 Overview of the Discovery Strategy

Consider a belief $X \rightarrow Y$ and a rule $A \rightarrow B$, where both the itemsets X and A are conjunctions of atomic conditions and both Y and B are single atomic conditions involving binary attributes. It follows from the definition of unexpectedness in Section 2 that if an association rule $A \rightarrow B$ is "unexpected" with respect to the belief $X \rightarrow Y$, then the following must hold:

(1) $B = \neg Y$.

(2) The rule $X, A \rightarrow B$ holds.

Hence, for every unexpected rule of the form $A \rightarrow B$, it has to be the case that the rule $X, A \rightarrow B$ also holds.

We propose the discovery algorithm *ZoomUAR* ("Unexpected Association Rules") that consists of two parts: *ZoominUAR* and *ZoomoutUAR*. Given a belief $X \rightarrow Y$, the strategy that the algorithm *ZoomUAR* adopts is to first discover (in Algorithm *ZoominUAR*) all significant rules of the form $X, A \rightarrow \neg Y$ and *then* consider (in Algorithm *ZoomoutUAR*) other more general and potentially unexpected rules of the form $X', A \rightarrow \neg Y$, where $X' \subset X$. The rules that *ZoominUAR* discovers are "refinements" to the beliefs such that the beliefs are contradicted. The rules that *ZoomoutUAR* discovers are *not* refinements, but more general rules that satisfy the conditions of unexpectedness. For example, if a belief is that "professional $\rightarrow$ weekend" (professionals tend to shop more on weekends than on weekdays), *ZoominUAR* may discover a refinement such as "professional, december $\rightarrow$ weekday" (in December, professionals tend to shop more on

weekdays than on weekends). *ZoomoutUAR* may then discover a more general rule "december → weekday", which is totally different from the initial belief "professional → weekend".We propose the discovery algorithm *ZoomUAR* ("Unexpected Association Rules") that consists of two parts: *ZoominUAR* and *ZoomoutUAR*.

4.2 Algorithm ZoominUAR

The inputs to this algorithm are a set of beliefs, *B*, and the dataset *D*. For each belief X → Y, ZoominUAR finds all unexpected association rules of the form X, A → ¬Y. It is important to note that our approach differs from other association rule algorithms mainly in how we generate candidate itemsets that need to be checked for support and not in the actual process of checking the transactions in the dataset to determine the supports for these itemsets. In [2] different algorithms that calculate supports for itemsets efficiently are presented. Our method can be easily integrated into any such efficient association rule algorithm. For simplicity, in this paper, we use the "shell" of the Apriori algorithm proposed in [2]. ZoominUAR is presented in Fig. 4.1.

For each belief, B, ZoominUAR first generates incrementally all large itemsets that may potentially generate unexpected rules. For example if a belief is X → Y then the search is initially for large itemsets that contain X and ¬Y since the set of unexpected rules generated by this algorithm is of the form X, A → ¬Y. The confidence of this rule is given by support(X, A, ¬Y)/ support(X, A). Hence each time the algorithm generates a candidate itemset I_1 containing the negation of the head of the belief, the algorithm should also generate a corresponding candidate itemset I_1' that contains all conditions in I_1 *except* the negation of the head of the belief. In Fig. 4.1 the notation C_k refers to a set of candidate itemsets that contain the negation of the head of the belief and C_k' refers to the corresponding set of candidate itemsets that do not contain the negation of the head of the belief. For example, for the belief X → Y, for every itemset of the form { X, A, ¬Y} in C_k there will be a corresponding itemset { X, A} in C_k'.

[5] This does not restrict the beliefs we consider. For example, if the head of the belief is day = Sunday, we handle this case by introducing a binary attribute that holds iff day=Sunday.

The first candidate itemsets generated (Step 2 in Fig. 4.1) in this case is just {X, ¬Y} and {X}. Once candidate itemsets are generated, steps 4 and 5 determine the support counts in dataset *D* for all the candidate itemsets currently being considered and selects the large itemsets in this set. Hence in the initial pass, if both {X, ¬Y} and {X} are found to be "large", then L_0 = { {X, ¬Y},{X} }.

```
Inputs: Beliefs Bel_Set, Dataset D, Thresholds min_support and min_conf
Outputs: For each belief, B, itemsets Items_In_UnexpRule_B

 1 forall beliefs B ∈ Bel_Set {
 2    C_0 = {{¬head(B),body(B)}}; C_0' = {{body(B)}}; k=0;
 3    while (C_k != ∅ ) do {
 4          forall candidates c ∈ C_k ∪ C_k', compute support(c)
 5          L_k = {x | x ∈ C_k ∪ C_k', support(x) ≥ min_support }
 6          k++
 7          C_k = generate_new_candidates(L_k-1, B);
 8          C_k' = generate_bodies(C_k , B);
 9    }
10    Let X = {x | x ∈ ∪L_i, x ⊇ ¬head(B) }
11    Items_In_UnexpRule_B = ∅
12    forall (x ∈ X) {
13          rule_conf = support(x)/support(x - ¬head(B))
14          if (rule_conf > min_conf) {
15                Items_In_UnexpRule_B = Items_In_UnexpRule_B ∪ {x}
16                Output Rule " x - ¬head(B) → ¬head(B) "
17          }
18    }
19 }
```

Figure 4.1 *Algorithm ZoominUAR*

In step 7, the function *generate_new_candidates(L_{k-1}, B)* generates the set C_k of new candidate itemsets to be considered in the next pass from the previously determined set of large itemsets, L_{k-1}, with respect to the belief B ("x → y") in the following manner:

(1) Initial condition (when k=1): To explain this step, consider the following example. Assume that for the belief x → y, L_0 = { {x, ¬y},{x} }, i.e. both the initial candidates were found to be large. Further assume that "p" and "q" are the only other attributes in the domain that are not

already present in any of the conditions in x or y and that "p" and "q" are both binary attributes. The next set of candidates to be considered would be C_1 = { {x, ¬y, p}, {x, ¬y, ¬p}, {x, ¬y, q}, {x, ¬y, ¬q}}, and C_1' = { {x, p}, {x, ¬p}, {x, q}, {x, ¬q}}.

In general for the belief B, the initial candidate itemset C_0 contains the single element {body(B),¬head(B)}. If this itemset is large, then the next set of candidate itemsets would be the sets C_1 = {{ body(B), ¬head(B), X } | where X is any atomic condition involving an attribute not present in either body(B) or ¬head(B)} and C_1' = {{ body(B), X } | where X is any atomic condition involving an attribute not present in either body(B) or ¬head(B)}. The algorithm precomputes all the unique values for each attribute in the dataset and uses these values to generate all possible attribute-value combinations that can be considered for X.

(2) Incremental generation of C_k from L_{k-1} when k > 1: This function is very similar to the *apriori-gen* function described in [2]. For example, assume that for the belief x → y, L_1 = { {x, ¬y, p}, {x, ¬y, q}, {x, p}, {x, q} } . Similar to the Apriori algorithm, the next set of candidate itemsets that contain x and ¬y is C_2 ={ {x, ¬y, p, q} } since this is the only itemset such that all its subsets of one less cardinality that contain x and ¬y are in L_1. We would also need the support of the itemset {x, p, q} to (eventually) determine the confidence of the rule *x, p, q* → ¬*y*. Hence the function *generate_bodies(C_2,B)* generates C_2' = {{x, p, q}}.

In general, an itemset X is in C_k if and only if for the belief B, X contains ¬head(B) and body(B) and all subsets of X with one less cardinality, containing ¬head(B) and body(B), are in L_{k-1}. The condition that X contains the negation of the head of the belief B is just an artifact of the fact that for every itemset of the form *A,body(B),¬head(B)* in L_{k-1} there will be a corresponding itemset *A,body(B)* that will also be in L_{k-1}. Once C_k is generated as described above, the function *generate_bodies(C_k,B)* generates the set C_k' by considering each itemset in C_k and dropping ¬head(B) from the itemset.

Once all such large itemsets have been generated, steps 10 through 16 of the algorithm generate unexpected rules of the form $(x,A \rightarrow \neg y)$ where $\{x,\neg y,A\}$ and $\{x,A\}$ are large itemsets generated.

4.3 An Example of ZoominUAR

Consider the following example. Assume that we have a dataset of purchases at a supermarket (Fig. 4.2a) containing the following 4 binary attributes: (1) day of shopping (weekend /weekday) (2) whether the shopper is employed (3) whether diapers were purchased and (4) whether beer was purchased. Further assume that we have a belief that shoppers who buy diapers tend to buy beer (*diaper → beer*). Fig. 4.2b illustrates the iterations of ZoominUAR and the unexpected rules generated given the constraints that any itemset in a rule should have a support of at least 3 transactions and that the minimum confidence of a rule should be 60%.

DATA
(9 transactions)

Bought Diapers (D = 1)
Bought Beer (B = 1)
Weekend (W = 1)
Employed (E = 1)

D	B	W	E
1	1	1	1
1	1	1	0
1	1	1	0
1	1	0	1
1	0	0	0
1	0	0	0
1	0	0	0
1	1	1	1
0	1	0	0

Itera-tion #	Itemsets To Check	Sup	Large Itemset? (sup. > 3)	Rules Generated (confidence > 0.6)
1	{diaper, not_beer}	3	✓	None - confidence (3/8) of
	{diaper}	8	✓	*diap.* → *not_beer* is < 0.6
2	{diaper, not_beer, weekend}	0	✖	None
	{diaper, weekend}	4	✓	
	{diaper, not_beer, not_weekend}	3	✓	***diap., weekday → not_beer***
	{diaper, not_weekend}	4	✓	**(conf. = 3/4 = 0.75)**
	{diaper, not_beer, employed}	0	✖	None
	{diaper, employed}	3	✓	
	{diaper, not_beer, not_employed}	3	✓	***diaper , not_employed. →***
	{diaper, not_employed}	5	✓	***not_beer* (conf. = 3/5)**
3	{diaper, not_beer, not_weekend, not_employed}	3	✓	***diaper , not_employed, weekday → not_beer***
	{diaper, not_weekend, not_employed}	3	✓	**(conf. = 3/3)**

Figure 4.2a (left). *Example data containing 9 transactions, each consisting of four fields.*
Figure 4.2b (right). *Iterations of ZoominUAR (right table) corresponding to the belief "diapers → beer"*

Our approach differs from Apriori in that in the first iteration we start with itemsets that are derived from the belief (since ZoominUAR focuses on discovering rules of the form *diaper, X* $\rightarrow$ *beer*). We observe from the data that the belief is true in general (confidence of the rule *diaper* $\rightarrow$ *beer* is 5/8). However, in the second iteration we discover two interesting rules that during weekdays or when the shopper is not employed, then the purchase of diapers implies that beer is *not* purchased. Further refinement (third iteration) results in a stronger rule (conf. 100%) that when a shopper who is not employed shops on weekdays and buys diapers then the shopper does not buy beer. Observe that:

(1) To compute the confidence of the rule *diaper, X* $\rightarrow$ *not_beer* we need the supports of *both* the itemsets {diaper, X, not_beer} and {diaper, X}.

(2) In each iteration we consider itemsets containing one more condition and the corresponding rules generated are therefore more specific ("*zooming in*").

(3) In the third iteration for example, we do not even consider the itemset {diaper, not_beer, weekend, unemployed} since a subset of this itemset, {diaper, not_beer, weekend}, did *not* have minimum support (the first itemset considered in iteration #2). If any subset of an itemset does *not* have enough support, then the itemset in consideration also cannot have minimum support [2].

4.4 Algorithm ZoomoutUAR

ZoomoutUAR considers each unexpected rule generated by ZoominUAR and tries to determine all the other more general rules that may be unexpected. Given a belief X $\rightarrow$ Y and an unexpected rule X, A $\rightarrow \neg$Y, ZoomoutUAR tries to find more general association rules of the form X', A $\rightarrow \neg$Y , where X' $\subset$ X, and check if they satisfy minimum confidence requirements. Such rules satisfy the following properties:

- They are unexpected since the intersection[6] of this rule with the belief results in the rule X, $A \rightarrow \neg Y$, which is already known to hold.
- These rules are more general in the sense that they have at least as much support as the rule X, $A \rightarrow \neg Y$.
- The itemsets { X', A} and { X', A, $\neg Y$ } are guaranteed to satisfy the minimum support requirement (though we still have to determine their exact support in *D*) since the itemsets { X, A} and { X, A, $\neg Y$ } are already known to satisfy the minimum support requirement.

Inputs: Beliefs *Bel_Set*, Dataset *D*, Thresholds *min_support'* and *min_conf*, For each belief, B, itemsets $Items\_In\_UnexpRule_B$

```
1  forall beliefs B {
2    new_candidates = ∅
3    forall (x ∈ Items_In_UnexpRule_B ) {
4         Let K = {k| k ⊂ x, k ⊇ x-body(B)}
5         Let K' = {k| k ⊂ x - ¬head(B), k ⊇ x-body(B)}
6         new_candidates = new_candidates ∪ K ∪ K'
7    }
8    find_support(new_candidates)
9    Let X = {x | x ∈ new_candidates, x ⊇ ¬head(B) }
10   forall (x ∈ X) {
11        rule_conf = support(x)/support(x - ¬head(B))
12        if (rule_conf > min_conf) {
13             Items_In_UnexpRule_B = Items_In_UnexpRule_B ∪ {x}
14             Output Rule " x - ¬head(B) → ¬head(B) "
15        }
16   }
17 }
```

Figure 4.3. *Algorithm ZoomoutUAR*

The algorithm ZoomoutUAR is presented in Fig. 4.3. For each belief, B, from the previous algorithm ZoominUAR we have the set of all large itemsets that contain both ¬head(B) and body(B). The general idea is to take each such large itemset, *I*, and find the supports for all the subsets of *I* that are obtained by dropping from *I* one or more attributes that are in body(B). From

[6] As described in part (b) of the definition of unexpectedness in Section 2

the supports of all such *new* itemsets[7] considered here, ZoomoutUAR derives the rest of the more general unexpected rules. Step 4 of the algorithm creates candidate itemsets from the set of all large itemsets that contains the head of the unexpected rule being considered. As explained previously, for every itemset created in step 4, step 5 creates a corresponding new itemset that does not contain the head of the unexpected rule considered. Hence in steps 3 through 6 the algorithm generates all the new candidate itemsets for which supports have to be calculated. In one pass over D, step 8 determines the supports for all these candidate itemsets. Once all the new large itemsets have been determined, steps 9 through 14 of the algorithm generates unexpected rules in a similar manner as the latter part (Steps 10 through 16) of the previous algorithm ZoominUAR.

4.5 Completeness of ZoomUAR

In this section we present the theorem that ZoomUAR discovers all unexpected rules and provide a sketch of the proof.

Theorem. For any belief $A \rightarrow B$, ZoomUAR discovers *all* unexpected rules of the form $X \rightarrow Y$, where X and A are conjunctions of atomic conditions and Y and B are single atomic conditions involving binary attributes.

Sketch of the Proof. Consider the belief $A \rightarrow B$ and *any* unexpected rule $X \rightarrow Y$ (with support and confidence values greater than the specified threshold values) where both the itemsets X and A are conjunctions of atomic conditions and both Y and B are single atomic conditions involving binary attributes. From the definition of unexpectedness (Section 2) it follows that:

(1) $Y = \neg B$.

(2) The rule $X, A \rightarrow \neg B$ holds.

Therefore, the rule $X, A \rightarrow \neg B$ has support and confidence values greater than the specified threshold values. More specifically, the itemset $\{ X, A, \neg B \}$ has adequate support. To prove the theorem, we will first show that **ZoominUAR**:

(a) Generates the itemset $\{ X, A, \neg B \}$, and

(b) Derives the rule $X, A \rightarrow \neg B$ from the itemset $\{ X, A, \neg B \}$.

---

[7] These itemsets were not considered in ZoominUAR since all candidate itemsets considered there contain body(B).

If A is a subset of X, this completes the proof[8], since the rule $X, A \rightarrow \neg B$ is equivalent to the rule $X \rightarrow Y$ given that $Y = \neg B$. If A is *not* a subset of X, we will show that **ZoomoutUAR**:

(c) Generates the itemset $\{ X, \neg B \}$ and

(d) Derives the rule $X \rightarrow \neg B$ from the itemset $\{ X, \neg B \}$. Since $Y = \neg B$, this rule is the unexpected rule $X \rightarrow Y$.

Since X is a conjunction of atomic conditions, assume that $X = \{X_1, X_2, ..., X_N\}$. Since the itemset $\{X, A, \neg B\}$ is guaranteed to have adequate suppport, all subsets of $\{X, A, \neg B\}$ will also have adequate support. First, since we start with the belief $A \rightarrow B$, step 2 of **ZoominUAR** (Fig. 4.1) generates the itemset $\{A, \neg B\}$. In the first iteration of ZoominUAR (k=0), the itemset $\{A, \neg B\}$ will be determined to have adequate support. Further, in this iteration, step 7 of ZoominUAR generates all candidate itemsets $\{P, A, \neg B\}$ where P is a single atomic condition. Hence, this candidate set also contains $\{X_i, A, \neg B\}$ for i = 1 to N. In the next iteration (k=1), all the itemsets $\{X_i, A, \neg B\}$ will be determined to have adequate support in step 4. The next set of candidate itemsets generated (by *generate_new_candidates*) will contain itemsets of the form $\{P, Q, A, \neg B\}$ such that all subsets of this itemset that contain $\{A, \neg B\}$ have been determined to be large in the previous iteration. All itemsets of the form $\{X_i, X_j, A, \neg B\}$ will be generated as candidates since the subsets $\{X_i, A, \neg B\}$ and $\{X_j, A, \neg B\}$ are known to be large from the previous iteration. Extending the same argument in subsequent iterations, it can be shown that ZoominUAR generates the itemset $\{X_1, X_2, ..., X_N, A, \neg B\}$ in the iteration in the Nth iteration. Since this itemset has adequate support, $\{X, A, \neg B\}$ will be an element in the set *Items_In_UnexpRule*.

Since ZoominUAR determines the set $\{X, A, \neg B\}$ to be large, step 13 of ZoominUAR considers the rule $X, A \rightarrow \neg B$ and determines that the rule holds. If A is a subset of X, this completes the proof, since the rule $X, A \rightarrow \neg B$ is equivalent to the rule $X \rightarrow Y$ given that $Y = \neg B$.

[8] If A is a subset of X, then the condition X, A is the same as the condition X. For example, if A is "income = high" and X is "income = high, age = low", then X *AND* A is equivalent to X.

If A is *not* a subset of X, since $\{X, A, \neg B\}$ is an element in the set *Items_In_UnexpRule* Step 4 of **ZoomoutUAR** (Fig. 4.2) generates the itemset $\{X, \neg B\}$ by dropping one or more attributes from the body (A) of the belief $A \rightarrow B$. Step 5 similarly generates the itemset $\{X\}$. The itemsets $\{X\}$ and $\{X, \neg B\}$ are, therefore, elements in the set *new_candidates* generated in step 6 of ZoomoutUAR. Since these are guaranteed to have adequate support, step 14 of ZoomoutUAR generates the rule $X \rightarrow \neg B$. From the observation that $Y = \neg B$ it follows that ZoomoutUAR generates the rule $X \rightarrow Y$.

The theorem presented in this section states that ZoomUAR discovers all unexpected rules. Moreover, it is clear that ZoomUAR discovers *only* unexpected rules and no other rules. Therefore, ZoomUAR discovers a rule if and only if it is unexpected.

5. Handling Multiple Beliefs Efficiently in ZoominUAR

Algorithm ZoominUAR shown in Figure 4.1 discovers unexpected rules for each belief independently. In this section we present extensions to ZoomUAR to exploit efficiency issues when dealing with multiple beliefs in parallel.

Iteration	**Itemsets Considered for the belief A → B**	**Itemsets Considered for the belief C → B**
1	{A, ¬B}	{C, ¬B}
2	{A, ¬B, C} , {A, ¬B, ¬C}, {A, ¬B, D}, {A, ¬B, ¬D}	{C, ¬B, A}, {C, ¬B, ¬A}, {C, ¬B, D}, {C, ¬B, ¬D}
3	{A, ¬B, C, ¬D}	{C, ¬B, A, D}

Figure 5.1 *Example itemsets considered by ZoominUAR for the two beliefs* A → B and C → B

Consider the following example. Assume that the only conditions in a domain are A, B, C, D and their logical negations. Consider two beliefs, A → B and C → B. When the algorithm attempts discovers unexpected rules for each belief independently, the Figure 5.1 lists the itemsets considered by ZoominUAR in a hypothetical case (the underlined itemsets in these tables are assumed to represent the "large" itemsets in that specific iteration).

Observe that:

(a) In the second iteration, the same itemset {A, ¬B, C} is considered twice: once each when itemsets are generated for the two beliefs, A → B and C → B. This could result in discovering the same rule (e.g. A, C → ¬B) by starting from two different beliefs.

(b) When itemsets are considered for the belief A → B, the third iteration considers only the itemset {A, ¬B, C, ¬D} for support since this is the only itemset that satisfies the condition that all its subsets containing {A, ¬B} are "large" (the underlined itemsets in the previous iteration). However, one of its subsets {¬B, C, ¬D} does not have support as determined in the second iteration for the *other* belief C → B. Hence, the itemset {A, ¬B, C, ¬D} is guaranteed *not* to have the minimum support and therefore does not even need to be considered for checking its support.

```
Inputs: Beliefs Bel_Set, Dataset D, Threshold min_support,
Outputs: The set of large itemsets used by the rest of ZoominUAR to generate
the  itemsets Items_In_UnexpRule_B (as in Fig. 4.1)

1   forall beliefs B_i ∈ Bel_Set {
2      C_0[i] = {{¬head(B_i),body(B_i)}}; C_0'[i] = {{body(B_i)}};
3   }
4   k = 0;
5   while (∃i : C_k[i] != ∅ ) do {
6      for (j = 1 to numbeliefs) {
7         forall candidates c ∈ C_k[j] ∪ C_k'[j], get support(c)
8         L_k[j]={x | x ∈ C_k[j] ∪ C_k'[j],support(x) ≥ min_support }
9      }
10     k++
11     for (j = 1 to numbeliefs) {
12        C_k[j] = generate_new_candidates(L_k-1[], B);
13        C_k'[j] = generate_bodies(C_k [j], B);
14     }
15  }
```

Figure 5.2 *Extension to the candidate building phase of Algorithm ZoominUAR*

We incorporated these observations into the extended version of ZoominUAR. In Figure 5.2 we present the modified candidate building phase of ZoominUAR. The rest of the algorithm is the same as shown in Figure 4.1.

6. Experiments

We tested our algorithm on consumer purchase data from a major market research firm. We pre-processed this data by combining different data sets, made available to us by this firm, into *one* table containing 36 different attributes. These attributes pertain to the item purchased by a shopper at a store, together with certain characteristics of the store and the demographic data about the shopper and his or her family[9]. Some demographic attributes include age and sex of the shopper, occupation, income and marital status of the household head and the presence of children in the family and the size of the household. Some transaction-specific attributes include type of item purchased, coupon usage (whether the shopper used any coupons to get a lower price or not), the availability of store coupons or manufacturer's coupons and presence of advertisements for the product purchased in the store.

While we generated this combined data set, we also restricted the purchasing records only to the class of carbonated beverages, i.e., each record in this data set refers to a purchase of some carbonated beverage by a shopper. The resulting dataset had 87437 records, each consisting of 36 *discrete* fields. The levels of discrete attributes range from 2 to 12 distinct values.

6.1 Discovering Unexpected Patterns.

We compiled 15 beliefs about the data in this domain which fall into three groups: (1) Usage of coupons, e.g. "*young shoppers with high income tend not to use coupons*". (2) Purchase of diet vs. regular drinks, e.g. *"shoppers in households with children tend to purchase regular beverages more than diet"*. (3) Day of shopping, e.g. *"professionals tend to shop more on weekends than on weekdays"*. Some of these beliefs were solicited from experts and others were based on prior analyses of data. These beliefs were certainly not exhaustive about such a

[9] We would like to point out that this is unnormalized data that contains in one file both transaction and demographic data.

complex application in the sense that they do not contain all possible beliefs that a person may have about consumer purchase data. The set of beliefs that we used were selected just for illustrative purposes. In general, a much more complete set of beliefs can be obtained by learning them from the data as discussed in [18].

#	Belief	Some Unexpected Rules for these Beliefs
1	occupation = retired → coupon_usage = yes	occupation = retired → coupon_usage = no [c = 0.9, s = 11%]
2	occupation = professional → day = weekend	occupation = professional, household_size = large → day = weekday [c = 0.6, s = 1%] occupation = professional, month = december → day = weekday [c = 0.6, s = 1%]
3	children = yes → drink = regular	children = yes, store_advertisement = large → drink = diet [c = 0.64, s = 1%]

Figure 6.1. *Some Unexpected Rules Derived from Consumer Purchase Data.*

Fig. 6.1 illustrates some of the unexpected rules discovered using our algorithm. The first rule in Fig. 6.1 is that shoppers who are retired do not use coupons, which is a *direct* contradiction of our belief. More subtle cases are when the unexpected rules do not *directly* contradict our beliefs. For example, we believed that professionals shop more on weekends than on weekdays (belief #2 in Fig. 6.1). Though the belief holds on the data, we find that, during December or when the household size is large, the belief is contradicted: professionals tend to shop more on weekdays in these cases. Though unexpected, *ex-post* these rules seem to make sense given that there are usually more holidays in December or that large households may require the shopper to shop more often on demand than at convenient times. We also believed that households that had children tend to buy more of regular than diet beverages (belief #3). Though this did seem to hold on the data, we found that when there are large advertisements in the store they bought more diet than regular drinks.

6.2 Comparison with Apriori.

In addition to our belief-driven algorithm ZoomUAR we tested a standard association rule generating algorithm, Apriori [2] on the consumer purchase data. ZoomUAR generated about 600 unexpected rules while Apriori generated over 40,000 rules. To compare the interestingness of the rules generated, for illustrative purposes we list a few rules generated by each method in Figure 6.2. For Apriori we selected some of the strongest rules (all with almost a 100% confidence!) and for ZoomUAR we manually selected some rules since we are dealing with a much smaller set of rules.

Rules from ZoomUAR	Rules from Apriori
1. professional, december → weekday (0.6)	1. weekday → no_display_in_lobby (1.0)
2. professional, large_household → weekday (0.6)	2. weekend → no_display_in_lobby (1.0)
3. children, store_advertisement → diet (0.6)	3. january → no_display_in_lobby (1.0)
4. male, young → diet (0.7)	4. february → no_display_in_lobby (1.0)
5. retired → no_coupon_usage (0.9)	5. march → no_display_in_lobby (1.0)
6. old, low_income → no_coupon_usage (0.9)	6. april → no_display_in_lobby (1.0)

Figure 6.2. *Comparison of rules generated from ZoomUAR and Apriori*

The rules generated from ZoomUAR (Figure 6.2) are not statistically very strong, but are interesting since they were unexpected with respected to some of our expectations. In contrast, some of the top few rules generated by Apriori were extremely strong in the statistical sense (close to a 100% rule confidence!). However these rules, as they turned out, were an artifact of the data - for all the records in the data, there were no product displays in the lobby of the store during the purchase. Hence, trivially, any rule $X \rightarrow$ *no_display_in_lobby* would have a 100% confidence.

6.3 Discussion.

As shown in the theorem in Section 4.5, for any belief, A → B ZoomUAR discovers *all* unexpected rules of the form X → Y, where X and A are conjunctions of atomic conditions and Y and B are single atomic conditions involving binary attributes. Apriori (extended for discrete attributes) on the other hand discovers all association rules. The rules that ZoomUAR discovers are therefore a subset of the rules that Apriori discovers. This subset consists of the set of all unexpected rules, and is the most interesting subset of rules if unexpectedness is used as the measure of interestingness. Are there some patterns that Apriori discovers and that ZoomUAR does not that could be "interesting"? Inasmuch as unexpectedness is the single measure of interestingness, this can never be the case (based on the theorem in Section 4.5). However from a more general perspective, there may be other subjective measures of interestingness (such as actionability [3,12,16,17]) that could result in ZoomUAR "missing" some of the "interesting" rules. We would like to make three observations in this regard. First, ZoomUAR is not intended to discover all "interesting" rules. Rather, it is an algorithm that discovers all *unexpected* rules. Hence, ZoomUAR by design, discovers only the subset of interesting rules that satisfy unexpectedness. Second, even for another subjective measure such as "actionability" the only "missing" rules are actionable, but expected rules. It has been conjectured [16,17,18] that most actionable rules are unexpected and hence this subset is small. Third, inasmuch as it is possible to explicitly characterize *all* facets of interestingness, it may be possible to develop methods that discover all interesting patterns. Our work is just a step in this direction. Characterizing all the facets of interestingness and developing methods to discover all interesting patterns are important areas of future research in data mining.

7. Conclusions

In this paper we proposed a new definition of unexpectedness of a rule with respect to a belief and presented an algorithm that finds unexpected association rules from data using this measure. We tested our methods on consumer purchase data from a market research firm and found some unexpected patterns with respect to our belief set. We also compared our approach with a standard association rule generating algorithm (Apriori) across the two dimensions:

interestingness of rules and number of rules generated. We conclude that our method discovers, generally, fewer rules and avoids discovering many obvious or irrelevant rules as Apriori does. This means that our approach provides more focused and, therefore, more efficient search for interesting rules than Apriori.

In future work we plan to extend our algorithm to discover unexpected patterns of a more general nature than association rules. We also plan to apply our method in the context of knowledge refinement based on the discovery of unexpected patterns. More generally, the contribution of this research will be procedures for making data-mining more intelligent and useful for the decision maker. Our approach should lead to the development of decision support systems that provide more relevant patterns in the data to the user, patterns that confirm or challenge the user's beliefs and domain knowledge. We believe that such a breakthrough is required if data-mining is to achieve its potential in business applications.

References

[1] Agrawal, R., Imielinski, T. and Swami, A., 1993. Mining Association Rules Between Sets of Items in Large Databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pp. 207-216.

[2] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Verkamo,A.I., 1995. Fast Discovery of Association Rules. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. eds., *Advances in Knowledge Discovery and Data Mining*. AAAI Press.

[3] Adomavicius, G., and Tuzhilin, A., 1997. Discovery of Actionable Patterns in Databases: The Action Hierarchy Approach. In *Proc. of the Third International Conference on Knowledge Discovery and Data Mining.*

[4] Adriaans, P. and Zantinge, D., 1996. Data Mining. Addison Wesley Longman.

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

[6] Brin, S., Motwani, R., Ullman, J.D., and Tsur, S., 1997. Dynamic Itemset Counting and Implication Rules for Market Basket Data. *Procs. ACM SIGMOD Int. Conf. on Mgmt. of Data, pp.255-264.*

[7] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., 1996. From Data Mining to Knowledge Discovery: An Overview. In Fayyad, U.M.,Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. eds., *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press.

[8] Liu, B. and Hsu, W., 1996. Post-Analysis of Learned Rules. In *Proc. of the Thirteenth National Conference on Artificial Intelligence (AAAI '96)*, pp. 828-834.

[9] Liu, B., Hsu, W. and Chen, S, 1997. Using General Impressions to Analyze Discovered Classification Rules. In *Proc. of the Third International Conference on Knowledge Discovery and Data Mining (KDD 97),* pp. 31-36.

[10] Merckt, T.V., 1997. *http://www.cs.su.oz.au/~thierry/ckdd.html*

[11] Piatetsky-Shapiro, G., 1991. Discovery, Analysis and Presentation of Strong Rules. In Piatetsky-Shapiro, G. and Frawley, W.J eds., *Knowledge Discovery in Databases*. AAAI/MIT Press.

[12] Piatetsky-Shapiro, G. and Matheus, C.J., 1994. The Interestingness of Deviations. In *Procs. of the AAAI-94 Workshop on Knowledge Discovery in Databases*, pp. 25-36.

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

[14] Reich, R.B., 1992. The Work of Nations: Preparing Ourselves for 21st Century Capitalism. New York, Alfred A. Knopf.

[15] Suzuki, E., 1997. Autonomous Discovery of Reliable Exception Rules. In *Proc. of the Third International Conference on Knowledge Discovery and Data Mining*, pp. 259-262.

[16] Silberschatz, A. and Tuzhilin, A., 1995. On Subjective Measures of Interestingness in Knowledge Discovery. In *Proc. of the First International Conference on Knowledge Discovery and Data Mining*, pp. 275-281.

[17] Silberschatz, A. and Tuzhilin, A., 1996. What Makes Patterns Interesting in Knowledge Discovery Systems. *IEEE Transactions on Knowledge and Data Engineering*. Special Issue on Data Mining, vol. 5, no. 6, pp. 970-974.

[18] Tuzhilin, A. and Silberschatz, A., 1996. A Belief-Driven Discovery Framework Based on Data Monitoring and Triggering. *Working Paper #IS-96-26, Dept. of Information Systems, Leonard N. Stern School of Business, NYU.*