# Removing trivial associations in association rule discovery*

Geoffrey I. Webb and Songmao Zhang
School of Computing and Mathematics, Deakin University
Geelong, Victoria 3217, Australia

## Abstract

Association rule discovery has become one of the most widely applied data mining strategies. Techniques for association rule discovery have been dominated by the frequent itemset strategy as exemplified by the Apriori algorithm. One limitation of this approach is that it provides little opportunity to detect and remove association rules on the basis of relationships between rules. As a result, the association rules discovered are frequently swamped with large numbers of spurious rules that are of little interest to the user. This paper presents association rule discovery techniques that can detect and discard one form of spurious association rule: trivial associations.

## 1 Introduction

We characterize the association rule discovery task as follows. A *dataset* is a finite set of *records* where each record is an element to which we apply Boolean predicates called *conditions*. An *itemset* is a set of conditions. The name *itemset* derives from association rule discovery's origins in market basket analysis where each condition denotes the presence of an item in a market basket. $cs(I)$ denotes the set of records from a dataset that satisfy all conditions in itemset $I$.

An *association rule* consists of itemsets called the *antecedent* and *consequent* and associated statistics describing the frequency with which the two co-occur within the data set. An association rule with antecedent $A$, consequent $C$, and statistics $S$ is denoted as $A \rightarrow C[S]$.

The statistics employed may vary considerably. We utilize the following (where $\mathcal{D}$ is the dataset from which associations are to be discovered):

$coverage = \frac{|cs(antecedent)|}{|\mathcal{D}|}$.

$support = \frac{|cs(antecedent \cup consequent)|}{|\mathcal{D}|}$.

$confidence = \frac{support}{coverage}$.

$lift = \frac{confidence}{|cs(consequent)|/|\mathcal{D}|)}$.

The task involves finding all association rules that satisfy a set of user defined constraints with respect to a given dataset.

The frequent itemset strategy, as exemplified by the Apriori algorithm [1], has become the standard approach to association rule discovery. This strategy first discovers all *frequent itemsets*. A frequent itemset is an itemset whose support exceeds a user defined threshold. The association rules are then generated from the frequent itemsets. If there are relatively few frequent itemsets this approach can be very efficient. However, one limitation of this approach is that it provides little opportunity to detect and remove association rules on the basis of relationships between rules. As a result, the association rules discovered are frequently swamped with large numbers of spurious rules that are of little interest to the user. This paper presents association rule discovery techniques that can detect and discard one form of spurious association rules: trivial associations.

Consider an association between three items, say tomatoes, carrots, and cucumber, $\{tomatoes, carrots\} \rightarrow \{cucumber\}$ [$coverage = 0.100$, $support = 0.050$, $confidence = 0.500$, $lift = 2.00$]. Suppose that another item, always appears in a record whenever the antecedent appears, say *lettuce*. In this case, the following association is entailed $\{tomatoes, carrots, lettuce\} \rightarrow \{cucumber\}$ [$coverage = 0.100$, $support = 0.050$, $confidence = 0.500, lift = 2.00$]. This association adds nothing of interest to most analyses beyond the first (and perhaps

*Prepublication draft of Webb, G. I. and S. Zhang (2002). Removing Trivial Associations in Association Rule Discovery. In Congress Proceedings, NAISO Academic Press, Canada/The Netherlands.

$\{tomatoes, carrots\} \rightarrow \{lettuce\}$ [$coverage = 0.100$, $support = 0.100$, $confidence = 1.000$, $lift = 2.00$]).

To formalise this notion, an association $X \rightarrow Y[S]$ is *trivial* if and only if there exists another association $Z \rightarrow Y[S]$ such that $Z$ is a proper subset of the conditions in $X$. Note, that the consequent and the statistics must be identical for both the trivial association and the more general form that makes it trivial.

OPUS_AR is an association rule discovery that provides an alternative to the frequent itemset approach by finding associations without first finding frequent itemsets [7]. This avoids the need to retain the set of frequent itemsets in memory, a requirement that makes the frequent itemset strategy infeasible for dense data [3]. It also enables comparisons between association rules to be performed during search enabling properties that arise due to the relationship of one association to another, such as triviality, to be utilized during search.

This paper presents techniques for detecting and discarding trivial associations during search. It is demonstrated that these techniques often reduce the time taken to find associations while also removing the trivial associations from the list of associations returned to the user.

## 2 The Apriori algorithm

The Apriori algorithm discovers associations in a two-step process. First, it finds the frequent itemsets $\{I \subseteq \mathcal{C} : \frac{|cs(I)|}{|\mathcal{D}|} \geq min\_support\}$, where $\mathcal{C}$ is the set of all available conditions, $\mathcal{D}$ is the dataset, and $min\_support$ is a user defined minimum support constraint. In the second stage the frequent itemsets are used to generate the association rules. The minimum support constraint on the frequent itemsets guarantees that all associations generated will satisfy the minimum support constraint. Other constraints, such as minimum confidence are enforced during the second stage.

The frequent itemset strategy limits the number of rules that are explored, and caches the support values of the frequent items so that there is no need to access the dataset in the second step. It is very successful at reducing the number of passes through the data. The frequent itemset approach has become the predominant approach to association rule discovery. However, the frequent itemset approach is only feasible for sparse data. For dense datasets where there are numerous frequent itemsets, the overheads for maintaining and manipulating the itemsets are too large to make the system efficient and feasible [3]. This is also apparent in the experiments presented below. Dense datasets are common in applications other than basket data analysis or when basket data is augmented by other customer information. Another problem of Apriori is that it lists numerous association rules to the user and it may be very difficult for the user to identify the interesting rules manually. Take the covtype dataset for example. Covtype has 581,012 records containing 125 items. The number of the association rules generated by Apriori with the minimum support set to 0.01, minimum confidence 0.8, and maximum itemset size 5 is 88,327,710. Since the Apriori algorithm generates itemsets by considering features of itemsets in isolation, the inter-relationships between the itemsets are not taken into account. In consequence, many association rules generated may not be of interest to the user.

## 3 The OPUS_AR algorithm

OPUS_AR extends the OPUS search algorithm to association rule discovery [7]. To simplify the search problem, the consequent of an association rule is restricted to a single condition. Association rules of this restricted form are of interest for many data mining applications.

Whereas OPUS supports search through spaces of subsets, the association rule search task requires search through the space of pairs $\langle I \subseteq conditions, c \in conditions \rangle$, where $I$ is the antecedent and $c$ the consequent of an association. OPUS_AR achieves this by performing OPUS search through the space of antecedents, maintaining at each node a set of potential consequents, each of which is explored at each node.

The algorithm relies upon there being a set of user defined constraints on the acceptable associations. These are used to prune the search space. Such constraints can take many forms, ranging from the traditional association rule discovery constraints on support and confidence to a constraint that only the $n$ associations that maximize some statistic be returned. To provide a general mechanism for handling a wide variety of constraints, we denote associations that satisfy all constraints *target associations*.

Note that it may not be apparent when an association is encountered whether or not it is a target. For example, if we are seeking the 100 associations with the highest lift, we may not know the cutoff value for lift until the search has been completed. Hence, while we may be able to determine in some circumstances that an association is not a target, we may not be able to determine that an association is a target until the search is completed. To accommodate this, pruning is only invoked when it is determined that areas of the search space cannot contain a target. All associations encountered are recorded unless the system can determine that they are not targets. However, these associations may be subsequently discarded as progress through the search space reveals that they cannot be targets. When seeking the $n$ best associations with respect to some measure, we can determine that a new association is not a target if its value on that measure is lower than the value of the $n^{th}$ best recorded so far, as the value of the $n^{th}$ best for the search space cannot be lower than the value of the $n^{th}$ best for the subset of the search space examined so far.

Table 1 displays the algorithm that results from applying the OPUS search algorithm [6] to obtain efficient search for this search task. The algorithm is presented as a recursive procedure with three arguments: **CurrentLHS**, the set of conditions in the antecedent of the rule currently being considered; **AvailableLHS**, the set of conditions that may be added to the antecedent of rules to be explored below this point; **AvailableRHS**, the set of conditions that may appear on the consequent of a rule in the search space at this point and below. The initial call to the procedure sets CurrentLHS to {}, and AvailableLHS and AvailableRHS to the set of conditions that are to be considered on the antecedent and consequent of association rules, respectively.

The algorithm OPUS_AR is a search procedure that starts with the associations with one condition in the antecedent and searches through successive associations formed by adding conditions to the antecedent. It loops through each condition in AvailableLHS, adds it to CurrentLHS to form the NewLHS. For the NewLHS, it loops through each condition in AvailableRHS to check if it could be the consequent for NewLHS. After the AvailableRHS loop, the procedure is recursively called with the arguments NewLHS, NewAvailableLHS and NewAvailableRHS. The two latter arguments are formed by removing the pruned conditions from AvailabeLHS and AvailableRHS, respectively. Step 2.1.3(b.1)ii.A checks if the current association is a potential target and if yes Step 2.1.3(b.1)ii.A.1 records it. When seeking the $n$ associations with the highest value of lift that satisfy all other constraints, the $n^{th}$ highest value of lift recorded at Step 2.1.3(b.1)ii.A.1 is used as a lower bound on the $n^{th}$ highest value of lift for the search space. When the search is completed, the $n^{th}$ highest value of lift recorded will be the $n^{th}$ high-

Table 1: The OPUS_AR algorithm

```
OPUS_AR(CurrentLHS, AvailableLHS, AvailableRHS)
 1. SoFar := {}
 2. FOR EACH P in AvailableLHS
 2. 1 IF pruning rules cannot determine that
    ∀x ⊆ AvailableLHS: ∀y ∈ AvailableRHS:
    ¬target(x ∪ CurrentLHS ∪ {P} → y) THEN
 2. 1.1 NewLHS := CurrentLHS ∪ {P}
 2. 1.2 NewAvailableLHS := SoFar - P
 2. 1.3 IF pruning rules cannot determine that
    ∀x ⊆ NewAvailableLHS: ∀y ∈ AvailableRHS:
    ¬target(x ∪ NewLHS → y) THEN
    (a) NewAvailableRHS := AvailableRHS - P
    (b) IF pruning rules cannot determine ∀y ∈
        NewAvailableRHS: ¬target(NewLHS → y)
        THEN
    (b. 1) FOR EACH Q in NewAvailableRHS
         i. IF pruning rules determine that ∀x
            ⊆ NewAvailableLHS: ¬target(x ∪
            NewLHS → Q) THEN
              A. NewAvailableRHS :=
                 NewAvailableRHS - Q
        ii. ELSE IF pruning rules cannot
            determine that ¬target(NewLHS
            → Q) THEN
              A. IF NewLHS → Q is a potential
                 target THEN
                 A.1 record NewLHS → Q
                 A.2 tune the settings of the
                 measures
              B. IF pruning rules determine
                 that ∀x ⊆ NewAvailableLHS:
                 ¬target(x ∪ NewLHS → Q) THEN
                 NewAvailableRHS :=
                 NewAvailableRHS - Q
    (c) IF NewAvailableLHS ≠ {} and
        NewAvailableRHS ≠ {} THEN
            OPUS_AR(NewLHS, NewAvailableLHS,
            NewAvailableRHS)
    (d) SoFar := SoFar ∪ {P}
```

est value of lift for the search space. During search, the pruning rules can discard associations that have lower lift than the lower bound established by the $n^{th}$ highest value of lift recorded at Step 2.1.3(b.1)ii.A.1.

To discard trivial association during search all that is required is to incorporate another test at Step 2.1.3(b.1)ii.A. At this stage, triviality can be added as another condition that prevents an association from being a target and sections of the search space can potentially be ignored when it can be determined that they may only contain trivial rules. Any potential target association already recorded at Step 2.1.3(b.1)ii.A.1. can not be trivial with respect to the current association since for any condition set, OPUS_AR always investigates all of its proper subsets before it. Therefore Step 2.1.3(b.1)ii.A only needs to check if the current association is trivial with respect to one of the recorded potential target associations.

# 4 Search for Nontrivial Association Rules

After giving a formal description of nontrivial association rule discovery based on OPUS_AR, we will present one pruning rule and one data access saving rule adopted in OPUS_AR for effectively removing trivial associations.

## 4.1 Formal description

We first give a formal description of the association rule discovery based on OPUS_AR, then define trivial association rules, and accordingly nontrivial association rule discovery based on OPUS_AR is presented.

**Definition 1.** *A association rule discovery task based on OPUS_AR (abbreviated as ARO) is a 4-tuple* $(\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$*, where*

$\mathcal{C}$**:** *nonempty set of conditions;*

$\mathcal{D}$**:** *nonempty set of records, called the dataset, where for each record* $d \in \mathcal{D}$*,* $d \subseteq \mathcal{C}$*. For any* $S \subseteq \mathcal{C}$*, let* $cs(S) = \{d | d \in \mathcal{D} \wedge S \subseteq d\}$*, and let* $cover(S) = \frac{|cs(S)|}{|\mathcal{D}|}$*;*

$\mathcal{A}$**:** *set of association rules, where each association rule takes the form*

$$X \rightarrow Y[coverage, support, confidence, lift]$$

*where* $X \subset \mathcal{C}$*,* $X \neq \emptyset$*,* $Y \subset \mathcal{C}$*,* $|Y| = 1$*,* $X \cap Y = \emptyset$*, and* $coverage, support, confidence,$ *and* $lift$ *are statistics for the association rule, satisfying* $coverage(X \rightarrow Y) = cover(X)$*,* $support(X \rightarrow Y) = cover(X \cup Y)$*,* $confidence(X \rightarrow Y) = \frac{support(X \rightarrow Y)}{coverage(X \rightarrow Y)}$*, and* $lift(X \rightarrow Y) = \frac{confidence(X \rightarrow Y)}{cover(Y)}$*;*

$\mathcal{M}$**:** *constraints, composed of maxAssocs denoting the maximum number of target association rules (which will consist of the association rules with the highest values for lift of those that satisfy all other constraints),* $maxLHSsize$ *denoting maximum number of conditions allowed in the antecedent of association rule,* $minCoverage$ *denoting the minimum coverage,* $minSupport$ *denoting the minimum support,* $minConfidence$ *denoting the minimum confidence, and* $minLift = \max(1.0, \beta(RS, maxAssocs))$*, where* $RS$ *is the set of associations* $\{R : coverage(R) \geqslant minCoverage \wedge support(R) \geqslant minSupport \wedge confidence(R) \geqslant minConfidence\}$*, and* $\beta(Z, n)$ *is the lift of the* $n^{th}$ *association in* $Z$ *sorted from highest to lowest by lift. An association rule* $X \rightarrow Y[coverage, support, confidence, lift]$ *is a target iff it satisfies* $|X| \leqslant maxLHSsize$*,*$coverage(X \rightarrow Y) \geqslant minCoverage$*,* $support(X \rightarrow Y) \geqslant minSuport$*,* $confidence(X \rightarrow Y) \geqslant minConfidence$*, and* $lift(X \rightarrow Y) \geqslant minLift$*.*

**Definition 2.** *Suppose* $ARO = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$*. For any association rule* $X \rightarrow Y \in \mathcal{A}$*, it is trivial iff there exists an association rule* $X_1 \rightarrow Y \in \mathcal{A}$ *satisfying* $X_1 \subset X$ *and* $coverage(X_1 \rightarrow Y) = coverage(X \rightarrow Y)$*.*

**Definition 3.** *For any* $ARO = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$*, it is a nontrivial association rule discovery task based on OPUS_AR, denoted by* $ARO^*$*, iff that every association rule is not trivial is added to the constraints* $\mathcal{M}$*.*

We give two properties about $ARO$ in the following. Obviously all the properties $ARO$ has also apply to $ARO^*$.

**Theorem 1.** *Suppose* $ARO = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$*. For any* $S_1 \subseteq \mathcal{C}$*,* $S_2 \subseteq \mathcal{C}$*, and* $S_1 \subseteq S_2$*,* $cs(S_2) \subseteq cs(S_1)$ *holds. This is to say,* $cover(S_2) \leqslant cover(S_1)$ *holds.*

*Proof.* For any $d \in cs(S_2)$, according to Definition 1, $S_2 \subseteq d$ holds. Since $S_1 \subseteq S_2$, $S_1 \subseteq d$ holds. Hence $d \in cs(S_1)$. So $cs(S_2) \subseteq cs(S_1)$ holds. □

**Theorem 2.** *Suppose* $ARO = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$. *For any association rules* $S_1 \rightarrow S_3$ *and* $S_1 \cup S_2 \rightarrow S_3$, *if*

$$coverage(S_1 \rightarrow S_3) = coverage(S_1 \cup S_2 \rightarrow S_3) \quad (1)$$

*the following hold.*

$$support(S_1 \rightarrow S_3) = support(S_1 \cup S_2 \rightarrow S_3) \quad (2)$$

$$confidence(S_1 \rightarrow S_3) = confidence(S_1 \cup S_2 \rightarrow S_3) \quad (3)$$

$$lift(S_1 \rightarrow S_3) = lift(S_1 \cup S_2 \rightarrow S_3) \quad (4)$$

*Proof.* Since (3) and (4) hold if (2) hold, so we only need to prove (2), which is as follows.

$$cover(S_1 \cup S_3) = cover(S_1 \cup S_2 \cup S_3) \quad (5)$$

From (1), i.e., $cover(S_1) = cover(S_1 \cup S_2)$, and Definition 1, we have

$$|cs(S_1)| = |cs(S_1 \cup S_2)| \quad (6)$$

From Theorem 1,

$$cs(S_1) \supseteq cs(S_1 \cup S_2) \quad (7)$$

From (6) and (7), we get

$$cs(S_1) = cs(S_1 \cup S_2) \quad (8)$$

For any $d \in \mathcal{D} \wedge S_1 \cup S_3 \subseteq d$, $S_1 \subseteq d$ and $S_3 \subseteq d$ hold. From $S_1 \subseteq d$ and (8), we get $S_1 \cup S_2 \subseteq d$. From $S_3 \subseteq d$, $S_1 \cup S_2 \cup S_3 \subseteq d$ holds. Hence

$$cs(S_1 \cup S_3) \subseteq cs(S_1 \cup S_2 \cup S_3) \quad (9)$$

From Theorem 1, we have

$$cs(S_1 \cup S_2 \cup S_3) \subseteq cs(S_1 \cup S_3) \quad (10)$$

From (9) and (10), $cs(S_1 \cup S_3) = cs(S_1 \cup S_2 \cup S_3)$ holds. Hence (5) is proved. □

From Theorem 2 it follows that if one association is trivial with respect to another association, these two associations share not only the same coverage, but also the same support, confidence and lift.

## 4.2 Pruning the condition added to antecedent

This pruning rule at Step 2.1 prunes the condition in $AvailableLHS$ before it is added to the $CurrentLHS$. It is based on the following theorem.

**Theorem 3.** *Suppose* $ARO^* = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$. *For any association rule* $X \rightarrow Y$, *if* $|X| \geq 2$ *and there exists* $P \in X$ *satisfying* $cover(\{P\}) = 1$, $X \rightarrow Y$ *is not a target.*

*Proof.* Let $X = X_1 \cup \{P\}$ where $X_1 \subset \mathcal{C}$. Since $cover(\{P\}) = 1$, $|cs(\{P\})| = |\mathcal{D}|$. From $cs(\{P\}) \subseteq \mathcal{D}$, we get

$$cs(\{P\}) = \mathcal{D} \quad (11)$$

From Theorem 1,

$$cs(X) = cs(X_1 \cup \{P\}) \subseteq cs(X_1) \quad (12)$$

For any $d \in \mathcal{D} \wedge X_1 \subseteq d$, from (12), $P \subseteq d$ holds. Therefore $X_1 \cup \{P\} \subseteq d$ holds. This means the following is proved.

$$cs(X_1) \subseteq cs(X) = cs(X_1 \cup \{P\}) \quad (13)$$

From (13) and (14), we get

$$cs(X_1) = cs(X) = cs(X_1 \cup \{P\}) \quad (14)$$

So $cover(X_1) = cover(X_1 \cup \{P\})$, i.e., $coverage(X \rightarrow Y) = coverage(X_1 \rightarrow Y)$. According to Definition 2, $X \rightarrow Y$ is trivial due to the fact that $X_1 \rightarrow Y$ exists. $X \rightarrow Y$ is not a target. □

From this theorem, we get the following pruning rule.

**Pruning 1.** *In OPUS_AR for* $ARO^* = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$, *for any condition* $P \in AvailableLHS$, *if* $CurrentLHS \neq \emptyset$ *and* $cover(\{P\}) = 1$, $P$ *can be pruned from* $NewAvailableLHS$.

According to the theorem above, any association rule containing $CurrentLHS \cup \{P\}$ in the antecedent is trivial, so $P$ can be pruned from $NewAvailableLHS$.

### 4.3 Saving data access for the associations with $NewLHS$ as antecedents

The saving rule at Step 2.1.3(b) for all the associations with $NewLHS$ as antecedents, where $NewLHS = CurrentLHS \cup \{P\}$, $P \in AvailableLHS$, functions according to the relation between $CurrentLHS$ and $P$.

**Saving 1.** *In OPUS_AR for $ARO^* = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$, for $NewLHS = CurrentLHS \cup \{P\}$ where $P \in AvailableLHS$, if $cover(CurrentLHS) = cover(NewLHS)$, there is no need to access data to evaluate all the associations with $NewLHS$ as antecedents, as they are not targets.*

All the associations with $NewLHS$ as antecedents are trivial since for each $NewLHS \rightarrow Q$ where $Q \in AvailableRHS$, there exists $CurrentLHS \rightarrow Q$ sharing the same coverage value and is evaluated before $NewLHS$ in OPUS_AR. This saving rule saves the data access for evaluating all the associations with $NewLHS = CurrentLHS \cup \{P\}$ as antecedents, however $P$ can not be pruned from $NewAvailableLHS$ since there may exist nontrivial associations having less coverage than $cover(CurrentLHS)$ by containing other conditions except $NewLHS$ in the antecedents.

## 5 Experiments

In order to evaluate the efficiency of OPUS_AR, experiments are performed on ten large datasets from the UCI ML and KDD repositories [4, 2]. These datasets are listed in Table 2.

We compare the performance with the publicly available apriori system developed by Borgelt [5]. In all the experiments OPUS_AR seeks the top 1000 associations on lift within the constraints of minimum confidence set to 0.8, minimum support set to 0.01, and the maximum number of conditions in antecedent of an association set to 4. The same minimum support, minimum confidence, and maximum antecedent size are used for Apriori, thus the maximum itemset size is 5 for Apriori because itemsets are required that contain up to 4 antecedent conditions as well as the single consequent condition. The experiments were performed on a Linux server with 2 CPUs each 933MHz in speed, 1.5G RAM, and 4G virtual memory.

Table 2: Datasets for experiments

| **name** | **records** | **attributes** | **values** |
|---|---|---|---|
| covtype | 581012 | 55 | 125 |
| ipums.la.99 | 88443 | 61 | 1883 |
| ticdata2000 | 5822 | 86 | 709 |
| connect-4 | 67557 | 43 | 129 |
| letter recognit. | 20000 | 17 | 74 |
| pendigits | 10992 | 17 | 58 |
| shuttle | 58000 | 10 | 34 |
| splice junction | 3177 | 61 | 243 |
| mush | 8124 | 23 | 127 |
| soybean-large | 307 | 36 | 119 |

The "trivial associations allowed" column of Table 3 lists the CPU times and data access numbers on the datasets for OPUS_AR searching for associations no matter they are trivial or not, and "trivial associations filtering out" column shows the efficiency of OPUS_AR searching for nontrivial association rules. The column "no of trivial associations filtered out" lists the number of trivial associations that appear in the top 1000 associations found when trivial associations are allowed (and hence which do not appear in the 1000 associations discovered when trivial associations are not allowed). This is the number of spurious and uninteresting associations that the user is saved from considering when this new approach to association rule discovery is employed. For half the datasets, more than three quarters of the 1000 best associations returned by conventional association rule discovery are trivial.

The CPU times of running Borgelt's Apriori system on the ten datasets are listed in the "Apriori" column of Table 3. We can see clearly that on every dataset OPUS_AR is more efficient than Apriori both for keeping and removing trivial associations. The infeasibility of Apriori for dense datasets is also demonstrated by that for "ticdata2000," Apriori runs out of memory when processing itemsets of size 4.

## 6 Conclusions

OPUS_AR provides an alternative to the frequent itemset approach to association rule discovery. Our experiments have demonstrated that OPUS_AR can utilize constraints on relationships between associa-

Table 3: Efficiency and data access numbers of OPUS_AR filtering none and filtering trivial associations and efficiency of Apriori

| datasets | OPUS_AR | | | | Apriori |
|---|---|---|---|---|---|
| | trivial associations allowed | | trivial associations filtered out | | |
| | CPU time | no of data accesses | CPU time | no of data accesses | no of trivial associations filtered out | |
| covtype | 0:26:41 | 5,502,241 | 0:23:54 | 4,356,827 | 719 | 77:56:3 |
| ipums.la.99 | 0:9:7 | 16,812,342 | 0:7:46 | 14,683,371 | 937 | 19:45:5 |
| ticdata2000 | 0:9:53 | 118,309,620 | 0:7:11 | 99,419,406 | 698 | Not enough memory |
| connect-4 | 0:3:18 | 8,197,934 | 0:2:26 | 6,390,096 | 852 | 3:15:26 |
| letter-recognition | 0:0:6 | 1,589,631 | 0:0:6 | 1,584,171 | 46 | 0:0:35 |
| pendigits | 0:0:3 | 1,069,519 | 0:0:3 | 1,069,557 | 29 | 0:0:23 |
| shuttle | 0:0:2 | 100,963 | 0:0:2 | 103,991 | 169 | 0:0:7 |
| splice junction | 0:6:4 | 719,884,792 | 0:6:3 | 719,433,780 | 0 | 0:12:50 |
| mush | 0:0:1 | 470,293 | 0:0:2 | 686,306 | 775 | 0:1:45 |
| soybean-large | 0:0:1 | 787,477 | 0:0:1 | 1,058,286 | 767 | 0:3:30 |

tion rules to prune the search space. Pruning trivial associations in this manner both removes large numbers of uninteresting associations from the list of associations delivered to the user and provides modest improvements in compute time.

# References


[1] R. Agrawal, T. Imielinski, & A. Swami. Mining associations between sets of items in massive databases. In *Proc. ACM-SIGMOD-93*, pp. 207–216, 1993.

[2] S. D. Bay. The UCI KDD archive. Irvine, CA: University of California, Department of Information and Computer Science, 2001.

[3] R. J. Bayardo. Efficiently mining long patterns from databases. In *Proc. ACM-SIGMOD-98*, pp. 85–93, 1998.

[4] C. Blake & C. J. Merz. UCI repository of machine learning databases. University of California, Department of Information and Computer Science, Irvine, CA., 2001.

[5] C. Borgelt. apriori. (Computer Software) http://fuzzy.cs.Uni-Magdeburg.de/ borgelt/, February 2000.

[6] G. I. Webb. OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995.

[7] G. I. Webb. Efficient search for association rules. In *KDD-2000*, pp. 99–107, Boston, MA, 2000.