Expert-Driven Validation of Rule-Based User Models in Personalization Applications[1]

Gediminas Adomavicius
Computer Science Department
New York University
adomavic@cs.nyu.edu

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

Abstract

In many e-commerce applications, ranging from dynamic Web content presentation, to personalized ad targeting, to individual recommendations to the customers, it is important to build personalized profiles of individual users from their transactional histories. These profiles constitute models of individual user behavior and can be specified with sets of rules learned from user transactional histories using various data mining techniques. Since many discovered rules can be spurious, irrelevant, or trivial, one of the main problems is how to perform post-analysis of the discovered rules, i.e., how to validate user profiles by separating "good" rules from the "bad." This validation process should be done with an explicit participation of the human expert. However, complications may arise because there can be very large numbers of rules discovered in the applications that deal with many users, and the expert cannot perform the validation on a rule-by-rule basis in a reasonable period of time. This paper presents a framework for building behavioral profiles of individual users. It also introduces a new approach to expert-driven validation of a very large number of rules pertaining to these users. In particular, it presents several types of validation operators, including rule grouping, filtering, browsing, and redundant rule elimination operators, that allow a human expert validate many individual rules at a time. By iteratively applying such operators, the human expert can validate a significant part of all the initially discovered rules in an acceptable time period. These validation operators were implemented as a part of a one-to-one profiling system. The paper also presents a case study of using this system for validating individual user rules discovered in a marketing application.

Keywords: personalization, profiling, rule discovery, post-analysis, validation.

[1]This paper substantially augments and improves the preliminary version that appeared as a poster paper in the Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99) [AT99].

1 Introduction

In various e-commerce applications, ranging from dynamic Web content presentation, to personalized ad targeting, to individual recommendations to the customers, *personalization* has become an important business problem [PR93, Per99]. For example, the personalized version of Yahoo (myYahoo) provides to its customers personalized content, such as local weather or interesting events in the area where the customer lives. As another example, Amazon.com and Moviecritic.com provide recommendations on what books to read and movies to see respectively. In general, there is a very strong interest in the industry in personalized (one-to-one) marketing applications [PR93, AKY98] and in recommender systems [CAC97, Kau98, Bau99, SNP99] that provide personal recommendations to individual users for products and services that might be of interest to them. The advantages of these personalized approaches over more traditional segmentation methods are well documented in the literature [PR93, Per99, AKY98].

One of the key issues in developing such e-commerce applications is the problem of constructing accurate and comprehensive *profiles* of individual customers that provide the most important information describing who the customers are and how they behave. This problem is so important for building successful e-commerce applications that some authors propose that companies treat customer profiles as key economic assets in addition to more traditional assets such as plant, equipment and human assets [Hag99, HS99]. Although some work on how to construct personal user profiles has been published in the academic literature (and we will review it below), most of the work has been done in the industry so far.

There are two main approaches to addressing the profiling problem developed by different companies. In the first approach, taken by such companies as Engage Technologies [www.engage.com] and Personify [www.personify.com], profiles are constructed from the customers' demographic and transactional data and contain important *factual* information about the customers. Examples of such factual information include (a) demographic attributes, such as age, address, income and a shoe size of a customer, and (b) certain facts extracted from his or her transactional data, such as that the average and maximal purchase amounts of that customer over the last year were $23 and $127 respectively, or that the favorite newspaper of a particular Travelocity customer is the New York Times and her favorite vacation destination is Almond Beach Club in Barbados. This factual data comprises the profile of a customer and is typically stored in a relational table.

According to the other approach, taken by such companies as Art Technology Group [www.atg.com] and BroadVision [www.broadvision.com], customer profiles contain not only factual information but also *rules* that describe on-line behavioral activities of the customers. However, these rules are defined by experts (e.g., a marketing manager working on a particular marketing application). For example, a manager may specify that if a customer of a certain type visits the Web site of the on-line groceries shopping company ShopTillUStop.com on Sunday evenings, that customer should be shown the discount coupons for diapers. This approach differs from the previous approach in that the profiles contain behavioral rules in addition to the factual information about the customer. However, these behavioral rules are not constructed in a truly one-to-one manner since these rules are specified by the expert rather than learned from the data and are applicable only to *groups* of customers.

In addition to the developments in the industry, the profiling problem was also studied in the data mining academic community in [FP96, FP97, ASY98, AT99, Cha99]. In particular, [FP96, FP97] studied this problem within the context of fraud detection in the cellular phone industry. This was done by learning rules pertaining to individual customers from the cellular phone usage data using the rule learning system RL [CP90]. However, these discovered rules were used not for the purpose of understanding the personal behavior of individual customers, but rather to instantiate generalized profilers that are applicable to several customer accounts for the purpose of learning fraud conditions.

[ASY98] study the problem of on-line mining of customer profiles specified with association rules, where the body of a rule refers to the demographic information of a user, such as age and salary, and the head of a rule refers to transactional information, such as purchasing characteristics. Moreover, [ASY98] present a multidimensional indexing structure for mining such rules. The proposed method provides a new approach to deriving association rules that *segment* users based on their transactional characteristics. However, it does not derive behavior of an *individual* user in a one-to-one fashion [PR93].

Still another approach to the profiling problem was presented by [Cha99] in the context of providing personalized Web search. In this approach the user profile consists of a Web Access Graph summarizing Web access patterns by the user, and a Page Interest Estimator characterizing interests of the user in various Web pages. Although the approach presented by [Cha99] goes beyond building simple factual profiles, these profiles are specialized to be used in specific Web-related applications, i.e., to provide personalized Web search. This

means that they do not attempt to capture all aspects of the on-line behavior of individual users. One specific consequence of this specialization is that [Cha99] does not use behavioral rules as a part of a user profile.

In [AT99], we presented an initial approach to the profiling problem that we expand and improve in this paper. In particular, in this paper we present a framework for building behavioral profiles of individual users. These behavioral profiles contain not only factual information about the users, but also capture more comprehensive behavioral information using conjunctive rules that are learned from user transactional histories using various data mining methods. However, there are caveats to this approach due to the nature of personalization applications. In particular, as will be explained in the paper, the behavioral rules learned about individual users can be unreliable, irrelevant, or obvious. Therefore, post-analysis, including *rule validation*, becomes an important issue for building accurate personalized profiles of users. The second contribution of this paper lies in developing a new approach to validating the discovered rules during the post-analysis stage of the data mining process. This validation process is performed by the domain expert who can iteratively apply various rule validation operators. In particular, we describe different validation operators and demonstrate how these operators are integrated into a unifying framework. Development of specific validation operators, in particular, rule grouping method based on attribute hierarchies, constitutes the third contribution of this paper. Finally, the paper describes a case study of testing the proposed validation method on a marketing application.

The "quality" of rules stored in user profiles can be defined in several ways. In particular, rules can be "good" because they are (1) statistically valid, (2) acceptable to a human expert in a given application, (3) "effective" in the sense that they result in certain benefits obtained in an application. In this paper, we focus on the first two aspects, i.e., statistical validity and acceptability to an expert. The third aspect of rule quality is a more complex issue, and we do not address it in this paper, leaving it as a topic for future research.

The rule validation problem in the post-analysis stage of the data mining process has been addressed before in the data mining community. In particular, there has been work done on specifying filtering constraints that select only certain types of rules from the set of all the discovered rules; examples of this research include [KMR$^+$94, LH96, LHM99]. In these approaches the user specifies constraints but does not do it *iteratively*. In contrast to this, it has been observed by several researchers, e.g. [BA96, FPSS96, ST96a, PJ98, LBA98, AT99,

Sah99], that knowledge discovery should be an iterative process that involves an explicit participation of the domain expert, and we apply this point of view to the rule validation process.

The rest of the paper is organized as follows. In Section 2, we present our approach to profiles and profile construction. The profile validation process is described in Section 3, and specific validation operators are presented in Section 4. In Section 5 we describe how to do incremental validation. In Section 6 we describe the case study of using our profiling system in a market research application. Finally, we discuss additional issues related to the profile construction problem in Section 7.

2 A Proposed Approach to Profiling

2.1 Defining User Profiles

In order to explain what user profiles are and how they can be constructed, we first focus on the data that is used for constructing these profiles.

Data Model. Various e-commerce personalization applications can contain different types of data about individual users. However, this data can be classified in many applications into two basic types – *factual* and *transactional*, where the factual data describes who the user *is* and the transactional data describes what the user *does*.

For example, in a marketing application based on purchasing histories of users, the factual data would be the demographic data of users, such as name, gender, birth date, and salary. The transactional data would consist of records of purchases that the user made over a period of time. A purchase record would include such attributes as the date of purchase, product purchased, product characteristics, amount of money spent, use or no use of a coupon, value of a coupon if used, discount applied, etc.

Profile Model. A profile is a collection of information that describes a user. One of the open issues in the profile construction process is what information should be included in a user profile. In their simplest form, user profiles contain *factual* information that can be described as a set of individual facts that, for example, can be stored in a record of a relational database table. These facts may include demographic information about the user, such as name, address, date of birth, and gender, that are usually taken from the user description data. The facts can also be derived from the transactional and item description data. Examples of such facts are *"the favorite beer of user ALW392 is Heineken"*, *"the biggest purchase*

made by ALW392 was for $237", *"the favorite movie star of ALW392 is Harrison Ford."* The construction of factual profiles is a relatively simple and well-understood problem, and keyword-based factual profiles have been extensively used in recommender systems.

A user profile can also contain a *behavioral* component that describes behavior of the user learned from his or her transactional history. One way to define user behavior is with a set of conjunctive rules, such as association [AMS+96] or classification rules [BFOS84]. Examples of rules describing user behavior are: *"when user ALW392 comes to the Web site Y from site Z, she usually returns back to site Z immediately"*, *"when shopping on the NetGrocer.com Web site on weekends, user ALW392 usually spends more than $100 on groceries"*, *"whenever user ALW392 goes on a business trip to Los Angeles, she stays there in expensive hotels."* The use of rules in profiles provides an intuitive, declarative and modular way to describe user behavior and was advocated in [FP97, AT99]. These rules can either be defined by domain experts, as is done in systems developed by BroadVision and Art Technology Group, or derived from the transactional data of a user using various data mining methods. We describe this derivation process in the next section.

2.2 Profile Construction

Since we focus on personalization applications, rule discovery methods should be applied *individually* to the transactional data of *every* user, thus, capturing truly personal behavior of each user.

Such rules can be discovered using various data mining algorithms. For example, to discover association rules, we can use *Apriori* [AMS+96] and its numerous variations. Similarly, to discover classification rules, we can use *CART* [BFOS84], *C4.5* [Qui93], or other classification rule discovery methods. We would like to point out that our approach *is not restricted* to any specific representation of data mining rules and their discovery methods.

One of the serious problems with many rule discovery methods is that they tend to generate large numbers of patterns, and often many of them, while being statistically acceptable, are trivial, spurious, or just not relevant to the application at hand [PSM94, ST96b, LH96, BMUT97, Ste97, PT98, PT99]. Therefore, post-analysis of discovered rules becomes an important issue, since there is a need to *validate* the discovered rules. For example, assume that a data mining method discovered the rule stating that, whenever customer ALW392 goes on a business trip to Los Angeles, she mostly stays in expensive hotels there. In partic-

ular, assume that ALW392 went to Los Angeles 7 times over the past 2 years and 5 out of 7 times stayed in expensive hotels. Before this rule can be placed into ALW392's profile, it needs to be validated, since it may not be immediately clear whether this rule really captures the behavior of ALW392, or whether it constitutes a spurious correlation or is simply not relevant to the application at hand. In the next section we present methods for validating behavioral rules in user profiles.

3 Validation of User Profiles

A common way to perform the post-analysis of data mining results is to let the *domain expert* perform this task, and several data mining systems support this capability. For example, MineSet [BKK97] provides a wide range of visualization techniques allowing the end-user to examine visually the results discovered by its data mining tools and thus evaluate the quality of these results.

In our approach, individual rules discovered during the data mining stage are validated by the expert, and, depending on how well they represent the actual behaviors of the users, some rules are "accepted" and some "rejected" by the expert. Then the accepted rules form the behavioral profiles of users.

One of the main issues about validating individual rules of users by a human expert is scalability. In many e-commerce personalization applications the number of users tends to be very large. For example, the number of registered users at major Web sites is measured in millions. If we discover a hundred rules per customer on average, then the total number of rules for such sites would be measured in hundreds of millions. Therefore, it would be impossible for a human expert to validate all the discovered rules on a one-by-one basis in such applications.

We address this problem by providing a framework allowing the human expert validate *large numbers* of rules (instead of individual rules) at a time with relatively little input from the expert. This is done by applying different *rule validation operators* that are described in Section 4. Then rule validation becomes an *iterative* process and is described in Figure 1. In particular, the profile building activity is divided into two phases. In Phase I, the data mining phase, rules describing behaviors of individual users are generated from the users' transactional data as was described in Section 2.2.

Phase II constitutes the rule validation process. Rule validation, unlike rule discovery

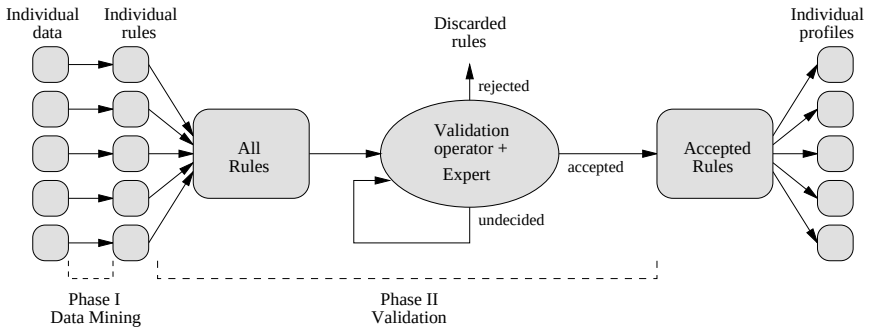

Figure 1: The profile building process.

(Phase I), is not performed separately for each user, but rather for *all* users at once. The reason we propose performing rule validation collectively (rather than individually) for all users is that there are usually many similar or even identical rules across different users. For example, the rule *"when shopping on the NetGrocer.com Web site on weekends, user ALW392 usually spends more than $100 on groceries"* can be common to many users. In addition, although rules *"when user ALW392 comes to our Web site from site Y, she usually returns back to site Y immediately,"* and *"when user KTL158 comes to our Web site from site Z, she usually returns back to site Z immediately,"* are not identical, they are quite "similar" and can be examined by the expert together. The collective rule validation allows one to deal with such common rules once, thus significantly reducing validation effort. Therefore, in the beginning of Phase II, rules from all the users are collected into one set. Each rule is tagged with the ID of the user to which it belongs, so that each accepted rule could be put into the profile of that user at the end of the validation phase.

After rules from all users are collected into one set, the rule validation process is performed as a second part of Phase II. This process is described in Figure 2. All rules discovered during Phase I (denoted by R_{all} in Figure 2) are considered unvalidated. The human expert selects various validation operators and applies them successively to the set of unvalidated rules. The application of each validation operator results in validation of some of the rules. In particular, some rules get accepted and some rejected (sets O_{acc} and O_{rej} in Figure 2). Then the next validation operator would be applied to the set of the remaining unvalidated rules (set R_{unv}). This validation process stops when the *TerminateValidationProcess* condition is met. This condition is set by the human expert and is discussed later in this section. After the validation process is stopped, the set of all the discovered rules (R_{all}) is split into

```
Input:   Set of all discovered rules R_all.
Output:  Mutually disjoint sets of rules R_acc, R_rej, R_unv,
         such that R_all = R_acc ∪ R_rej ∪ R_unv.

(1)  R_unv := R_all, R_acc := ∅, R_rej := ∅.
(2)  while (not TerminateValidationProcess()) begin
(3)     Expert selects a validation operator (say, O) from the set of
        available validation operators.
(4)     O is applied to R_unv.  Result:  disjoint sets O_acc and O_rej.
(5)     R_unv := R_unv − O_acc − O_rej, R_acc := R_acc ∪ O_acc, R_rej := R_rej ∪ O_rej.
(6)  end
```

Figure 2: An algorithm for the rule validation process.

three disjoint sets: accepted rules (R_{acc}), rejected rules (R_{rej}), and possibly some remaining unvalidated rules (R_{unv}). At the end of Phase II all the accepted rules are put into the behavioral profiles of their respective users. This is possible, because all the rules have been tagged with the user ID in the beginning of Phase II as described above.

As was already stated above and shown in Figure 2, various validation operators are successively applied to the set of the unvalidated rules until the stopping criterion *TerminateValidationProcess* is reached. The stopping criterion can be specified by the expert and may include such conditions as (a) only few rules remain unvalidated, (b) only few rules are being validated at a time by one or several validation operators, and (c) the total elapsed validation time exceeds the predetermined validation time.

In this section we described the overall validation process. We present the detailed description of various specific validation operators in the next section.

4 Validation Operators

As stated in Section 3, validation operators provide a way for the domain expert to examine multiple rules at a time. This examination process can be performed in the following two ways. First, the expert may already know some types of rules that he or she wants to examine and accept or reject based on the prior experience. Therefore, it is important to provide capabilities allowing him or her to specify such types of rules in advance. In this section, we present template- and interestingness-based filtering operators that serve this purpose. Second, the expert may not know all the relevant types of rules in advance, and it is important to provide methods that *group* discovered rules into classes that he or she

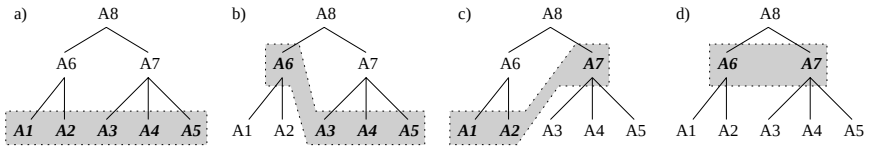

Figure 3: An example of an attribute hierarchy for similarity-based grouping.

can subsequently examine and validate. In this section we also present the similarity-based rule grouping operator that serves this purpose. In addition, we describe other operators that can be used in the validation process, including visualization, statistical analysis, and browsing operators.

Although our validation methods are general and can be applied to several forms of conjunctive rules, we focus mainly on association rules with discrete values in this paper.

4.1 Similarity-based rule grouping

As pointed out in Section 3, there can be many "similar" rules among all the discovered rules, and it would be useful for the domain expert to evaluate all these similar rules together rather than individually. In order to do this, some similarity measure that would allow grouping similar rules together needs to be specified.

In this paper, we propose a method to specify such a similarity measure using *attribute hierarchies*. An attribute hierarchy is organized as a tree by the human expert in the beginning of the validation process.[2] The leaves of the tree consist of all the attributes of the data set to which rule discovery methods were applied, i.e., all the attributes that can potentially be present in the discovered rules. The non-leaf nodes in the tree are specified by the human expert and are obtained by combining several lower-level nodes into one parent node. For instance, Figure 3 presents an example of such a hierarchy, where nodes A1 and A2 are combined into node A6 and nodes A3, A4 and A5 into node A7, and then nodes A6 and A7 are combined into node A8. Another example of an attribute hierarchy is presented in Figure 9. We call non-leaf nodes of an attribute hierarchy *aggregated attributes*.

The attribute hierarchy is used for determining similar rules and grouping them together.

[2]We would like to point out that in certain domains, e.g., groceries, such hierarchies may already exist, and some well-known data mining algorithms, such as [CP90, SA95], explicitly assume the existence of attribute (or, more generally, feature) hierarchies. Alternatively, attribute hierarchies may possibly be constructed automatically in certain other applications. However, automatic construction of such hierarchies is beyond the scope of this paper.

More specifically, the semantics of the similarity-based grouping operator is defined as follows.

1. **Specifying rule aggregation level**. Rules are grouped by specifying the level of rule aggregation in the attribute hierarchy which is provided by the human expert. Such a specification is called a *cut*, and it forms a subset of all the nodes of the tree (leaf and non-leaf), such that for every path from a leaf node to the root, exactly one node on such path belongs to this subset. Therefore, given a cut, every leaf node has its corresponding *cut node*. Given a cut C, we define for any leaf node X_i its corresponding cut node $cut_C(X_i)$ as follows:

$$cut_C(X_i) = \begin{cases} X_i, & if \;\; X_i \in C \\ cut_C(parent(X_i)), & otherwise \end{cases}$$

   Figure 3 presents several different cuts of an attribute hierarchy that are represented by shaded regions. For example, for the cut from Figure 3(c), $cut_{3c}(A2) = A2$ and $cut_{3c}(A3) = A7$. Moreover, the cut node of any leaf node can be calculated in constant time by implementing a straightforward lookup table for that cut.

2. **Aggregating rules**. Given a cut C, a rule $X_1 \wedge ... \wedge X_k \Rightarrow X_{k+1} \wedge ... \wedge X_l$ is *aggregated* by performing the following *syntactic* transformation:

$$\begin{array}{l} cut_C(X_1 \wedge ... \wedge X_k \Rightarrow X_{k+1} \wedge ... \wedge X_l) \\ cut_C(X_1) \wedge ... \wedge cut_C(X_k) \Rightarrow cut_C(X_{k+1}) \wedge ... \wedge cut_C(X_l) \end{array} \quad =$$

   where $cut_C(X_i)$ maps each leaf node of the attribute hierarchy into its corresponding cut node as described in Step 1 above. The resulting rule is called an *aggregated rule*. Since several different leaf nodes can have the same cut node, sometimes after aggregating a rule we can get multiple instances of the same aggregated attribute in the body or in the head of the rule. In this case we simply eliminate those extra instances of an attribute. Consider, for example, the rule $A2 \wedge A3 \wedge A4 \Rightarrow A5$. By applying cut (c) from Figure 3 to this rule, we will get the aggregated rule $A2 \wedge A7 \wedge A7 \Rightarrow A7$, and by removing duplicate terms $A7$ in the body of the rule we finally get $A2 \wedge A7 \Rightarrow A7$.[3]

[3]Note that, while the just obtained aggregated rule $A2 \wedge A7 \Rightarrow A7$ may look like a tautology, it is not. As mentioned above, aggregated rules are obtained from the originally discovered rules using purely syntactic transformations. Therefore, the above mentioned aggregated rule does not make any logical statements about the relationship between attributes $A2$ and $A7$ in the given data, but simply denotes the *class* of rules of the particular syntactic structure.

Initial rule set S	Rule groups obtained from rule set S using cuts :					
	cut 3(b)		cut 3(c)		cut 3(d)	
$A1 \Rightarrow A3$	$A6 \Rightarrow A3$	(3)	$A7 \Rightarrow A7$	(2)	$A6 \Rightarrow A7$	(3)
$A1 \wedge A2 \Rightarrow A3$	$A3 \wedge A6 \Rightarrow A5$	(2)	$A2 \wedge A7 \Rightarrow A1$	(2)	$A6 \wedge A7 \Rightarrow A7$	(3)
$A1 \wedge A2 \wedge A3 \Rightarrow A5$	$A3 \Rightarrow A5$	(1)	$A2 \wedge A7 \Rightarrow A7$	(2)	$A6 \wedge A7 \Rightarrow A6$	(2)
$A2 \wedge A3 \Rightarrow A4$	$A3 \wedge A5 \Rightarrow A4$	(1)	$A1 \Rightarrow A7$	(1)	$A7 \Rightarrow A7$	(2)
$A2 \wedge A3 \Rightarrow A5$	$A3 \wedge A6 \Rightarrow A4$	(1)	$A2 \Rightarrow A7$	(1)		
$A2 \Rightarrow A3$	$A4 \wedge A6 \Rightarrow A6$	(1)	$A1 \wedge A2 \Rightarrow A7$	(1)		
$A2 \wedge A4 \Rightarrow A1$	$A5 \wedge A6 \Rightarrow A6$	(1)	$A1 \wedge A2 \wedge A7 \Rightarrow A7$	(1)		
$A3 \Rightarrow A5$						
$A2 \wedge A5 \Rightarrow A1$						
$A3 \wedge A5 \Rightarrow A4$						

Figure 4: Grouping a Set of Rules Using Several Different Cuts from Figure 3 (the number of rules in groups is specified in parentheses).

Given a cut, the computational complexity of a single rule aggregation is linearly proportional to the size of the rule (i.e., total number of attributes in the rule), as will be described later.

3. **Grouping rules**. Given a cut C, we can group a set of rules S into groups by applying C to every rule in S as described in Step 2 above. When a cut is applied to a set of rules, different rules can be mapped into the same aggregated rule. For example, consider rules $A2 \wedge A3 \wedge A4 \Rightarrow A5$ and $A2 \wedge A5 \Rightarrow A3$. After applying cut (c) from Figure 3 to both of them, they are mapped into the same rule $A2 \wedge A7 \Rightarrow A7$. More generally, we can group a set of rules based on the cut C as follows. Two rules R_1 and R_2 belong to the same group *if and only if* $cut_C(R_1) = cut_C(R_2)$. Naturally, two different aggregated rules represent two disjoint groups of rules. As an example, Figure 4 presents the results of grouping a set of rules based on the attribute hierarchy and several different cuts shown in Figure 3.

The grouping operator described above allows the user to group rules into sets of similar rules, where similarity is defined by the expert who selects a specific cut of the attribute hierarchy. Moreover, instead of examining and validating individual rules inside each group, the user can examine the group of these rules as a whole based on the aggregated rule (that is common for all the rules in the group) and decide whether to accept or reject *all* the rules in that group at once based on this aggregated rule.

So far, we assumed that the leaves in the attribute hierarchies are specified by the attributes of the data set. However, we also consider the case when attribute hierarchies include *values* and *aggregated values* of attributes from the data set. For example, assume that a data set has attribute *Month*. Then Figure 5 presents an attribute hierarchy with 12

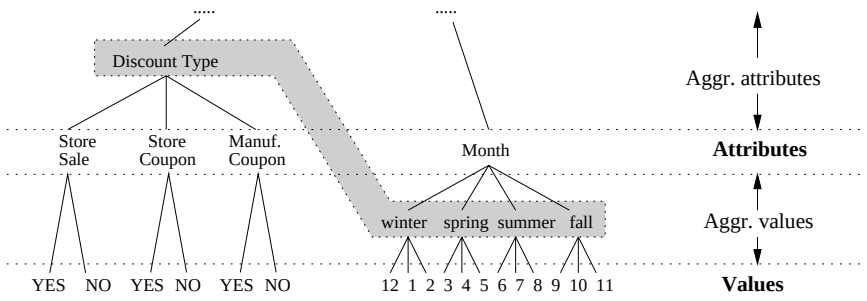

Figure 5: A fragment of attribute hierarchy which includes attribute values.

values as the leaves representing specific months of the year that are grouped together into four aggregated values: *winter*, *spring*, *summer*, and *fall*.

For these extended hierarchies, cuts can include not only attribute and aggregated attribute nodes, but also value and aggregated value nodes. For example, consider the extended attribute hierarchy presented in Figure 5 that includes 12 values for the attribute *Month* and the boolean values for the attributes *StoreSale*, *StoreCoupon*, and *ManufCoupon*. Also consider the cut from Figure 5 specified with a shaded line, and the following three rules: (1) *Month*=3 $\Rightarrow$ *StoreSale*=YES, (2) *Month*=5 $\Rightarrow$ *ManufCoupon*=NO, (3) *Month*=10 $\Rightarrow$ *StoreSale*=YES. The cut presented in Figure 5 maps rules (1) and (2) into the same aggregated rule *Month=spring* $\Rightarrow$ *DiscountType*. However, rule (3) is mapped into a different aggregated rule *Month=fall* $\Rightarrow$ *DiscountType* by the cut. Therefore rule (3) will be placed into a different group than rules (1) and (2).

The grouping operator based on attribute hierarchies provides a flexible way for the expert to group rules according to the granularity important to that expert. This provides the expert with the ability to evaluate larger or smaller number of groups of similar rules based on his or her preferences and needs. Moreover, an efficient algorithm that implements the grouping operator has been developed and is presented in Figure 6. The procedure `GROUP` performs the grouping using a single pass over the set of discovered rules (the `foreach` loop statement in lines 3-7 in Figure 6). For each rule r in the input rule set R (line 3) we compute its aggregated rule r' using the procedure `AGGR_ATTRS` (lines 5-6).

The procedure `AGGR_ATTRS` (lines 11-15) performs the aggregation of a set of attributes. Using the mapping cut_C, each element of an attribute set is aggregated in constant time. Moreover, since the attribute set `AttrSet` is implemented as a hash table, an insertion of an aggregated attribute into the resulting set A' (line 13, inside the loop) also takes constant

```
1    GROUP ( RuleSet R, Map cut_C ) {
2        GroupSet G := ∅;
3        foreach r from R {
4            r' := new Rule;
5            r'.body := AGGR_ATTRS(r.body, cut_C);
6            r'.head := AGGR_ATTRS(r.head, cut_C);
7            G := G ∪ r';
8        }
9        return G;
10   }
11   AGGR_ATTRS( AttrSet A, Map cut_C ) {
12       AttrSet A' := ∅;
13       foreach a from A { A' := A' ∪ cut_C[a]; }
14       return A';
15   }
```

Figure 6: Algorithm for similarity-based rule grouping.

time. Therefore, the total running time of the procedure `AGGR_ATTRS` is linear in the size of the attribute set.

As the result, the running time of a rule aggregation (lines 5-6) is linear in the size of the rule (i.e., total number of attributes in the body and the head of the rule). Also, since the group set `GroupSet` is implemented as a hash tree data structure (similar to the one described by [Sri96]), an insertion of a group into the resulting group set G (line 7) is also linear in the size of the rule. Consequently, the running time of the whole grouping algorithm is linear in the total size of the rules to be grouped. Note also that, besides the computational space needed to store the resultant rule groups, the algorithm uses virtually no additional computational space (except for several local variables).

In summary, the grouping algorithm presented in Figure 6 scales up well, which is very important for personalization applications dealing with very large numbers of rules.

There have been related approaches to rule grouping proposed in the literature [LSW97, WTL98] that consider association rules in which both numeric and categorical attributes can appear in the body and only categorical attributes in the head of a rule. However, [LSW97] take a more restrictive approach by allowing only two numeric attributes in the body and one categorical attribute in the head of a rule, whereas [WTL98] allow any combination of numeric and categorical attributes in the body and one or more categorical attributes in the head of a rule. Both of the approaches merge adjacent intervals of numeric values in a bottom-up manner, where [LSW97] utilize a clustering approach to merging and [WTL98] maximize certain interestingness measures during the merging process. It is interesting to

observe that interval merging can also be supported in our rule grouping operator by letting a domain expert specify the cuts at the value and aggregated-value levels of the attribute hierarchy (as shown in Figure 5).

However, in order to allow the domain expert to validate very large numbers of rules within a reasonable amount of time, personalization applications require more powerful grouping capabilities that go beyond the interval merging techniques for attribute values. Therefore, our approach differs from [LSW97, WTL98] in that it allows the grouping of rules with different structures, at different levels of the attribute hierarchy and also not only for numerical but for categorical attributes as well. Moreover, the domain expert has the flexibility to specify the relevant cuts in the attribute hierarchy, whereas the interval merging approaches do the merging automatically based on the built-in heuristics.

Still another related approach to grouping is proposed by [TKR+95] where a distance between two association rules is defined as the number of transactions on which two rules differ. Using this distance measure, [TKR+95] group all the rules into appropriate clusters. One of the limitations of this approach lies in that the distance measures selected for rule clustering are somewhat arbitrary. Moreover, it is not clear how to describe concisely the rule cluster to the user for the purpose of evaluation, since rules belonging to the same cluster may have substantially different structures. In contrast, in our proposed similarity-based grouping approach every rule cluster is uniquely represented by its aggregated rule (common to all rules in that cluster), that is concise and descriptive.

4.2 Template-based rule filtering

Another validation operator is *template-based rule filtering* that allows the expert to specify in general terms the types of rules that he or she either wants to accept (*accepting* template) or reject (*rejecting* template). After a template is specified, unvalidated rules are "matched" against it. Rules that match an accepting template are accepted and put into user profiles, and rules that match a rejecting template are rejected. Rules that do not match a template remain unvalidated.

The formal definition of the template-based filtering operator is provided with the BNF specification, the top-most fragment of which is presented in Figure 7. This specification language allows one to define various constraints that the expert can impose on:

- *The syntactic structure of the body (antecedent) and the head (consequent) of the rule.*

During the rule validation process, restrictions can be placed on combinations of attributes and, possibly, their values that can appear in the rule using the following set-like notation:

$$rule\_part \;\; set\_op \;\; \{ \; A_1, \; A_2, \; ..., \; A_N \; \}$$

where *rule_part* can be either BODY, HEAD, or RULE, and it specifies the part of the rule (antecedent, consequent, or the whole rule, respectively) on which the restriction is being placed; *set_op* is a set comparison operator, such as $=, \neq, \subset, \subseteq, \supset, \supseteq$; $\{A_i\}_{i=1..N}$ is a comparison set, i.e, a set of attributes to be compared (using *set_op*) with the set of attributes appearing in the *rule_part* of each rule. This template matches the rules for which set comparison yields true. For example, if *rule_part* is BODY and *set_op* is $\subseteq$ then this template matches the rules whose bodies have only the attributes from the set $\{A_i\}_{i=1..N}$. Moreover, the comparison set $\{A_i\}_{i=1..N}$ can be extended to include not only attribute names, but also a value or a set of values that a given attribute can have. In particular, each element of a comparison set can be described as A_i, or $A_i \; = \; val$, or even $A_i \; = \; \{ \; val_1, \; val_2, \; ... \; \}$.

Using hash-based data structures for storing rule templates, we can implement the filtering algorithm that runs in time linear in the total size of the rules to be filtered.

- *Basic statistical parameters of the rule.* During the rule validation process, restrictions on basic statistical parameters (e.g., support and confidence for association rules) can be imposed using the following template:

$$\textbf{STATS} \; \{ \; par_1 \; op_1 \; val_1, \quad par_2 \; op_2 \; val_2, \quad ... \; \}$$

where par_i is the name of a statistical parameter (e.g., *conf* for confidence, *supp* for support); op_i is a comparison operator, such as $>, \geq, <, \leq, =, \neq$; and val_i is a value of a statistical parameter. This template matches the rules, parameters of which satisfy all the specified restrictions $par_i \; op_i \; val_i$. Examples of such restrictions are $conf < 80\%$ and $supp \geq 35\%$.

The computational complexity of this filter is linear in the number of rules since each rule requires a constant time to check if it satisfies the constraint.

- *The factual information about a user for whom the rule was discovered.* As mentioned in Section 2.1, we assume that the factual information of each user can be stored as a record in a relational table. Our template specification language allows one to formulate a restriction on the factual information of users for the purpose of obtaining only the rules that belong to this "restricted" set of users. Formally, such a template is specified as follows:

  FACTS { `restriction` }

  This type of filter works in two steps. First, the following SQL statement that returns a set of "qualifying" users (i.e., users that satisfy the restriction) is generated and executed:[4]

  SELECT *UserId* **FROM** *FactualData* **WHERE** `restriction`

  And, second, the rule set is filtered to include *only* the rules of the users returned by the SQL query described above.

In our template-based filtering operator, each of the above templates can be used individually or several templates can be combined into one using boolean operations AND, OR, and NOT. The filtering semantics of such a template combination is defined as follows. Rule r matches template $NOT\ T$ if r does not match T; rule r matches template $T_1\ AND\ T_2$ if r matches both T_1 and T_2; finally, rule r matches template $T_1\ OR\ T_2$ if r matches at least one of T_1, T_2.

The proposed language is related to the data mining query languages, such as the ones described in [KMR+94, SOMZ96, HFW+96, LHC97, SVA97, MPC98, IV99], among them M-SQL [IV99] and the template language of [KMR+94] being the closest to our proposal. In this paper, we enhanced and combined various features of these two languages into one integrated language and included features arising from idiosyncrasies of personalization applications, such as the existence of individual data mining rules and of factual information about the users. Some examples of filtering operators are provided below.

1. Accept all the rules that refer to Grand Union stores:

   ACCEPT : **RULE** $\supset$ { *Store* = *GrandUnion* }

[4] Therefore, the syntax of the `restriction` element in the FACTS filter allows any expression that is acceptable in the WHERE clause of an SQL SELECT statement.

template	→	*action* : *tmpl_expression*
action	→	**ACCEPT** | **REJECT**
tmpl_expression	→	*atom_tmpl* | *atom_tmpl* *logic_oper* *tmpl_expression*
atom_tmpl	→	*inverse* *pos_atom_tmpl*
logic_oper	→	**AND** | **OR**
inverse	→	ϵ | **NOT**
pos_atom_tmpl	→	*rule* | *stats* | *facts*
rule	→	*rule_part* *set_oper* { *trans_term_list* }
rule_part	→	**BODY** | **HEAD** | **RULE**
stats	→	**STATS** { *stat_term_list* }
facts	→	**FACTS** { *fact_term_list* }
set_oper	→	$=$ | $\neq$ | $\subset$ | $\subseteq$ | $\supset$ | $\supseteq$
trans_term_list	→	*trans_term* | *trans_term*, *trans_term_list*
trans_term	→	*attr_term* | *aggr_attr_term*
attr_term	→	*attr_name* | *attr_name* *compar_oper* *value*
		| *attr_name* $=$ *value_set*
stat_term_list	→	*stat_term* | *stat_term*, *stat_term_list*
stat_term	→	*stat_name* | *stat_name* *compar_oper* *stat_value*
stat_name	→	**supp** | **conf**
...	...	...

Figure 7: A fragment of the template specification language.

2. Reject all the rules that have attribute *Product* in the body (possibly among other attributes) and the head of the rule has either *DayOfWeek* or *Quantity* = *Big* in it:

   REJECT : **BODY** $\supseteq$ { *Product* }
   AND HEAD $\subset$ { *DayOfWeek*, *Quantity* = *Big* }

3. Accept all the rules that involve any combination of attributes *DayOfWeek* (only when value is *Mon* or *Wed*), *TimeOfDay*, and *Product*, in the body of the rule, that also have confidence greater than 65%:

   ACCEPT : **BODY** $\subseteq$ { *DayOfWeek* = { *Mon*, *Wed* }, *TimeOfDay*, *Product* } **AND STATS** { **conf** $>$ 65% }

4. Reject all the rules that have the attribute *Product* present in their bodies and, possibly, *DayOfWeek* or *TimeOfDay* (but no other attributes besides these):

   REJECT : **BODY** $\supseteq$ { *Product* }
   AND BODY $\subseteq$ { *DayOfWeek*, *TimeOfDay*, *Product* }

5. Reject all the rules that refer to the purchase of a luxury car for the low-income users:

   REJECT : **RULE** $\supseteq$ { *Product* = *LuxuryCar* }
   AND FACTS { *YearlyIncome* = *Low* }

6. The filtering operator can take advantage of an attribute hierarchy that was described in Section 4.1 and was used in the similarity-based grouping operator. That is, aggregated attributes and aggregated values can also be used in a template. For example, if

we would like to accept all the rules that involve any type of discount in the body and specify any spring month in the head (based on the attribute hierarchy from Figure 5), we would use the following template:

ACCEPT : **BODY** $\supseteq$ { *DiscountType* }
AND HEAD = { *Month* = *spring* }

As we have shown above, the template-based filtering operator is computationally inexpensive. Therefore, as with the similarity-based rule grouping operator, this operator also scales well for very large numbers of rules.

4.3 Interestingness-based rule filtering

As described above, our proposed template-based rule filtering operator allows the domain expert to accept or to reject the discovered rules based on their structure, statistical parameters, and factual characteristics of the users. In addition to this, we propose using a filtering operator that selects only the most "interesting" rules according to some interestingness criteria.

There has been much research done in recent years quantifying "interestingness" of a rule, and several metrics have been proposed and used as a result of this work. Among "objective" metrics, besides confidence and support [AIS93], there are gain [FMMT96], variance and chi-squared value [Mor98], gini [MFM+98], strength [DT93], conviction [BMUT97], sc- and pc-optimality [BA99], etc. "Subjective" metrics include unexpectedness [ST96b, LH96, Suz97, PT98] and actionability [PSM94, ST96b, AT97].

Any of these metrics can be used as a part of the interestingness-based filtering operator, and the validation system can support different interestingness criteria. Moreover, the domain expert can specify interestingness-based filters using a syntax similar to the syntax of the template-based filters. For example, the filter

ACCEPT : **INTERESTINGNESS** { **gain** > 0.5, **unexpected** }

specifies that all the high-gain and unexpected rules should be accepted. Moreover, the uniform syntax for both template-based and interestingness-based filter specifications allows to combine filters of both types into one. For example, the following template accepts all

actionable rules that mention the purchase of a luxury car in the body of the rule:

ACCEPT : **BODY** $\supseteq$ { *Product* = *LuxuryCar* }
AND INTERESTINGNESS { **actionable** }

We would like to point out that such interestingness-based filters can be added to the profile validation system as external modules, thus making the system more versatile. The efficiency of such interestingness-based filters depends on their inherent complexity (i.e., some interestingness measures are inherently more complex to calculate than others) and their particular implementation.

Redundant rule elimination. One class of non-interesting rules are *redundant* rules. For example, consider the association rule "*Product = AppleJuice* $\Rightarrow$ *Store = Grand Union (supp=2%, conf=100%)*" that was discovered for customer ALW392. This rule appears to capture a specific aspect of the customer behavior: customer ALW392 buys apple juice *only* at Grand Union, and we may add it to his behavioral profile. However, assume, that is was also determined from the data that this customer does *all* of his shopping at Grand Union. Then the above mentioned rule constitutes a special case of this finding.

The redundant rule elimination filter finds all the redundant rules and removes them from the user profiles. In other words, this operator eliminates the rules that, by themselves, do not carry any new information about the behavior of a user. One particular case of redundancy occurs when the consequent Y of a high-confidence rule $X \Rightarrow Y$ has a high support. For instance, following the previous example, the rule "*Product = AppleJuice* $\Rightarrow$ *Store = GrandUnion (supp=2%, conf=100%)*" would be removed from the profile of user ALW392 and only the fact *"Store = GrandUnion (supp=100%)"* (i.e., this customer shops only at Grand Union) will be kept.

The computational complexity of such redundant rule elimination filter is linear in the number of rules to be filtered, because for each rule we only have to check whether its consequent has a very high support measure. This check can be done in constant time using a lookup table that holds a most frequent value of each attribute (along with its actual frequency). There is no extra work needed to create such table, since it can be obtained as a by-product of a rule discovery algorithm (e.g., Apriori) from the set of frequent 1-itemsets.

We implemented the redundant rule elimination operator described above as a part of the validation system. However, we would like to point out that this redundant rule elimination operator constitutes only one type of such operator, and that other types of such operators

based on ideas presented in [AY98, BAG99, BA99, LHM99] can also be used in the rule validation process.

4.4 Other Validation Operators

Although rule grouping and filtering proved to be the most useful and frequently used validation operators as is demonstrated in Section 6, they can be complemented with various other validation operators. We briefly describe some of these operators below.

- **Visualization Operators**. Allow the expert to view the set of unvalidated rules or various parts of this set in different visual representations (histograms, pie charts, etc.) and can give the expert insights into what rules are acceptable and can be included in profiles.

- **Statistical Analysis Operators**. Statistical analysis operators can compute various statistical characteristics (value frequencies, attribute correlation, etc.) of unvalidated rules. This allows the expert to have many different "views" of these rules, therefore helping him or her during the rule validation process.

- **Browsing Operators**. As mentioned above, visualization and statistical analysis operators allow the expert to have "aggregated" views of the unvalidated rules through various visual representations and statistical characteristics. Browsing operators, on the other hand, can help the expert to inspect individual rules directly.

  Browsing operators are especially useful when combined with the similarity-based grouping operator described in Section 4.1. Instead of browsing through individual rules and manually validating (accepting or rejecting) them on the one-by-one basis, the expert can apply the grouping operator and then browse the resulting groups (aggregated rules) and manually validate the selected groups.

  Browsing operators can have some additional capabilities, such as being able to sort the content to be browsed in various ways. For example, it might be helpful for the expert to be able to sort rules by the user ID or by some interestingness measure, sort groups by their size, etc.

5 Incremental Profiling

In most e-commerce applications user transactional histories usually change over time since users continue their browsing and purchasing activities. Therefore, user behavioral profiles usually change over time, and there is a need to keep these profiles current by removing behavioral rules that are no longer valid and adding new rules that characterize user's emerging behaviors.

A straightforward approach to maintaining user profiles would be to rebuild them periodically "from scratch." However, this is, clearly, a very computationally intensive and time consuming process, especially since profiles often do not change significantly with new data.

An alternative approach would be to develop efficient *incremental* profile construction techniques that would adjust user profiles based on the new data without rebuilding them from scratch. One way to accomplish this would be to keep track of the sequence of all the validation operations $\{O_i\}_{i=1..N}$ that were performed during the initial profile validation process. Then, when new incremental data ΔD is added to the initial dataset D, the previously used data mining algorithm can be applied to the dataset $D \cup \Delta D$ to discover all the new rules R_{new}. After that, each of the previously used validation operators O_i can be applied to the set of rules R_{new} in the *same* sequence as they were applied during the initial validation process. We would like to point out that this technique provides for *automatic* incremental validation of user profiles without any additional participation of the domain expert (until he or she decides to revisit the sequence of validation decisions).

Moreover, this incremental validation method can be improved further by using one of the existing incremental rule discovery techniques [CHNW96, FAAM97, TBAR97] instead of using the "from-scratch" rule discovery method considered before. Data monitoring triggers, such as the ones proposed in [TS96, AT97], can also be used for this purpose.

6 Case Study

We implemented the methods presented in Sections 3 and 4 in the 1:1Pro system.[5] The 1:1Pro system takes as inputs the factual and transactional data stored in a database and generates a set of validated rules capturing personal behaviors of individual users following the approach presented in Section 3 and illustrated in Figure 1. The 1:1Pro system can use

[5] 1:1Pro stands for One-to-One Profiling System.

Validation operator	Number of rules: accepted	rejected	unvalidated
1. Redund. elimination	0	186,727	836,085
2. Filtering	0	290,427	545,658
3. Filtering	0	268,157	277,501
4. Filtering	6,711	0	270,790
5. Filtering	0	233,013	37,777
6. Grouping (1,046 gr.)	16,047	1,944	19,786
7. Grouping (6,425 gr.)	4,120	863	14,803
Final:	26,878	981,131	14,803

Figure 8: Example of a validation process for a marketing application: promotion sensitivity analysis.

any relational DBMS to store user data and various data mining tools for discovering rules describing personal behaviors of users. In addition, 1:1Pro can incorporate various other tools that can be useful in the rule validation process, such as visualization and statistical analysis tools as mentioned in Section 4.

The current implementation of 1:1Pro uses association rules to represent behaviors of individual users. However, as pointed out before, our methods can support other types of conjunctive rules. Also, the current implementation of 1:1Pro supports similarity-based grouping, template-based filtering, redundant rule elimination, and browsing operators.

We tested 1:1Pro on a "real-life" marketing application that analyzes the purchasing behavior of customers. The application included data on 1903 households that purchased different types of beverages over a period of one year. The data set contained 21 fields characterizing purchasing transactions, including the information about the time of purchase, product purchased, amount spent, coupons used, and related advertisements seen. The whole data set contained 353,421 records (on average 186 records per household). The data mining module of 1:1Pro executed a rule discovery algorithm on the individual household data for *each* of the 1903 households and generated 1,022,812 association rules in total, on average about 537 rules per household. Minimal values for the rule support and confidence were set at 20% and 50%, respectively.

Three case studies of user profile validation were performed for this application. In the first case study, we performed promotion sensitivity analysis, i.e., analysis of customer responses to various types of promotions, including advertisements, coupons, and various types of discounts. As a part of this application, we wanted to construct customer profiles that reflect different types of individual customer behaviors related to promotional activities.

Since we are very familiar with this application, we assumed the role of the domain experts. In the second case study, we performed seasonality analysis, i.e., we constructed customer profiles that contain individual rules describing seasonality-related behaviors of customers, such as the types of products that a customer buys under specific temporal circumstances (e.g., only in winter, only on weekends) and the temporal circumstances under which a customer purchases specific products. In the third case study, we asked a marketing expert to perform the seasonality analysis from her point of view. To illustrate the validation process, we describe the first case study in detail below. We also report the results from the other two case studies in this section.

As mentioned above, we performed the role of experts in the promotion sensitivity analysis and validated the 1,022,812 discovered rules ourselves using the sequence of validation operators presented in Figure 8. As shown in Figure 8, we first applied the redundant rule elimination operator that examined the heads of all the rules and removed those rules whose heads by themselves are "implied" by the data in the sense explained in Section 4.3. It turned out that this operator rejected about 18% from the set of all the discovered rules, namely 186,727. Then we applied the filtering operator (operator 2 in Figure 8) that rejects all the rules with household demographics-related information in their heads. As a result of this filtering operation, the number of unvalidated rules was reduced from 836,085 to 545,658. After that, we applied several additional filtering operators (operators 3, 4 and 5 in Figure 8). One of them (operator 3) rejected rules where either body or head contains only the market research company-specific attributes without any other information. Another filtering operator (operator 4) accepted rules that state direct relationship between kinds of products purchased and various promotions, i.e., rules that have product information (possibly among other attributes) in the body and promotion-related information (discount, sale, coupon used, or advertisement seen) in the head. Another filtering operator (operator 5) rejected all the rules that do not have any promotion-related information in the body as well as in the head of the rule. By applying all these filtering operators, we reduced the number of unvalidated rules to 37,777. Then we applied two grouping operators, using the attribute hierarchy, a fragment of which is presented in Figure 9. First, we applied grouping operator using the cut presented in Figure 9(a) to get fewer, but more aggregated (therefore, less descriptive) groups (operator 6 in Figure 8). This operator grouped the remaining 37,777 unvalidated rules into 1,046 groups, where the biggest group contained 2,364 rules and the

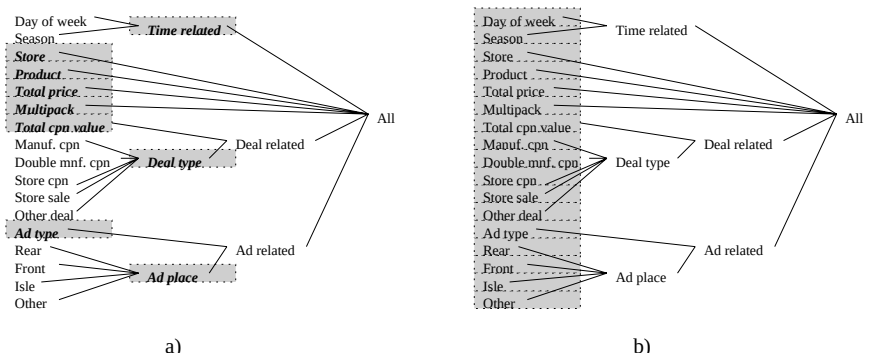

Figure 9: Fragment of an attribute hierarchy used in a marketing application.

smallest group had just 1 rule in it. We inspected the 50 biggest groups and were able to validate 38 of them (31 accepted and 7 rejected), which brought the unvalidated rule count down to 19,786. We were unable to decide on whether to accept or reject the remaining 12 groups (out of 50) and left them as "undecided" for further analysis. Finally, we applied another grouping operator (operator 7) to the remaining unvalidated rules using the cut presented in Figure 9(b). We obtained 6,425 groups. The biggest group had 237 rules but about 80% of groups contained 5 rules or less. Again, we inspected 50 biggest groups and validated 47 of them (34 accepted and 13 rejected). As the result, we validated 4,983 more rules.

We stopped the validation process at this point because there were no large groups that we could validate as a whole and it started taking us more and more time to validate smaller and less "understandable" groups. The whole validation process, including expert and computing time, took about 1.5 hours,[6] during which we validated 98.5% of the initially discovered rules (only 14,803 rules out of 1,022,812 remained unvalidated). The total number of accepted and rejected rules constituted 2.6% and 95.9% respectively of the initially discovered rules. The total number of rules accepted and put into profiles was 26,878 (on average, about 14 rules per household profile).

We performed the validation process described above on all the 1,022,812 rules generated by the rule discovery algorithm. Alternatively, we could have specified constraints, for example, using the methods proposed by [SVA97] or [BAG99], on the types of rules that we are interested in prior to the data mining stage. As a result of this, fewer data mining

[6]This time includes several minutes of computing time and the remainder constitutes the time for the expert to browse through the rules, think, and decide on the validation operators to be applied. This time does not include rule discovery and the construction of attribute hierarchies.

rules would have been generated, and there would have been no need to apply some of the elimination filters described in this case study. For example, we could have specified the constraints corresponding to the validation operators (1) and (2) in Figure 8 before applying a rule discovery algorithm. As a result, we would have generated only 545,658 rules, all of them satisfying these two conditions, and there would have been no need to apply validation operators (1) and (2) in the post-analysis stage. Although very useful, the constraint specification approach cannot replace rule validation in the post-analysis stage of the knowledge discovery process. We will elaborate on this further in Section 7.

In addition to the analysis of customer responses to promotions described in detail above, we used the same set of discovered rules to perform another related market research task – seasonality analysis. In particular, in the second case study, we constructed customer profiles that contain individual rules describing seasonality-related behaviors of customers, such as the types of products that a customer buys under specific temporal circumstances. It took us about one hour to perform this task. As the result, we validated 97.2% of the 1,022,812 discovered rules, where 40,650 rules were accepted and 953,506 rules were rejected.

For the third case study, we asked a marketing analyst to perform seasonality analysis with 1:1Pro. She started the analysis with applying redundant rule elimination and several template-based filtering rejection operators to the rules (e.g., reject all the rules that are not referring to the *Season* or the *DayOfWeek* attributes). After that, she grouped the remaining unvalidated rules, examined several resulting groups, and then stopped the validation process. At that point, she felt that there is nothing more to reject and decided to accept all the remaining unvalidated rules.[7] As a result, she accepted 42,496 rules (4.2% of all the discovered rules) and spent about 40 minutes on the whole validation process.

The results of all the three case studies are summarized in Figure 10.

We received the following feedback from the marketing expert at the end of the validation process. First, she liked the flexibility of 1:1Pro and the ability to apply a variety of validation operators in the analysis. In particular, she liked our grouping and filtering operators, but felt that we should provide better ways for presenting results, including certain visualization capabilities. Second, we observed that her validation "style" was to keep rejecting groups of irrelevant rules and accept all the remaining rules when there was nothing left to reject further. Such style can be explained by the fact that the expert was only marginally familiar

[7]Although she accepted all the remaining rules, we personally felt that if she continued the validation process she could have found some more "bad" rules.

Number of rules	Case Study I		Case Study II		Case Study III	
Rejected	981,131	(95.9%)	953,506	(93.2%)	980,316	(95.8%)
Accepted	26,878	(2.6%)	40,650	(4.0%)	42,496	(4.2%)
Unvalidated	14,803	(1.5%)	28,656	(2.8%)	0	(0.0%)

Figure 10: Summary of case studies.

with 1:1Pro and did not utilize fully its capabilities to reject and accept groups of rules in an interleaving manner. Third, we discussed the issue of the "quality" of the validated rules. The marketing expert felt that the rule evaluation process is inherently subjective because different marketing experts have different opinions, experiences, understanding the specifics of the application, etc. Therefore, she believed that different marketing experts would arrive at different evaluation results using the validation process described in this paper because of the various biases that they have.

7 Discussion

The experiments performed on a medium-size problem (1903 households, 21 fields, and 1,022,812 discovered rules) reported in the previous section produced encouraging results: based on the first case study, we managed to validate 98.5% of 1,022,812 rules in only 1.5 hours of inspection time. The results of this and other case studies produce several important observations and raise several questions.

"Quality" of generated rules. One of the central questions is how "good" the profiles are that were generated by the domain expert. In other words, would it be possible for the domain expert to discard "good" and retain "bad" rules in the user profiles during the validation process. As was pointed out in Section 1, the terms "good" and "bad" can take different meanings, such as statistical validity, acceptability by an expert, and effectiveness. Generating statistically valid rules is the prerogative of data mining algorithms and objective interestingness metrics (as described in Section 4.3) that can be applied to the discovered rules in the post-analysis stage. The problem of validating the rules by an expert was considered in this paper. As was pointed out in Section 6, there is no single objectively "correct" set of validated rules that the expert should be able to discover because different experts have different evaluation biases. One possible approach lies in assigning a certain metric to the rules and then measuring the quality of validated rules according to this metric. For example, in the context of recommender systems, one can measure the quality

of discovered and validated rules in terms of the quality of recommendations that these rules generate.[8] However, this approach deals with the rule effectiveness issues. As pointed out in Section 1, the problem of generating effective rules has not been addressed in this paper and is left as a topic of future research.

Scalability. Our experimental results demonstrate that 1:1Pro can handle medium-size problems well. An interesting question is how well our approach would scale up to large problems having millions of users and dozens of attributes. If the number of attributes increases, then the rule mining methods, such as Apriori, will generate exponentially larger number of rules and would constitute a bottleneck of the profile generating process (rather than the rule validation phase). If the number of attributes is fixed and the number of users grows, then an application of validation operators should scale up linearly with the total number of users. This is the case, because, as demonstrated in Section 4, validation operators run in time linear in the total size of the rules, and we observed that the number of discovered rules grows linearly with the number of users.[9]

Constraint-based rule generation vs. post-analysis. In our experiments we applied a rule discovery algorithm to generate *all* the association rules for pre-specified confidence and support levels and then applied several filtering operators to remove "uninteresting" rules from this set (e.g., as shown in Figure 8). Alternatively, we could have applied a constraint-based version of association rule discovery methods, such as the ones presented in [SVA97, BAG99]. As a result, we could have obtained the number of rules smaller than 1,022,812 produced by the unconstrained rule discovery algorithm.

Although the constraint-based approach reported in [SVA97, BAG99] provides a partial solution to the validation problem by reducing the total number of rules generated during the initial data mining stage, it does not provide the complete solution for the following reason. It is very hard to figure out *all* the relevant constraints *before* the data mining algorithms are launched. The human expert, most likely, will be able to come up with many important filters only after inspecting data mining results using browsing, grouping, or visualization operators. Alternatively, an expert can make a mistake and specify a filter that happens to be too strict (i.e., rejects too many rules). If such constraint was specified before mining, the whole rule discovery algorithm would have to be reexecuted with the correct constraint,

[8] In fact, we have started looking into this issue in [TA99] and are planning to conduct this research by using recommender systems and judging the quality of profiles via the quality of resulting recommendations.

[9] Although we have not conducted rigorous experiments to prove this point.

which is more computationally expensive than to reexecute a correct filtering operator in the post-analysis phase. The benefits of iterative analysis of data mining results are also pointed out by several researchers, including [FPSS96, ST96a, PJ98, LBA98, Sah99].

Therefore, neither the post-analysis nor the pre-specification of constraints works best as a stand-alone method, and the two approaches should be combined into one integral method. The main question pertaining to this combination is what kinds of constraints should be pre-specified by the user for the rule generation phase and what functionality should be left for the post-analysis phase. This topic was addressed by several researchers within the rule discovery context [PJ98, GVdB99]. We are currently working on extending this line of work to the personalization problem.

Examination of groups of rules. One of the main features of our approach is the ability for the domain expert to examine groups of rules and to decide whether to accept or reject a group as a whole. One of the concerns for such method is that the domain expert can make mistakes by accepting "bad" and rejecting "good" rules. This issue is addressed in 1:1Pro by providing the capability for the domain expert to evaluate a group of rules *recursively* in case the expert is unable to decide whether or not to accept or reject this group as a whole. In other words, the expert can apply validation operators just to this particular group of rules and examine its subgroups. By examining smaller subgroups, the expert can then make more reliable decisions.

Future research. This paper opens several directions for future work. One of such directions includes studies of measures of effectiveness of discovered rules and development of efficient algorithms for discovering such rules. Moreover, the marketing expert pointed to us that some additional validation operators should be added to our system, and we plan to work on this issue. Finally, we plan to study tradeoffs between constraint-based generation and post-analysis of rules in the context of personalization applications.

References

[AIS93] R. Agrawal, T. Imielinsky, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference*, pages 207–216, 1993.

[AKY98] C. Allen, D. Kania, and B. Yaeckel. *Internet World Guide to One-to-One Web Marketing.* John Wiley & Sons, 1998.

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, chapter 12. AAAI Press, 1996.

[ASY98] C. C. Aggarwal, Z. Sun, and P. S. Yu. Online generation of profile association rules. In *Proc. of the Fourth Int'l Conference on Knowledge Discovery and Data Mining*, August 1998.

[AT97] G. Adomavicius and A. Tuzhilin. Discovery of actionable patterns in databases: The action hierarchy approach. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, August 1997.

[AT99] G. Adomavicius and A. Tuzhilin. User profiling in personalization applications through rule discovery and validation. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 1999.

[AY98] C. C. Aggarwal and P. S. Yu. Online generation of association rules. In *Proceedings of the Fourteenth International Conference on Data Engineering*, 1998.

[BA96] 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.

[BA99] R. J. Bayardo and R. Agrawal. Mining the most interesting rules. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 1999.

[BAG99] R. J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. In *Proceedings of the 15th International Conference on Data Engineering*, March 1999.

[Bau99] P. Baudisch, editor. *CHI'99 Workshop: Interacting with Recommender Systems*, May 1999. http://www.darmstadt.gmd.de/rec99/.

[BFOS84] L. Breiman, J. H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth Publishers, 1984.

[BKK97] C. Brunk, J. Kelly, and R. Kohavi. Mineset: An integrated system for data mining. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, August 1997.

[BMUT97] S. Brin, R. Motwani, J.D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the ACM SIGMOD Conference*, 1997.

[CAC97] *Communications of the ACM*, 40(3):56–89, 1997. Special issue on Recommender Systems.

[Cha99] P. K. Chan. A non-invasive learning approach to building web user profiles. In *Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*, August 1999.

[CHNW96] D. Cheung, J. Han, V. Ng, and C. Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proceedings of 1996 International Conference on Data Engineering*. IEEE Computer Society, 1996.

[CP90] S. Clearwater and F. Provost. Rl4: A tool for knowledge-based induction. In *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, 1990.

[DT93] V. Dhar and A. Tuzhilin. Abstract-driven pattern discovery in databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6), 1993.

[FAAM97] R. Feldman, Y. Aumann, A. Amir, and H. Mannila. Efficient algorithms for discovering frequent sets in incremental databases. In *Proceedings of the Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'97)*, May 1997.

[FMMT96] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. In *Proceedings of the 1996 ACM SIGMOD International Conference on the Management Of Data*, pages 13–23, 1996.

[FP96] T. Fawcett and F. Provost. Combining data mining and machine learning for efficient user profiling. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, August 1996.

[FP97] T. Fawcett and F. Provost. Adaptive fraud detection. *Journal of Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

[FPSS96] 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.

[GVdB99] B. Goethals and J. Van den Bussche. A priori versus a posteriori filtering of association rules. In *Proceedings of the 1999 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 1999.

[Hag99] J. Hagel. Keynote address at the personalization summit. San Francisco. November 16, 1999.

[HFW+96] 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.

[HS99] J. Hagel and M. Singer. *Net Worth: Shaping markets when customers make the rules.* Harvard Business School Press, 1999.

[IV99] T. Imielinski and A. Virmani. MSQL: A query language for database mining. *Journal of Data Mining and Knowledge Discovery*, 3(4), 1999.

[Kau98] H. Kautz, editor. *Recommender Systems. Papers from 1998 Workshop. Technical Report WS-98-08.* AAAI Press, 1998.

[KMR+94] 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.

[LBA98] Y. Lee, B. G. Buchanan, and J. M. Aronis. Knowledge-based learning in exploratory science: learning rules to predict rodent carcinogenicity. *Machine Learning*, 30:217–240, 1998.

[LH96] B. Liu and W. Hsu. Post-analysis of learned rules. In *Proceedings of the AAAI Conference*, pages 828–834, 1996.

[LHC97] B. Liu, W. Hsu, and S. Chen. Using general impressions to analyze discovered classification rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, August 1997.

[LHM99] B. Liu, W. Hsu, and Y. Ma. Pruning and summarizing the discovered associations. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 1999.

[LSW97] B. Lent, A. N. Swami, and J. Widom. Clustering association rules. In *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K*, pages 220–231. IEEE Computer Society, 1997.

[MFM+98] Y. Morimoto, T. Fukuda, H. Matsuzawa, T. Tokuyama, and K. Yoda. Algorithms for mining association rules for binary segmentations of huge categorical databases. In *Proceedings of the 24th VLDB Conference*, pages 380–391, 1998.

[Mor98] S. Morishita. On classification and regression. In *Proceedings of the First International Conference on Discovery Science*, 1998.

[MPC98] R. Meo, G. Psaila, and S. Ceri. An extension to SQL for mining association rules. *Journal of Data Mining and Knowledge Discovery*, 2(2):195–224, 1998.

[Per99] Personalization summit. San Francisco. November 14-16, 1999.

[PJ98] F. Provost and D. Jensen. Evaluating knowledge discovery and data mining. In *Tutorial for the Fourth International Conference on Knowledge Discovery and Data Mining*, August 1998.

[PR93] D. Peppers and M. Rogers. *The One-to-One Future*. Doubleday, 1993.

[PSM94] G. Piatetsky-Shapiro and C. J. Matheus. The interestingness of deviations. In *Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases*, 1994.

[PT98] B. Padmanabhan and A. Tuzhilin. A belief-driven method for discovering unexpected patterns. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, August 1998.

[PT99] B. Padmanabhan and A. Tuzhilin. Unexpectedness as a measure of interestingness in knowledge discovery. *Decision Support Systems*, 27(3):303–318, 1999.

[Qui93] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[SA95] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proceedings of the 21st International Conference on Very Large Databases*, September 1995.

[Sah99] S. Sahar. Interestingness via what is not interesting. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 1999.

[SNP99] I. Soboroff, C. Nicholas, and M. J. Pazzani, editors. *ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation*, August 1999. http://www.cs.umbc.edu/~ian/sigir99-rec/.

[SOMZ96] 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.

[Sri96] R. Srikant. *Fast Algorithms for Mining Association Rules and Sequential Patterns*. PhD thesis, University of Wisconsin, Madison, 1996.

[ST96a] 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.

[ST96b] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(6), December 1996.

[Ste97] C. Stedman. Data mining for fool's gold. *Computerworld*, 31(48), 1997.

[Suz97] E. Suzuki. Autonomous discovery of reliable exception rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, August 1997.

[SVA97] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, August 1997.

[TA99] A. Tuzhilin and G. Adomavicius. Integrating user behavior and collaborative methods in recommender systems. In *CHI'99 Workshop. Interacting with Recommender Systems*, May 1999.

[TBAR97] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka. An efficient algorithm for the incremental updation of association rules in large databases. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, August 1997.

[TKR+95] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila. Pruning and grouping discovered association rules. In *ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, April 1995.

[TS96] A. Tuzhilin and A. Silberschatz. A belief-driven discovery framework based on data monitoring and triggering. Technical Report IS-96-26, Stern School of Business, New York University, December 1996.

[WTL98] K. Wang, S. H. W. Tay, and B. Liu. Interestingness-based interval merger for numeric association rules. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, August 1998.