Clustering and Approximate Identification of Frequent Item Sets

Selim Mimaroglu and **Dan A. Simovici**
University of Massachusetts at Boston,
Department of Computer Science,
Boston, Massachusetts 02125, USA,
{smimarog,dsim}@cs.umb.edu

Abstract

We propose an algorithm that computes an approximation of the set of frequent item sets by using the bit sequence representation of the associations between items and transactions. The algorithm is obtained by modifying a hierarchical agglomerative clustering algorithm and takes advantage of the speed that bit operations afford. The algorithm offers a very significant speed advantage over standard implementations of the Apriori technique and, under certain conditions, recovers the preponderant part of the frequent item sets.

Introduction

The identification of frequent item sets and of association rules is a major research direction in data mining that has been examined in a large number of publications.

In (Afrati, Gionis, & Mannila 2004) it is shown that most variants of concise approximations of collections of frequent item sets are computationally hard. The same reference provides approximative feasible solutions. In the same direction, we present an approximative algorithm for finding frequent item sets that makes use of hierarchical clusterings of items through their representations as bit sequences. We show that the identification of similar attributes allows a rapid identification of sets of attributes that have high levels of support.

The representation of attributes using bit sequences was introduced in (Yen & Chen 2001). We use this representation technique to develop a fast retrieval algorithm of a high proportion of frequent item sets, particularly in data with low density or high density. Since representing a data set as bit sequences is very space-efficient large amounts of data can be accommodated.

The other major idea of the paper is to use a modification of an agglomerative clustering technique to gradually generate frequent item sets. In some cases, this approach can save computational effort by grouping and dropping similar item sets. Clustering frequent item sets aiming at compressing the size of the results of the mining process was addressed in (Xin *et al.* 2005). Our purpose is different in that it is directed towards providing, under certain conditions, good approximations of frequent item sets that are expressed concisely as relatively small sets of clusters of items and can be obtained at significantly lower computational cost. In a different direction, mining approximate frequent item sets in the presence of noise has been explored in (Liu *et al.* 2006).

All sets are assumed to be finite unless we specify otherwise. A *transaction data sequence* is a finite sequence $T = (t_1, \ldots, t_n)$ of subsets of a set $I = \{i_1, \ldots, i_m\}$. The elements of I are referred to as *items*, and the members of T are referred to as *transactions*. A set of items K *occurs in a transaction* t_p if $K \subseteq t_p$. The *support sequence* of an item set $K \in \mathcal{P}(I)$ is the subsequence $\mathsf{sseq}_T(K)$ of the sequence of transactions that consists of those transactions t_i such that $K \subseteq t_i$. The *support count* of the set of items K in the transaction data set T is the length of the support sequence

$$\mathsf{suppcount}_T(K) = |\mathsf{sseq}_T(K)|.$$

The support of K in T is the number:

$$\mathsf{supp}_T(K) = \frac{\mathsf{suppcount}_T(K)}{|T|}.$$

If $\mathsf{supp}_T(K) \geq \sigma$ we say that K is a *σ-frequent item set in* T.

After a discussion of the metric space of bit sequences we examine the properties of the clusters of item sets. In the final part of the paper we describe the actual algorithm and present experimental results.

The metric space of bit sequences

To use bit sequences we start from the observation that transactions data set have a natural tabular representation. Consider a $0/1$-table τ whose columns correspond to the items $i_1, \ldots, i_m$; the rows of τ correspond to the transactions of T. If τ_{pq} is the entry of τ located in row p and column q, then

$$\tau_{pq} = \begin{cases} 1 & \text{if } i_q \in t_p, \\ 0 & \text{otherwise.} \end{cases}$$

The column of τ that corresponds to an item i is a bit sequence $\mathbf{b}^i$. Clearly, we have $\mathbf{b}^i \in \{0,1\}^n$. The notion of bit-sequence has been frequently used in the literature for discovery of association rules (see (Yen & Chen 2001;

Shenoy *et al.* 2000; Burdick, Calimlim, & Gehrke 2001)). Bit sequences are used in our clustering approach.

The notion of bit sequence of an item can be extended to bit sequences for item sets. Namely, if $L = \{i_{\ell_1}, \ldots, i_{\ell_r}\}$ is an item set, then the characteristic bit sequence of the set $\mathsf{sseq}_T(L)$ is be bit sequence $\mathbf{b}^L$ given by:

$$\mathbf{b}^L = \mathbf{b}^{i_{\ell_1}} \wedge \cdots \wedge \mathbf{b}^{i_{\ell_r}}. \tag{1}$$

We have $\mathbf{b}^L_p = 1$ if and only if $L \subseteq t_p$.

For a bit sequence $\mathbf{b} \in \{0,1\}^n$ denote by $\| \mathbf{b} \|$ the number $\sqrt{\mathbf{b} \cdot \mathbf{b}^{tr}} = \sqrt{\sum_{p=1}^n b_p}$. The row that is obtained by transposing the column $\mathbf{b}$ is denoted by $\mathbf{b}^{tr}$. The *norm of* $\mathbf{b}$, $\| \mathbf{b} \|$, equals the Euclidean norm of $\mathbf{b}$ considered as a sequence in $\mathbb{R}^n$ because $b^2 = b$ for every $b \in \{0,1\}$.

The operations $\wedge, \vee$ and $'$ are defined on the set $\{0,1\}$ by:

$$a \wedge b = \min\{a,b\}, a \vee b = \max\{a,b\}, a' = 1 - a,$$

for $a, b \in \{0,1\}$. These operations are extended componentwise to bit sequences and the set of bit sequences $\mathbf{B}^n = \{0,1\}^n$ equipped with the operations $\vee, \wedge$ and $'$ is a Boolean algebra having $\mathbf{0}^{tr} = (0, \ldots, 0)$ as its least element, and $\mathbf{1}^{tr} = (1, \ldots, 1)$ as its greatest element.

It is also useful to consider the symmetric difference defined on $\{0,1\}$ by:

$$a \oplus b = \begin{cases} 1 & \text{if } a \neq b, \\ 0 & \text{otherwise,} \end{cases}$$

for $a, b \in \{0,1\}$. This operation can be extended componentwise to bit sequences. As a result we have a Boolean ring structure $(\mathbf{B}^n, \oplus, \wedge, \mathbf{0})$.

For sequences in $\{0,1\}^n$ the usual metric in $\mathbb{R}^n$ defined by $d(\mathbf{b}, \mathbf{b}') = \| \mathbf{b} - \mathbf{b}' \|$ can also be expressed using the operation "$\oplus$" by:

$$d(\mathbf{b}, \mathbf{b}') = \| \mathbf{b} \oplus \mathbf{b}' \|.$$

The following equalities are immediate consequences of the definition of $\mathbf{b}^L$:

$$\begin{aligned} |\mathsf{sseq}_T(L)| &= \mathsf{suppcount}_T(L) = \| \mathbf{b}^L \|^2 \\ \mathbf{b}^L \wedge \mathbf{b}^K &= \mathbf{b}^{KL}, \\ \mathbf{b}^L \oplus \mathbf{b}^K &= \mathbf{b}^{K \oplus L}. \end{aligned}$$

Here KL denotes the union of the item sets K and L.

The Jacquard-Tanimoto metric δ defined on $\mathcal{P}(S)$, the set of subsets of a set S, by $\delta(U,V) = \frac{|U \oplus V|}{|U \cup V|}$ for $U, V \in \mathcal{P}(S)$ can be transferred to the set of bit sequences by defining

$$\delta(\mathbf{b}^K, \mathbf{b}^L) = \frac{|\mathbf{b}^K \oplus \mathbf{b}^L|}{|\mathbf{b}^K \vee \mathbf{b}^L|}.$$

It is easy to see that

$$\delta(\mathbf{b}^K, \mathbf{b}^L) = 1 - \frac{\| \mathbf{b}^K \wedge \mathbf{b}^L \|^2}{\| \mathbf{b}^K \vee \mathbf{b}^L \|^2}.$$

Clustering Item Sets

The core of our algorithm is a modified version of hierarchical clustering that seeks to identify item sets that are close in the sense of the Jacquard-Tanimoto metric. The diameter of an item set K is defined as the largest distance between two members of the set. We give a lower bound on the probability that a transaction contains an item set K; this lower bound depends on the diameter of the item set and on the level of support of the items of K.

Theorem 1 *Let T be a transaction data set over the set of attributes I and let K be an attribute cluster of diameter $\mathsf{diam}_\delta(K)$. The probability that a transaction t contains the item set K is at least $(|K| - 1)(1 - \mathsf{diam}_\delta(K)) \min_{i \in K} \mathsf{supp}(i) - (|K| - 2)$.*

Proof. Let $K = i_1 \cdots i_r$ be an item set. The probability that K is contained by a transaction t is $P(\{t[i_1] = \cdots = t[i_r] = 1\})$. Let $Q_{ii'}$ be the event that occurs when $t[i] = t[i'] = 1$ for two attributes i, i'. Since $\{t[i_1] = \cdots = t[i_r] = 1\}) = Q_{i_1 i_2} \cap \cdots \cap Q_{i_{r-1} i_r}$ it follows, by Boole's inequality, that

$$P(\{t[i_1] = \cdots = t[i_r] = 1\}) \geq \sum_{j=1}^{r-1} P(Q_{i_j i_{j+1}}) - (r-2)$$

Suppose that the data set T contains N transactions. To evaluate $P(Q_{ii'}) = P(t[i] = t[i'] = 1)$ observe that

$$|\{t \mid t[i] = t[i'] = 1\}| = |\mathsf{sseq}_T(i) \cap \mathsf{sseq}_T(i')|,$$

so

$$P(\{t \mid t[i] = t[i'] = 1\}) = \frac{|\mathsf{sseq}_T(i) \cap \mathsf{sseq}_T(i')|}{N}.$$

Since

$$\begin{aligned} \delta(i,i') &= \frac{|\mathsf{sseq}_T(i) \oplus \mathsf{sseq}_T(i')|}{|\mathsf{sseq}_T(i) \cup \mathsf{sseq}_T(i')|} \\ &= 1 - \frac{|\mathsf{sseq}_T(i) \cap \mathsf{sseq}_T(i')|}{|\mathsf{sseq}_T(i) \cup \mathsf{sseq}_T(i')|}, \end{aligned}$$

we have

$$\begin{aligned} &|\mathsf{sseq}_T(i) \cap \mathsf{sseq}_T(i')| \\ &\quad = (1 - \delta(i,i'))|\mathsf{sseq}_T(i) \cup \mathsf{sseq}_T(i')|, \end{aligned}$$

so

$$\begin{aligned} &P(\{t \mid t[i] = t[i'] = 1\}) \\ &\quad = \frac{(1 - \delta(i,i'))|\mathsf{sseq}_T(i) \cup \mathsf{sseq}_T(i')|}{N} \\ &\quad \geq (1 - \delta(i,i')) \min\{\mathsf{supp}(i), \mathsf{supp}(i')\}. \end{aligned}$$

Thus, we have:

$$\begin{aligned} &P(t[i_1] = \cdots = t[i_r] = 1) \\ &\quad \geq \sum_{j=1}^{r-1} P(Q_{i_j i_{j+1}}) - (r-2) \\ &\quad \geq (r-1)(1 - \mathsf{diam}_\delta(K)) \min_{i \in K} \mathsf{supp}(i) - (r-2) \end{aligned}$$

which is the needed inequality.

Theorem 1 shows that the tighter the cluster (that is, the smaller its diameter), the more likely it is that two tuples will have equal components for all attributes of the cluster. Therefore, if the attributes $K = i_1 \cdots i_r$ form a cluster there is a substantial probability that the support of the item set K will be large.

The Approximative Frequent Item Set Algorithm

Let T be a transaction data set having the attributes $i_1 \cdots i_m$ and containing n tuples. The algorithm proposed in this paper is a modification of the standard agglomerative clustering algorithm applied to the collection of sets of items. The distance between sets of items are defined by the transaction data set, as indicated earlier.

The standard approach in agglomerative clustering is to start from a distance defined on the finite set C of objects to be clustered. Then, the algorithm proceeds to construct a sequence of partitions of C: $\pi_1, \pi_2, \ldots$, such that $\pi_1 = \{\{c\}|c \in C\}$ and each partitions π_{i+1} is obtained from the previous partition π_i by fusing two of its blocks chosen using a criterion specific to the variant of algorithm. For example, in the *single link* variant one defines for every two blocks $B, B' \in \pi_i$ the number:

$$\delta(B, B') = \min\{d(x, y)|x \in B, y \in B'\}.$$

One of the pairs (B, B') that has the minimal value for $\delta(B, B')$ is selected to be fused.

Let T be a transaction data set on a set of items I and δ be a metric on $\mathcal{P}(I)$, the set of subsets of I. Define $\mathcal{M}_{T,\sigma}$ as the family of pairs of subsets of I given by

$$\begin{aligned}\mathcal{M}_{T,\sigma} &= \{(X, Y)|X, Y \in \mathcal{P}(I), \mathsf{supp}_T(X) \geq \sigma, \\ &\quad \mathsf{supp}_T(Y) \geq \sigma \text{ and } \mathsf{supp}_T(XY) < \sigma\}.\end{aligned}$$

A pair of item sets (U, V) is said to be (δ, σ)-*maximal* in a collection of clusters $\mathcal{C}$ on I if the following conditions are satisfied:

1. $(U, V) \in \mathcal{M}_{T,\sigma}$, and
2. $\delta(U, V) = \min\{\delta(X, Y) \mid (X, Y) \in \mathcal{M}_{T,\sigma}\}$.

The clustering process works with two collections of item sets: the collection of current item sets denoted by CIS, and the final collection of clusters denoted by FIS. An item set K is represented by its characteristic sequence $\mathbf{b}^K$ and all operations on item sets are actually performed on bit sequences.

At each step the algorithm seeks to identify a pair of item sets in CIS that are located at minimal distance such that the support of each of the item sets is at least equal to a minimum support σ.

If $\mathsf{CIS} = \{L_1, \ldots, L_n\}$, then, as in any agglomerative clustering algorithm we examine a new potential cluster $L = L_i \cup L_j$ starting from the two closest clusters L_i and L_j; the fusion of L_i and L_j takes place *only if the support of the cluster L is above the threshold σ*, that is, only if the pair (L_i, L_j) is not (δ, σ)-maximal. In this case, L replaces L_i and L_j in the current collection of clusters.

Input: a transaction data set T and a minimum support σ;
Output: an approximation of the collection of σ-frequent item sets;
Method: initialize $\mathsf{FIS} = \emptyset$;
initialize CIS to contain all one-attribute clusters $\{i\}$;
such that $\mathsf{supp}_T(\{i\}) \geq \sigma$;
repeat
find in CIS the closest two clusters L_i, L_j;
compute the bit sequence of the candidate cluster $L = L_i \cup L_j$ as $\mathbf{b}^L = \mathbf{b}^{L_i} \wedge \mathbf{b}^{L_j}$;
if $\mathsf{supp}_T(L) \geq \sigma$
then
merge L_i and L_j into L;
add L to CIS;
else /* this means that (L_i, L_j) is (δ, σ)-maximal */
begin
cross-expand L_i and L_j;
add L_i and L_j to FIS;
end;
remove L_i, L_j from CIS;
until CIS contains at most one cluster;
if one cluster L remains in CIS
then add L to FIS;
output $\bigcup\{\mathcal{P}_1(K)|K \in \mathsf{FIS}\}$.

Figure 1: Approximative Frequent Item Set Algorithm AFISA

If (L_i, L_j) is a (δ, σ)-maximal pair, then we apply a special post-processing technique to the two clusters that we refer to as *cross-expansion*. This entails expanding L_i by adding to it each maximal subset H_j of L_j such that $\mathsf{supp}(L_i \cup H_j) \geq \sigma$ and expanding L_j in a similar manner using maximal subsets of L_i.

The approximative frequent item set algorithm (AFISA) is given in Figure 1. $\mathcal{P}_1(K)$ denotes the set of all non-empty subsets of the set K.

Each cluster produced by the above algorithm is a σ-frequent item set, and each of its subsets is also σ-frequent.

It is interesting to examine the probability that bit sequences $\mathbf{b}^K$ and $\mathbf{b}^L$ of two item sets K, L that have the combined support $\mathsf{supp}_T(KL)$ are situated at a distance that exceeds a threshold d. Suppose that the data set T consists of n tuples, $T = \{t_1, \ldots, t_n\}$. To simplify the argument we assume that the data set is random. Our evaluation process is a variant of the one undertaken in (Agrawal *et al.* 1996). Further, suppose that each transaction contains an average of c items; thus, the probability that a transaction contains an item i is $p = \frac{c}{m}$, where m is the number of items.

For each transaction t_h consider the random variable X_h defined by $X_h = 1$ if exactly one of the sets K and L is included in t_h and 0 otherwise for $1 \leq h \leq n$. Under the

assumptions made above we have:

$$X_h : \begin{pmatrix} 1 & 0 \\ \phi_{k,l}(p) & 1-\phi_{k,l}(p) \end{pmatrix},$$

where $k = |K|$ and $l = |L|$ and $\phi_{k,l}(p) = p^k(1-p^l) + p^l(1-p^k)$. Assuming the independence of the random variables $X_1, \ldots, X_n$ the random variable $X = X_1 + \cdots + X_n$, which gives the number of transactions that contain exactly one of the sets K and L, has a binomial distribution with the parameters n and $\phi_{k,l}(p)$. Thus, the expected value of X is $n\phi_{k,l}(p)$ and that the value of $\phi_{k,l}(p)$ is close to 0 when p is close to 0 or to 1. Note that

$$\delta(\mathbf{b}^K, \mathbf{b}^L) = \frac{X}{X + \mathsf{supp}(KL)},$$

so, we have $\delta(\mathbf{b}^K, \mathbf{b}^L) > d$ if $X > \frac{\mathsf{supp}_T(KL)}{\frac{1}{d}-1}$. By Chernoff's inequality (see (Alon & Spencer 2000)) we have:

$$P\left(X > n\phi_{k,l}(p) + a\right) < e^{-\frac{2a^2}{n}}.$$

If we choose $a = \frac{\mathsf{supp}_T(KL)}{\frac{1}{d}-1} - n\phi_{k,l}(p)$ we obtain:

$$P\left(X > \frac{\mathsf{supp}_T(KL)}{\frac{1}{d}-1}\right) < e^{-\frac{2}{n}\left(\frac{\mathsf{supp}_T(KL)}{\frac{1}{d}-1} - n\phi_{k,l}(p)\right)^2}$$

Thus, the probability that $\delta(\mathbf{b}^K, \mathbf{b}^L) > d$ (which is the probability that AFISA will fail to join the sets K and L) is small for values of p that are close to 0 or close to 1 because $\phi_{k,l}(p)$ is small in this case. This suggests that our algorithm should work quite well for low densities or for high densities and our experiments show that this is effectively the case.

Experimental Results

We used for experiments a 32-bit, Pentium 4 3.0 GHz processor with 2GB of physical memory; only 1.5GB of that memory was assigned to AFISA or ARTool when testing.

Performance gains on bit sequence operations are reflected in our test results as much faster execution times (up to 146,000 times faster in some cases).

On a synthetic data set containing 491,403 transactions having 100 items and an average of 10 items/transaction we applied the Apriori algorithm and the FPGrowth (using ARTool, the implementation provided by (Cristofor 2002)), and our own algorithm (AFISA). The number of item sets returned in each case is shown below:

support	number of FIS		
	AFISA	Apriori	FPGrowth
0.02	118	1000	1000
0.05	77	188	188
0.1	40	47	47
0.2	10	10	10
0.3	4	4	4
0.4	0	0	0
0.5	0	0	0

The corresponding running time is shown next:

support	Running Time (ms)		
	AFISA	Apriori	FPGrowth
0.02	2141	421406	149375
0.05	2110	244578	139359
0.1	2063	218625	132969
0.2	2047	140406	126797
0.3	2031	141282	127640
0.4	2032	71047	64597
0.5	2032	69594	66515

A series of experiments were performed on data sets containing 1000 transactions and 20 items, by varying the density of the items, that is, the number of items per transaction. Below we give the results recorded for several such densities.

As anticipated by the theoretical evaluation the superiority of AFISA is overwhelming at low or high densities. For example, for 18.2 items/transaction we recorded the following results:

supp.	number of FIS		Running Time (ms)	
	AFISA	Apriori	AFISA	Apriori
0.02	1048575	1048575	15	2195000
0.05	1048575	1048575	32	2473641
0.1	1048575	1048575	15	2342484
0.2	1048575	1048575	16	2309437
0.3	1048575	1048575	31	2526141
0.4	1048575	1048575	32	2522750
0.5	393216	1001989	15	2216375

For a density of 1.8 items per transaction the results are similar: at reasonable high levels of support all frequent item sets are recovered, and the time is still much lower than the Apriori time, even though the advantage of AFISA is less dramatic.

supp.	number of FIS		Running Time (ms)	
	AFISA	Apriori	AFISA	Apriori
0.02	22	30	15	125
0.05	16	16	15	94
0.1	6	6	15	94
0.2	1	1	15	47
0.3	0	0	15	47
0.4	0	0	15	47
0.5	0	0	15	47

In the mid range of densities (at 10.2 items/transaction) the time of AFISA is still superior; however, the recovery of frequent item sets is a lot less impressive until a rather high level of support is reached. The results of this experiment are enclosed below.

supp.	number of FIS		Running Time (ms)	
	AFISA	Apriori	AFISA	Apriori
0.02	28677	149529	15	102593
0.05	2580	38460	15	8328
0.1	784	9080	15	2500
0.2	206	1315	16	1203
0.3	89	371	16	922
0.4	49	112	15	734
0.5	23	45	16	578

We also conducted experiments on several UCI data sets (Blake & Merz 1998); we include typical results obtained on the ZOO data set.

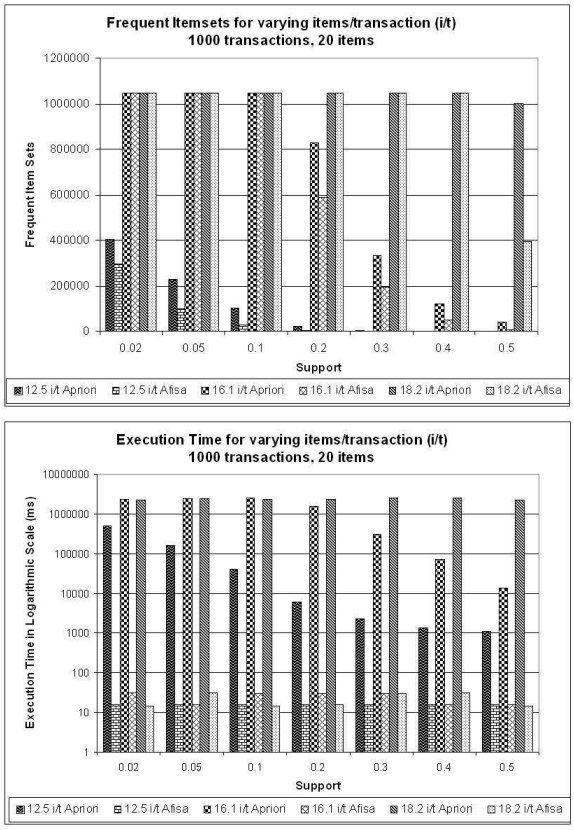

Figure 2: Frequent item sets and execution time for a synthetic data set having 1K rows and 20 items

supp.	number of FIS		Running Time (ms)	
	AFISA	Apriori	AFISA	Apriori
0.04	1016	1741	10	203
0.05	768	1472	10	187
0.1	290	855	10	125
0.2	267	329	10	109
0.3	31	126	10	78
0.4	16	27	9	47
0.5	13	13	9	31

We used the synthetic data generator that is available from IBM Almaden Research Center through the "IBM Quest Data Mining Project". The tests were performed for several values of the number of items per transaction (i/t). These results are shown in Figure 2.

It is clear that for every value of the number of items per transaction (i/t), execution time of AFISA is superior. For values of i/t close to the extremes (below 10% or above 90%) AFISA either produced the same amount of frequent item sets as Apriori or the difference was negligible. This is especially useful for analyzing sparse data which can be produced, for example, by customer transactions in supermarkets.

Conclusions

Clustering bit sequences representing item sets is an efficient technique for the approximate detection of frequent item sets in data sets that have high or low densities. The technique recovers item set in the mid-range of densities at a lower rate; however, the speed of the algorithm makes the algorithm useful, when a complete recovery of the frequent item set is not necessary.

We intend to expand our experimental studies to data sets that cannot fit into memory. We believe that AFISA will still be much faster than Apriori because using bit sequences is both time and space efficient; also, operations on bit sets are done very fast by the current modern processors.

References

Afrati, F.; Gionis, A.; and Mannila, H. 2004. Approximating a collection of frequent sets. In *Proceedings of KDD*, 12–19.

Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; and Verkamo, A. I. 1996. Fast discovery of association rules. In Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R., eds., *Advances in Knowledge Discovery and Data Mining*. Menlo Park: AAAI Press. 307–328.

Alon, N., and Spencer, J. H. 2000. *The Probabilistic Method*. New York: Wiley-Interscience, second edition.

Blake, C. L., and Merz, C. J. 1998. *UCI Repository of machine learning databases*. http://www.ics.uci.edu/∼mlearn/MLRepository.html: University of California, Irvine, Dept. of Information and Computer Sciences.

Burdick, D.; Calimlim, M.; and Gehrke, J. 2001. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proc. of the 17th Int. Conf. on Data Engineering*, 443–452.

Cristofor, L. 2002. ARtool: Association rule mining algorithms and tools. http://www.cs.umb.edu/∼laur/ARtool/.

Liu, J.; Paulsen, S.; Xu, X.; Wang, W.; Nobel, A.; and Prins, J. 2006. Mining approximate frequent itemset in the presence of noise: algorithm and analysis. In *Proceedings of the 6th SIAM Conference on Data Mining (SDM)*, 405–416.

Shenoy, P.; Haritsa, J. R.; Sudarshan, S.; Bhalotia, G.; Bawa, M.; and Shah, D. 2000. Turbo-charging vertical mining of large databases. In *Proceedings of SIGMOD*, 22–33.

Xin, D.; Han, J.; Yan, X.; and Cheng, H. 2005. Mining compressed frequent-pattern sets. In *Proc. 2005 Int. Conf. on Very Large Data Bases (VLDB'05)*, 709–720.

Yen, S. J., and Chen, A. 2001. A graph-based approach for discovering various types of association rules. *IEEE Transactions on Knowledge and Data Engineering* 13:839–845.