

# Discovering Frequent Itemsets in the Presence of Highly Frequent Items

Dennis P. Groth<sup>1</sup> and Edward L. Robertson<sup>2</sup> \*

<sup>1</sup> School of Informatics, Indiana University, Bloomington, IN 47405, USA.  
dgroth@indiana.edu

<sup>2</sup> Computer Science, Indiana University, Bloomington, IN 47405, USA.  
edrbtn@cs.indiana.edu

**Abstract.** This paper presents new techniques for focusing the discovery of frequent itemsets within large, dense datasets containing highly frequent items. The existence of highly frequent items adds significantly to the cost of computing the complete set of frequent itemsets. Our approach allows for the exclusion of such items during the candidate generation phase of the Apriori algorithm. Afterwards, the highly frequent items can be reintroduced, via an inferencing framework, providing for a capability to generate frequent itemsets without counting their frequency. We demonstrate the use of these new techniques within the well-studied framework of the Apriori algorithm. Furthermore, we provide empirical results using our techniques on both synthetic and real datasets - both relevant since the real datasets exhibit statistical characteristics different from the probabilistic assumptions behind the synthetic data. The source we used for real data was the U.S. Census.

## 1 Introduction

Association rule discovery has been an extremely active area for database researchers for a number of years. Since the initial description of the Apriori algorithm [2], research has focused on numerous optimization strategies, with many researchers reporting the dramatic increase in candidate itemset generation as minimum support levels are decreased. This increase in candidate sets translates into a significant amount of effort expended to determine the frequency of the candidate sets. From a user's standpoint furthermore, if the support level is set too low, standard techniques might generate millions of possible associations from even moderately sized datasets, the result of which can only be interpreted as noise. These circumstances will undoubtedly lead to a need to data mine the results of association rule generation itself!

Within the context of association rule discovery, an itemset is a collection of items that occur together in transactions. The discovery of these itemsets is solved by the Apriori algorithm, which finds all itemsets within a dataset of transactions (sometimes called baskets), such that the frequency of the itemset

---

\* The authors were supported by NSF Grant IIS-0082407

exceeds user-provided threshold frequency. In many applications the Apriori algorithm quickly finds the frequent itemsets. However, certain datasets make the application of the Apriori algorithm intractable, due to a combinatorial explosion of the number of itemsets discovered.

In contrast with optimization strategies strictly focused on identifying and thus avoiding the counting of itemsets that are not frequent, this paper describes techniques for excluding itemsets that are certain to be frequent. Our goal is to avoid verifying whether an itemset is frequent for a particular minimum support level when we can determine in advance that the itemset is certain to be frequent. In such cases, we can simply output the certainly frequent itemsets without verification. Our approach does involve a tradeoff between the time involved in computing the frequent itemsets and the totality of the frequent itemsets that are discovered. As a result, we accept a subset of the frequent itemsets in exchange for a significant improvement in computation time. For the datasets we are using, generating the complete set of frequent itemsets is unreasonable, since there are billions of frequent itemsets.

For the purposes of experimentation, we incorporated these new techniques within the framework of the well-studied Apriori algorithm. However, we envision that these techniques can be added to other algorithms in a straightforward fashion. Using these new techniques, we present empirical results for both synthetic and real datasets. For real data, we utilized U.S. Census data, which presents significant challenges due to the size of the dataset and the large number of items. Of particular significance in census data is the existence of a number of items that occur with high frequency.<sup>3</sup> The density of such data has been identified as a major problem, leading to combinatorial explosion.[4] It is in this context of such highly frequent items that we can identify the itemsets that are certain to be frequent. Instead, we introduce an inference mechanism for generating the certainly frequent itemsets – without counting them.

The paper is structured as follows. We provide a brief coverage of other research related to optimizing the discovery of frequent itemsets in Section 2. In Section 3 we introduce the RangeApriori algorithm, providing the ability to selectively generate subsets of the frequent itemsets contained in a dataset. Section 4 describes our inference model for generating frequent itemsets. Experimental results are reported in Section 5.

## 2 Related Work

There have been numerous optimization strategies applied to the problem of discovering frequent itemsets. In [5], the authors apply significance testing to associations, essentially searching for correlation using a chi-squared test. Another approach involves sampling [14], in which a subset of the dataset is analyzed for likely associations. These associations from the sample are then verified in the complete dataset. An approach based on partitioning the input is found in [12]. An approach that avoids counting subsets of large frequent itemsets is presented

---

<sup>3</sup> For this research we consider items occurring in greater than 50% of transactions to be highly frequent.

in [3, 1]. An approach to dealing with dense data and the resulting combinatorial explosion of possible rules is described in [4].

In [6], an innovative counting scheme is employed, which begins counting larger itemsets based on partially reading the dataset. The goal is to reduce the number of passes through the data, since for large datasets, the I/O costs are significant. Research in feature reduction and product hierarchies [13, 8] is aimed at controlling the potential exponential blowup of candidate itemsets. Separate and distinct from these, are approaches utilizing SQL to discover itemsets [11, 10].

In [9], the authors describe results using multiple minimum support levels, the goal of which is to discover rare itemsets. An axiomatization of frequent itemsets is presented in [7].

### 3 The Apriori Algorithm

For the sake of brevity, we utilize the standard framework for frequent itemsets relative to the Apriori algorithm. Let  $I$  be a set of unique item identifiers, and  $\mathcal{I} \subseteq I$  be an itemset. We assume a dataset  $D$  of transactions (or baskets)  $t$ . Transactions contain a subset of the items, which we refer to as  $t.I$ . The frequency of an itemset is denoted as  $Count(\mathcal{I}) = |\{t \in D | \mathcal{I} \subseteq t.I\}|$ . The ratio  $Count(\mathcal{I})/|D|$  is called the support of itemset  $\mathcal{I}$ , denoted  $support(\mathcal{I})$ . An itemset  $\mathcal{I}$  is considered frequent if  $support(\mathcal{I})$  exceeds a user provided minimum support value, referred to as *minsup*.

The Apriori algorithm uses a level-wise approach to discovering frequent itemsets, starting with the singleton itemsets by reading in each transaction and counting the number of occurrences of each item. After this counting phase, items with support less than the minimum support value are pruned and the remaining items are combined in all possible pairs. These pairs are called the candidate itemsets, and are counted by making another pass through the data. The process continues until no new candidate itemsets are generated. The Apriori algorithm relies upon the following property, which is called the Apriori Principle:

$$\mathcal{I} \text{ is frequent} \iff \forall \mathcal{J} \subset \mathcal{I}, \mathcal{J} \text{ is frequent.}$$

Given a minimum support level  $s$ , we define  $F^s$  to be the frequent itemsets occurring in at least  $s$  percent of transactions in  $D$ . For a particular level  $k$ , let  $F_k^s$  be the frequent itemsets of size  $k$ . Clearly, it is the case that  $F^s = \bigcup_{i=1}^k F_i^s$ . Similarly, we define  $C^s$  to be the candidate itemsets generated, and  $C_k^s$  to be the  $k$ -candidate sets. Note that while  $F_k^s$  is defined only in terms of the data, the definition of  $C_k^s$  depends upon the algorithm.

**Lemma 1.** *Given a dataset  $D$  and a minimum support level  $s$ , and a level  $k$ :*

$$\begin{aligned} F^s &\subseteq C^s \\ F_k^s &\subseteq C_k^s \end{aligned}$$

**Proof:** The inclusions follow from the Apriori algorithm's definition of candidate and frequent itemsets. ■

**Proposition 1.** (*Support monotonicity*)

Given a dataset  $D$  and two support levels  $s_1$  and  $s_2$ , such that  $s_1 < s_2$ , then:

$$\begin{aligned} F^{s_1} &\supseteq F^{s_2} \\ C^{s_1} &\supseteq C^{s_2} \end{aligned}$$

**Proof:** The inclusion  $F^{s_1} \supseteq F^{s_2}$  follows from the definition of minimum support.  $C^{s_1} \supseteq C^{s_2}$  is based on the Apriori algorithm. ■

A consequence of Proposition 1 is that the number of candidates increases as support levels decrease. In addition, it immediately follows that increasing the support level is non-monotonic.

### 3.1 The RangeApriori Algorithm

While the Apriori algorithm takes advantage of the downward closed property of frequent itemsets, we augment the typical lower support threshold with an upper bound on support. We refer to this additional constraint as the RangeApriori Principle, which is defined below.

**Definition 1** *RangeApriori Principle*

Given a dataset  $D$  and a minimum support level  $s_1$  and a maximum support level  $s_2$ , we define  $F^{s_1, s_2}$  as follows:

$$F^{s_1, s_2} = \{\mathcal{I} \in I : \forall \mathcal{J} \subseteq \mathcal{I}, s_1 \leq \text{support}(\mathcal{J}) < s_2\}$$

In other words,  $\mathcal{I} \in F^{s_1, s_2}$  only if **all** subsets of  $\mathcal{I}$  are also contained in  $F^{s_1, s_2}$ . ■

Note that this approach may exclude some frequent itemsets from being discovered. That is, it is not always the case that  $F^{s_1, s_2} = F^{s_1} - F^{s_2}$ .

For a particular level  $k$ , let  $F_k^{s_1, s_2}$  be the frequent itemsets of size  $k$  within the two support levels, again subject to the RangeApriori Principle (Definition 1). Candidate generation occurs in a level-wise fashion, similar to the Apriori algorithm. However, since we are interested in generating only those frequent itemsets with support bounded by the lower and upper support parameters, our approach defines candidates according to ranges as well.

**Definition 2** *Candidate Ranges*

Given a dataset  $D$  and a minimum support level  $s_1$  and a maximum support level  $s_2$ , we define  $C^{s_1, s_2}$  as follows:

$$C^{s_1, s_2} = \{\mathcal{I} \in I : \forall \mathcal{J} \subset \mathcal{I}, s_1 \leq \text{support}(\mathcal{J}) < s_2\}$$

The candidates of size  $k$  are denoted by  $C_k^{s_1, s_2}$ . ■

The application of an upper bound requires only a slight modification to the Apriori algorithm. In particular, it suffices to prune the singleton itemsets, since every larger itemset can only be comprised of items with support within the

```

1)  $C_2 = \emptyset$ ;
2)  $minsup = s_1 \cdot |D|$ ;
3)  $maxsup = s_2 \cdot |D|$ ;
4) for (int  $i = 1$ ;  $i < NumberOfItems$ ;  $i++$ )
5)   if ( $ItemFrequency[i] \geq minsup$  and  $ItemFrequency[i] < maxsup$ )
6)     for (int  $j = i + 1$ ;  $j \leq NumberOfItems$ ;  $j++$ )
7)       if ( $ItemFrequency[j] \geq minsup$  and  $ItemFrequency[j] < maxsup$ )
8)          $C_2 = C_2 \cup \{\{i, j\}\}$ ;
9) return  $C_2$ ;

```

**Fig. 1.** RangeApriori Algorithm - Level 2 Candidate Generation

supplied ranges. The input for the algorithm is the two support levels as well as the dataset. We modify the Apriori algorithm slightly during the generation of candidates of size 2. We provide pseudo-code for the modifications in Figure 1.

After the second level, the RangeApriori algorithm uses a standard implementation of the Apriori algorithm. Notice that the algorithm does not need to perform the maximum support level test (Line 7) beyond level 2, since the frequency of any itemset is bounded by the smallest frequency of any of its subsets. Our algorithm generalizes to Apriori when  $s_2 = \infty$ .

The following example (1) illustrates the concept of minimum and maximum support, as well as the success rate of the Apriori algorithm and the RangeApriori algorithm.

*Example 1.*

Let  $D$  be the following dataset<sup>4</sup>

$tID$	$A$	$B$
1	a	c
2	a	c
3	a	d
4	b	d

The following describes applications of the Apriori algorithm at both 50% and 60% minimum support levels.

$$\begin{aligned}
C_1^{50} &= \{\{a\}, \{b\}, \{c\}, \{d\}\} \\
F_1^{50} &= \{\{a\}, \{c\}, \{d\}\} \\
C_2^{50} &= \{\{a, c\}, \{a, d\}, \{c, d\}\} \\
F_2^{50} &= \{\{a, c\}\}
\end{aligned}$$

$$\begin{aligned}
C_1^{60} &= \{\{a\}, \{b\}, \{c\}, \{d\}\} \\
F_1^{60} &= \{\{a\}\}
\end{aligned}$$

<sup>4</sup> Note that we have added a tuple ID to the representation of the dataset for clarity purposes.

The following describes the candidate and frequent itemsets discovered using the RangeApriori algorithm, as well as the success rate:

$$\begin{aligned} C_1^{50,60} &= \{\{a\}, \{b\}, \{c\}, \{d\}\} \\ F_1^{50,60} &= \{\{c\}, \{d\}\} \\ F_1^{60,\infty} &= \{\{a\}\} \end{aligned}$$

Note that the itemset  $\{a, c\}$  is contained in  $F^{50}$ , but not in  $F^{50,60}$ . The reason, of course, is that  $\{a\}$  is contained in  $F^{60,\infty}$ , and by the Range Apriori Principle (Definition 1),  $\{a, c\}$  cannot be contained in  $F^{50,60}$ . ■

Using the RangeApriori algorithm involves a trade-off between the quantity of the information discovered with the effort required. For certain applications, RangeApriori is certainly sufficient. Its usefulness might be in *probing* the search space or for hypothesis testing. In Section 4, we will show how to infer the existence of frequent itemsets by combining the results of multiple applications of the RangeApriori algorithm.

## 4 Inferring Frequent Itemsets

In the previous section we introduced the RangeApriori algorithm, which was shown to find a subset of the frequent itemsets that would be found using the Apriori algorithm. In this section we introduce two methods for inferring frequent itemsets. We begin with an additive inference method.

### Lemma 2. Additive Inference Rule

Given support levels  $s_1$  and  $s_2$ , such that  $s_1 < s_2$ , then:

$$F^{s_1} \supseteq F^{s_1, s_2} \cup F^{s_2}$$

**Proof:** The inclusion  $F^{s_1} \supseteq F^{s_1, s_2}$  follows from the definition of  $F^{s_1, s_2}$  and the RangeApriori principle.  $F^{s_1} \supseteq F^{s_2}$  follows from Proposition 1. ■

*Example 2.*

We will utilize the same  $D$  that was used in example 1:

$tID$	$A$	$B$
1	a	c
2	a	c
3	a	d
4	b	d

We will again set  $s_1 = 50$  and  $s_2 = 60$ . The following describes the candidate and frequent itemsets discovered using the RangeApriori algorithm:

$$\begin{aligned}
C_1^{50,60} &= \{\{a\}, \{b\}, \{c\}, \{d\}\} \\
F_1^{50,60} &= \{\{c\}, \{d\}\} \\
F_1^{60,\infty} &= \{\{a\}\}
\end{aligned}$$

Using Lemma 2, we can combine the frequent itemsets:

$$F^{50} \supseteq \{\{a\}, \{c\}, \{d\}\}$$

■

Notice that Lemma 2 allows for the combination of frequent itemsets generated from multiple executions of the RangeApriori algorithm. For example,  $F^{s_1} \supseteq F^{s_1, s_2} \cup F^{s_2, s_3} \cup F^{s_3}$ . We next show, in the presence of highly frequent items, that we are able to infer larger itemsets.

One could ask how often are items encountered with support levels greater than 50 percent. Our experimental results using U.S. Census data found 96 items with support greater than 50 percent. While these 96 items represent a very small fraction of the 7523 total items in the dataset, their effect on the total number of frequent itemsets results in an intractable problem. High frequency items often combine with lower frequency items, which creates a huge number of candidates that must be counted. Other examples of high frequency items occurring in market basket data include seasonal purchases, such as candy before halloween, or wrapping paper before Christmas. In addition, whenever a subset of a large dataset is utilized, the subset may contain certain items that are much more frequent in the subset than in the full dataset.

The following definition describes the *Cross-Union* operator, which is used to combine sets of sets. In particular, we will make use of this operator to combine sets of frequent itemsets.

**Definition 3** *Cross-Union*

Let  $A$  and  $B$  be sets of sets, then

$$A \bowtie B = \{a \cup b \mid a \in A \text{ and } b \in B\}$$

■

While Lemma 2 allowed for additive inferencing of frequent itemsets, it does not provide any capability for inferring larger itemsets. Our approach is based on a minimal overlapping principle. In particular, if two items each have support greater than 50% in the dataset, then the pair must also exist in the dataset. We do not claim that they exist at the 50% support level (eventhough it is possible), but rather, we know they exist. The following Lemma characterizes the support level that we can expect to hold for the pair of items.

**Lemma 3.** *Multiplicative Inference Rule*

Let  $s_1, s_2, s_3, s_4$  be level of support parameters used with the RangeApriori algorithm, and let  $s^\Delta = s_1 + s_3 - 100$ , then:

$$\text{If } s^\Delta > 0, \text{ then } F^{s^\Delta} \supseteq F^{s_1, s_2} \bowtie F^{s_3, s_4}.$$

Proof: By pigeon-holing, the inclusion follows from the definition of  $\bowtie$ . ■

It is clear that, if  $s_1$  and  $s_3$  are over 50%, there must be at least  $s^\Delta\%$  of transactions in  $D$  containing  $F^{s_1, s_2} \bowtie F^{s_3, s_4}$ .

Lemma 3 provides significant power for inferring frequent itemsets, allowing for discovery of frequent itemset with lower support levels than previously had been feasible. The following example shows Lemmas 2 and 3 in action.

*Example 3.*

Let  $D$  be the same dataset used in the previous example.

$tID$	$A$	$B$
1	a	c
2	a	c
3	a	d
4	b	d

Suppose we are interested in support levels of at least 10%, then the following describes an application of the Apriori algorithm to  $D$ :

$$\begin{aligned}
 C_1^{10} &= \{\{a\}, \{b\}, \{c\}, \{d\}\} \\
 F_1^{10} &= \{\{a\}, \{b\}, \{c\}, \{d\}\} \\
 C_2^{10} &= \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}\} \\
 F_2^{10} &= \{\{a, c\}, \{a, d\}, \{b, d\}\}
 \end{aligned}$$

Now, using the RangeApriori algorithm, we will again set  $s_1 = 50$  and  $s_2 = 60$ . The following describes the candidate and frequent itemsets discovered using the RangeApriori algorithm. Remember, RangeApriori retains the count of all singletons from the first pass through the data.

$$\begin{aligned}
 C_1^{50,60} &= \{\{a\}, \{b\}, \{c\}, \{d\}\} \\
 F_1^{50,60} &= \{\{c\}, \{d\}\} \\
 F_1^{60,\infty} &= \{\{a\}\} \\
 F_1^{10,50} &= \{\{b\}\}
 \end{aligned}$$

Using Lemma 3, we can directly compute  $F^{50,60} \bowtie F^{60,\infty}$ :

$$F^{10} \supseteq \{\{a, c\}, \{a, d\}\}$$

Then, by using Lemma 2, we can combine all of the frequent itemsets that we have discovered:

$$F^{10} \supseteq \{\{a\}, \{b\}, \{c\}, \{d\}, \{a, c\}, \{a, d\}\}$$

The previous example demonstrates the real power of combining inferencing with the RangeApriori algorithm. We were able to generate more frequent itemsets than the number of candidates that were considered. ■

#### 4.1 Approximating Confidence for Association Rules

Once the frequent itemsets have been discovered, further effort must be spent in order to generate association rules. An association rule is an implication  $X \Rightarrow Y$ , such that  $X, Y \in I$ . The confidence of a rule is  $\text{conf}(X \Rightarrow Y) = \text{Count}(X \cup Y) / \text{Count}(X)$ . Typically, users are interested in high confidence rules. Using our approach, the confidence levels need to be approximated for the frequent itemsets that are inferred. This section describes the bounds on the confidence values, which can be used for either post-pruning of the itemsets, or as a means of presentation to an expert for further examination.

First, we separate the frequent itemsets discovered by RangeApriori into two sets. The first set contains all itemsets that are discovered using the traditional Apriori technique of generating candidate sets and then scanning the dataset to verify whether the itemsets are frequent. The notation we will use is for this set is  ${}^V F^{s_1, s_2} \subseteq F^{s_1, s_2}$ , the *verified* frequent itemsets. Verified itemsets have their actual frequencies computed during the running of the algorithm. The second set contains all itemsets inferred by using the additive and multiplicative inference rules. The notation we will use is for this set is  ${}^I F^{s_1} \subseteq F^{s_1}$ , the *inferred* frequent itemsets. Note that the RangeApriori principle (Definition 1) is not enforced for inferred itemsets.

Let  $\mathcal{I} = X \cup Y$  be a frequent itemset and  $X \Rightarrow Y$  be a rule. If  $X \cup Y \in {}^V F^{s_1, s_2}$ , we know the counts for  $X$  and  $X \cup Y$ , so we can simply compute the confidence of the rule as  $\text{conf}(X \Rightarrow Y) = \text{Count}(X \cup Y) / \text{Count}(X)$ .

When  $X \cup Y \in {}^I F^s$ , we know  $s \cdot |D| \leq \text{Count}(X \cup Y) \leq \min(\text{Count}(X), \text{Count}(Y))$ . Since  $X \cup Y$  is inferred, it must be the case that either  $X$  and/or  $Y$  are also inferred. We describe the case where  $X$  is inferred. Using Lemma 2 and Lemma 3, we know that  $X \in {}^I F^{s'}$ , for some  $s'$ . The Lemmas are applied in decreasing values of support. Consequently,  $s' \geq s$ , which also follows from  $\text{support}(X) \geq \text{support}(X \cup Y)$ . Since  $X \in {}^I F^{s'}$ , the lower bound for  $\text{Count}(X)$  is  $s' \cdot |D|$ . To determine an upper bound on  $\text{Count}(X)$ , we consider the counts of each  $X' \subseteq X$ , such that  $X' \in {}^V F^{s'', s''}$ . In other words, we find the smallest verified count for any subset of  $X$ . Let  $t' \leq \text{Count}(Y) \leq t''$  be the bounds defined in similar fashion for  $Y$ . The bounds for  $\text{conf}(X \Rightarrow Y)$  are expressed by:

**Definition 4** *Confidence Bounds*

$$\frac{s}{s''} \leq \text{conf}(X \Rightarrow Y) \leq \frac{\min(s', t')}{s'}$$

**Proof:** From the definition of  $s$ ,  $s'$ ,  $s''$ , and  $s''$  it is clear that  $\frac{s}{s''} \leq \frac{\min(s', t')}{s'}$ . To show that  $\text{conf}(X \Rightarrow Y)$  lies within the bounds first assume, to the contrary, that  $\text{conf}(X \Rightarrow Y) > \frac{\min(s', t')}{s'}$ . However, for our assumption to hold it must be the case that  $\text{Count}(X) < \text{Count}(X \cup Y)$  - an obvious contradiction. To prove the lower bound a similar argument hold. ■

The following example illustrates the approximation of confidence.

*Example 4.*

Consider the following verified frequent itemsets from Example 3:

$$\begin{aligned} V F_1^{10,50} &= \{\{b\}\} \\ V F_1^{50,60} &= \{\{c\}, \{d\}\} \\ V F_1^{60,\infty} &= \{\{a\}\} \end{aligned}$$

The following itemsets were inferred using Lemmas 2 and 3:

$${}^I F^{10} \supseteq \{\{a\}, \{b\}, \{c\}, \{d\}, \{a, c\}, \{a, d\}\}$$

Consider the rule  $a \Rightarrow c$ . In this case  $s = 10$ , since  $(a \cup c) \in {}^I F^{10}$ . Likewise,  $s' = 10$ , since  $a \in {}^I F^{10}$ .  $s'' = 60$ , since  $a \in V F^{60,\infty}$ . Finally,  $t' = 10$ , since  $c \in {}^I F^{10}$ . Substituting these values into the inequality defined in Definition 4:

$$\frac{10}{60} \leq \text{conf}(a \Rightarrow c) \leq \frac{\min(10, 10)}{10}$$

In this example, the actual confidence is  $\text{Count}(a \cup c)/\text{Count}(a) = 2/3$ , which lies within the bounds. ■

## 5 Experimental Results

In this section we provide results from experiments of RangeApriori on both synthetic and non-synthetic data. Our focus with this implementation is accurate counting of candidate and frequent sets, and not on overall running time of the algorithm. For these purposes, we report the number of candidates generated, the number of frequent itemsets discovered, the number of frequent itemsets inferred, and the number of passes through the dataset. Note that for reporting purposes, certain graphs will describe the support level as a ratio.

### 5.1 Synthetic Data

For synthetic data, we utilized the synthetic dataset generator from the IBM Quest Research Group<sup>5</sup> to create datasets in the same fashion as [2]. We report experimental results for the T10.I4.D100K dataset. Since there are no highly frequent items, RangeApriori is unable to infer any frequent itemsets. We can use the algorithm, however, to probe the dataset with a range of support values. Figure 2 reports the number of frequent sets found by each algorithm. The minimum support range used for Apriori was 0.001. The RangeApriori algorithm used a support range of 0.001-0.02.

Our use of synthetic data was undertaken in order to create a comparison between the RangeApriori and Apriori algorithms. However, since RangeApriori is discovering a subset of the frequent itemsets, it appears that RangeApriori is best used as a probe of the dataset when item frequencies are low.

<sup>5</sup> <http://www.almaden.ibm.com/cs/quest//syndata.html>

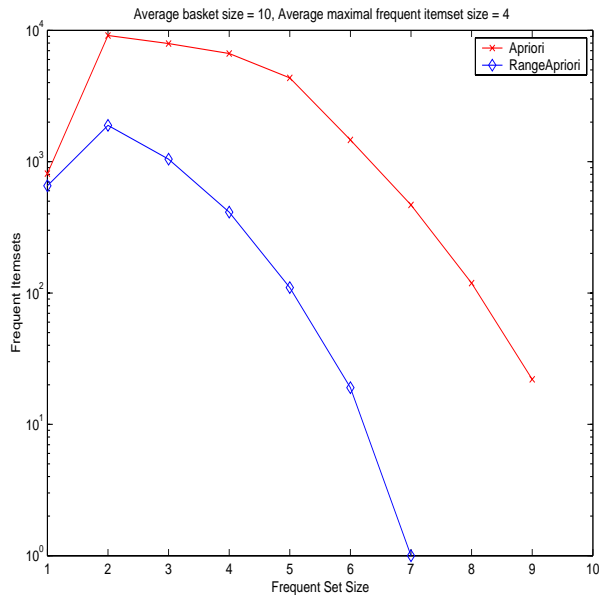


Fig. 2. Frequent sets generated by RangeApriori compared to Apriori

## 5.2 Census Data

Much more profitable results were obtained using the RangeApriori algorithm against non-synthetic data. For data, Public Use Microdata Samples (PUMS) was selected as input. We chose the same sample that was used by [6, 5]. The data represents a five percent sample of the 1990 U.S. Census data for Washington, D.C., and contains 30,370 entries with 122 attributes.

Before applying the RangeApriori algorithm to the PUMS data, the following transformations were performed. For monetary attributes, the transformation was the result of the ceiling of the logarithm of the value. Then, a unique integer was assigned to each possible value in the data. In total, the converted PUMS data was 16.6 MB, with 7523 unique items.

In contrast with the synthetic data, the census data had more items and a different distribution of singleton items. The data is similar to the synthetic data with respect to the infrequent items. However, there are 96 items occurring in more than fifty percent of transactions.

While we performed a variety of experiments, we provide a representative result due to space limitations. We performed experiments with multiple ranges to determine how many itemsets we could infer at the 10% support level. Rather than restrict the number of attributes, all 122 attributes were used. We first counted the singleton items with the RangeApriori algorithm, for the ranges shown in Table 1.

These ranges were selected in order to take advantage of the multiplicative inference rule, which allowed the algorithm to infer a significant number of rules

Minimum Support	Maximum Support	$ F_1 $
0.10	0.20	53
0.20	0.30	11
0.30	0.40	13
0.40	0.50	14
0.50	0.60	10
0.60	0.70	3
0.70	0.80	6
0.80	0.90	29
0.90	1.00	44
1.00	$\infty$	4

**Table 1.** Frequency of singleton items in PUMS data

contained in  $F^{10}$ . As evidenced by the census data, there were 4 items with 100% coverage in the data. Clearly, combining these items with lower frequency items does not make sense.

We then completed the discovery of frequent itemsets for the ranges up to [50-60). The count of candidate and frequent itemsets is shown in Table 2.

Minimum Support	Maximum Support	$ C_2 $	$ F_2 $	$ C_3 $	$ F_3 $	$ C_4 $	$ F_4 $	$ C_5 $	$ F_5 $
0.10	0.20	1378	25	30	30	25	25	11	1
0.20	0.30	55	2						
0.30	0.40	78	4	1	1				
0.40	0.50	91	4						
0.50	0.60	45	4	1	1				

**Table 2.** Itemset counts for  $k > 1$ .

If one considers only the items contained in  $F_1^{90,100}$ , there are  $\binom{44}{2}$  pairs contained in  $F^{80}$ ,  $\binom{44}{3}$  triples contained in  $F^{70}$ , etc. It should be noted that this number represents a lower bound on the number of frequent itemsets, since the dataset does contain some verified itemsets in  $F^{80,90}$ , etc. The following formula describes the number of frequent sets that we avoided counting for the items in  $F_1^{90,100}$ :

$$\sum_{i=1}^9 \binom{44}{i} = 932,778,774$$

Note that the formula does depend on the ranges that were discovered with the RangeApriori algorithm. It is clear that, by using the RangeApriori algorithm

with inferencing, the number of frequent sets that we avoided counting is huge. It is no wonder that previous attempts at analyzing this census data were not successful at low minimum support levels.

## 6 Conclusion and Future Work

In this paper we have introduced the RangeApriori algorithm, which discovers frequent itemsets within a minimum and maximum support range. We have shown that our approach is successful at avoiding the counting of itemsets that are certain to be frequent. For highly frequent items, we have provided a mechanism for inferring frequent itemsets, as well as an approximation of their confidence levels. Note that the inference capability allows for the reintroduction of highly frequent items without incurring the combinatorial explosion involved in counting their frequency. While it is certainly possible to output all of the inferred frequent itemsets, we believe that this capability is best encapsulated as a function that can respond to user queries. This functionality can be embedded within a system supporting an interactive exploration of the associations. Constructing such a system is (certainly) in our future plans.

## Acknowledgements

The authors wish to thank Memo Dalkilic, Chris Giannella, Paul Purdom, Dirk Van Gucht and Cathy Wyss for several helpful suggestions.

## References

1. AGARWAL, R. C., AGGARWAL, C. C., AND PRASAD, V. V. Depth first generation of long patterns. In *SIGKDD 2000, Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 20-23, 2000, Boston, MA, USA* (2000), ACM Press, pp. 108–118.
2. AGRAWAL, R., AND SRIKANT, R. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile* (1994), J. B. Bocca, M. Jarke, and C. Zaniolo, Eds., Morgan Kaufmann, pp. 487–499.
3. BAYARDO JR., R. J. Efficiently mining long patterns from databases. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA* (1998), L. M. Haas and A. Tiwary, Eds., ACM Press, pp. 85–93.
4. BAYARDO JR., R. J., AGRAWAL, R., AND GUNOPULOS, D. Constraint-based rule mining in large, dense databases. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Australia* (1999), IEEE Computer Society, pp. 188–197.
5. BRIN, S., MOTWANI, R., AND SILVERSTEIN, C. Beyond market baskets: Generalizing association rules to correlations. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA* (1997), J. Peckham, Ed., ACM Press, pp. 265–276.

6. BRIN, S., MOTWANI, R., ULLMAN, J. D., AND TSUR, S. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA* (1997), J. Peckham, Ed., ACM Press, pp. 255–264.
7. CALDERS, T., AND PAREDAENS, J. Axiomatization of frequent sets. In *Proceedings of the 8th International Conference on Database Theory* (London, UK, January 4-6 2001), pp. 204–218.
8. HAN, J., AND FU, Y. Discovery of multiple-level association rules from large databases. In *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland* (1995), U. Dayal, P. M. D. Gray, and S. Nishio, Eds., Morgan Kaufmann, pp. 420–431.
9. LIU, B., HSU, W., AND MA, Y. Mining association rules with multiple minimum supports. In *Knowledge Discovery and Data Mining* (1999), pp. 337–341.
10. MEO, R., PSAILA, G., AND CERI, S. A new sql-like operator for mining association rules. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India* (1996), T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, Eds., Morgan Kaufmann, pp. 122–133.
11. SARAWAGI, S., THOMAS, S., AND AGRAWAL, R. Integrating mining with relational database systems: Alternatives and implications. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA* (1998), L. M. Haas and A. Tiwary, Eds., ACM Press, pp. 343–354.
12. SAVASERE, A., OMIECINSKI, E., AND NAVATHE, S. B. An efficient algorithm for mining association rules in large databases. In *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland* (1995), U. Dayal, P. M. D. Gray, and S. Nishio, Eds., Morgan Kaufmann, pp. 432–444.
13. SRIKANT, R., AND AGRAWAL, R. Mining generalized association rules. In *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland* (1995), U. Dayal, P. M. D. Gray, and S. Nishio, Eds., Morgan Kaufmann, pp. 407–419.
14. TOIVONEN, H. Sampling large databases for association rules. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India* (1996), T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, Eds., Morgan Kaufmann, pp. 134–145.