Efficient Discovery of Error-Tolerant Frequent Itemsets in High Dimensions

Cheng Yang
Stanford University
Dept. of Computer Science
Stanford, CA 94305, USA
yangc@cs.stanford.edu

Usama Fayyad[1]
digiMine, Inc.
10500 NE 8th Street
Bellevue, WA 98004, USA
usama@digimine.com

Paul S. Bradley[2]
digiMine, Inc.
10500 NE 8th Street
Bellevue, WA 98004, USA
paulb@digimine.com

ABSTRACT

We present a generalization of frequent itemsets allowing for the notion of errors in the itemset definition. We motivate the problem and present an efficient algorithm that identifies error-tolerant frequent clusters of items in transactional data (customer-purchase data, web browsing data, text, etc.). The algorithm exploits sparseness of the underlying data to find large groups of items that are correlated over database records (rows). The notion of transaction coverage allows us to extend the algorithm and view it as a fast clustering algorithm for discovering segments of similar transactions in binary sparse data. We evaluate the new algorithm on three real-world applications: clustering high-dimensional data, query selectivity estimation and collaborative filtering. Results show that the algorithm consistently uncovers structure in large sparse databases that other traditional clustering algorithms fail to find.

Keywords

Error-tolerant frequent itemset, clustering, high dimensions, query selectivity estimation, collaborative filtering.

1. PRELIMINARIES AND MOTIVATION

Progress in database technology has provided the foundation that made large stores of transactional data ubiquitous. Such stores are common in commerce (products purchased by customers), web site analysis (sites visited by users), text processing (words occurring in documents), etc. The *frequent itemset* problem is that of determining which items occur together frequently in a transaction. We generalize the definition of frequent itemsets to tolerate error, and propose an algorithm for finding all such error-tolerant frequent itemsets. We then discuss an efficient algorithm approximating the complete algorithm. The primary motivation for this generalization is to find frequent groups of transactions (user groups, web sessions, etc.) instead of focusing on just the items themselves, allowing for the discovery of groups of similar transactions that share most items. We believe these to be more general and intuitive characterizations of groups of transactions.

The frequent itemset generalization, based on relaxing the exact matching criteria in the standard frequent itemset definition and allowing a transaction to violate some conditions, is motivated by the following example. Consider customer purchase data over 50 products (P1, P2,…, P50). Table 1 shows the counts of customers with corresponding purchase records for P1 through P5 (1 for "purchased", 0 for "not purchased", other products are not of interest for this example). Let the total number of customers (rows) in the database be 10,000. For simplicity, suppose that no other customers in the database purchased these 5 products. Notice that for any minimum support value $\kappa > 0$, the itemset {P1, …, P5} is not frequent. In fact, for a support level of 5%, none of the 5 items would appear in any frequent itemset enumeration. Note, however, that 5.7% of the transactions contain 4 of the 5 products. If products P1,…,P5 are different brands of soda then 5.7% of the customers purchase a significant portion of the 5 brands. This pattern may be useful for the data analyst but would be undiscovered by traditional frequent itemset approaches due to the harsh definition of support. By relaxing the definition of frequent itemsets to be error-tolerant, one could identify this cluster of customers who, purchase "most" of the products {P1,…,P5}.

1.1 Problem Statement

The intuition is made specific with the following definition and problem statement. Adopting the notation of [AMSTV96], let $I = \{i_1, i_2, \ldots, i_d\}$ be the full set of items over a database D consisting of transactions T, where each transaction is a subset of the full itemset I. Each transaction T may be viewed as record with d attributes $\{i_1, i_2, \ldots, i_d\}$ taking values in {0,1}, where a 1 occurs over the attributes specifically listed in the transaction T (hence viewing the database as a table with one record for each transaction and d columns). This view of the data often results in a very sparse table (i.e. the majority of the table elements have value 0). The *support* of an itemset is the fraction of total

Count	P1	P2	P3	P4	P5	Other Products
100	1	1	1	1	0	…
100	0	1	1	1	1	…
80	1	1	1	0	1	…
90	1	0	1	1	1	…
200	1	1	0	1	1	…
…	…	…	…	…	…	…

Table 1: Counts of customer-purchase data patterns

transactions in the database containing the given itemset. The *frequent itemset* problem is that of finding all itemsets with support greater than a minimum threshold κ (called *minimum support* or *minsup*) [RG99]. Note that for a single transaction T to contribute to the support of a given itemset, it must contain the entire itemset. We relax this exact matching criterion to yield a more flexible definition of support and consequently of error-tolerant itemsets, eventually leading to an algorithm for clustering rows in sparse binary databases.

We have found that this algorithm is capable of identifying the presence of structure (clusters) in large sparse databases that traditional clustering algorithms consistently fail to find. This leads to both a more effective (and faster) method of clustering, as well as an effective way of determining the number of clusters: an open problem for most classical clustering algorithms [DH73,CS96,BFR98].

Definition 1: Error-Tolerant Itemset [ETI] (informal): An itemset $E \subseteq I$ is an *error-tolerant itemset* having error ε and support κ with respect to a database D having n transactions if there exists at least κn transactions in which at least a fraction 1-ε of the items from E are present.

Problem Statement: Given a sparse binary database D of n transactions (rows) and d items (columns), error tolerance $\varepsilon > 0$, and minimum support κ in [0,1], determine all error-tolerant itemsets (ETIs) over D.

The fundamental difference between this problem and traditional frequent itemsets is a relaxation in support criteria. An error threshold $\varepsilon = 0$ collapses Definition 1 to the standard frequent itemset definition. For $\varepsilon > 0$, the problem is to efficiently determine itemsets satisfying Definition 1. For example, the itemset E = {P1,P2,P3,P4,P5} from Table 1 is an error-tolerant itemset with support κ = 5.7% and ε = 0.2. Note that the support for this itemset can also be interpreted as those transactions containing 4 of 5 of the items in E.

Definition 1 is not confined to binary {0,1} data, but can be extended to find error-tolerant itemsets over transactional databases with categorical-valued attributes (more than 2 values). Continuous-valued attributes may be preprocessed with a discretization algorithm [FI93]. We discuss generalizations in Section 6, but this paper focuses on the binary case. Note that the definition does admit degenerate cases that need to be handled.

Degenerate case: Table 2 illustrate a degenerate case of ETIs, where {1 2 3 4 5} can be regarded as an ETI with κ=50% and ε=20%, even though item 5 does not contribute anything to the ETI formation. Although it is technically a valid ETI, in the following sections we will develop algorithms to avoid such cases since they are not interesting in practice.

1	2	3	4	5	6
1	1	1	1	1	
1	1	1	1		
1	1	1	1		
1	1	1	1		1
				1	
				1	

Table 2

We define *maximal* ETIs as ETIs whose supersets are not ETIs. Sometimes both maximal and non-maximal ETIs are of interest when finding clusters of similar transactions (clustering rows). For example, a large group of people buy 5 products {P1,P2,P3,P4,P5}, and another large group of people buy only {P1,P2,P3}. Here, {P1,P2,P3,P4,P5} is a maximal ETI, but {P1,P2,P3} is not. However, it would be useful to identify both ETIs since they each represent a large group of customers and provide useful information. We will show how the ETIs can be used to efficiently uncover such structure. First, we discuss the possibility of ETIs occurring by chance.

1.2 Are ETIs Random Artifacts?

Before studying properties of ETIs and their efficient extraction, we address the question of whether finding such item sets is of interest, and whether ETI discovery could be a side effect of correlations that happen to appear in data by chance. It should be obvious from the definition of ETIs that as the error ε is increased, the expected number of items in ETIs should increase, which raises the question of the validity of such patterns. Essentially, is one really finding structure in data, or simply discovering random correlations in large data sets?

Lemma: Assume a binary $n \times d$ sparse matrix over {0,1} is generated at random, while the probability that an entry is 1 is p. Then the probability of a frequent error-tolerant item set with r items appearing in it with support κ and error ε is not greater than $\binom{d}{r}\binom{n}{\kappa n} p^{(1-\varepsilon)\kappa n r}(1-p)^{\varepsilon \kappa n r}$.

Proof: see [YFB00].

An application of this lemma to some realistic assumptions over market-basket type data quickly shows that this probability is very small. For example, for p = 0.15, ε = 0.2, κ = 0.01, N = 1,000,000, D = 500 and r = 5, using Stirling's approximation [GKP89] for the combinatorial terms one obtains that the probability of finding an ETI with 5 items by chance is approximately 10^{-9300} (for r=10 items this probability drops down to $10^{-43,000}$). Hence the identification of submatrices with many "1"s is indeed interesting as such submatrices are extremely unlikely to occur by chance, especially when the number of columns involved is large.

2. FINDING ERROR-TOLERANT FREQUENT ITEMSETS

In binary {0,1} datasets, an error-tolerant frequent itemset (ETI) is represented as a set of dimensions (called *defining dimensions*) where "1" appears with high probability among a set of rows. Formally, we give the following two ETI definitions, a strong one and a weak one:

ETI Definition 1 (strong): A strong ETI consists of a set of items, called defining dimensions DD $\subseteq I$, such that there exists a subset of transactions R $\subseteq T$ consisting of at least κn transactions and, for each $r \in$ R, the fraction of items in DD which are present in r is at least 1-ε.

ETI Definition 2 (weak): A weak ETI consists of a set of items, called defining dimensions DD $\subseteq I$, such that there exists a subset of transactions R $\subseteq T$, $|\mathrm{R}| \geq \kappa n$ and,

$$\frac{\sum_{x\in \mathrm{R}} \sum_{d \in \mathrm{DD}} d(x)}{|\mathrm{R}| \cdot |\mathrm{DD}|} \geq (1-\varepsilon).$$

Here $d(x)$ is 1 if item d occurs in x and 0 otherwise and for a set S, |S| indicates the cardinality of S. The weak definition requires that the data sub-matrix defined by the records in R and the columns

1	2	3	4	5	6
1	1	1	1	1	
1	1	1	1	1	
1	1	1	1		
1	1		1	1	
1	1	1	1	1	
			1	1	1
			1	1	1
				1	1
			1	1	1
			1		1

Table 3

in DD be mostly "1"s, with the fraction of "0"s not greater than ε. It is clear that anything that satisfies the strong definition above also satisfies the weak definition, but not vice versa.

In both definitions above, we say that the set DD *defines* the ETI, and we call κ the *support threshold* and ε the *error threshold*. A set of defining dimensions DD is called *maximal* if and only if DD defines an ETI and no superset of DD defines an ETI. For example, in Table 3 with κ=40% and ε=20%, {1 2 3 4 5} defines a maximal strong ETI, {1 2 3 4} defines a non-maximal strong ETI, and {4 5 6} defines a maximal weak ETI (but not a strong ETI). Our ultimate goal is to find strong ETIs, but the notion of weak ETIs will also be needed in one of the proposed algorithms.

2.1 Properties

Lemma 1: If a set DD of m dimensions defines a weak ETI, then there exists a set DD' of m-1 dimensions such that DD'$\subset$ DD and DD' also defines a weak ETI. (In other words, it is possible to remove one defining dimension from DD and still maintain a weak ETI).

Sketch of proof: Assume to the contrary that there does not exist such a set DD'. Then, over the set of records R that make DD a weak ETI, any subset of DD with one item removed will not form a weak ETI and therefore will not have enough number of "1"s. This leads to the contradiction that DD does not have enough number of "1"s to make a weak ETI. See [YFB00] for details.

Lemma 2: Given a set of m dimensions, their eligibility to be defining dimensions for a weak ETI can be tested with one pass over the database.

Proof: The test can be done by the following algorithm, which makes one pass over the database:

While scanning the database once, we keep m+1 counters C_0, C_1, …, C_m where C_i is the number of data points (records) that have i "1"s out of m candidate dimensions, i = 0,1,2,…, m. From these counters, we find the maximum t such that: $\sum_{i=t}^{m} C_i \geq \kappa \cdot n$, where n is the total number of records in the database. With this t value, let $C_t^{'} = \sum_{i=t}^{m} C_i - \kappa \cdot n$. Define

$$\delta = \frac{1}{\kappa \cdot n \cdot m}\left[\left(\sum_{i=t+1}^{m}(m-i)C_i\right) + (m-t)(C_t - C_t^{'})\right].$$

Then, the m candidate dimensions are eligible to be defining dimensions for a weak ETI if and only if $\delta \leq \varepsilon$, and δ is referred to as the *error rate* of this weak ETI. The algorithm works as follows: it ranks all rows based on the number of "1"s out of m candidate dimensions, picks the top κn rows, and checks the frequency δ of "0"s occurring among the κn rows over m candidate dimensions. Such ranking ensures that, if we picked any other set of κn rows, the frequency of "0" occurring among those rows over m candidate dimensions would have been at least δ. Hence, if $\delta > \varepsilon$, no weak ETI exists. If $\delta \leq \varepsilon$, these κn rows and m dimensions form a weak ETI, so a weak ETI exists.

2.2 The Exhaustive Algorithm

The lemmas in the previous section suggest the following algorithm to find maximal weak or strxong ETIs, which parallels the *a-priori* algorithm [AIS93, AS94]:

Exhaustive Growing Algorithm:

1. Find all dimensions d_i where the global count of "1"s is at least κn (1-ε). Each of these dimensions forms a singleton set {d_i} which defines a weak ETI. We call each of these singleton sets a "seed". Set i = 1.
2. For every seed that contains i dimensions, grow it by adding a dimension to it so that the new seed still defines a weak ETI (obtained with one pass over the database as in the proof of Lemma 2). If one or more such dimension can be found, keep all of the new seeds (each of which contains i+1 dimensions).
3. Increment i and repeat step 2 until no more growing is possible.
4. (to find maximal strong ETIs) Among all seeds, pick those satisfying the strong ETI definition (done in one pass over the database in a straightforward way). Output the maximal strong ETIs.

Lemma 1 ensures that any weak ETI can be built iteratively by adding new dimensions to smaller weak ETIs. Therefore, this algorithm will find all weak or strong ETIs.

2.3 Approximate Approach: Greedy Growing

Time complexity of the exhaustive growing algorithm is exponential in the maximum number of ETI defining dimensions. To reduce complexity, we modify it to a polynomial time greedy method that finds most of the ETIs. In practice, it is observed that the chance of the approximate approach missing ETIs is very low. We characterize these situations below.

Algorithm modifications are summarized in three stages. In the first stage, we show how to reduce complexity. In the second stage we show how missed ETIs can be recovered by doing a few iterations. This second stage also efficiently identifies clusters of similar transactions. In the third stage (Section 3), the overall scheme can be scaled with a sampling and validation framework.

2.3.1 Heuristics to Reduce Complexity

First, we make three changes to step 2 of the exhaustive growing algorithm: (i) when looking for a dimension to grow a seed, consider only those dimensions that have been picked in step 1, i.e., dimensions that have enough "1"s to form singleton weak ETIs; (ii) when testing whether a dimension can be added to a seed, we require the expanded seed still define a weak ETI, and that the new dimension have at most a fraction ε of "0"s within the weak ETI; (iii) when two or more dimensions are candidates for seed growth, keep only one. Throw away the old seed once it has been grown to a new seed.

The first two changes remove from consideration those dimensions that have too few "1"s globally or too few "1"s in a weak ETI. Even though those dimensions could potentially be in a weak ETI as dimension 5 in Table 2 (which could be part of a weak ETI {1 2 3 4 5}), they are not likely to make any interesting contributions to the result in real-world applications.

To implement the second change, we need to augment the algorithm given in the proof of Lemma 2. In addition to the counters $C_0, C_1, \ldots, C_m$, we keep an extra set of m+1 counters $Z_0, Z_1, \ldots, Z_m$, where Z_i keeps track of how many "0"s there are in the new candidate dimension d, over those data points that have i "1"s out of m existing candidate dimensions. These counters, together with the other counters $C_0, C_1, \ldots, C_m$, can be updated in the same pass over the database. Then, the fraction of "0"s along the new dimension d within the weak ETI is approximately the fraction

$$\frac{(\sum_{i=t+1}^{m} Z_i) + \max(0, Z_t - C_t^{'})}{\kappa \cdot n}.$$

1	2	3	4	5	6
1	1	1	1		
1	1	1	1		
1	1	1	1		
		1	1	1	1
		1	1	1	1
		1	1	1	1
1	1			1	1
1	1			1	1
1	1			1	1

Table 4

The third change ensures that the total number of seeds at any time would not exceed the total number of seeds we started with, reducing the exponential behavior to polynomial. When two or more candidate dimensions are found, we use the heuristics that picks one that incurs the smallest error in the new weak ETI. Other heuristics are possible. Although the third change dramatically reduces the amount of time and memory required, it may cause some ETIs to be missed, such as {1,2,5,6} in Table 4 (with κ=33% and any ε). We address this issue by considering an iterative extension to the algorithm.

2.3.2 Iterative Scheme to Improve Approximation

We make two more changes. First, after steps 1, 2 and 3 of the exhaustive algorithm (in which step 2 is modified as in section 2.3.1), a database scan is made to check if each row is covered by the discovered ETIs. (A row r is covered by an ETI if the fraction of items in the ETI which are present in r is at least 1-ε.) Then steps 1, 2, and 3 are performed again over all rows that are not covered by any ETI. We repeat this process until no more ETIs can be found. Each pass of steps 1, 2 and 3 is called a "round". It is observed that, typically, no ETIs are found after 2 or 3 rounds.

Starting with the second round, we replace the support threshold κ with a smaller value κ/λ, λ > 1, except at the very last step (corresponding to step 4 in the exhaustive growing algorithm), where we use the original κ value to pick out strong ETIs. Increasing λ reduces the probability of missing ETIs, but at the same time increases run time and memory requirements. Typically, we use λ=2. Returning to our example of Table 4, this would enable the algorithm to discover the missing ETI {1,2,5,6}.

2.3.3 Summary of Greedy Growing Algorithm (GGA)

As above, we have converted the Exhaustive Growing Algorithm into the Greedy Growing Algorithm, which is summarized below.

1. Set of candidate dimensions = {all dimensions whose count of "1"s in the database is at least κn(1-ε)/λ} (λ is initialized to 1 but will be set to 2 after the first round.)
2. If no candidate dimension exists, go to step 8.
3. Each candidate dimension forms a singleton seed.
4. Grow every seed by trying to add one best candidate dimension to it, while maintaining weak ETIs with support threshold κ/λ and error threshold ε.
5. Repeat step 4 until no seed can be grown further.
6. Add all fully-grown seeds to the set of potential ETIs.
7. Remove from the database all rows covered by any potential ETI. Set λ=2 and go to step 1.
8. Restore the original database, count the number of rows covered by every potential ETI, and remove those ETIs that are covered by fewer than κn rows. Output all remaining ETIs.

2.3.4 Analysis

GGA is able to find all maximal strong ETIs in the database except: (i) ETIs consisting of a dimension whose global count of "1"s is smaller than κn(1-ε)/λ; (ii) ETIs consisting of a dimension whose frequency of "0"s within the ETI is greater than ε; (iii) ETIs consisting of fewer than κn(1-ε)/λ unique rows (a row is unique to an ETI if it does not belong to any other ETI), and no unique dimension (a dimension is unique to an ETI if it does not occur in any other ETI).

Cases (i) and (ii) correspond to the degenerate case shown in Table 2. An example of case (iii) is shown in Table 5, with κ=30% and ε=0. Here the GGA algorithm (with λ=2) finds ETIs {1,2,3} and {3,4,5}, but not ETI {2,3,4}, which consists of only 1 unique row and no unique dimension. It is observed that this case is not particularly important in our primary application: identifying similar clusters of transactions.

1	2	3	4	5
1	1	1		
1	1	1		
1	1	1		
	1	1	1	
	1	1	1	1
		1	1	1
		1	1	1

Table 5

1	2	3	4
1	1	1	1
1	1	1	1
1	1	1	1
1	1		
1	1		
1	1		
1	1		

Table 6

Also, a side-effect of the iterative scheme of the algorithm is that it may find some non-maximal strong ETIs, as illustrated in Table 6, with κ=40% and ε=35%. In Table 6, both {1,2,3,4} and {1,2} will be identified as strong ETIs in successive iterations, although {1,2} is not strictly maximal (however, it covers many rows that do not overlap with the rows covered by ETI {1,2,3,4}). An additional step can be added to GGA to remove such non-maximal ETIs if desired, but in some real-world applications one would prefer not to remove them, as we will see in section 4.

Worst-case running time of the Greedy Growing Algorithm is O($cd'h^2$), where: c = # of ETIs, d' = average # of defining dimensions in ETIs, and h = # of high-support dimensions (dimensions whose global count of "1"s is at least κn(1-ε)/λ. There are h seeds. Typically h is small. Each seed may grow by adding one of h possible candidate dimensions, at each of d' growing steps. In the worst case, c iterations are needed to find all ETIs (one in each iteration). So O($cd'h^2$) is the worst-case time complexity. However, in most cases, all ETIs are found in 1 or 2 iterations, in which case the running time is only O($d'h^2$).

Memory requirement is O(hd'+cd'+d), where d is the total number of dimensions. We need O(hd') space to store all seeds while they are being grown, O(cd') space to store all ETIs found, and an additional O(d) space as a buffer to count the number of "1"s in every dimension.

3. ITERATIVE SAMPLING AND VALIDATION

GGA makes multiple passes over the database and can be quite slow when the database is large. In this section, we extend the algorithm by considering iterative sampling and validation scheme, wrapped around the original GGA algorithm. The sampling framework results in the algorithm SGGA.

3.1 Framework

Instead of running GGA on the entire database, we randomly sample two independent non-overlapping samples of the database, run GGA on one sample and validate the results on the other. Then, we return to the original database and check if any rows are still uncovered by the ETIs discovered so far. If so, we repeat the process on the remaining database until no more new ETIs can be discovered. This is illustrated in Figure 1.

3.2 Validation Schemes

We randomly sample the database into two independent non-overlapping samples T (training set) and V (validation set) and run GGA on the set T to get a list of ETIs. Some of the ETIs may contain spurious defining dimensions which exist due to sampling. Also, some of the ETIs may be spurious.

Identifying Spurious Defining Dimensions of an ETI: An artifact of sampling to be avoided is adding a dimension to an ETI description if it happens to occur by chance with the other defining dimensions over the random sample T. Suppose the first defining dimension of an ETI description was really random, but we observe that it occurs frequently along with the remaining defining dimensions in the ETI S. We identify this situation by first considering the ETI with the first defining dimension removed, then count the number of points from the validation set V satisfying the reduced ETI description. Determining whether or not the first defining dimension is spurious is done by comparing the frequency of observing a "1" in the removed dimension over points satisfying the reduced ETI description with the frequency of observing a "1" over the entire validation set. Let the removed

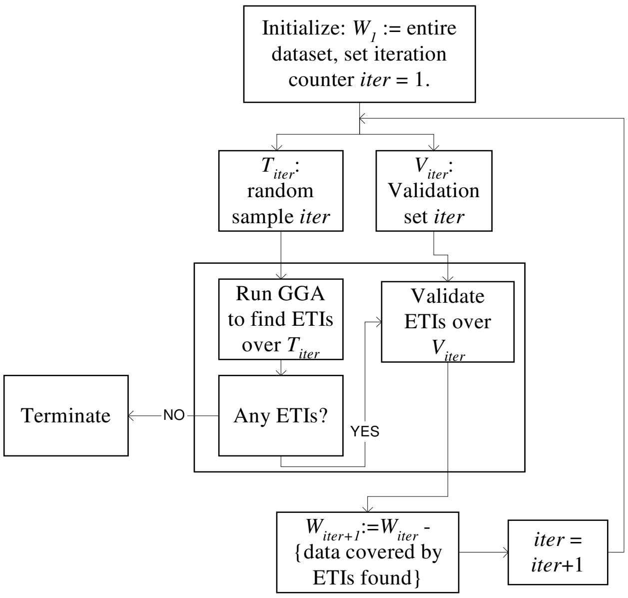

Figure 1: Flowchart of SGGA: sampling and validation scheme

defining dimension for ETI S be d_1. Let $p(d_1 \mid \{S - d_1\})$ be the fraction of "1"'s in dimension d_1 over the data in the validation set V belonging to the reduced ETI with defining dimensions in S minus d_1. Let $p(d_1)$ be the fraction of "1"'s in dimension d_1 over the entire validation set V.

If $p(d_1 \mid \{S - d_1\})$ is within 1 standard deviation from the value $p(d_1)$, then we consider dimension d_1 as a spurious defining dimension and remove it from the description of S. The standard deviation of $p(d_1)$ is given by: $s(d_1) = \sqrt{\frac{p(d_1)\left(1 - p(d_1)\right)}{|T|}}$.

Specifically, dimension d_1 is removed from the defining dimensions for ETI S if:

$$p(d_1) - s(d_1) \le p(d_1 \mid \{S - d_1\}) \le p(d_1) + s(d_1).$$

We do this for every defining dimension of every ETI, and prune out defining dimensions that appear spurious.

Identifying Spurious ETIs: To filter out ETIs found by chance over a random sample from the database, T, we consider the number of rows of T which match the ETI to be a random variable. Suppose the ETI has n points in it and the random sample from T consists of $|T|$ points. If the ETI truly exists in the database, then given another random sample V, one would expect to observe approximately the same proportion of points belonging to the ETI in V. If the proportion of points belonging to the ETI in V is too low, then one may conclude that the ETI is spurious and needs to be adjusted. This motivates the mechanism for removing spurious ETIs found over random samples.

Let $n(T)$ be the number of data points from T which belong to the ETI. We view the proportion of points in a random sample belonging to S as a random variable p. We estimate the value of p over T as: $p = [n(T)/|T|]$. The standard deviation in this estimate is given by: $s = \sqrt{\frac{p(1-p)}{|T|}}$. We consider an ETI to be valid if the fraction of data points belonging to it over the validation set, $n(V)$, is not less than the standard deviation of the expected fraction in V: $\frac{n(V)}{|V|} \ge \frac{n(T)}{|T|} - \sqrt{\frac{p(1-p)}{|T|}}$.

ETIs which do not satisfy the above criteria must be adjusted (if possible) so that they are valid. We do so by iteratively removing from an ETI the defining dimension with the least number of "1"s, until the ETI satisfies the criteria, or until the ETI becomes empty.

4. APPLICATIONS AND EVALUATION METHODOLOGY

We have tested SGGA on both synthetic data and real-world data. For synthetic data, since we know the "true" ETIs in the database, we evaluate performance of our algorithm based on the number of true ETIs found and number of extra (false) ETIs found. For real data, we evaluate performance based on three applications: clustering, query selectivity estimation and collaborative filtering prediction, which are described below. For real-data experiments, we also include results of random initial grouping for comparison.

We omit the comparison with traditional (non-error-tolerant) frequent itemsets for the following two reasons. First, when applied to the task of clustering transactions over sparse {0,1} data matrices, frequent itemsets generate too many small clusters. Secondly, it is observed frequent itemsets perform poorly when modeling sparse, {0,1}-valued database specifically to support query selectivity estimation and collaborative filtering.

4.1 Determining Initial Mixture Models

ETIs are extremely effective in identifying structure in sparse {0,1} data. This structure can be easily exploited by using ETIs to form an initial statistical model of the underlying data. This initial model may then be further fit to data using an iterative optimization procedure such as the Expectation-Maximization (EM) algorithm [DLR77, CS96, BFR98]. EM estimates the probability density function of the data as a mixture of densities (binomial distributions for {0,1} data). Each cluster corresponds to one component of the mixture model and specifies a distribution for its population. For an observation (transaction) x, the probability that x is a member of cluster C, is proportional to $\Pr(x \mid C) = \prod_{i \in I} \Pr(x_i \mid C)$. Hence the model for each cluster is simply the product of the probabilities of observing each attribute (item). Note that the assumption of conditional independence within clusters (implied by the equality above) is different from assuming attribute independence. The conditional independence assumption, typically applied when clustering discrete data in the statistics, learning, and pattern recognition literature [CS96, BFR98,DLR77], is a much more powerful model than the global independence model. In the case of binary {0,1} data, we model each attribute with a binomial probability distribution within a cluster or component. In this case $\Pr(x_i \mid C)$ is simply $\Pr(x_i = 1 \mid C)$ if item i occurs in the record x or $(1 - \Pr(x_i = 1 \mid C))$ if item i does not appear in x.

The utility of the mixture model statistical summary of a given database has been demonstrated in data mining tasks such as speeding up nearest neighbor queries [BFG99] and approximating OLAP aggregate queries [SFB99]. The mixture models used in query selectivity estimation in [SFB99] were generated over continuous-valued data only. The query selectivity task addressed here is focused on sparse {0,1}-valued data. The current query selectivity estimation study here was partially motivated by unsatisfactory results when using standard methods to compute mixture models over sparse {0,1}-valued data. It is observed that mixture models computed by EM when initialized with ETIs are superior in this case (see Section 5.2.3).

While EM provides many advantages for clustering data, including the fact that it produces a statistically meaningful model, the quality of the solution it produces is determined completely by the initial model. The state of the art of initializing EM over {0,1} data is via random restarts [MH98]. When applied to high-dimensional data, EM suffers from two problems in particular: clusters often go empty (no data records assigned to them); and different clusters converge on the same distribution (two or more clusters become identical). In either case, the algorithm effectively identifies fewer clusters in the data. Certainly this should happen if the user asked the algorithm to find more clusters than truly exist in the data. However, in practice these problems surface frequently when clustering high-dimensional sparse data when more structure actually exists. We demonstrate that ETIs consistently uncover more structure in the data than EM is capable of discovering with a multitude of repeated random initializations.

Each ETI is treated as a potential cluster. To construct a binomial model for each ETI (which will be the initial binomial distribution of that cluster) a pass is made over the data, collecting counts of items for all transactions that belong to the ETI. A transaction belongs to an ETI if the fraction of items in the ETI which are present in the transaction is at least 1-ε. Since ETIs can overlap on items, some transactions could belong the multiple ETIs. A transaction t that belongs to a set of ETIs gets fractional membership in each ETI proportional to the number of defining dimensions in that ETI. Hence longer ETIs, when they match a transaction, claim "more" of the transaction than shorter ETIs. With a single pass over the database, the fractional membership of each transaction in each ETI is computed and the corresponding item counts for items appearing in the transaction are incremented by the fractional membership value. Typically there is only one matching ETI. At the completion of the scan, the counts for each ETI are normalized to probabilities, and an initial cluster model for each ETI is produced.

This initial model is then refined via EM. If EM converges with stable clusters (few clusters have gone empty and the clusters are distinct), it is an indicator that the ETIs detected real structure in the data. Hence a quick way to compare a clustering solution based on ETIs with one obtained from a random starting model is to compare the number of non-empty clusters when EM converges. See results in Section 5.2.2.

4.2 Query Selectivity Estimation

Having derived a statistical model based on ETI initialization, one can evaluate the fit of the model to the data. An application measuring this directly is to estimate a `count(*)` query using the model and compare the difference between the model-based estimate and the true value of the query. A comparison is made between the statistical model estimated from ETI initialization and the statistical model estimated from random initialization.

Queries with h conjunctive items are generated with respect to a statistical cluster model as follows.

1. Cluster C is selected with probability given by the number of transactions belonging to cluster C divided by the total number of transactions.
2. Generate a distribution over each attribute $i \in I$ given by: $p_{i,C} = \dfrac{\Pr(x_i = 1 \mid C)}{\sum_{i \in I} \Pr(x_i = 1 \mid C)}$.
3. Select h items randomly according to the computed probabilities $p_{i,C}$, yielding the `where`-clause of a `count(*)` query.

An aggregate query, such as the number of shoppers who bought items 1, 5, and 16 together, is estimated by a probabilistic model by computing the probability of the event that items 1, 5, and 16 occur together in each cluster and summing these values weighted by the size of each cluster. Not only is this an extremely fast operation involving access only to the model and no access to the data, but it can also be a very accurate estimator for an underlying model that fits the data well. See results in Section 5.2.3.

4.3 Collaborative Filtering Prediction

Given a set of items, the collaborative filtering prediction task is that of predicting other items that typically occur with the given set [MRK97,R*97,RV97]. For example, given that a set of web pages have been browsed by a user, predict other web-pages that are likely to be browsed by the user in this session. By using the given set of items (viewed as a partial record, since more items may be added), it is possible to associate the record with a cluster. Once the record has been associated with a cluster (possibly with fractional membership), it is possible to predict other items that would most likely appear with the given set based on the values of $\Pr(x_i \mid C)$. Intuitively, an item i with a large corresponding value of $\Pr(x_i \mid C)$ would be likely to occur along with the given itemset. See [BHK98] for a detailed description. Results are given in Section 5.2.4.

5. RESULTS

5.1 Synthetic Data Experiments

We generated 70 different synthetic datasets while controlling the following parameters: total number of ETIs, average number of points in each ETI, average support of each ETI, number of dimensions, number of defining dimensions in each ETI, probability of a defining dimension being 1, probability of a non-defining dimension being 1, number of random dimensions, probability of a random dimension being 1, and degree of overlap of defining dimensions in different ETIs. For each dataset we measured the quality of result and running time. Results have shown that our algorithm is able to efficiently discover all true ETIs and very few false ETIs in almost every case, except when the noise level in the data (attributes with values generated at random, without correlation with other attributes) is very high. Also, scalability tests on synthetic data show that our algorithm performs well on large datasets. Details of the synthetic data experiments can be found in [YFB00]. Due to space limitations, we omit them here.

5.2 Results on Real Data

Error Tolerant Itemsets are particularly suited to identifying structure for cluster initialization over sparse {0,1} high-dimensional data. In this respect, we evaluate ETIs in this application over 4 real sparse {0,1} databases. ETI-initialized cluster models are evaluated in comparison to randomly-initialized cluster models in 3 areas: 1) preservation of the number of clusters in the model (see Section 5.2.2); 2) ability of the model to approximate `count(*)` queries over the database (see Section 5.2.3); 3) a collaborative filtering prediction task (see Section 5.2.4).

5.2.1 Real Databases

Web-1: This database consists of the browsing behavior of 516,511 users over 218 web-page categories. This data was obtained from a major internet service provider/web portal.

Web-2: This database consists of the browsing behavior of 602,479 users over 9822 websites. This data set was obtained from one of the largest web-content providers on the internet.

Product-Purchase-1: This database consists of the products purchased by 29989 users. The number of possible products is 297. This was obtained from a small e-commerce site that sells software products.

Product-Purchase-2: This database consists of purchasing behavior of 703,510 customers purchasing a subset of 32,301 products. This data set was obtained from a major software/hardware retailer.

5.2.2 Number of Clusters

For a given minimum support value κ, SGGA was applied to the database. Upon termination, the ETIs constructed were used to initialize the binomial cluster model. EM [BFR98] was then applied to the database to refine the given initial model. EM was also applied to the database from 5 random initial binomial cluster models. The number of clusters in the random models was initially set to the number of ETIs returned by SGGA. We then compare the final number of clusters from the ETI-initialized model with the average number of clusters from the 5 random initial models. Results are summarized for the 4 datasets in Figure 2. The random initial models had the number of clusters equal to the corresponding ETI-value. *No clusters went empty when initialized via ETIs.* Random values are averages over 5 random initial models; error bars are 1 standard deviation.

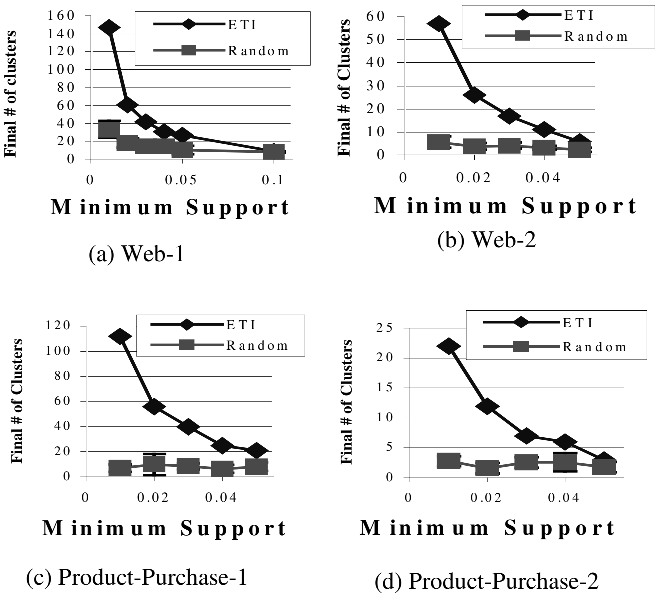

Figure 2: Final number of clusters after EM converges

Note that when EM is initialized via ETIs, no empty clusters are observed. In contrast, when initializing EM with a random cluster model with the same number of clusters as the corresponding ETI model, *EM converges with many empty clusters*. For minimum support κ = 0.01, ETI-initialized models uncovered as many as 16 times as many clusters as randomly initialized models on Product-Purchase-1 database. On average over the 4 databases tested, for minimum support κ = 0.01, ETI-initialized models uncovered 9.6 times as many clusters as the randomly initialized models.

5.2.3 Query Selectivity Estimation

Similar to Section 5.2.2, for a given minimum support value κ, SGGA was applied to the database. The ETIs constructed were used to initialize the binomial cluster model as in the previous section. As above, we compare against initializing EM with 5 different random binomial cluster models. The number of clusters in the random models was initially set to the number of ETIs returned by SGGA. We then randomly generated 50 `count(*)` queries with 2 items in the `where` clause, based on the ETI-initialized model as described in Section 4.2. Experiments were performed with 2 and 3 conjunctions. It was observed that more than 3 conjuncts usually result in zero or a very small count result. When a query has a small result (e.g. less than 5), it is known that the estimate of these small queries with statistical models is poor [SFB99]. Estimates of `count(*)` queries with only 1 attribute in the "`where`" clause were deemed uninteresting as these

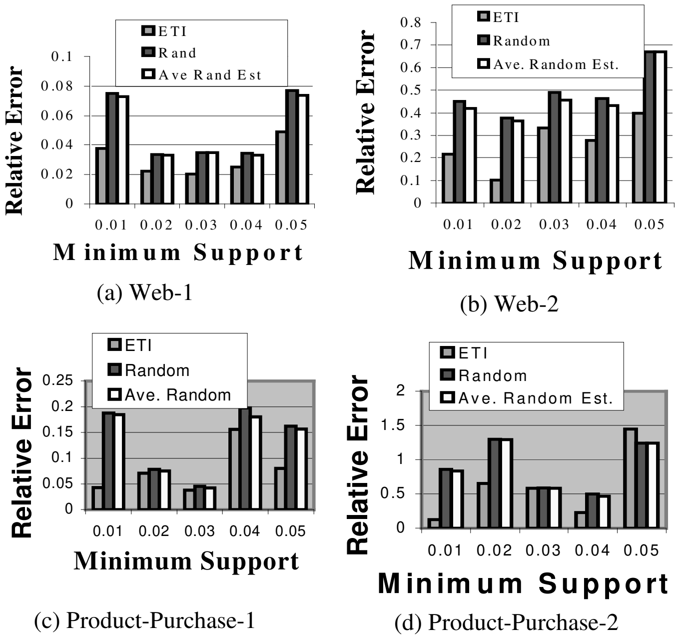

Figure 3: Average query selectivity error for 4 real databases.

queries can be effectively estimated without modeling dependency among attributes. For very long queries, a constant zero is a good estimator, hence uninteresting. Suppose the "`where`" clause of the query of interest consists of items *i* and *j*. The probability that these two items appear in a given cluster *C* is $\Pr(x_i \mid C) \cdot \Pr(x_j \mid C)$. Let $N(C)$ be the number of records in cluster *C*, the query :

```
"select count(*) from DB.Table where (i = 1)
and (j = 1)"
```

is then simply approximated by the cluster model by: $\sum_C N(C) \cdot \Pr(x_i \mid C) \cdot \Pr(x_j \mid C)$.

Average results over 50 two conjunct queries (see Section 4.2 for description of query generation) for the ETI-initialized model are given in Figure 3 noted as "ETI". Results over 3-conjunct queries yielded similar results. Average results over the same 50 queries and over the 5 randomly initialized models are noted in Figure 3 as "Random". The method noted as "Ave. Random Est." in Figure 3 refers to the following: for each individual query, we obtain the 5 approximations with the random models, then average these to get an approximate result for the specific query in question. Relative error in Figure 3 is obtained by taking the absolute value of the difference between the true result and the approximation and dividing by the true result. Over all databases, we discarded queries where the true result was less than or equal to 5.

Over all 4 databases the ETI-initialized models were better query estimators than the randomly initialized models, except over the Product-Purchase-2 database with minimum support $\kappa = 0.05$. For this value of minimum support, the number of ETIs found was 3 and the average number of clusters from random initialization was 1.8. Hence these models are very similar. We note that data which does not match any ETI does not contribute to the initial cluster model. So in this case where there are only 3 ETIs we conjecture that there is a non-trivial portion of the data space which was not appropriately modeled by the ETI-initialized cluster model and multiple randomly initialized cluster models could capture more of this space.

5.2.4 Collaborative Filtering Prediction

We tested the utility of cluster models initialized via ETIs in the collaborative filtering task of predicting items that a transaction may contain based on a set of given items occurring in the transaction. We compared the predictive accuracy of ETI-initialized cluster models and randomly initialized models using a subset of data not used in the clustering process. Recall that for databases Web-1, Web-2 and Product-Purchase-2, clustering was performed over a random sample of 100,000 records. For these databases, an additional 10,000 records were held out and used to score the predictive accuracy of the models. For the Product-Purchase-1 database, having 29989 records, 10% of the records were set aside for scoring and the cluster models were built on the remaining 90%.

The scoring procedure was the following. For each record in the hold-out set, we remove one of the items occurring in the record. Call this record with 1 item removed the *partial transaction*. The partial transaction is assigned to clusters (with fractional membership) based on the remaining items. Let $w(C)$ be the fractional membership assignment for cluster *C*. The $w(C)$ values satisfy: $\sum_C w(C) = 1$. For every missing item in the partial transaction (including the item removed), we ask for a prediction for that item. For missing item *j*, the prediction is given by: $p(j) = \sum_C w(C) \cdot \Pr(x_j \mid C)$.

The missing items are sorted in descending order by their $p(j)$ values. If the item which was explicitly removed occurs in the top 10 of the sorted list, the score is incremented by 1. Accuracy is then the score divided by the number of items held out over the entire hold-out set. Accuracy scores are summarized for the 4 databases in Figure 4 for ETI-initialized cluster models and average collaborative filtering predictive accuracy over 5 randomly initialized cluster models. Error bars associated with the random results are 1 standard deviation.

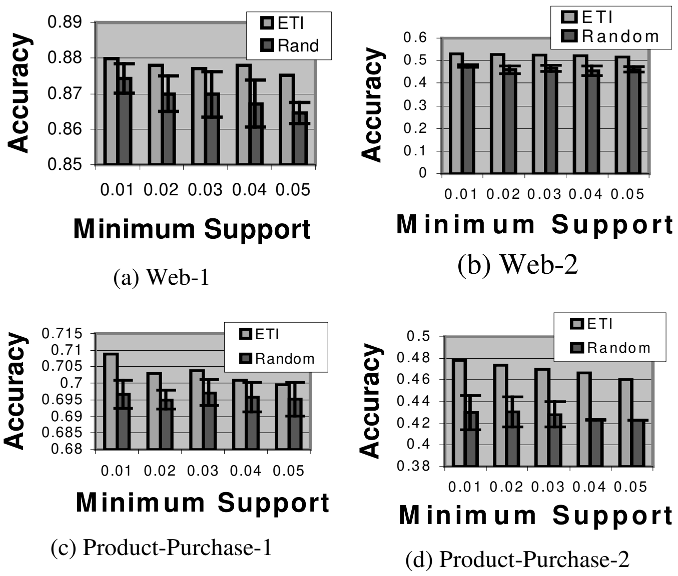

Figure 4: Collaborative filtering predictive accuracy

6. GENERALIZATION TO CATEGORICAL DATA

Note that GGA and SGGA do not fundamentally rely on the assumption that the data is binary. The algorithms can be generalized to handle categorical data as well. For efficiency, we assume that each attribute has one value that is somehow distinguished as the "default" value. In transaction data, that value is implicitly the "zero" value. Assuming that the "default" value for each attribute is pre-dominant (i.e. the data can be efficiently represented in sparse format), we can generalize GGA and SGGA to count combinations of attribute values that do not involve the "default" value for each attribute. Note that the treatment is almost identical algorithmically by viewing each non-default value of a multi-valued categorical attribute as a new binary attribute. As long as we have an efficient mechanism for counting frequent combinations of attribute values, the algorithm carries through and can be applied to finding clusters in categorical data. This is similar to the algorithm presented in [SA96].

7. RELATED WORK

Frequent itemsets were first developed by Agrawal et al. in the *a-priori* algorithm for association rule mining [AIS93, AS94, AMSTV96]. The key optimization in finding frequent itemsets was based on the fact that, if an itemset of length m has enough support, then any of its subset of length m-1 must also have enough support. This property enables building frequent itemsets in a bottom-up manner. As we introduce the notion of errors into the definition of frequent itemsets, this property no longer holds. A similar but weaker property exists as discussed earlier, but it was not sufficient to ensure fast discovery of error-tolerant frequent itemsets, hence the additional optimization schemes developed in this paper.

One problem that arises in *a-priori* is that the algorithm scales exponentially with longest pattern length. Many variants have been proposed to address this issue. Zaki et al. [ZPOL97] developed the algorithms MaxEclat and MaxClique which "look ahead" during initialization so that long frequent itemsets are identified early. Bayardo [B98] presented an optimized search method called Max-Miner that prunes out subsets of long patterns of frequent itemsets that are discovered early. Gunopulos et al. [GMS97] suggested iteratively extending a working pattern until failure, using a randomized algorithm, which is similar to the idea we used in our algorithm to grow itemsets in a greedy fashion.

Much work has been done in automatic clustering methods. Traditionally, clustering is done by finding a set of centroids in a high-dimensional space [CS96, DLR77, NH94, ZRL96], and cluster membership is determined by some distance function to the centroids. This leads to cluster shapes similar to spheres. Later work by Guha et al. [GRS98] was able to handle arbitrarily-shaped clusters by using several representative points to define a cluster. They also used a sampling scheme to reduce I/O costs. Recognizing that most clusters are defined on subspaces rather than the entire high-dimensional space, Agrawal et al. [AGGR98] presented a method to build subspace clusters in a bottom-up way, using the property that if a collection of points form a cluster in a k-dimensional space, they must also form a cluster in all of its (k-1)-dimensional subspaces (where a cluster is defined as a dense region in the subspace). As presented in this paper, our algorithm to find error-tolerant frequent itemsets may be used independently as an efficient subspace clustering algorithm, with a more general definition of clusters than that used in [AGGR98]. It can also be used as an effective initialization method for existing cluster refinement algorithms such as EM.

Discrete clustering algorithms, as opposed to generalized frequent itemsets, include CACTUS [GGR99], STIRR [GKR98], and ROCK [GRS99]. The first two require the computation of a similarity matrix between all attributes (items), which takes $O(d^2)$ time (d = number of attributes). CACTUS uses a more efficient refinement method on the computed similarities and hence is faster. ETIs are a generalization to frequent itemsets, and cluster initialization is just an application. In fact, ETIs could be used as a preprocessor to these algorithms, reducing the similarity matrix needed and hence alleviating the primary bottleneck. ROCK requires a distance metric between transactions and is cubic in their number. In general, clustering is much more time consuming than extracting ETIs.

8. CONCLUDING REMARKS

We have presented a generalization to the standard frequent itemset problem and an efficient and scalable algorithm to find error-tolerant frequent itemsets (ETIs). ETIs describe simpler and more intuitive frequent structures in data. Starting with an exhaustive approach which guarantees that all such ETIs will be found, we developed an efficient approximation which runs in polynomial time and produces good results. We demonstrated that this method can be used as a fast initialization method for clustering algorithms such as EM, and generates far more stable models than existing techniques. Query selectivity estimation and collaborative filtering are two other useful applications of our algorithm.

One possible future direction is to study the extension of the algorithm to continuous-valued domains. Approaches suggested in [SA96] are applicable, but there may be other methods as well. We would like to explore its use to identify which attribute similarities to focus on in algorithms driven by such similarity matrices [GGR99,GKR98,GRS99]. It would also be interesting to explore other sampling schemes to improve performance. Furthermore, much of the previous work that made use of traditional (error-free) frequent itemsets can now be reconsidered in this new framework.

9. ACKNOWLEDGEMENTS

We gratefully acknowledge Jeong Han Kim and Dimitris Achlioptas for discussions and assistance regarding the probability of error-tolerant itemsets occurring by chance. We thank Ilya Vinarsky for help with implementation and experiments.

10. REFERENCES

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *Proc. of the ACM SIGMOD Conf.*, 1998.

[AIS93] R. Agrawal, T. Imielinski and A. Swami. Mining Association Rules Between Sets of Items in Large Databases. In *Proc. of the ACM SIGMOD Conf.*, 1993, pp. 207-216.

[AMSTV96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A. I. Verkamo. Fast Discovery of Association Rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R.

Uthurusamy (eds.), *Advances in Knowledge Discovery and Data Mining*, pp. 307-328, AAAI Press, Menlo Park, CA, 1996.

[AS94] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th International Conference on Very Large Databases*, 1994.

[B98] R. J. Bayardo Jr. Efficiently Mining Long Patterns from Databases. In *Proc. of the 1998 ACM SIGMOD Int'l Conf. on Management of Data*, pp. 85-93, 1998.

[BFG99] K. P. Bennett, U. M. Fayyad and D. Geiger. Density-based Indexing for Approximating Nearest Neighbor Queries. *Proc. KDD-99*, ACM Press, 1999.

[BFR98] P. S. Bradley, U. M. Fayyad and C. Reina. Scaling EM (Expectation-Maximization) Clustering to Large Databases. *Technical Report MSR-TR-98-35*, Microsoft Research, 1998.

[BHK98] J. Breese, D. Heckerman and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *Technical Report MSR-TR-98-12*, Microsoft Research, 1998.

[CS96] P. Cheeseman and J. Stutz. Bayesian Classification (AutoClass): Theory and Results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurasamy (eds.) *Advances in Knowledge Discovery and Data Mining*, pages 153-180. AAAI Press, Menlo Park , CA, 1996.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society B*, 39:1-38, 1973.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

[FI93] U.M. Fayyad and K.B. Irani. "Multi-interval Discretization of Continuous-valued Attributes for Classification Learning." *Proc. of the 13th Intl. Joint Conf. on Artificial Intelligence. IJCAI-93*: Chambery, France (1993).

[GGR99] V. Ganti, J. Gehrke, and R. Ramakrishnan. "CACTUS – Clustering Categorical Data Using Summaries". *Proc. of KDD-99,* ACM Press, 1999.

[GKP89] R. L. Graham, D. E. Knuth and O. Patashnik. *Concrete Mathematics*. Addison Wesley, Reading, MA, 1989.

[GKR98] D. Gibson, J. Kleinburg, and P. Raghavan. "Clustering categorical data: an approach based on dynamical systems". *Proc. VLDB-98,* pp. 311-323. 1998.

[GMS97] G. Gunopulos, H. Mannila and S. Saluja. Discovering All Most Specific Sentences by Randomized Algorithms. In *Proc. Of the 6th Int'l Conf. On Database Theory*, pp. 215-229, 1997.

[GRS98] S. Guha, R. Rastogi and K. Shim. CURE: An efficient algorithm for clustering large databases. In *Proceedings of the ACM SIGMOD conference*, 1998.

[GRS99] S. Guha, R. Rastogi, K. Shim. "A Robust Clustering Algorithm for Categorical Attributes". *Proc. ICDE-99*, IEEE Press, 1999.

[MH98] M. Meila and D. Heckerman. An Experimental Comparison of Several Clustering and Initialization Methods. *Technical Report MSR-TR-98-06*, Microsoft Research, 1998.

[MRK97] B. Miller, J. Riedl and J. Konstan. Experiences with GroupLens: Making Usenet Useful Again. In *Proc. USENIX 1997 Tech. Conf.*, pp. 219-231, Anaheim, CA, 1997.

[NH94] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the International Conference on Very Large Databases*, 1994.

[RG99] R. Ramakrishnan and J. Gehrke. Principles of Database Management (2nd Edition). 1999.

[R*97] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnew. In *Proc. ACM 1994 Conf. Computer Supported Cooperative Work,* pp. 175-186, New York. ACM, 1997.

[RV97] P. Resnick and H. Varian. Recommender Systems. *Comm. of the ACM*, 40(3):56-58, 1997.

[SFB99] J. Shanmugusundaram, U. M. Fayyad and P. S. Bradley. Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions. In *Proc. 5th Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 223-232, 1999.

[SA96] R. Srikant and R. Agrawal. Mining Quantitative Association Rules in Large Relational Tables. In *Proceedings of the ACM SIGMOD Conference*, 1996.

[YFB00] C. Yang, U. Fayyad and P. S. Bradley. Efficient Discovery of Error-Tolerant Frequent Itemsets in High Dimensions. *Technical Report MSR-TR-2000-20*, Microsoft Research, 2000.

[ZPOL97] M. J. Zaki, S. Parthasarathy, M. Ogihara and W. Li. New Algorithms for Fast Discovery of Association Rules. In *Proc. of the Third Int'l Conf. On Knowledge Discovery in Databases and Data Mining*, pp. 283-286, 1997.

[ZRL96] T. Zhang, R. Ramakrishnan and M. Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD Conference*, 1996, pp. 103-114.

[Z49] G.E. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley Press, Inc, 1949.

[1] Work was done while author was at Microsoft Research.

[2] Work was done while author was at Microsoft Research.