A short poster version to appear in KDD-2001

Real World Performance of Association Rule Algorithms

Zijian Zheng
Blue Martini Software
2600 Campus Drive
San Mateo, CA 94403, USA
+1 650 356 4223
zijian@bluemartini.com

Ron Kohavi
Blue Martini Software
2600 Campus Drive
San Mateo, CA 94403, USA
+1 650 356 4113
ronnyk@bluemartini.com

Llew Mason
Blue Martini Software
2600 Campus Drive
San Mateo, CA 94403, USA
+1 650 356 4136
lmason@bluemartini.com

ABSTRACT

Association rule discovery has been an active research area over the past few years with several new proposals for algorithms that improve the running time for generating association rules or frequent itemsets. Several new algorithms were shown by their authors to run faster then previously existing algorithms, although benchmarks were typically done on artificial datasets. Unlike classification algorithms, for which several large evaluations were done by third parties, there have been no such evaluations for the correctness and runtime performance of association algorithms. This study compares five well-known association rule algorithms using three real-world datasets and an artificial dataset from IBM Almaden. The experimental results confirm the performance improvements previously claimed by the authors on the artificial data, but some of these gains do not carry over to the real datasets, indicating overfitting of the algorithms to the IBM artificial dataset. More importantly, we found that the choice of algorithm only matters at support levels that generate more rules than would be useful in practice. For support levels that generate less than 1,000,000 rules, which is much more than humans can handle and is sufficient for prediction purposes where data is loaded into RAM, Apriori finishes processing in less than 10 minutes. On our datasets, we observed super-exponential growth in the number of rules. On one of our datasets, a 0.02% change in the support increased the number of rules from less than a million to over a billion, implying that outside a very narrow range of support values, the choice of algorithm is irrelevant.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Applications – *Data Mining*. I.2.6 [**Artificial Intelligence**]: Learning.

General Terms

Algorithms, Measurement, Performance, Experimentation.

Keywords

Data Mining, Association Rules, Benchmark, Comparisons, Frequent Itemsets, Market Basket Analysis, Affinity Analysis.

---

*A short version of this paper is published in Proceedings of the Seventh ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY: ACM,*2001.

1. INTRODUCTION

Practically anyone wishing to do affinity analysis on products, whether at a physical store or at an online store, will evaluate the use of association algorithms. In addition, online sites want to use such algorithms to evaluate page views that are associated in a session in order to improve the layout of the site or recommend related products to visitors. Because association algorithms are sound and complete algorithms, they should, at least in theory, generate the same output for the same parameters (e.g., support and confidence) on the same data. Whether you build your own association algorithm, whether you wish to license one, or you wish to research and develop a new algorithm, it is important to know how existing algorithms perform.

Over the last several years, the problem of efficiently generating large numbers of association rules has been an active research topic in the data mining community. Several different algorithms have been developed with promising results. However, the authors typically only show the performance advantages of their new algorithms using artificial datasets provided by IBM Almaden. To the authors' best knowledge, there has been no large third-party benchmark in the area of association rule discovery, while several such comparisons have been performed in other related areas of machine learning and data mining. For example, the Statlog project compared 22 different classification algorithms (including decision tree algorithms, rule algorithms, and neural networks) on 23 datasets [9]. MLC++ has several algorithms and a large comparison was done with 22 algorithms on eight datasets [5]. Lim et. al. compared 33 classification algorithms in terms of prediction accuracy, complexity, and training time on 32 datasets [7].

This paper has three main contributions. Firstly, we provide the first objective evaluation and comparison of several well-known association rule algorithms on real-world e-commerce and retail datasets. We are also donating one of these e-commerce datasets for use in the research community. Secondly, we show that the artificial datasets from IBM Almaden have very different characteristics from our real-world datasets. Optimizing algorithms for these artificial datasets can mislead research effort if algorithms will be applied to real-world datasets similar to ours. We are not against benchmark datasets; in fact we encourage more benchmarks and hope to see more comparisons with our donated dataset. However, the community has worked for several years optimizing association algorithms against variants of one artificial dataset and, as we show in the rest of the paper, recent comparisons that show improvement against these artificial data

do not show similar improvements on our real-world datasets. Thirdly, and perhaps most interestingly, we show that the association rule algorithms exhibit similar and surprising performance characteristics on our datasets. We demonstrate that for association rule generation, the choice of algorithm is irrelevant for a large range of choices of the minimum support parameter. For support levels that generate less than 100,000 rules, which is a very conservative upper bound for humans to sift through even considering pruning un-interesting rules, Apriori finishes on all datasets in less than 1 minute.[1] For support levels that generate less than 1,000,000 rules, which is sufficient for prediction purposes where data is loaded into RAM, Apriori finishes processing in less than 10 minutes. When the minimum support is smaller, and hence the number of frequent itemsets and association rules is very large, most algorithms either run out of memory or run over our 150GB of allowed disk space due to the huge number of frequent itemsets.

We first give a brief description of association rule discovery and related concepts in Section 2. Then, in Sections 3, 4 and 5, we describe the association rule algorithms and the datasets used in the benchmarks, respectively. Section 6 reports our experimental results and findings, followed by a discussion of the correctness of the algorithms in Section 7. We give conclusions in Section 8. The appendices contain tables giving detailed information about the datasets and benchmark results.

2. ASSOCIATION RULE GENERATION

An association is a rule of the format: LHS → RHS, where *LHS* and *RHS* stand for Left Hand Side and Right Hand Side respectively. These are two sets of items (or equivalently conjunctions of conditions) and do not share common items. The rule can be read as "IF LHS THEN RHS". A set of items is called an *itemset*.

The goal of association rule discovery is to find associations among items from a set of transactions, each of which contains a set of items. Not all of the association rules discovered within a transaction set are interesting or useful. Generally the algorithm finds a subset of association rules that satisfy certain constraints. The most commonly used constraint is *minimum support*. The support of a rule is defined as the support of the itemset consisting of both the LHS and the RHS.[2] The support of an itemset is the percentage of transactions in the transaction set that contain the itemset. An itemset with a support higher than a given minimum support is called *frequent itemset*. Similarly, a rule is frequent if its support is higher than the minimum support. Minimum confidence is another commonly used constraint for association rules. The *confidence* of a rule is defined as the ratio of the support of the rule and the support of the LHS. It is equivalent to the probability that a transaction contains the RHS if the transaction contains the LHS. A rule is confident if its confidence is higher than a given minimum confidence. *Lift* is another important measure for association rules. It is defined as the confidence of the rule divided by the probability that a transaction contains the RHS, indicating how much more confident we can be regarding that a transaction contains the RHS if we know the transaction contains the LHS. Generally speaking, a rule with high support, high confidence, and high lift is good. However, there is a trade-off among these three measures in practice.

Most association rule algorithms generate association rules in two steps:

1. Generate all frequent itemsets; and
2. Construct all rules using these itemsets.

The foundation of this type of algorithm is the fact that any subset of a frequent itemset must also be frequent, and that both the LHS and the RHS of a frequent rule must also be frequent. Therefore, every frequent itemset of size *n* can result in *n* association rules with a single item RHS. The first step, generating all frequent itemsets, is expensive in terms of computation, memory usage and I/O resources. Much of the research effort in association rule discovery has been devoted to improving the efficiency of this first step. The second step, generating rules from all of the frequent itemsets, is relatively straightforward, but it can still be very expensive when solving real-world problems.

We have only briefly described the most basic concepts of association rule discovery. For more detailed information, see related technical publications [1][2][4][12][13].

3. ALGORITHMS

In this section we describe the software implementations of the association rule algorithms used in our experiments. The five algorithms evaluated were *Apriori*, *Charm*, *FP-growth*, *Closet* and *MagnumOpus*. We provide references to articles describing the details of the algorithm when available and also specify the algorithms' parameter settings used in our experiments (if any). We started the experiments several months ago and published preliminary results to the authors of the algorithms. Several authors provided us with an updated version of their code to fix bugs and/or improve the performance. We reran our experiments with the new versions and noted below when updated versions were received.

Apriori: Apriori is Christian Borgelt's implementation of the well-known Apriori association rule algorithm [1][2]. The source code in C for this implementation is available under the "GNU Lesser General Public License" from http://fuzzy.cs.uni-magdeburg.de/~borgelt/.

Apriori takes transactional data in the form of one row for each pair of transaction and item identifiers. It first generates frequent itemsets and then creates association rules from these itemsets. It can generate both association rules and frequent itemsets. Apriori supports many different configuration settings. In our experiments, we used the percentage of transactions that satisfy both the LHS and the RHS of a rule as the support. We also specified that Apriori should load the entire dataset into memory

[1] Liu, Hsu, and Ma [8] report an average reduction of a factor 72 from all association rules to pruned positively correlated rules, and a factor 386 from all association rules to direction setting rules. Even with these pruning and summarization techniques, we still expect about 1389 pruned positive correlated rules and 259 direction setting rules, which in our opinion represents the maximum rule base size a business person will be willing to sift through.

[2] Sometimes this is defined using only the LHS.

rather than making multiple database scans.[3] Running Apriori using multiple database scans would be significantly slower.

Charm: Charm is an algorithm for generating closed frequent itemsets for association rules from transactional data [13]. A closed frequent itemset is a subset of the corresponding frequent itemset. This subset is necessary and sufficient to capture all of the information about the frequent itemset. The closed frequent itemsets are the smallest representative subset of a frequent itemset without loss of information (under the lattice definition in Zaki's paper). For the formal definition of a closed frequent itemset, see [13]. All possible association rules can be generated from the association rules created from only the closed frequent itemsets.

Charm takes transactional data in the form of one row for each single complete transaction, with a customer identifier, a transaction identifier, the number of items in the transaction, and a list of items. The Charm implementation used in our experiments was obtained from Mohammed Zaki on February 25, 2001, which was an improved version of an earlier version after our preliminary experimental results were shared with the author. This implementation of Charm only generates the closed frequent itemsets, and not the association rules.

FP-growth: FP-growth is an algorithm for generating frequent itemsets for association rules from Jiawei Han's research group at Simon Fraser University. It generates all frequent itemsets satisfying a given minimum support by growing a frequent pattern tree structure that stores compressed information about the frequent patterns. In this way, FP-growth can avoid repeated database scans and also avoid the generation of a large number of candidate itemsets [4].

Jiawei Han and Jian Pei provided the FP-growth implementation used in our experiments. We received the final version of this implementation on February 5, 2001, which significantly improved the earlier version we used after our preliminary experimental results were shared with the authors.

Closet: Closet is another frequent itemset generator for association rules from Jiawei Han's research group. Like Charm, it generates only the closed frequent itemsets using the minimum support constraint. For the details of this algorithm, see [10].

Again, Jiawei Han and Jian Pei provided the Closet implementation used in our experiments. We received this implementation on September 21, 2000. Both FP-growth and Closet take transactional data in the form of one row for each single complete transaction, with the number of items in the transaction followed by a list of items. These implementations of FP-growth and Closet only generate the frequent itemsets, and not the association rules.

MagnumOpus: MO [12] is the command line application shipped with the beta release of `MagnumOpus1.2`, a commercial system for association rule discovery. The main unique technique used in MagnumOpus is the search algorithm based on OPUS [11], a systematic search method with pruning. It considers the whole search space, but during the search, effectively prunes a large area of search space without missing search targets provided that the targets can be measured using certain criteria. Based on this technique, MagnumOpus can efficiently find top-N association rules with respect to a search criterion such as support or lift, which can be very useful in some applications.

Geoff Webb provided the most recent implementation of MO used in our experiments on February 1, 2001. MO directly generates association rules from a dataset based on a specified search preference. In addition to transactional data, it can also process the data format of the C5 rules induction engine, that is, MO can efficiently generate association rules from non-transactional datasets or transactional datasets augmented with other information. MO generates the top-N association rules based on sorting rules by the coverage, leverage, lift, strength, or support (see http://www.rulequest.com/MagnumOpus-info.html for the definitions of these measures). MO only generates rules where the RHS is a single item, while the LHS can be any size. To speed up rule generation, MO can take the following constraints as parameters: maximum size of the LHS, minimum coverage, minimum support, minimum leverage, minimum lift, and minimum strength. To compare with other association rule algorithms, we ran MO with the following settings: search based on support, all-associations, the default value (1 case) for minimum coverage, the default value (0 cases) for minimum leverage, minimum lift, and minimum strength. The maximum size of the LHS was set to 1000 to effectively remove this constraint. The minimum support was varied throughout the experiments. To evaluate whether MO is efficient when generating the top-N association rules, we ran the same set of experiments replacing "all-associations" with "max-associations=1000" to generate only the top 1000 rules. This configuration is indicated in the results section using MO-1000. We did not test MO with other pruning constraints such as strength, which is claimed to improve performance significantly, but changes the resulting associations generated.

4. DATASETS

We now describe the four datasets, *IBM-Artificial*, *BMS-POS*, *BMS-WebView-1* and *BMS-WebView-2*, used in our experiments. To make it easier to bridge our benchmarks with previously published experimental results we included the IBM-Artificial dataset, typically designated T10I4D100K, which is often used in the association rule research community. This dataset was generated using a transaction data generator obtained from IBM Almaden.[4]

The BMS-POS dataset contains several years worth of point-of-sale data from a large electronics retailer. Since this retailer has so many different products, we used product categories as items. Each item thus represents a category, rather than an individual product. The transaction in this dataset is a customer's purchase transaction consisting of all the product categories purchased at one time. The goal for this dataset is to find associations between product categories purchased by customers in a single visit to the retailer.

[3] Changing the main memory setting will not help Apriori finish the failed runs as the transactional dataset itself is not large.

[4] The data generator is available from http://www.almaden.ibm.com/cs/quest//syndata.html#assocSynData.

The BMS-WebView-1 and BMS-WebView-2 datasets contain several months worth of clickstream data from two e-commerce web sites. Each transaction in these datasets is a web session consisting of all the product detail pages viewed in that session. That is, each product detail view is an item. The goal for both of these datasets is to find associations between products viewed by visitors in a single visit to the web site. We are making the BMS-WebView-1 dataset available to the research community. This dataset comes from a small dot-com company called Gazelle.com, a legwear and legcare retailer, which no longer exists; a portion of their data was used in the KDD-Cup 2000 competition [6].[5]

Table 1 Dataset characteristics

	Transac-tions	Distinct Items	Maximum Trans. Size	Average Trans. Size
IBM-Artificial	100,000	870	29	10.1
BMS-POS	515,597	1,657	164	6.5
BMS-WebView-1	59,602	497	267	2.5
BMS-WebView-2	77,512	3,340	161	5.0

Table 1 characterizes the four datasets in terms of the number of transactions, the number of distinct items, the maximum transaction size, and the average transaction size. Figure 1 shows the distributions of transaction sizes in the four datasets. Part of the detailed transaction size distribution data is available in Table 15 and Table 16 in Appendix D. It is very clear that the artificial dataset has a very different transaction size distribution from the three real-world datasets, while the distributions of the three real-world datasets are similar.

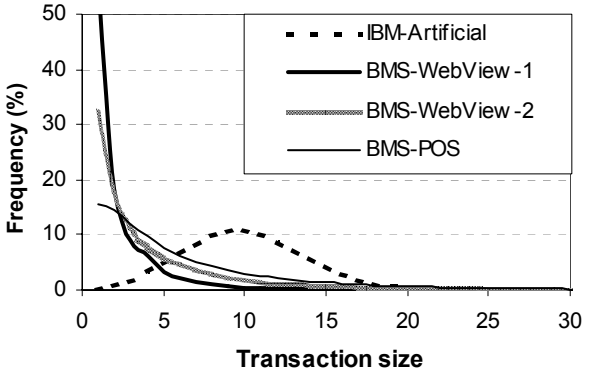

Figure 1. Dataset transaction size distribution.

It is worth mentioning that all four benchmark datasets contain sparse data, since most association rules discovery algorithms were designed for these types of problems [12]. On non-sparse datasets, the number of frequent itemsets and the number of association rules grow even faster than what we see in these experiments. For example, Zaki [13] showed that the number of rules with single RHS grows from 1,846 to 170,067 when the minimum support reduces from 97% to 90% on the Connect domain from UCI [3]. Such high minimum supports are uninteresting for many real world applications.

[5] After receiving a username and password from the KDD-Cup 2000 home page at http://www.ecn.purdue.edu/KDDCUP, the dataset can be downloaded from http://www.ecn.purdue.edu/KDDCUP/data/BMS-WebView-1.dat.gz.

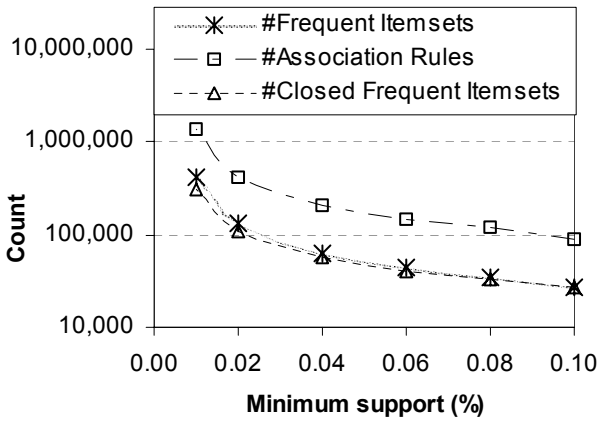

Figure 2. The number of frequent itemsets and association rules at different minimum support levels on IBM-Artificial. The two lines for frequent itemsets and closed frequent items overlap at many data points.

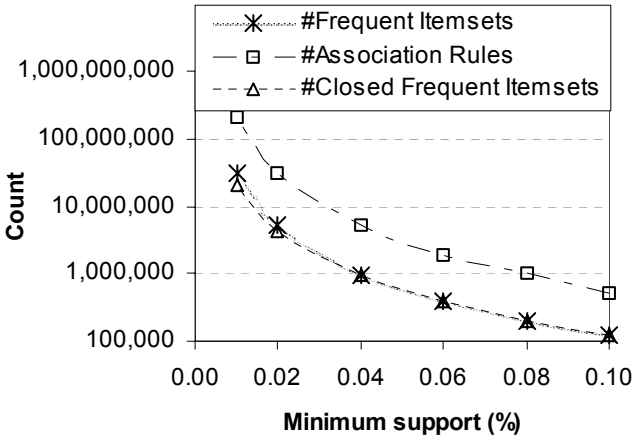

Figure 3. The number of frequent itemsets and association rules at different minimum support levels on BMS-POS. The two lines for frequent itemsets and closed frequent items overlap at many data points.

5. SUPER EXPONENTIAL GROWTH

Figures 2 through 5 show how the number of frequent itemsets, the number of association rules, and the number of closed frequent itemsets increase as the minimum support is reduced on each of the four datasets. The detailed numbers are available in Tables 3 through 6 in Appendix A. In all of these figures the number of closed frequent itemsets is the number produced by Charm. We can see that the number of rules and frequent itemsets

increases *super-exponentially* on the three real-world datasets, in contrast to the IBM-Artificial dataset, where the number of rules and frequent items only increases exponentially (probably because of the significantly larger average transaction size for the artificial dataset). This, in conjunction with the difference between the transaction size distributions, shows that the real-world datasets have fundamentally different characteristics to the artificial

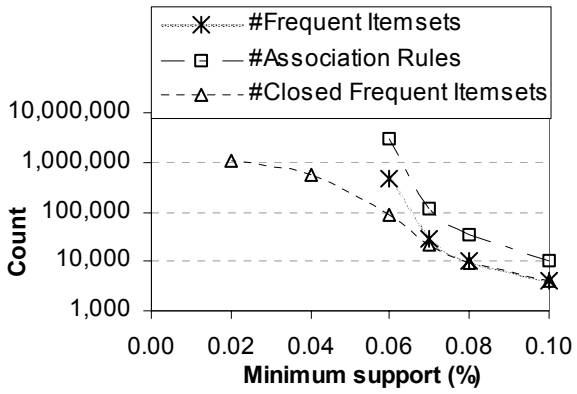

Figure 4. The number of frequent itemsets and association rules at different minimum support levels on BMS-WebView-1. The two lines for frequent itemsets and closed frequent items overlap at many data points.

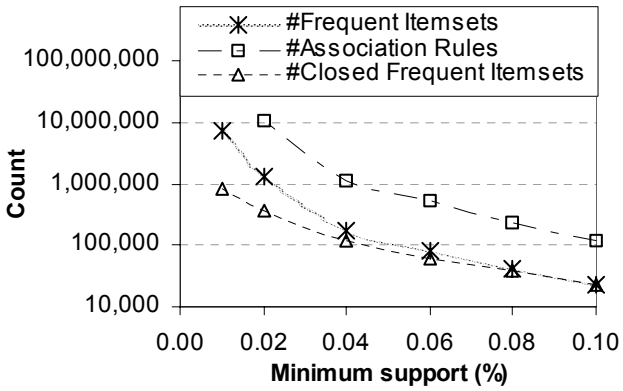

Figure 5. The number of frequent itemsets and association rules at different minimum support levels on BMS-WebView-2. The two lines for frequent itemsets and closed frequent items overlap at many data points.

datasets typically used to benchmark the performance of association rule algorithms.

Figures 6 and 7 show the ratio of the number of frequent itemsets over the number of closed frequent itemsets on BMS-POS and BMS-WebView-2 as two examples. The exponential growth in the ratio when the minimum support decreases indicates the big potential advantage of creating closed frequent itemsets over frequent itemsets in terms of execution time and memory needed.

In Tables 3 through 6 in Appendix A, "Failed" indicates that we do not have that value for the cell as the corresponding algorithm failed due to bugs, out of memory, or over our 150GB disk space limit. The three numbers marked using "*" in Table 5 were estimated to give a feeling of how the number of frequent itemsets can grow on a domain like BMS-WebView-1. We could not get the true numbers for them as the algorithms ran over the disk space limit at these minimum support levels on this dataset. To estimate these numbers, we ran FP-growth once more at the

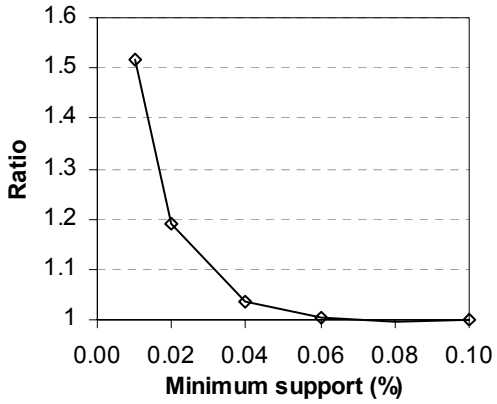

Figure 6. The ratio of the number frequent itemsets over the number of closed frequent itemsets on BMS-POS

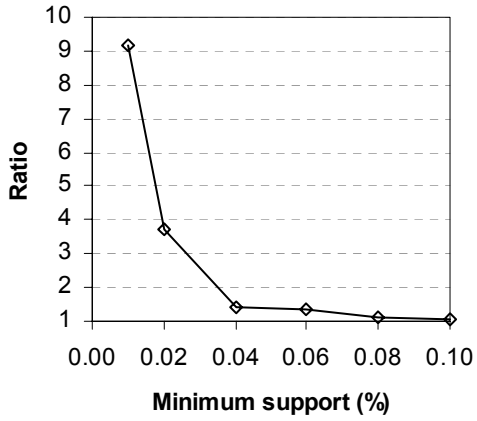

Figure 7. The ratio of the number of frequent itemsets over the number of closed frequent itemsets on BMS-WebView-2

minimum support level of 0.058%, resulting in 1,177,607 frequent itemsets. We applied Log_{10} three times on the number of frequent itemsets, resulting in a super-exponential curve. To make a conservative estimation, we created a linear function using the last three points (corresponding to the support levels 0.07%, 0.06%, and 0.058%) on the curve. The three estimated numbers were computed using this function. We have three pieces of evidence indicating that the estimations are conservative: 1. The linear function is used to estimate the super-exponential curve. 2. The estimated value from the function for the level 0.058% is 1,078,571, much smaller than the true value 1,177,607. 3. The file containing the 1,177,607 frequent itemsets generated by FP-growth was 57MB, so the file containing the 3,782,775 frequent

itemsets estimated at the level of 0.055% should be 183MB. However, when running FP-growth at this level, the frequent itemsets file was over 150GB when it ran out of disk space, indicating a much faster growth. All these results suggest that for some real-world datasets, when the minimum support is low, there are too many frequent itemsets to be useful.

6. EXPERIMENTS

Each of the five algorithms described in Section 3 was tested on the four datasets described in Section 4. The performance measure was the execution time (seconds) of the algorithms on the datasets with the following minimum support settings 1.00%, 0.80%, 0.60%, 0.40%, 0.20%, 0.10%, 0.08%, 0.06%, 0.04%, 0.02%, and 0.01%. The minimum confidence was always set to zero. That is, we required no minimum confidence for the generated association rules. Since some of the algorithms could only generate frequent itemsets, and some others could directly generate association rules, we measured the execution time for both creating the frequent itemsets and for creating the association rules whenever possible. Note that time for generating the association rules includes the computation for generating the frequent itemsets.

The computer used to run the experiments had dual 550MHz Pentium III Xeon processors and 1 GB of memory. The operating system used was Windows NT 4.0. To make the time measurements more reliable, no other application was running on the machine while the experiments were running. Although none of the algorithms supported parallel processing, the second processor helped to stabilize the measured results since system processes could run on the other processor. To verify the stability of the timing, we ran Apriori five times to generate the frequent itemsets for BMS-WebView-1 with a minimum support level of 0.06%. The average running time was 110.5 seconds, with a standard deviation of only 0.1 seconds.

6.1 Generating Frequent Itemsets

Figures 8 though 11 show performance curves for the four algorithms that generate frequent itemsets, namely Closet, FP-growth, Charm, and Apriori, on IBM-Artificial, BMS-POS, BMS-WebView-1, and BMS-WebView-2 respectively. The vertical axis is on a logarithmic scale (base 10). For each figure, the bottom chart shows performance curves for the full range of minimum supports from 0.01% to 1.00%, while the top chart highlights the sub-range of the minimum supports from 0.01% to 0.10%. The detailed values for these results are listed in Tables 7 through 10 in Appendix B. MagnumOpus is not included in these results because it cannot generate frequent itemsets.

The first thing to notice about these results is that when the minimum support is large,[6] such as 0.10% for BMS-WebView-1 or 0.40% for BMS-POS, all algorithms finish within a minute, and hence the choice of algorithm in these cases is irrelevant. In the remaining analysis, we focus only on the results where the minimum supports ranges from 0.01% to 0.10% (the top chart in each figure).

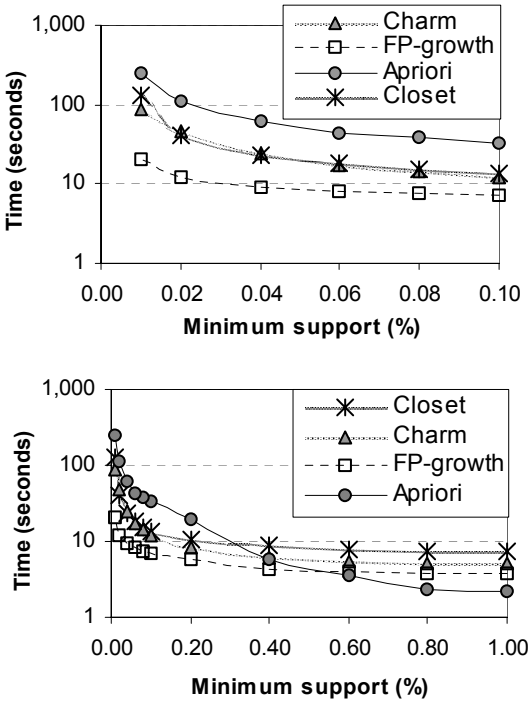

Figure 8. Running time of the algorithms for generating frequent itemsets for IBM-Artificial.

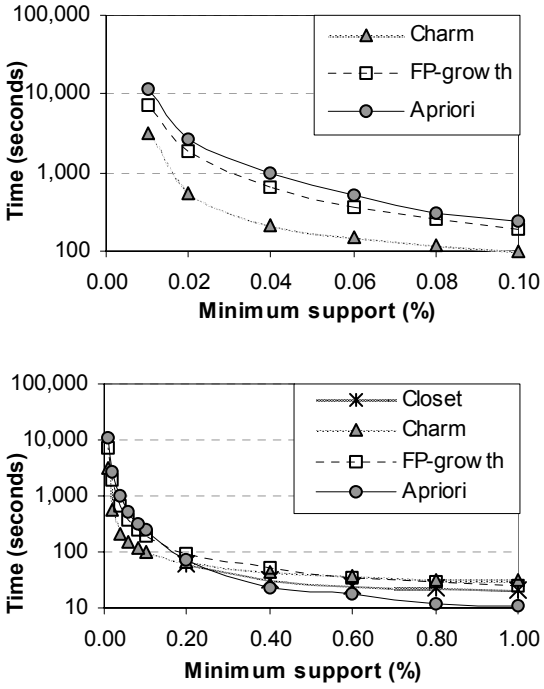

Figure 9. Running time of the algorithms for generating frequent itemsets for BMS-POS.

[6] The definition of large is obviously dependent on the dataset.

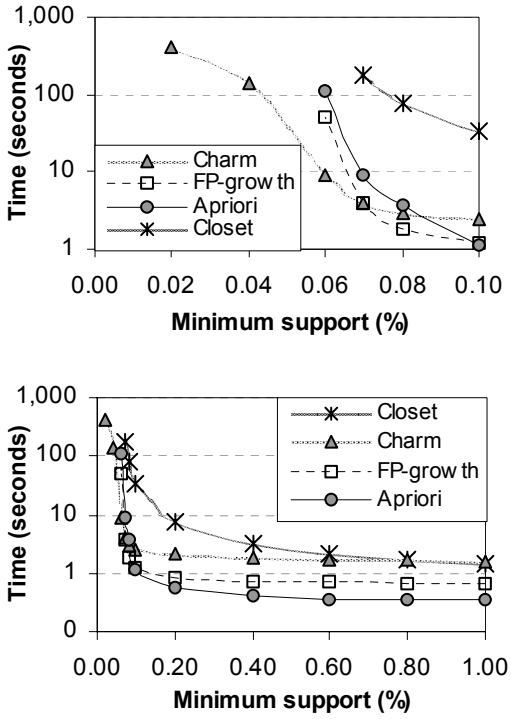

Figure 10. Running time of the algorithms for generating frequent itemsets for BMS-WebView-1.

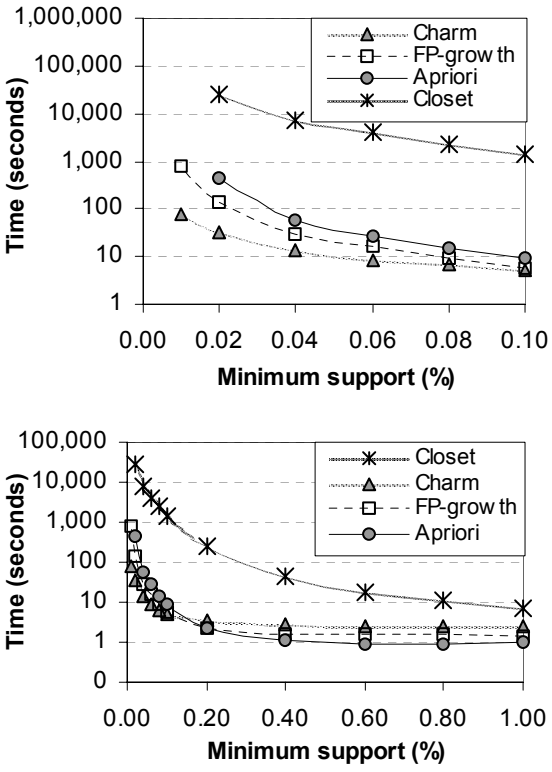

Figure 11. Running time of the algorithms for generating frequent itemsets for BMS-WebView-2.

For the artificial dataset, every algorithm outperforms Apriori by a significant margin for minimum support values less than 0.10%. FP-growth and Closet are one order of magnitude faster than Apriori when the minimum support reaches 0.02%, which is consistent with the results reported for previous experiments [4][10][13]. It is clear that the performance improvement of FP-growth over Apriori increases as the minimum support decreases, indicating that FP-growth scales better than Apriori. The results also show that FP-growth is consistently and significantly faster than Charm and Closet. Charm is faster than Closet at all minimum support levels except 0.02% and 0.04%, which differs from the results reported in [10] where Closet is faster than Charm. One explanation for this is that we are using a new version of Charm which has some newly developed techniques, while Closet is the old version. Based on our experimental results, the latest implementation is significantly faster than previous versions. To summarize, these results on the artificial dataset support the conclusion that the new algorithms have significant performance improvements over Apriori.

From the results for the three real-world datasets (Figures 9, 10, and 11) we make the following observations:

1. For BMS-WebView-1, Closet is much slower than every other algorithm for minimum support ranging from 0.10% to 0.06%. At minimum support levels below 0.06%, Closet used more than the 1GB available memory. The same trend exists on BMS-WebView-2. For BMS-POS, Closet ran out of memory once the minimum support level reached 0.10%. These results suggest Closet does not scale well on the real-world datasets, which is contrary to the results on the artificial dataset.
2. For all of the real-world datasets, FP-growth is faster than Apriori, but the differences are not as large as on the artificial dataset.
3. For all of the real-world datasets, Charm is much faster than Apriori. The improvement is one order of magnitude on BMS-POS when the minimum support is 0.02% or 0.01%, and on BMS-WebView-2 when the minimum support is 0.06%, 0.04%, or 0.02%. It is two orders of magnitude faster on BMS-WebView-1 when the minimum support is 0.06%. On these datasets, Charm is also faster than FP-growth.
4. Table 2 summarizes the rankings of the four algorithms for generating frequent itemsets on the four datasets for both high minimum supports and low minimum supports used in the experiments. It is clear that with high minimum supports, Apriori is always faster than the others, and Charm is always slower than Apriori and FP-growth. The reason could be that the frequent itemsets are very short (with 2 or 3 items) and the numbers of frequent itemsets are small with these high minimum supports (see Table 3 through Table 6), and Apriori can handle this type of problem quickly with very little overhead while others have some overhead. It seems that Charm's overhead is larger than FP-

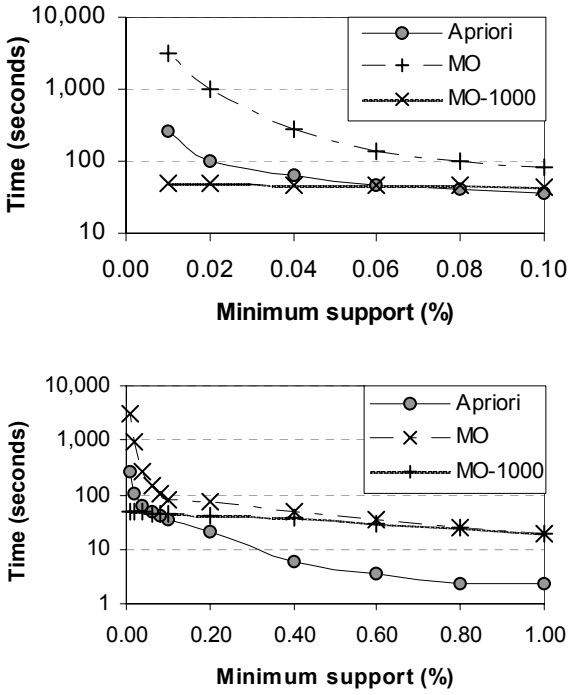

Figure 12. Running time of the algorithms for generating association rules for IBM-Artificial.

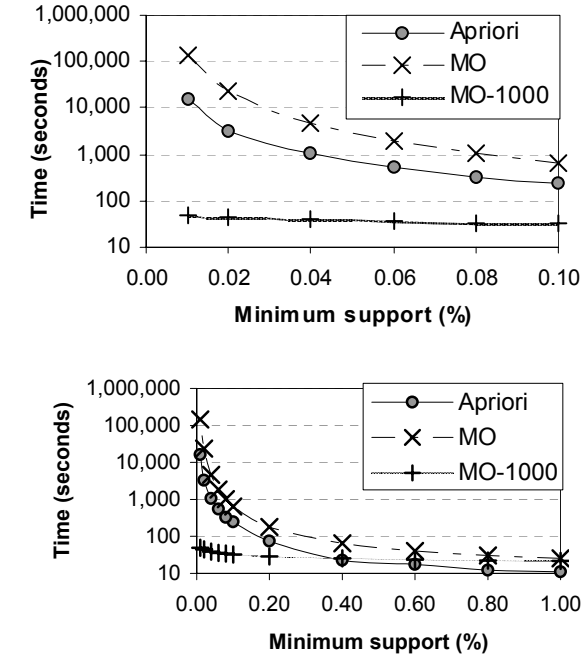

Figure 13. Running time of the algorithms for generating association rules for BMS-POS.

growth's. On the three real-world datasets with low minimum supports, the ranking is always Charm > FP-growth > Apriori > Closet. This shows the advantages of the techniques in Charm and FP-growth when there are many long frequent itemsets. Generating significantly smaller numbers of frequent itemsets (only closed) may partially explain the fast execution of Charm.

5. None of the algorithms can handle too many frequent itemsets. On BMS-WebView-1, with a minimum support of 0.04%, both Apriori and FP-growth ran out of disk space after writing over 150GB of data to disk. Since Charm generated significantly less closed frequent itemsets than all frequent itemsets on this dataset, it continued to work until the minimum support reached 0.01% when it ran out of memory. Even if a much larger disk was available, it is questionable as to how one could make use of so many frequent itemsets. Since most algorithms (MagnumOpus being a notable exception) generate all frequent itemsets as the basis for generating associations, all these algorithms will fail on this initial step of the overall process. Our results indicate that if we need association rules for low level of support, we need better techniques to filter association rules and avoid generating frequent itemsets in order to output a manageable number of associations.

Table 2. Rankings (left is better) of the algorithms for generating frequent itemsets on the four datasets with high minimum supports and low minimum supports (Ap: Apriori, FP: FP-growth, Ch: Charm, Cl: Closet)

	High Min-Support	Low Min-Support
IBM-Artificial	Ap > FP > Ch > Cl	FP > Ch > Cl > Ap
BMS-POS	Ap > Cl > FP > Ch	Ch > FP > Ap > Cl
BMS-WebView-1	Ap > FP > Cl > Ch	Ch > FP > Ap > Cl
BMS-WebView-2	Ap > FP > Ch > Cl	Ch > FP > Ap > Cl

6.2 Generating Association Rules

Figures 12 through 15 show performance curves of the two algorithms that can generate association rules, MagnumOpus and Apriori, on the four datasets, IBM-Artificial, BMS-POS, BMS-WebView-1, and BMS-WebView-2, respectively. Since MagnumOpus has an option for generating the top-N association rules, we include MO-1000 for generating the top-1000 rules sorted by support. The vertical time axis is on a logarithmic scale (base 10). The time includes the running time for both creating the frequent itemsets and creating the association rules. In each figure, the bottom chart shows performance curves for the full range of the minimum support from 0.01% to 1.00%, while the top chart highlights the sub-range of the minimum support from 0.01% to 0.10%. The detailed values of these results are listed in Tables 11 through 14 in Appendix C. Charm, FP-growth, and Closet are not included in these results because they do not generate association rules. As mentioned in the previous section, we focus on the results where the minimum support ranges from 0.01% to 0.10%.

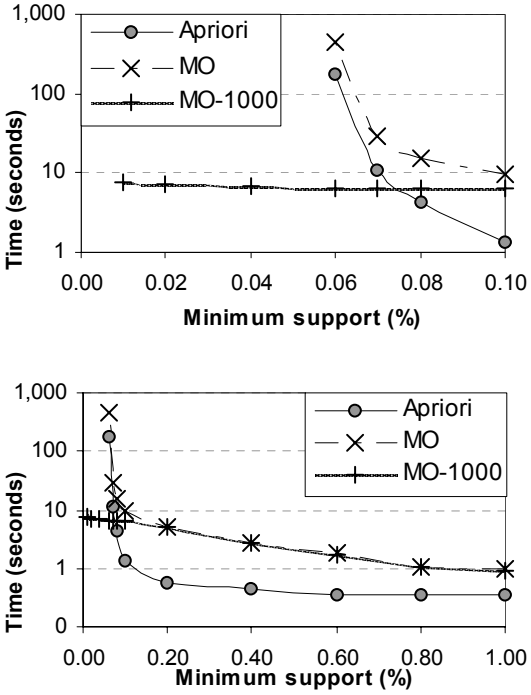

Figure 14. Running time of the algorithms for generating association rules for BMS-WebView-1.

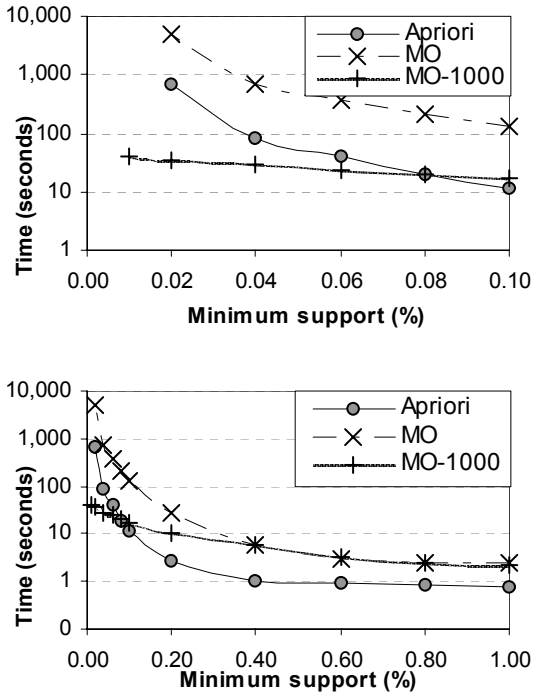

Figure 15. Running time of the algorithms for generating association rules for BMS-WebView-2.

We make the following observations from these results:

1. Unrestricted MagnumOpus (MO) is significantly slower than Apriori on all of the datasets. This is not surprising, because these datasets are sparse and MagnumOpus was not designed for this type of data [12].
2. Both Apriori and MagnumOpus ran out of disk space while generating the rules for BMS-WebView-1 when the minimum support level reached 0.04%. This was due to the fact that the number of rules created exceeded the available disk space of 150GB.
3. MO-1000's running time grows very slowly as the minimum support level decreases, indicating that it provides a solution when the number of association rules is very large and only the top-N rules are needed (based on some criteria such as lift or confidence).

7. CORRECTNESS OF THE ALGORITHMS

Apriori and FP-growth generated the same frequent itemset in every experiment where we verified the generated itemsets by hand.

From Tables 4 and 5, we can see that Charm generates more closed frequent itemsets than there are frequent itemsets. The reason for this disparity seems to be that it rounds a real-valued number down to an integer in evaluating the absolute minimum support, while all of the other algorithms round up. For example, on BMS-WebView-1 with a minimum support of 1.00%, Apriori generates 77 frequent itemsets while Charm generates 78 closed frequent itemsets. We compared these two sets by hand, and found that the 77 itemsets created by Apriori are contained within the 78 itemsets generated by Charm. The one extra itemset is {32201} and has a support of 596. This BMS-WebView-1 dataset has 59,602 transactions and a minimum support of 1.00% corresponds to 596.02. Apriori rounds this up to 597, while Charm rounds it down to 596. Charm therefore generates a frequent set that is sometimes slightly below the minimum support.

Charm and Closet do not match in terms of the number of closed frequent itemsets in some experiments, and the difference is large in some cases. For example, for the IBM-Artificial dataset, with a minimum support of 0.40%, Charm and Closet generated 1997 and 1992 closed frequent itemsets respectively. However, with a minimum support of 0.01% they generated 303,610 and 283,397 closed frequent items respectively, a difference of 20,213. Note that there can be no rounding problem for this dataset since it has exactly 100,000 transactions. Therefore, one or both of these implementations seems to generate incorrect closed frequent itemsets in some cases.

For generating association rules, MagnumOpus and Apriori do not match in terms of the number of rules. One explanation is that Apriori generates association rules with an empty LHS while MagnumOpus does not. However, even taking this into account, the number of generated rules still doesn't match. For example, on the IBM-Artificial dataset, with a minimum support of 0.40%,

MagnumOpus generated 3,649 rules, while Apriori generated 4,278 rules with 3,648 rules having a non-empty LHS. The two numbers differ by one, suggesting that one or both of these implementations seems to generate incorrect rules in some cases.

These differences should clearly raise concern since these algorithms should all be sound and complete and generate exactly the same associations for the same input parameters. Authors consistently showed improvements in execution time but rarely mentioned correctness.

8. CONCLUSIONS

To better understand the performance characteristics of association rule algorithms in the e-commerce and retail domains, we benchmarked five well-known association rule algorithms (Apriori, FP-growth, Closet, Charm, and Magnum-Opus) using three real-world datasets (two e-commerce and one retail) and an IBM Almaden artificial dataset.

The results show that the association rule algorithms that we evaluated perform differently on our real-world datasets than they do on the artificial dataset. The performance improvements reported by previous authors can be seen on the artificial dataset, but some of these gains do not carry over to the real datasets, indicating that these algorithms overfit the IBM artificial dataset. The primary reason for this seems to be that the artificial dataset has very different characteristics, suggesting the need for researchers to improve the artificial datasets used for association rule research or use more real-world datasets. We donated one such dataset for research use.

We also found that the choice of algorithm only matters at support levels that generate more rules than would be useful in practice. For a support level that generates a small enough number of rules that a human could understand, Apriori finishes on all datasets in under a minute, so performance improvements are not very interesting. Even for support levels that generate around 1,000,000 rules, which is far more than humans can handle and is typically sufficient for prediction purposes, Apriori finishes processing in less than 10 minutes. Beyond this level of support, the number of frequent itemsets and association rules grows extremely quickly on the real-world datasets, and most algorithms quickly run out of either memory or reasonable disk space.

It would be interesting to see whether the artificial datasets have a closer similarity to real-world supermarket datasets (the traditional domain for market basket analysis using association rules). Also, when generating association rules, we didn't experiment with many of the different parameter settings (e.g., the minimum confidence level), and it would be interesting to see how these parameters impact performance and the number of associations generated.

9. ACKNOWLEDGMENTS

We would like to thank Mohammed Zaki for providing the Charm algorithm, Jiawei Han and Jian Pei for providing the FP-growth and Closet algorithms, and Geoff Webb for providing the MagunmOpus algorithm.

10. APPENDICES

A. Frequent Itemset Size, Association Rule Sizes, and Longest Frequent Itemset Length

Table 3. IBM-Artificial

Minimum Support (%)	Frequent Itemsets	Length of Longest Frequent Itemsets	Association Rules	Closed Frequent Itemsets
1.00	385	3	396	385
0.80	494	3	552	494
0.60	772	5	1,110	772
0.40	2,001	6	4,278	1,997
0.20	13,255	9	44,315	13,151
0.10	27,532	10	91,147	26,962
0.08	35,019	10	118,254	32,676
0.06	44,583	10	148,255	41,215
0.04	62,864	11	204,439	56,962
0.02	129,875	12	413,478	110,824
0.01	411,365	13	1,376,684	303,610

Table 4. BMS-POS

Minimum Support (%)	Frequent Itemsets	Length of Longest Frequent Itemsets	Association Rules	Closed Frequent Itemsets
1.00	1,099	2	2,793	1,099
0.80	1,695	2	4,543	1,695
0.60	2,915	3	8,421	2,915
0.40	6,646	3	21,234	6,656
0.20	27,641	4	103,449	27,703
0.10	122,449	9	530,353	122,369
0.08	200,595	9	989,411	201,098
0.06	382,663	10	1,837,824	380,911
0.04	973,385	11	5,061,105	938,534
0.02	5,174,660	12	30,702,323	4,348,009
0.01	31,669,064	14	214,300,568	20,888,059

Table 5. BMS-WebView-1

Minimum Support (%)	Frequent Itemsets	Length of Longest Frequent Itemsets	Association Rules	Closed Frequent Itemsets
1.00	77	2	87	78
0.80	105	2	122	105
0.60	162	3	195	162
0.40	286	3	404	288
0.20	798	4	1,516	808
0.10	3,991	6	10,360	3,977
0.08	10,286	9	33,514	9,522
0.07	27,403	11	110,925	21,142
0.06	461,521	15	3,011,836	86,506
0.04	* 6.82×10^{10}	Failed	Failed	567,905
0.02	* 1.08×10^{26}	Failed	Failed	1,038,951
0.01	* 1.78×10^{45}	Failed	Failed	Failed

Table 6. BMS-WebView-2

Minimum Support (%)	Frequent Itemsets	Length of Longest Frequent Itemsets	Association Rules	Closed Frequent Itemsets
1.00	81	3	108	82
0.80	138	3	203	138
0.60	257	4	452	257
0.40	676	5	1,504	679
0.20	3,683	7	12,665	3,743
0.10	23,294	10	119.335	22,256
0.08	40,855	11	230,996	37,285
0.06	80.925	13	510,233	59.209
0.04	168,375	15	1.096.720	118,434
0.02	1,316,614	17	10,448,483	351,722
0.01	7,313,361	20	Failed	799,764

B. Running Time for Generating Frequent Itemsets

All tables contain the time in seconds spent creating frequent itemsets.

Table 7. IBM-Artificial

Minimum Support (%)	Closet	Charm	FP-growth	Apriori
1.00	7.2	5.1	3.7	2.2
0.80	7.4	5.3	3.8	2.3
0.60	7.9	5.5	4.0	3.7
0.40	8.6	6.2	4.3	5.7
0.20	10.9	8.6	5.8	19.5
0.10	13.8	12.2	7.1	33.5
0.08	15.3	14.1	7.6	37.9
0.06	17.9	17.4	8.2	43.9
0.04	23.2	24.8	9.2	61.3
0.02	40.8	47.4	11.9	109.5
0.01	131.1	89.0	20.2	244.9

Table 8. BMS-POS

Minimum Support (%)	Closet	Charm	FP-growth	Apriori
1.00	20.7	30.8	24.5	11.0
0.80	22.5	32.2	29.0	12.2
0.60	25.3	36.8	35.4	17.3
0.40	30.9	45.5	50.3	22.4
0.20	62.0	66.0	94.5	73.6
0.10	Failed	101.3	192.1	237.5
0.08	Failed	117.7	256.9	311.5
0.06	Failed	146.8	367.2	516.6
0.04	Failed	211.2	641.3	969.2
0.02	Failed	540.6	1,916.7	2,684.5
0.01	Failed	3,134.1	7,191.5	11,150.2

Table 9. BMS-WebView-1

Minimum Support (%)	Closet	Charm	FP-growth	Apriori
1.00	1.4	1.6	0.7	0.4
0.80	1.7	1.7	0.7	0.4
0.60	2.2	1.8	0.7	0.4
0.40	3.2	1.8	0.7	0.4
0.20	7.4	2.1	0.8	0.6
0.10	33.1	2.5	1.2	1.2
0.08	77.8	3.0	1.8	3.7
0.07	180.9	4.0	3.8	9.2
0.06	Failed	8.8	52.1	110.4
0.04	Failed	144.1	Failed	Failed
0.02	Failed	408.7	Failed	Failed
0.01	Failed	Failed	Failed	Failed

Table 10. BMS-WebView-2

Minimum Support (%)	Closet	Charm	FP-growth	Apriori
1.00	6.9	2.5	1.5	1.0
0.80	10.6	2.5	1.5	0.9
0.60	18.5	2.7	1.5	0.9
0.40	45.9	2.9	1.7	1.1
0.20	241.9	3.5	2.3	2.4
0.10	1,458.3	5.2	5.9	9.1
0.08	2,380.8	6.6	8.8	14.4
0.06	3,950.8	8.5	15.7	27.8
0.04	7,529.8	13.4	29.0	58.2
0.02	26,662.8	33.5	136.5	423.8
0.01	Failed	78.5	792.5	Failed

C. Running Time for Creating Association Rules

All tables contain the time in seconds spent creating association rules.

Table 11. IBM-Artificial

Minimum Support (%)	Apriori	MO	MO-1000
1.00	2.3	18.9	18.7
0.80	2.4	25.8	23.8
0.60	3.6	33.6	29.9
0.40	5.8	50.4	37.0
0.20	20.2	71.7	41.9
0.10	34.8	82.9	44.8
0.08	39.8	100.5	45.7
0.06	47.5	141.2	46.4
0.04	65.0	272.0	47.4
0.02	102.4	970.9	48.2
0.01	268.0	3,137.0	48.7

Table 12. BMS-POS

Minimum Support (%)	Apriori	MO	MO-1000
1.00	11.1	25.5	21.8
0.80	12.3	30.6	22.3
0.60	17.4	40.3	23.6
0.40	22.7	65.2	25.5
0.20	75.4	179.1	28.7
0.10	246.6	643.8	32.4
0.08	327.6	1,015.0	34.1
0.06	550.0	1,864.5	35.9
0.04	1,064.8	4,576.8	39.0
0.02	3,300.5	23,559.7	44.7
0.01	16,084.3	143,701.8	49.5

Table 13. BMS-WebView-1

Minimum Support (%)	Apriori	MO	MO-1000
1.00	0.4	1.0	0.9
0.80	0.4	1.0	1.0
0.60	0.4	1.8	1.7
0.40	0.4	2.7	2.7
0.20	0.6	5.2	5.2
0.10	1.3	9.7	6.3
0.08	4.2	15.1	6.4
0.07	11.1	29.5	6.4
0.06	176.8	453.1	6.6
0.04	Failed	Failed	6.8
0.02	Failed	Failed	7.3
0.01	Failed	Failed	7.7

Table 14. BMS-WebView-2

Minimum Support (%)	Apriori	MO	MO-1000
1.00	0.8	2.4	2.3
0.80	0.9	2.4	2.4
0.60	0.9	3.1	3.2
0.40	1.1	5.9	5.8
0.20	2.6	27.0	10.1
0.10	11.4	127.2	17.1
0.08	19.3	205.6	19.9
0.06	40.2	377.7	23.8
0.04	84.7	712.8	28.7
0.02	695.5	4,860.6	35.4
0.01	Failed	Failed	40.1

D. Distribution of Dataset Transaction Sizes

Table 15. Distribution of transaction sizes for each dataset (I)

	Number of Transactions by Transaction Size				
	1	**2**	**3**	**4**	**5**
IBM-Artificial	128	545	1,607	3,287	4,849
BMS-POS	80,278	75,431	61,242	50,124	39,908
BMS-WebView-1	32,935	10,861	5,378	3,591	1,972
BMS-WebView-2	24,893	13,548	8,826	6,204	4,573

Table 16. Distribution of transaction sizes for each dataset (II)

	Number of Transactions by Transaction Size					
	6	**7**	**8**	**9**	**10**	**> 10**
IBM-Artificial	6,525	7,990	9,759	10,471	10,892	43,947
BMS-POS	32,229	26,050	22,237	18,236	15,378	94,484
BMS-WebView-1	1,220	908	639	420	315	1,363
BMS-WebView-2	3,624	2,734	2,139	1,736	1,401	7,834

11. REFERENCES

[1] Agrawal, R., Imielinski, T., and Swami, A. Mining associations between sets of items in massive databases. In Proceedings of the ACM-SIGMOD International Conference on Management of Data, 207-216.

[2] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A.I. Fast discovery of association rules. In U. Fayyad et al. (eds), Advances in Knowledge Discovery and Data Mining, Menlo Park, CA: AAAI Press, 307-328.

[3] Blake, C.L., and Merz, C.J. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Dept of Information and Computer Science, 1998.

[4] Han, J., Pei, J., and Yin, Y. Mining frequent patterns without candidate generation. In Proceedings of ACM-SIGMOD International Conference on Management of Data.

[5] Kohavi, R., Sommerfield, D., and Dougherty, J. Data mining using MLC++: A machine learning library in C++, International Journal on Artificial Intelligence Tools 6(4), 537-566. http://www.sgi.com/Technology/mlc.

[6] Kohavi, R., Brodley, C.E., Frasca, B., Mason, L., and Zheng, Z. KDD-Cup 2000 Organizers' Report: Peeling the Onion, SIGKDD Exploration 2(2), 2000, 86-93.

[7] Lim, T., Loh, W., and Shih, Y. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, to appear in Machine Learning (available from http://www.recursive-partitioning.com/Bibliography/Journal_Articles/Theory_and_Methodology/more2.html).

[8] Liu, B., Hsu, W., and Ma, Y. Pruning and summarizing the discovered associations. In Proceedings of the Fifth ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY: ACM, 125-134.

[9] Michie, D., Spiegelhalter, D.J., and Taylor, C.C. (eds.). Machine Learning, Neural and Statistical Classification (STATLOG Project), Herfordshire: Ellis Horwood.

[10] Pei, J., Han, J., and Mao, R. CLOSET: An efficient algorithm for mining frequent closed itemsets. In Proceedings of ACM_SIGMOD International Workshop on Data Mining and Knowledge Discovery.

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

[12] Webb, G.I. Efficient search for association rules. In Proceedings of the Sixth ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY: ACM, 99-107.

[13] Zaki, M.J. Generating non-redundant association rules. In Proceedings of the Sixth ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY: ACM, 34-43.