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

This study compares five well-known association rule algorithms using three real-world datasets and an artificial dataset. 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*.

Keywords

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

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 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 [3][6].

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

[1] Liu, Hsu, and Ma [5] 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.

memory or run over our 150GB of allowed disk space due to the huge number of frequent itemsets.

A long version of this paper including the detailed data description, experimental results, and a discussion about the algorithms' correctness is available [11].

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 and do not share common items. 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. 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, which is the minimum ratio of the support of the rule and the support of the LHS, is another commonly used constraint for association rules.

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 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 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][9][10].

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 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 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]. 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.

Charm: Charm is an algorithm for generating closed frequent itemsets for association rules from transactional data [10]. 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 frequent itemsets without loss of information (under the lattice definition in Zaki's paper). For the formal definition of a closed frequent itemset, see [10]. 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. 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 [2].

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 [7].

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. These implementations of FP-growth and Closet only generate the frequent itemsets, and not the association rules.

MagnumOpus: MO [9] 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 [8], 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

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

technique, MagnumOpus can efficiently find top-N association rules with respect to a search criterion such as support or lift.

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. 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 and all-associations. The maximum size of the LHS was set to 1000 to effectively remove this constraint. The minimum support was varied throughout the experiments. The default values for other parameters were used. 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.

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 data generator obtained from IBM Almaden.[3]

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. 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.

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 [4]. The access information is available from the KDD-Cup 2000 home page at http://www.ecn.purdue.edu/KDDCUP.

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

Table 1 Dataset characteristics

	Transac-tions	Distinct Items	Maximum Transaction Size	Average Transaction 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

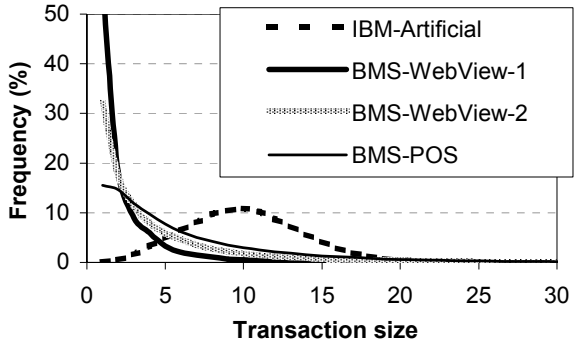

Figure 1. Dataset transaction size distribution.

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. Additional transaction size distribution data is available in [11]. 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.

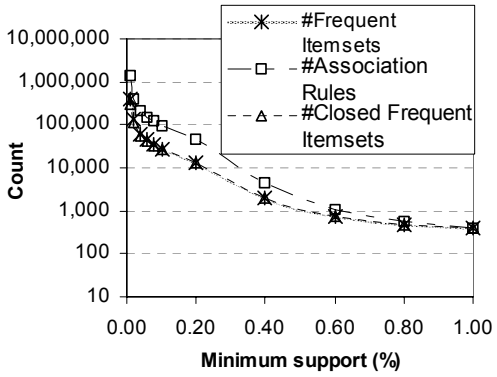

Figure 2. The number of frequent itemsets and association rules at different minimum support levels on IBM-Artificial. The two lines for frequent itemsets and closed frequent items overlap at many data points.

Figures 2 and 3 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 IBM-Artificial and BMS-POS. The figures on BMS-WebView-1 and

BMS-WebView-2 have similar trends to BMS-POS and are not shown here due to space limitations. The other figures and the detailed numbers are available in [11]. 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 datasets typically used to benchmark the performance of association rule algorithms.

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 [9]. 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 [10] 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. Such high minimum supports are uninteresting for many real world applications.

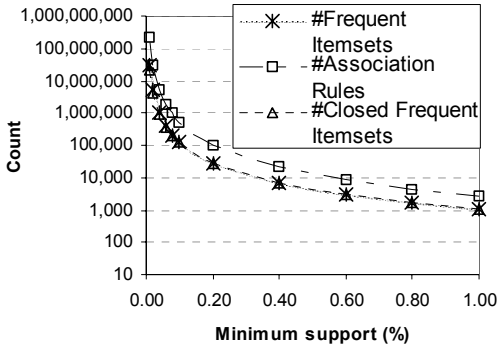

Figure 3. The number of frequent itemsets and association rules at different minimum support levels on BMS-POS. The two lines for frequent itemsets and closed frequent items overlap at many data points.

5. EXPERIMENTS

Each of the five algorithms described in Section 3 was tested on the four datasets described in the previous section. 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.

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.

5.1 Generating Frequent Itemsets

Figures 4 through 6 show performance curves with minimum supports ranging from 0.01% to 0.10% for the four algorithms that generate frequent itemsets, namely Closet, FP-growth, Charm, and Apriori, on IBM-Artificial, BMS-POS, and BMS-WebView-1 respectively. The vertical axis is on a logarithmic scale. The results on BMS-WebView-2 have similar trends to those on BMS-WebView-1, and are omitted due to space limitations. In the figures, a missing data point indicates that we do not have that value as the corresponding algorithm failed because it ran out of memory or disk space. MagnumOpus is not included in these results because it cannot generate frequent itemsets.

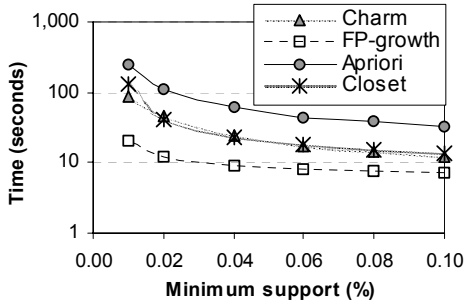

Figure 4. Running time of the algorithms for generating frequent itemsets for IBM-Artificial.

The first thing to notice about these results is that when the minimum support is large,[4] 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%.

For the artificial dataset, every algorithm outperforms Apriori by a significant margin for minimum support values less than 0.10%.

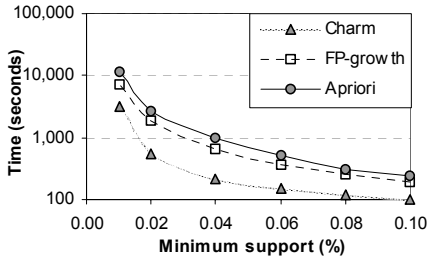

Figure 5. Running time of the algorithms for creating frequent itemsets on BMS-POS. Closet does not show up as it failed.

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

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 [2][7][10]. 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 [7] where Closet is faster than Charm. One explanation for this is that we are using a new version of

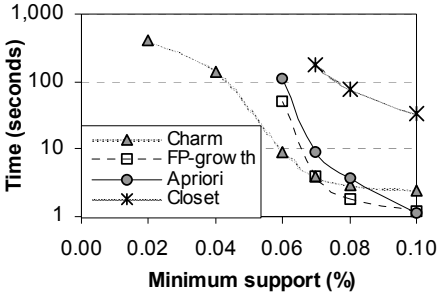

Figure 6. Running time of the algorithms for generating frequent itemsets for BMS-WebView-1.

Charm that 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 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 ran out of our 1GB of allowed 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 [11]), 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-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 Charm and FP-growth when there are many long frequent itemsets. Creating much smaller numbers of frequent itemsets (closed) may partially explain why Charm is so fast.

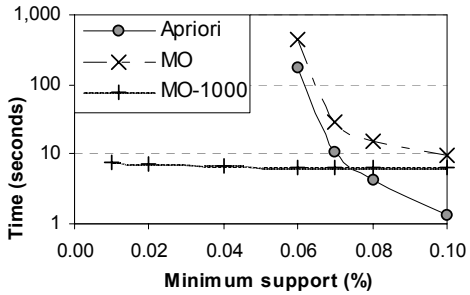

Figure 7. Running time of the algorithms for generating association rules for BMS-WebView-1.

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 (1%) and low minimum supports (< 0.1%) (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

5.2 Generating Association Rules

Due to space limitations, we only show performance curves of the two algorithms, MagnumOpus and Apriori, on the BMS-WebView-1 dataset in Figure 7. 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. The time includes the running time for both creating the frequent itemsets and creating the association rules. The figures for the other three domains and the detailed values of these results are available in [11]. 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%. From these results, we make the following observations:

1. MagnumOpus is significantly slower than Apriori on all of the datasets. This is not surprising, as these datasets are sparse and MagnumOpus was not designed for this type of data [9].

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 because 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 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).

6. 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 one of the IBM Almaden artificial datasets.

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 have donated such a 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 either run out of 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 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.

7. 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.

8. REFERENCES

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] Webb, G.I. OPUS: An efficient admissible algorithm for unordered search. JAIR, 3:431-465.

[9] 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.

[10] 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.

[11] Zheng, Z., Kohavi, R., and Mason, L. Real word performance of association rule algorithms (long version), 2001. Available at http://robotics.stanford.edu/~ronnyk/assocBenchmarkLong.pdf.