Mining for Strong Negative Associations in a Large Database of Customer Transactions

Ashok Savasere* Edward Omiecinski Shamkant Navathe
College of Computing, Georgia Institute of Technology, Atlanta GA 30332
{*ashok,edwardo,sham*}*@cc.gatech.edu*

Abstract

Mining for association rules is considered an important data mining problem. Many different variations of this problem have been described in the literature. In this paper we introduce the problem of mining for negative associations. A naive approach to finding negative associations leads to a very large number of rules with low interest measures. We address this problem by combining previously discovered positive associations with domain knowledge to constrain the search space such that fewer but more interesting negative rules are mined. We describe an algorithm that efficiently finds all such negative associations and present the experimental results.

1 Introduction

Wide spread use of computers in business operations and the availability of cheap storage devices have led to an explosive growth in the amount of data gathered and stored by most business organizations today. There has been a trend in recent years to search for interesting patterns in the data and use them for improved decision making. (e.g., see [3]). Data mining is defined as the process of finding hidden, nontrivial and previously unknown information from a large collection of data [10]. It has been recognized as one of the promising areas of research encompassing databases, statistics and machine learning [6, 12, 15].

Recently, there has been considerable interest in finding associations between items in a database of customer transactions, such as the sales data collected at super market check out counters [1, 2, 5, 7, 11, 8, 14, 4]. Association rules identify items that are most often bought along with certain other items by a significant fraction of the customers. For example, we may find that "95 % of the customers who bought bread also bought milk." Every rule must satisfy two user specified constraints: one is a measure of statistical significance called *support* and the other a measure of goodness of the rule called *confidence*. The support constraint ensures that the rule occurs relatively often to be considered useful. The confidence measures how well the rule predicts the association between the items. The support of a rule $X \Longrightarrow Y$ is defined as the fraction of transactions that contain $X \cup Y$, where X and Y are sets of items. The confidence is defined as the ratio $support(X \cup Y)/support(X)$. The goal is to find all rules that satisfy minimum support and minimum confidence. These types of rules specify the likelihood of finding Y in a customer basket given that it contains X. Applications of such rules include cross-marketing, attached mailing, catalog design, add-on sales, store layout, etc., [3].

In this paper we consider the complementary problem, i.e., what items a customer is not likely to buy given that he buys a certain set of items. We call these types of rules as the *negative association rules*. An example of such a rule is "60 % of the customers who buy potato chips do not buy bottled water." Such negative association rules can provide valuable information about customer buying patterns and help managers in devising better marketing strategies. In this paper, we consider negative associations in the context of retail sales data. However, the solutions developed here can be applied to other domains. To the best of our knowledge, we are unaware of any work that addresses the issue of mining negative rules from data. In this paper we discuss the difficulties of mining negative rules and describe an algorithm which efficiently mines negative rules in large datasets.

Finding negative associations is not straight forward due to the following reason: in a large retail store, there may be tens of thousands of items. Even if billions of customer transactions are considered, many of the item combinations may not appear even once. For example, if there are 50,000 items the possible combinations of items is $2^{50,000}$ a majority of which will not appear even once in the entire database. If the absence of a certain item combination is taken to mean negative association, then we can generate millions of negative association rules. However, most of these rules are likely to be extremely uninteresting. The problem is therefore one of finding only *interesting* negative rules. We call such rules *strong* negative rules. In the following section we define the problem more precisely and then discuss the notion of interestingness and the type of negative rules that fall within this definition.

1.1 Measure of Interestingness

The objective measure of interestingness of a rule is defined in terms of the "unexpectedness" of the rule[1]

*Current address: Data Mining Group, Tandem computers Inc., 14231 Tandem Blvd., Austin, TX 78728

[1]This is only one measure of interestingness. There may be other factors which make a particular rule intersting.

Simply stated, a rule is interesting if it contradicts or deviates significantly from our expectation based on previous belief. The previous belief is usually stated in terms of the a priori probabilities based on our knowledge of the problem domain. For example, in the retail marketing context, suppose there are 50,000 distinct items, and there are 10 million transactions each containing 5 items on an average. Without any prior knowledge we would expect all items are equally likely to be bought. Then the number of transactions containing a specific item is 1000. Now however, after scanning the customer transactions, if we find that 500,000 transactions contain that particular item, we say that we have discovered an interesting fact because it significantly deviates from our earlier expectation. In information theoretic terms the a priori probabilities represent our state of *ignorance* and the deviation of the a posteriori probabilities represent the degree of information gained.

In the case of negative rules we are interested in finding itemsets that have a very low probability of being bought with certain other items. That is, we are interested in cases where two specific sets of items appear very rarely in the same transaction. However, this poses two problems as follows.

1. In the absence of any knowledge about customer buying preferences, we expect the items to be bought independently of each other. In the above example the expected support for a specific pair of items is 1000/10 million $\times$ 1000/10 million = 1/100,000,000. Even if the actual support turns out to be zero, the deviation from expectation is extremely small. Consequently, a rule which states that these two items are negatively associated does not carry much information and hence uninteresting.

2. If we are looking for item combinations which have very low support, there will be a very large number of combinations having very low or even zero support. In the above example, even if we take only pairs of items there are approximately 2.5 billion combinations. Since we have only 10 million transactions, most of the combinations will not appear even once. Therefore we may generate billions of negative rules most of which will be useless. The problem gets even worse if we include combinations of larger sizes.

To summarize, mining for negative rules is impossible with a naive approach due to the following reasons: (a) We may find billions of negative associations in a large dataset; and (b) Almost all of these negative associations will be extremely uninteresting.

Even though the preceding argument shows that negative associations may be inherently uninteresting, there are many situations where this is not true. We motivate this with the following examples:

Example 1: Suppose we consider the sales of a particular brand of chips, say Ruffles and two brands of soft drinks, say Coke and Pepsi[2]. After scanning through a large number of transactions suppose we find that when customers buy Ruffles they also usually buy Coke but not Pepsi. We can then conclude that Ruffles has an interesting negative association with Pepsi. The reason this is interesting is that by considering only the associations between Ruffles and Coke, we *expected* the association between Ruffles and Pepsi also to be high. In other words, we reformulated our belief from one of independence to a positive association based on (a) our knowledge that Coke and Pepsi fall under the same "category" and therefore they can be expected to have similar types of associations with other items and (b) the positive association between Ruffles and Coke.

Example 2: After scanning the data suppose we find a strong positive association between the frozen yogurt category and the bottled water category. Suppose each of these categories contains two individual brands, say brands Bryers and Healthy Choice of frozen yogurt and brands Evian and Perrier of bottled water. For simplicity assume that the sales of each individual brand accounts for 50% of the sales of its category. Based on the association between the two categories we can reasonably expect each of the four combinations of frozen yogurt and bottled water brands to account for 25% of the support for {frozen yogurt, bottled water}. However, if we actually find the support for {Bryers, Perrier} accounts for only 2%, we can conclude that Bryers and Perrier are negatively associated.

Example 3: In the above example, we can also expect the support for {Frozen yogurt, Perrier} to be 50% of the support for {frozen yogurt, bottled water}. However, suppose we actually find this support to be 10%, we can deduce that most customers must be buying Evian when they buy either brands of frozen yogurt which indicates an interesting negative association between the frozen yogurt category and Perrier.

The preceding examples shows that we can indeed find certain interesting negative associations. In each of these examples we based our conclusion on the information already present in the data and additional domain knowledge which groups similar items together.

We use these ideas to mine for a class of negative rules which are interesting and likely to be useful. The basic idea behind our approach is to look at only those cases where we expect a high degree of positive association. If the actual support is found to be significantly smaller then we can conclude that they have a negative association. We make the following observations with respect to this approach.

1. **Domain knowledge.** The type of domain knowledge we depend on to mine the negative rules is a grouping of similar items. This information is most naturally conveyed by a taxonomy on the items. In this paper we assume such a tax-

[2]The actual brand names used in this paper are for the purpose of illustration only.

onomy is available. In most retail organizations, items are already classified under departments, categories, sub-categories, etc.

2. **Uniformity assumption.** One of the fundamental assumptions in our approach is that the items that belong to the same parent in a taxonomy are expected to have similar types of associations with other items. This is similar to the principle of uniformity of nature found in inductive reasoning.

3. **Class of negative rules.** Since we consider only those cases where an expected support can be computed based on the taxonomy, the class of negative rules we can generate is not all possible negative rules but a subset of those rules. If additional domain knowledge is incorporated, it may be possible to mine additional types of negative rules.

It should be noted that our approach solves the two problems described earlier. That is, we generate fewer negative rules and all those rules are likely to be interesting.

1.2 Previous Work

The problem of mining for association rules was introduced in [1]. Since then it has been recognized as one of the important types of data mining problems and many algorithms have been proposed to improve its performance, for e.g., [2, 7, 8, 11]. Related work also includes [9, 13]. In [14] the problem was extended to include taxonomies on the items and mining rules between different levels in the taxonomy. This enables more expressive types of rules to be mined by allowing a limited form of disjunction in the association rules, e.g., "(overcoats OR jackets) AND shoes $\Longrightarrow$ perfume." However, we are unaware of any work that tries to find negative associations in data. The closest work is the technique to prune uninteresting rules proposed in [14].

2 Problem Statement

Formally, the problem can be stated as follows: Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of m distinct literals called *items*[3]. Let $\mathcal{T}$ be a taxonomy on $\mathcal{I}$. Let $\mathcal{L} \subseteq \mathcal{I}$ be the set of leaf items in $\mathcal{T}$ and let $\mathcal{C} \subset \mathcal{I}$ be the set of internal nodes called categories. $\mathcal{D}$ is a set of variable length transactions over $\mathcal{L}$. Each transaction, T, *contains* a set of items $i_i, i_j, \ldots, i_k \subset \mathcal{L}$. A transaction also has an associated unique identifier called TID. In general, a set of items is called *itemsets*. The number of items in an itemset is called the *length* of an itemset. Itemsets of some length k are referred to as k–itemsets. For an itemset $X \cdot Y$, if Y is an m–itemset then Y is called an *m-extension* of X. An itemset $X \subset \mathcal{I}$, has *support(X)* $= s$, if the fraction of transactions in $\mathcal{D}$ containing X equals s. A *negative association rule* is an implication of the form $X \not\Longrightarrow Y$, where $X, Y \subset \mathcal{I}$, and $X \cap Y = \emptyset$. X is called the antecedent and Y is called the consequent of the rule. Every rule also has a rule interest measure. We define the interest measure *RI* of a negative association rule, $X \not\Longrightarrow Y$, as follows:

$$RI = \frac{\mathcal{E}[support(X \cup Y)] - support(X \cup Y)}{support(X)}$$

Where $\mathcal{E}$ [support(X)] is the *expected support* of an itemset X. We describe how expected support is computed in the following section. Note that the rule interest RI is negatively related to the actual support of the itemset $X \cup Y$. It is highest if the actual support is zero and zero if the actual support is same as the expected support reflecting our earlier definition of interestingness.

The problem of finding negative rules can be now be stated as follows: given a database of customer transactions $\mathcal{D}$ and a taxonomy $\mathcal{T}$ on the set of items, find all rules $X \not\Longrightarrow Y$ such that (a) support(X), and support(Y) are greater than minimum support *MinSup*; and (b) the rule interest measure is greater than *MinRI* where *MinSup* and *MinRI* are specified by the user. Condition (a) is necessary to ensure that the generated rule is statistically significant. For example, even if the rule perfectly predicts 10 out of 10 million cases, it is not very useful because it is not general enough.

The problem can be decomposed into two subtasks: (a) finding itemsets whose actual support deviates at least *MinSup* $\times$*MinRI* from their expected support. We call such itemsets as *negative itemsets* and (b) generating the negative rules after the negative itemsets are found. The first condition eliminates testing for rules which will have an RI less than *MinRI*. We consider each of these tasks in the following sections.

2.1 Finding Negative Itemsets

Finding negative itemsets involves following steps:

1. We first find all the generalized large itemsets in the data (i.e., itemsets at all levels in the taxonomy whose support is greater than the user specified minimum support).

2. We next identify the candidate negative itemsets based on the large itemsets and the taxonomy and assign them expected support.

3. In the last step, we count the actual support for the candidate itemsets and retain only the negative itemsets.

The first step of discovering generalized large itemsets has been discussed in [14, 4]. The second step is the most important step and is discussed in the next section. The last step is relatively straight forward as any data structures to count support can be used.

2.1.1 Generating Candidate Negative Itemsets

The candidate negative itemsets are generated based on the previously determined large itemsets. For each

[3] This description and the terminology used are largely based on [1] and [2].

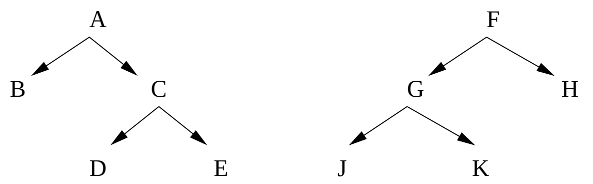

Figure 1: Taxonomy

large itemset l_k, we generate candidate itemsets composed of the immediate descendents and siblings of the items in l_k. The candidate itemsets will also be of size k. In general any itemset whose expected support can be computed based on the support of the large itemsets and whose computed expected support is greater than $MinSup \times MinRI$ is a candidate itemset.

We can identify the following cases for generating candidate itemsets. All the examples refer to the taxonomy shown in Figure 1 where {CG} has been found to be large. We show only pairs of items in these examples. However, they can be extended to any number of items. It should be noted that all the 1-itemsets in a candidate must have minimum support. Otherwise no rule will be produced for this itemset because both antecedent and the consequent of a rule must have minimum support. Therefore if all 1-itemsets of a candidate are not large, that candidate is immediately rejected.

Case 1: Here, the candidates are formed from the immediate children of large itemsets. For example, the candidates in this case will be {D J}, {D K}, ... , {E J}, {E K}, ... , etc. The expected supports are computed from the support of their parents. For example, the expected support of {D J} is computed as

$$\mathcal{E}[sup(DJ)] = \frac{sup(CG) \times sup(D) \times sup(J)}{sup(C) \times sup(G)}$$

In general, if $\{\hat{p}, \hat{q}, \ldots, \hat{t}\}$ is a large itemset,

$$\mathcal{E}[sup(p, q, \ldots, t)] = \frac{sup(\hat{p} \cup \hat{q} \cup \cdots \cup \hat{t}) \times sup(p) \times sup(q) \times \cdots \times sup(t)}{sup(\hat{p}) \times sup(\hat{q}) \times \cdots \times sup(\hat{t})}$$

where $\hat{p}, \hat{q}, \ldots, \hat{t}$ are parents of $p, q, \ldots, t$, respectively. The candidate itemsets are formed by taking every possible combination of the children of the items in the large itemset.

Case 2: The candidate itemsets are {C J}, {C K}, ... , {G D}, {G E}, ... , etc. The expected support for, say, {C J} is computed as

$$\mathcal{E}[sup(CJ)] = \frac{sup(CG) \times sup(J)}{sup(G)}$$

In general,

$$\mathcal{E}[sup(p, q, r, \ldots, t)] = \frac{sup(p \cup q \cup \hat{r} \cup \cdots \cup \hat{t}) \times sup(r) \times \cdots \times sup(t)}{sup(\hat{r}) \times \cdots \times sup(\hat{t})}$$

where $\hat{p}, \hat{q}, \ldots, \hat{t}$ are parents of $p, q, \ldots, t$, respectively. The candidate itemsets are generated as all combinations of the children of the items in the large itemset.

Case 3: In this case, the candidates are formed from the siblings of the items in the large itemset. For the above example, the candidates are {C H}, {C I}, and {B G}. The expected support for {C H} is given by

$$\mathcal{E}[sup(CH)] = \frac{sup(CG) \times sup(H)}{sup(G)}$$

In general,

$$\mathcal{E}[sup(p, q, r, \ldots, t)] = \frac{sup(p \cup q \cup r' \cup \cdots \cup t') \times sup(r) \times \cdots \times sup(t)}{sup(r') \times \cdots \times sup(t')}$$

where $p', q', \ldots, r'$ are siblings of $p, q, \ldots, t$, respectively.

It is possible that same candidates generated from different large itemsets. For example, we may generate the candidate {C H} from the large itemset {C G} according to case 3. Now, if the itemset {A F} is also large, then the candidate {C H} will be generated according to case 1. This may result in different values of expected support for the same candidate depending on how it is generated. In such situations the largest value of the expected support is chosen.

The main idea behind generating candidate negative itemsets is that the items close together in the taxonomy will have similar associations with other items. However, the candidates are generated only if we can assign an expected value of support based on the supports of other large itemsets. Therefore, even though we can identify many more cases where itemsets can be formed from the neighbors of large itemsets, they are not considered as candidates. For example, the following itemset combinations are not considered as candidates.

1. itemsets containing only the siblings of the items in the large itemset.
2. itemsets containing the immediate ancestors and immediate children of the items in the large itemset.
3. itemsets containing the immediate ancestors and siblings of the items in the large itemset.
4. itemsets containing immediate descendents and siblings of the items in the large itemset.

Items	*Support*
Bryers	20,000
Healthy Choice	10,000
Evian	10,000
Perrier	5,000
Frozen yogurt	30,000
Bottled water	20,000
Frozen Yogurt and Bottled water	15,000

Table 1: Supports for the example

Items	*Expected Support*	*Actual Support*
Bryers and Evian	6,000	7,500
Bryers and Perrier	4,000	800
Healthy Choice and Evian	3,000	4,200
Healthy Choice and Perrier	2,000	2,500

Table 2: Expected and actual supports

Example We illustrate the candidate and negative itemset generation using the an example. Figure 2 shows the taxonomy and Table 1 shows the supports for the individual brands (we have shown absolute values of support for simplicity). The minimum support is specified as 4,000.

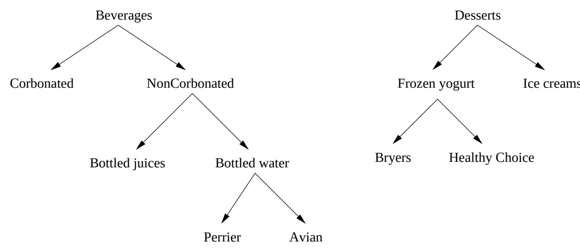

Figure 2: Taxonomy for the example

The negative candidates in this case are {Bryers, Evian}, {Bryers, Perrier}, {Healthy Choice, Evian} and {Healthy Choice, Perrier}. Their expected supports are shown in Table 2. Among these {Bryers, Evian} and {Healthy Choice, Evian} will already be found to be large. Hence they are not considered negative candidates.

If the actual supports are as shown in Table 2 and the minimum RI is specified as 0.5, then the only negative association rule will be Perrier $\not\Longrightarrow$ Bryers.

2.1.2 Number of Candidates

The actual number of candidates generated depends on the characteristics of the data and the taxonomy being used. However, we can estimate the number of candidates generated as

$$\sum_{1=1}^{k} \binom{k}{i} f^i + k(f-1)$$

where f is the average fan-out in the taxonomy and k is the itemset size. As can be seen, the number of candidates is exponential in the size of the candidates being considered. However, as the size is increased the number of large itemsets rapidly decreases. A large number of candidates generated can also be pruned by applying optimizations. For example, since all 1-itemsets contained in a candidate must be large, the candidates are generated by discarding all small 1-itemsets in the taxonomy. This has the effect of reducing the fanout and hence the candidates generated. Other optimizations will be discussed in Section 2.2.

2.1.3 Effect of the Taxonomy

The taxonomy is considered as the domain knowledge which is used to induce negative rules. Therefore the rules generated will be affected by the quality of the taxonomy. One of the implicit assumptions in our approach is that the items belonging to the same category are "substitute" items, i.e., customers may substitute one item in place of the other within that category. Taxonomies on a given set of items can be generated based on any arbitrary criterion. However, our approach works best if the taxonomy is generated based on the similarity of usage of the items. This also ensures that the items grouped under the same category are substitutes. Most real life taxonomies are, in general, of this nature. Therefore, the approach works satisfactorily in most situations.

Another issue that needs to be considered is the level of detail in the taxonomy. A taxonomy with finer "granularity," i.e., one in which categories are classified into smaller and smaller sub-categories leads to the generation of better rules compared to a shallow taxonomy where hundreds or thousands of items are grouped under the same category. This is to be intuitively expected because the information content of a fine granular taxonomy is more than a taxonomy of coarser granularity, leading to better domain knowledge and hence better quality of rules.

This can be explained in more practical terms as follows: we generate the negative candidate itemsets and assign them expected supports based on the supports of either their parents or siblings. In a taxonomy with finer granularity, each category will have fewer children and also fewer siblings compared to a taxonomy with coarser granularity. As the number of children or siblings in a category increases, the relative support of an individual child or sibling decreases. For example, in a category with two children the relative support of each child may be 50% on average. However, if there are 100 children the relative support will drop to 1%. Therefore, the expected supports which are computed from the relative supports of the items in a candidate will have relatively larger error terms in a taxonomy with coarser granularity leading to less accurate rules.

The second reason why fine granularity taxonomies are preferred is that the number of candidates generated increases rapidly with the increase in the average fanout. Fine granularity taxonomies alleviate this problem.

2.2 Algorithm

We assume that the transactions are in the form $\langle TID, i_j, i_k, \ldots, i_n \rangle$ and the complete taxonomy is available. Generating negative association rules involves finding all the negative large itemsets and generating the negative rules. We first consider the problem of finding negative large itemsets. As explained in section 2.1, generating negative large itemsets involves finding generalized large itemsets, generating negative candidates, counting support for the candidates and generating the negative large itemsets. To find all large itemsets, we can use one the algorithms, *Basic*, *Cumulate* or *EstMerge*, proposed in [14]. To generate the negative large itemsets we describe two algorithms below. Both the algorithms use similar approaches. The first algorithm is a straight forward implementation and the second algorithm incorporates some optimizations to improve the performance. Both the algorithms require multiple iterations.

2.2.1 Naive Algorithm

Each iteration of this algorithm consists of two phases. In the first phase of iteration k, we compute the generalized large itemsets of size k (using one of *Basic*, *Cumulate* or *EstMerge*). In the second phase, first the negative candidate itemsets of size k are generated as describe in section 2.1.1. Next, support for the candidates is counted by making a pass over the data. Therefore, this algorithm requires two passes over the data during each iteration for a total of $2 \times n$ passes, where n is the total number of iterations. The improved algorithm reduces the number of passes over the data as described below.

2.2.2 An Improved Algorithm

This algorithm incorporates two optimizations over the naive algorithm: first, all small 1-itemsets are deleted from the taxonomy; second, the instead of generating the negative itemsets during each iteration, they are generated in a single step after generating the large itemsets of all sizes. The first optimization reduces the number of negative candidates generated. The second optimization reduces the number of passes over the data from $2 \times n$ to $n+1$. The algorithm is shown in Figure 3.

L_1 = {large 1–itemsets};
$k = 2$; // k represents the pass number
// First generate all large itemsets
while $(L_{k-1} \neq \emptyset)$
begin
  // Generate new candidates of size k using Basic,
  // Cumulate or EstMerge
  C_k = GenCands (L_{k-1});
  forall transactions $t \in \mathcal{D}$
  begin
    C_t = subset(C_k, t);
    forall candidates $c \in C_t$
      c.count++;
  end
  $L_k = \{c \in C_k | c.\text{count} \geq \text{MinSup}\}$
  $k = k + 1$
end
// Now generate negative itemsets
Delete all small 1-itemsets from the taxonomy
$k = 2$; **while** $(L_k \neq \emptyset)$
begin
  // Generate negative candidates of size k as
  // described in section 2.1.1
  NC_k = GenNegCands (L_k);
  $NC = NC \cup NC_k$;
  $k = k + 1$;
end
forall transactions $t \in \mathcal{D}$
begin
  NC_t = subset(NC_k, t);
  forall candidates $c \in NC_t$
    c.count++;
end
$N_k = \{c \in NC_k | c.\text{count} < \text{MinSup} \times \text{MinRI}\}$

Figure 3: An improved algorithm

2.3 Generating Rules

Once the negative itemsets are generated finding all negative rules is straight forward. For a negative itemset n, we output the rule $a \not\Longrightarrow (n-a)$ where a, and $(n-a)$ are nonempty subsets of n having minimum support if its rule interest measure is greater than specified minimum, MinRI. Our algorithm for generating negative rules is an extension of the *apgenrules* algorithm described in [2]. The extensions handle the requirement that the antecedent and the consequents of the rule must be large. Note that if a turns out to be small none of the extensions need to be generated because they will not have the minimum support. Similarly if a rule $a \not\Longrightarrow (n-a)$ does not have minimum RI, then none of the subsets of a need to be considered because none of those rules will have minimum RI either. The rule generation algorithm is shown in Figure 4. The *apriori-gen* function is the join followed by the prune step described in [2].

2.4 Data Structures

The algorithm to generate negative association rules is very similar to generating generalized association rules with additional steps to generate negative itemsets. Since every 1-itemset of candidate must have minimum support, instead of testing every itemset for minimum support while generating candidates, it is considerably faster to compress the taxonomy by filtering out all items which do not have minimum support. Then no candidate in which one of the 1-item

forall negative itemsets n_k of size k, $k \geq 2$ **do**
  H_1 = {consequents of rules generated from n_k
    with one item in the consequent};
  call genrules (n_k, H_1, L_2, L_{k-2});
end
procedure genrules (n_k, H_m, L_{m+1}, L_{k-m-1})
// n_k: negative k-itemset, H_m: set of m-item consequents
  if ($k > m+1$) **then**
  begin
    H_{m+1} = apriori-gen (H_m);
    forall $h_{m+1} \in H_{m+1}$
      if ($h_{m+1} \in L_{m+1}$ **then**
        if (($n_k - h_{m+1}) \in L_{k-m-1}$) **then**
          $RI = (\mathcal{E}[sup(n_k)] - sup(n_k))/sup(n_k - h_{m+1})$;
          if ($RI \geq MinRI$) **then**
            output rule $(n_k - h_{m+1}) \not\Longrightarrow h_{m+1}$;
          else
            delete h_{m+1} from H_{m+1};
      else
        delete h_{m+1} from H_{m+1};
    end
    call genrules (n_k, H_{m+1});
  end

Figure 4: Rule Generation

subsets does not have minimum support will be generated. As mentioned earlier, same candidate may be generated in more than one way. Therefore all candidates are put in a hash table for fast lookup. Whenever a candidate is generated, first the hash table is checked. If the candidate is not found then the candidate is inserted in the hash table. Otherwise, the expected support for the candidate is set to larger of the two. During the rule generation process, the counts of the subsets of the negative itemset are required. All large itemsets are also placed in a hash table for fast lookup.

2.5 Memory Management

Since negative candidates of all sizes are generated at the same time and their support is tested in a single pass, it is possible that the number of candidates generated is too large to accommodate in available memory. However, if such a situation arises, the negative candidate generation process is stopped and the support for the candidates already generated is counted. The generated negative itemsets are either written back to the disk or if they are sufficiently small, are kept in the main memory. The candidate generation process is now continued. This will necessitate more than one pass over the database.

3 Experimental Results

In this section we describe the experimental results of our technique for generating negative associations. We performed the experiments using synthetic data on Sun SPARCstation 5 with 32 MB of main memory.

3.1 Synthetic Data

The synthetic data is generated such that it simulates customer buying pattern in a retail market environment. We have used the same basic method as described in [14]. However, to simulate the buying pattern more accurately we adapted the hierarchical model of consumer choice called the nested logit model to generate the data. In this model, consumers first decide on which category to buy and then decide which particular brand to buy within that category.

We first generate a taxonomy over the items. For any internal node, the number of children are picked from a Poisson distribution with mean set to F. This process is generated starting from the root level. Then at level 2 and so on until there are no more items. We next generate a set of potentially maximal large itemsets from which itemsets are assigned to a transaction. To generate the set of potentially maximal large itemsets, we first generate potentially maximal clusters of categories comprising of items one level above the leaf level. The clusters sizes are picked from a Poisson distribution with mean equal to average cluster size. Next for each cluster we generate a set of potentially maximal itemsets from the children of the items in the cluster. The number of such itemsets is picked from a Poisson distribution with mean set to average number of itemsets. The size of each itemset is also picked from a Poisson distribution with mean set to average large itemset size. Each cluster has an associated weight that determines the probability that this cluster will be picked. The weight is picked according to an exponential distribution with mean set to 1. The weights are normalized such that the sum of all weights equals 1. The itemsets associated with each cluster are also given weights which determine the probability this itemset will be picked once that particular cluster is picked. The weights are picked from an exponential distribution with mean set to 1. The weights are then normalized such that the sum of all weights of all itemsets for a cluster equal 1. The length of a transaction is determined by Poisson distribution with mean μ equal to $|T|$. Until the transaction size is less than the generated length, a cluster is picked according to its weight. Once the cluster is determined an itemset from that cluster is picked and assigned to the transaction. Not all items from the itemset picked are assigned to the transaction. Items from the itemset are dropped as long as an uniformly generated random number between 0 and 1 is less than a corruption level, c. The corruption level for itemset is determined by a normal distribution with mean 0.5 and variance 0.1. The transactions contain only leaf items from the taxonomy.

The parameters used in the generation of the synthetic data are shown in Table 3.

3.2 Performance

We generated two sets of data, "Short" and "Tall" based on different average fanouts in the taxonomy on the items. Both datasets contain the same number of items (leaf items in the taxonomy) and the same number of transactions. The parameter values used to generate the data are shown in Table 4.

$\|D\|$	Number of transactions
$\|T\|$	Average size of transactions
$\|C\|$	Average size of maximal potentially large clusters
$\|I\|$	Average size of maximal potentially large itemsets
$\|S\|$	Average number of itemsets for each cluster
$\|L\|$	Number of maximal potentially large clusters
N	Number of items
R	Number of roots
F	Fanout

Table 3: Parameters

Parameter	"Short"	"Tall"
$\|D\|$	50,000	50,000
$\|T\|$	10	10
$\|C\|$	5	5
$\|I\|$	5	5
$\|S\|$	3	3
$\|L\|$	2,000	2,000
N	8,000	8,000
R	10	10
F	9	3

Table 4: Data parameters

We ran both algorithms on the two data sets for various values of minimum support. The minimum RI was set to 0.5 in all cases. The results are shown in Figures 5 and 6. The times shown include both generation of negative itemsets and negative rules. Since our goal was to study the performance of generating negative associations, we have not included the time taken to generate the generalized large itemsets. The "Tall" dataset which has a taxonomy with smaller fanout took longer to complete than the "Short" dataset. The reason was the far larger number of generalized large itemsets that were generated for the "Tall" dataset. For example, at a support level of 1.5 %, 15,476 large itemsets were generated for the "Tall" dataset as opposed to 1,499 for "Short." If normalized for the number of generalized large itemsets, the times for the "Tall" dataset are much smaller than those of "Short" as per our expectations.

Next, we compared the effect of the different fanouts in the taxonomy. The number of negative candidates generated and the number of negative rules for each data set are shown in the Figure 7. To keep the results comparable we have normalized these numbers with respect to the number of large itemsets. As can be seen, the experiment confirmed that the number of candidates increases with the increase in fanout.

4 Conclusion

We introduced the problem of mining negative association rules in a large database of retail customer

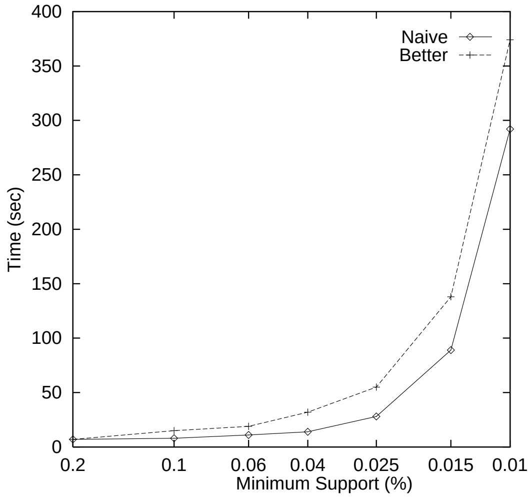

Figure 5: Execution times: "Short" data set

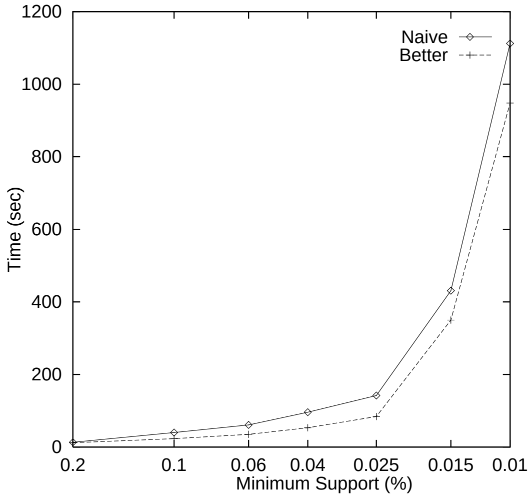

Figure 6: Execution times: "Tall" data set

transactions. Mining negative information is non-trivial and in the case of retail transactions data the problem becomes impossible to solve. Our approach is to use the grouping information such as a taxonomy over the items and the existing positive associations in the data to induce negative rules between items "close" to the items in the positive associations. This approach solves the problem pruning the combinatorial search space to a small subset of cases which have a high potential of being interesting. We present an algorithm for mining negative rules and improvements and study their performance on synthetic data.

4.1 Future Work

Many problems remain unsolved in the problem of mining negative rules. Two of the biggest problems are as follows.

- We assume only the taxonomy over the items as the domain knowledge available to mine for negative rules. However, there may be many other

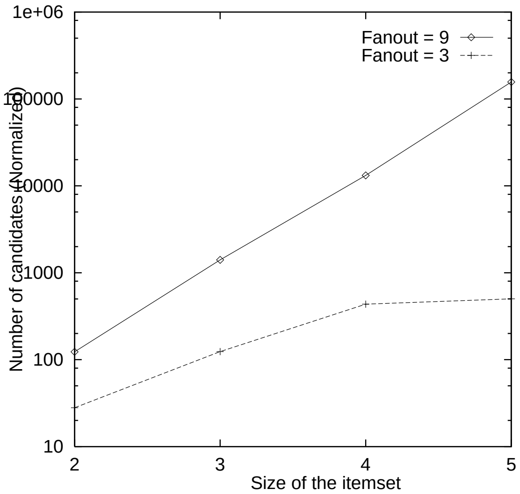

Figure 7: Number of negative candidates

types of information available. For instance, a knowledge of substitute items. How to incorporate other types of information to improve the quality of rules needs to be explored further.

- The number of candidates generated is exponential over the length of the large itemsets being considered. More efficient candidate generation techniques need to be developed.

Acknowledgment

The first author wishes to thank Dr. Rakesh Agrawal at IBM Almaden Research Center for suggesting the problem and for many insightful discussions.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, DC, May 26-28 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, August 29-September 1 1994.

[3] T. J. Blischok. Every transaction tells a story: Creating customer knowledge through market-basket analysis. *Chain Store Age Executive*, V71:50 – 57, March 1995.

[4] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of the VLDB Conference*, pages 420 – 431, September 1995.

[5] M. Houtsma and A. Swami. Set-oriented mining of association rules. In *Proceedings of the International Conference on Data Engineering*, Taipei, Taiwan, March 1995.

[6] R. Krishnamurthy and T. Imielinski. Practitioner problems in need of database research. *ACM SIGMOD Record*, 20(3):76–78, September 1991.

[7] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In *KDD-94: AAAI Workshop on Knowledge Discovery in Databases*, pages 181 – 192, Seattle, Washington, July 1994.

[8] J. S. Park, M-S. Chen, and P. S. Yu. An effective hash based algorithm for mining association rules. In *Proceedings of the ACM-SIGMOD Conference on Management of Data*, pages 229 – 248, San Jose, California, May 1995.

[9] G. Piatetsky-Shapiro. *Discovery, Analysis and Presentation of Strong Rules*, pages 229 – 248. AAAI Press/The MIT Press, Menlo Park, California, 1991.

[10] G. Piatetsky-Shapiro and W. J. Frawley, editors. *Knowledge Discovery in Databases*. MIT Press, 1991.

[11] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules. In *Proceedings of the VLDB Conference*, pages 432 – 444, Zurich, Switzerland, September 1995.

[12] A. Silberschatz, M. Stonebraker, and J. Ullman. Database systems: achievements and opportunities. *Communications of the ACM*, 34(10):110–120, October 1991.

[13] P. Smyth and R. M. Goodman. *Rule Induction Using Information Theory*, pages 159 – 177. AAAI Press/The MIT Press, Menlo Park, California, 1991.

[14] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proceedings of the VLDB Conference*, pages 407 – 419, September 1995.

[15] M. Stonebraker, R. Agrawal, U. Dayal, E. Nuehold, and A. Reuter. Database research at a crossroads: The vienna update. In *Proceedings of the 19th International Conference on Very Large Data Bases*, pages 688–192, Dublin, Ireland, August 1993.