Michael Hahsler

A model-based frequency constraint for mining associations from transaction data

Working Paper

A Model-Based Frequency Constraint for Mining Associations from Transaction Data

Hahsler, Michael
Michael.Hahsler@wu-wien.ac.at

Arbeitspapiere zum Tätigkeitsfeld
Informationsverarbeitung und Informationswirtschaft
Working Papers on
Information Processing and Information Management

Nr./No. 07/2004

Herausgeber / Editor:
Institut für Informationsverabeitung und Informationswirtschaft
Wirtschaftsuniversität Wien · Augasse 2-6 · 1090 Wien
Institute of Information Processing and Information Management
Vienna University of Economics and Business Administration
Augasse 2-6 · 1090 Vienna

Abstract

Mining associations is a popular method for finding interesting patterns in databases (e.g., items which are frequently purchased together). Normally support, the occurrence frequency of the items which form a pattern, is used as the primary indicator of the significance of a pattern and a user-specified support threshold is used to decided if a pattern should be further investigated. The application of support implies important assumptions which are rarely discussed or checked. Employing a minimum support threshold implies that the items occur in the database following a, possibly unknown, but stable process and that the items occur in the database with roughly similar frequencies. However, real-world transaction data is known to often possess a highly skewed frequency distribution with most items occurring infrequently while some items occurring extremely often. This situation leads to the rare item problem, where interesting patterns are not found since some of the associated items are too infrequent to satisfy the user-specified minimum support.

In this paper we develop an alternative to minimum support which utilizes knowledge of the process which generates transaction data and allows for highly skewed frequency distributions. We apply a simple stochastic model (the NB model), which is known for its usefulness to describe item occurrences in transaction data, to develop a frequency constraint. This model-based frequency constraint is used together with a precision threshold to find individual support thresholds for groups of associations. We develop the notion of NB-frequent itemsets and present two mining algorithms which find all NB-frequent itemsets in a database. In experiments with publicly available transaction databases we show that the new constraint can provide significant improvements over a single minimum support threshold and that the precision threshold is easier to use.

Keywords: Data mining, associations, interest measure, mixture models, transaction data.

Contents

1 Introduction

Mining associations and association rules between items in large databases is an area which is under intense research since *Apriori*, the first algorithm using the support-confidence framework, was presented in 1993 by Agrawal et al. [3]. The enormous interest in associations is due to their direct applicability for many practical purposes. Beginning with discovering regularities in transaction data recorded by point-of-sale systems to improve sales, association rules are nowadays also used to analyze Web usage patterns, for fraud and intrusion detection, mining genome data, and for many other applications.

An association or pattern is a set of items which occurs unexpectedly often together in the database and which provides useful and actionable insights into the structure of the database. Normally, support, the frequency of an association in the database, is used to find potentially useful associations. But using a minimum threshold on support implies the assumptions that all items occur in the database following the same, stable process and that all items occur with a roughly similar frequency. To check whether these assumptions hold for a particular application is important before employing support. However, in practice the assumptions are often neglected when the support-confidence framework or other constraints are proposed and discussed in current literature.

The assumption that the generation processes of the occurrence of items in the database are similar and stable (stationary) for all items during the analyzed period seems a reasonable assumption for most applications. Statistics already provides a multitude of stationary models which proved to be helpful to describe data which is mined for association rules (e.g., accident data, market research data including market baskets, data from medical and military applications, and biometrical data [15]). Recently, strong regularities were also found in data from Web usage (e.g. in [14], [5], and [18]) .

Violation of the assumption, that all items occur with similar frequencies in the database, has adverse effects on support. This needs special attention when dealing with transaction data since counting processes are known to generate power-law item frequency distributions. These massively violate the assumption and together with a minimum support threshold lead to the so-called *rare item problem* where associations which include items with low support are discarded although they might contain valuable information.

Although the effects of skewed item frequency distributions in transaction data are sometimes discussed (e.g. in [19] or [29]), most current approaches ne-

glect knowledge about the processes which underly the databases. In this paper we develop the notion of NB-frequent itemsets based on a model-based frequency constraint which uses no predefined support, but evaluates observed deviations from a baseline model to identify an individual frequency threshold for groups of associations. The proposed constraint has the following properties:

1. It utilizes knowledge of the process which underlies transaction data by applying a simple stochastic model which is known for its wide applicability.
2. It reduces the problem with rare items since the used stochastic model allows for highly skewed frequency distributions.
3. It is able to produce longer patterns without generating an enormous number of shorter, spurious patterns since the support required by the model decreases with pattern length.
4. It's parameter is a precision threshold which has a direct interpretation as a predicted error rate and seems to be less dependent on the structure of the database. This makes communicating and setting the parameter easier for domain experts.

The rest of the paper is organized as follows: First, we summarize mining associations and survey proposed alternatives for support. In section 3 we develop the model-based frequency constraint and show that the chosen model is useful to describe real-word transaction data. In section 4 we present algorithms to mine associations with the model-based constraint. In section 5 we investigate and discuss the behavior of the model-based constraint using several real-world and artificial transaction data sets.

2 Mining associations from transaction data

The problem of mining associations and rules from transaction data was introduced by Agrawal et al. [3] for mining association rules as: Let $I = \{i_1, i_2, ..., i_n\}$ be a set of n distinct items and $\mathcal{D} = \{t_1, t_2, ..., t_m\}$ a set of transactions called the database. Each transaction in $\mathcal{D}$ contains a subset of the items in I. A rule is defined as an implication of the from $X \Rightarrow Y$ where $X, Y \subseteq I$ and $X \cap Y = \emptyset$. The sets of items (for short itemsets) X and Y are called antecedent and consequent of the rule. Itemsets of length k are called k-itemsets.

To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The best known constraints are minimum thresholds on support and confidence often referred to as the support-confidence framework. Rules which satisfy both these constraints are called association rules.

2.1 Association rules

Agrawal et al. [3] define two measures for association rules: A measure of significance called *support* and a measure of strength called *confidence.* In the following we define these measures using probabilities which can be estimated from a database by counting the number of transactions an itemset $Z \subseteq I$ appears in and then dividing it by the total size of the database. For example, $P(Z)$ is estimated by $\frac{\text{freq}(Z)}{|\mathcal{D}|}$ where $\text{freq}(Z)$ denotes the frequency (number of occurrences) of itemset Z in database $\mathcal{D}$, and $|\mathcal{D}|$ is the number of transactions in the database.

Definition 1 (Support). *Support is defined on itemset Z as the probability that all its items are found together in transactions in the database:*

$$\text{supp}(Z) = P(Z) \tag{1}$$

An itemset Z is only considered for association rule mining if the constraint $\text{supp}(Z) \geq \sigma$ holds, where σ is a user-specified minimum support. Itemsets which satisfy the minimum support constraint are called *frequent itemsets.* Some authors refer to these itemsets as *large itemsets* [3] or *covering sets* [20]. Support is often also used in absolute terms as the frequency $\text{freq}(Z)$. We denote the support threshold on the frequency by $\sigma^{freq} = \sigma \cdot |\mathcal{D}|$.

The rational for minimum support is that items which appear more often in the database are more important since, e.g. in a sales setting they are responsible for a higher sales volume. However, this rational breaks down when some rare but expensive items contribute most to the store's overall earnings. Not finding patterns or rules for such items is known as the rare item problem which is inherent in frequency constraints such as minimum support. Support also favors smaller itemsets. By adding items to an itemset the probability of finding such itemsets in the database can only decrease or, in rare cases, stay the same. Consequently, with growing size the support of an itemset falls. Although this can be problematic for some applications, the fact that support cannot increase with growing itemset size enables algorithms to efficiently mine association rule.

Definition 2 (Confidence). *Confidence, the second measure introduced for association rules, is defined as the probability of finding itemset Y in a transaction under the condition that it also contains itemset X:*

$$\text{conf}(X \Rightarrow Y) = P(Y \mid X) = \frac{P(X \cup Y)}{P(X)} = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)} \tag{2}$$

As shown in formula 2 this conditional probability can be directly calculated from support values by dividing the support of the rule (support of the union of the items in the antecedent and in the consequent) by the support of the antecedent. A minimum confidence constraint $\text{conf}(X \Rightarrow Y) \geq \gamma$ is used to filter rules without the required strength.

A weakness of minimum confidence is that it favors rules with frequent consequents. In formula 2 it is easy to see that a higher count for Y (which results in a higher chance of seeing X and Y together, even if they are unrelated) directly translates into a higher confidence value. In a supermarket setting this means that confidence would mix the interesting rules into many uninteresting rules with items as consequents which many customers would buy anyway.

Association rule mining algorithms find all rules which satisfy both constraints, minimum support and minimum confidence, at the same time. This task is usually broken down into two subproblems:

1. Find the set $\mathcal{F} = \{Z \subseteq I | \text{supp}(Z) \geq \sigma\}$ of all frequent itemsets.

2. Generate for each frequent itemset $Z \in \mathcal{F}$ all possible rules $X \Rightarrow Y$ with $\text{conf}(X \Rightarrow Y) \geq \gamma$ where $Y \subset Z$ and $X = Z \setminus Y$.

Once all frequent itemsets are found, the solution of the second subproblem is straight forward. If we restrict the size of the consequent to one item, as it is done in the original problem description [3], each k-itemset in $\mathcal{F}$ produces k rules which have to be checked for minimum confidence. Conveniently, confidence can be directly calculated from the support values already produced for the first subproblem. However, since often a large number of frequent itemsets is discovered, generating and checking all possible rules can be still computationally challenging.

For all but very small or sparse databases, the first subproblem is computationally very expensive. The search space for frequent itemsets is the powerset $\mathcal{P}(I)$ which grows exponential with the number of items. An exhaustive search is infeasible for all applications which involve more than a very small number of items. However, the minimum support constraint possesses a special

property called *downward closure* [4] (some authors call this property also *anti-monotonicity* [23]) which can be used to make more efficient search possible.

Definition 3 (Downward closure). *A constraint is downward closed (anti-monotone) if, and only if, for each itemset which satisfies the constraint all subset also satisfy the constraint. That is,* const *is a downward closed constraint if*

$$\forall Y \subseteq X : \text{const}(X) \Rightarrow \text{const}(Y)$$

where X and Y are itemsets.

The minimum support constraint is downward closed since

$$\forall Y \subseteq X : \text{supp}(X) \geq \sigma \Rightarrow \text{supp}(Y) \geq \sigma,$$

i.e., if set X is supported at a threshold, also all its subsets Y must be supported at the same threshold. This property implies that (a) a itemset can only satisfy a downward closed constraint if all its subsets satisfy the constraint and that (b) if an itemset is found to satisfy a downward closed constraint all its subsets need no inspection since they must also satisfy the constraint. These facts are used by mining algorithms to reduce the search space which is often referred to as *pruning* or finding a border in the lattice representation of the powerset of all items.

Often a great number of frequent itemsets exist in a database. The main reason is that the downward closure property of support implies that for each frequent k-itemset there exist $2^k - 2$ frequent subsets. However, due to downward closure of support, all frequent subsets can be inferred from the k-itemset. This idea is used by mining only *maximal frequent itemsets*. A frequent itemset is maximal if it is no proper subset of any other frequent itemset [30]. Since the number of maximal frequent itemsets is normally by several orders smaller then the number all frequent itemsets, mining performance and storage requirement can improve significantly.

Another approach is to only mine *frequent closed itemsets.* An itemset is closed if no proper superset of the itemset is contained in each transaction in which the itemset is contained [22]. Frequent closed itemsets are a superset of the maximal frequent itemsets and a subset of all frequent itemsets. Their advantage over maximal frequent itemsets is that in addition to be able to infer all frequent itemsets, they also preserve the support information for all frequent itemsets which can be important for applying other measures.

2.2 Associations and rules without fixed support

Driven by the shortcomings of the support-confidence framework, especially of support's problems with rare items and skewed item frequency distributions, some researchers proposed alternative approaches for mining interesting patterns and rules.

Some researchers try to fix the short comings of support. For example, Liu et al. [19] deal with the rare item problem inherent in using a single fixed minimum support threshold by suggesting to mine association rules with multiple minimum support thresholds. An individual *minimum item support* threshold is assigned to each item and a rule's minimum support is defined to be the smallest minimum item support of all items in the rule. Liu et al. showed that, after sorting the items according to their minimum item support, a *sorted closure property* of minimum item support can be used to prune the search space. An open research question is how to determine the optimal values for the minimum item supports, especially in databases with many items where a manual assignment is not feasible.

Seno and Karypis [26] try to reduce support's tendency to favors smaller itemsets which naturally have higher counts than larger itemsets by proposing a support constraint that decreases as a function of itemset length. Especially, when mining maximal itemsets discovering a larger itemset instead of many of its subsets can improve performance. Seno and Karypis developed a property called *smallest valid extension* which is defined as the minimum size a superset of an infrequent itemset must have so that it would become frequent (according to the decreasing minimum support function) if all transactions which contain the infrequent itemset also contain the superset. In their paper Seno and Karypis exploit this property for pruning in their FP-tree-based algorithm LPMiner. The authors present the feasibility of their approach with experiments on artificial data sets and a support function which linearly decreases with the itemset length till it reaches an absolute minimum. An open question is how to choose an appropriate support function and its parameters.

Omiecinski [21] introduced several alternatives which avoid the need for support. The first two measures, *any-* and *all-confidence*, only use the confidence measure of the support-confidence framework. Any-confidence is defined as the confidence of the rule with the largest confidence which can be generated from an itemset. The author states that although finding all itemsets with a set any-confidence would enable us to find all rules with a given minimum confidence, any-confidence cannot be used efficiently as a measure of interestingness since

minimum confidence is not downward closed. The all-confidence measure is defined as the smallest confidence of all rules which can be produced from an itemset, i.e., all rules produced form an itemset will have a confidence greater or equal to its all-confidence value. Omiecinski shows that a minimum constraint on all-confidence is downward closed and, therefore, can be used for efficient mining algorithms without support. Finally, he introduces *bond* another downward closed measure which is defined as the ratio of the number of transactions which contain all items of an itemset to the number of transactions which contain at least one of these items.

Xiong et al. [29] argue that support-based pruning is not effective for data sets with skewed support distributions since many spurious patterns are generated while missing patterns with rare items. As an alternative the authors propose mining hyperclique patterns with a measure called h-confidence which is mathematically equivalent to Omiecinski's downward closed all-confidence measure [21]. In addition Xiong et al. show that h-confidence (and some similarity measures) posses a so-called *cross-support property*. Measures with this property avoid generating many patterns which contain items with substantially different support levels. The authors argue that these patterns tend to have a low pairwise correlation and therefore are often uninteresting.

Another family of approaches is based on using statistical methods to mine associations. The main idea is to identify associations as deviations from a baseline given by the assumption that items occur statistically independent from each other. The simplest measure to quantify the deviation is *interest* [7] (often also called *lift*). Interest for a rule $X \Rightarrow Y$ is defined as

$$\text{interest}(X \Rightarrow Y) = \frac{P(X \cup Y)}{P(X)P(Y)} \tag{3}$$

where the denominator is the baseline probability, the expected probability of the itemset under independence. Interest is usually calculated by the ratio n_{obs}/n_{exp} which are the observed and the expected occurrence counts of the itemset. The ratio is close to one if the itemsets X and Y occur in the database the same time as it would be expect under the assumption that they are independent. A value greater than one indicates a positive correlation between the items and values lesser than one indicate a negative correlation. Drawbacks of this simple ratio are:

1. The ratio does not possess the downward closure property needed for efficient mining. Interest can increase, decrease or stay constant by adding an

additional item to an itemset.

2. The ratio gives only values close to one, the value for uncorrelated items, for items which are very frequent even if they have a highly positive correlation. For example, two perfectly positive correlated items with probability 0.9 give a interest of only $0.9/(0.9 \cdot 0.9) = 1.11$.

3. When the count used to estimate the occurrence probabilities of items get small (e.g., because of a small sample size or because of a skewed item support distribution) the interest ratio tends to get very noisy producing very large values for some rare items which occur a few times together in the data set by accident.

Aggarwal and Yu [2] developed the measure *collective strength* which avoids the first two problems. Collective strength uses a *violation rate*, the fraction of transactions which contain some of the items in an itemset but not all. Collective strength for itemset Z is defined as

$$C(Z) = \frac{(1 - v(Z))E[v(Z)]}{v(Z)(1 - E[v(Z)])} \tag{4}$$

where $v(Z)$ is the violation rate and $E[\cdot]$ is the expected value for independent items. Collective strength gives 0 for items with a perfectly negative correlation, ∞ for items with a perfectly positive correlation, and 1 if the items co-occur as expected under independence. DuMouchel and Pregibon [11] argue that for items with medium to low probabilities the observations of the expected values of the violation rate is dominated by the proportion of transactions which do not contain any of the items in Z. For such itemsets collective strength produces values close to one, even if the itemset appears together several times more often than expected. This is especially problematic for transaction data with skewed support distributions where most items have a relatively low occurrence probability.

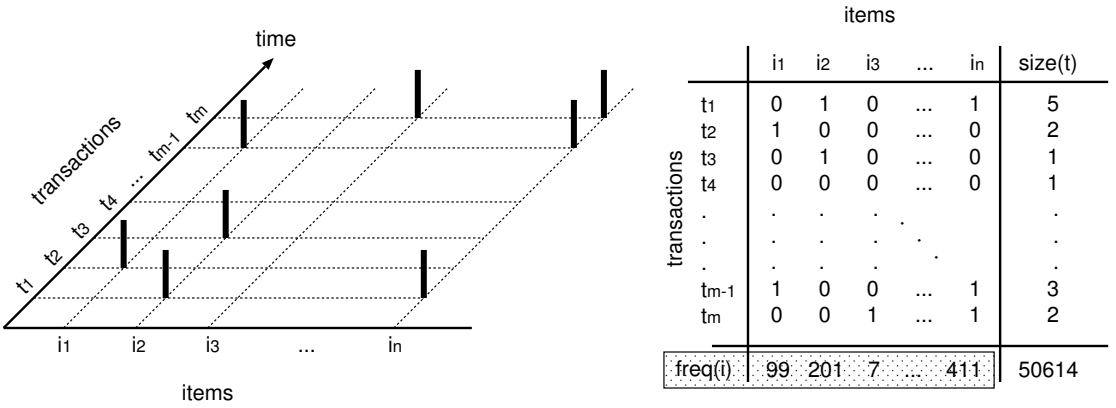
Silverstein et al. [27] suggested mining dependence rules using the χ^2 test for independence between items on 2×2 contingency tables. The authors use the fact that the test statistic can only increase with the number of items to develop mining algorithms which rely on this *upward closure property*. DuMouchel and Pregibon [11] pointed out that more important than the test statistic is the p-value which can due to the increasing number of degrees of freedom of the χ^2 test increase or decrease with itemset size. This invalidates the upward closure property and makes mining associations based on only this statistic inefficient. Furthermore, Silverstein et al. [27] mention that a significant problem of the ap-

proach is the normal approximation used in the χ^2 test which can skew results unpredictably for contingency tables with cells with low expectation.

DuMouchel and Pregibon [11] presented an approach which uses empirical Bayes estimation to smooth the interest ratios (the observed count to baseline ratios) to produce more reliable estimates for items with low counts. It is assumed that each observed count n_{obs} is drawn from a Poisson distribution with its own unknown mean μ. The true interest ratio is then $\lambda = \mu / n_{exp}$ instead of the simple observed interest ratio of n_{obs}/n_{exp}. For all itemsets of the same size the posterior distributions of their values of λ are estimated together using the additional prior information that all values of λ are assumed to be distributed according to a continuous density function. For this function DuMouchel and Pregibon suggest a mixture of two Gamma distributions. Its 5 parameters can be estimated by the maximum likelihood method from the observed data. The authors suggest to sort itemsets according to the value of the geometric mean of their posterior λ distributions, or, as a more conservative choice, by the value of the lower 95% Bayes confidence limit of the posterior λ distributions. The conservative estimate has the property that for large observed counts the estimated ratio is very close to the observed interest while for small observed counts, which could be the result of noise, the estimated ratio tends to reduce (shrink) the observed effect. In [11] DuMouchel and Pregibon concentrate on the statistical and not on the computational issues of finding associations. Although, they include a minimum support constraint, it is stated that this minimum support is chosen rather small. In the example in [11] which analyzes international calling behavior between $n = 228$ countries, a minimum support of 4 observations in 20 million transactions or $\sigma = 5.7 \cdot 10^{-7}$ is used and only results for itemsets of size $k = 3$ are reported. Since at such low minimum support values combinatorial explosion for larger itemset sizes is inevitable, the computational aspects of the approach needs further research.

3 Developing a model-based frequency constraint

In this section we build on the idea of discovering associated items with the help of observed deviations of co-occurrences from a baseline. In contrast to previous research done in this area, we will not estimate the degree of deviation at the level of an individual item or itemset, but we will evaluate the deviation for the set all possible 1-extensions of an itemset together to find a local frequency constraint for these extensions. A 1-extension of an k itemset is an itemset of size $k + 1$ which



items

transactions	i1	i2	i3	...	in	size(t)
t1	0	1	0	...	1	5
t2	1	0	0	...	0	2
t3	0	1	0	...	0	1
t4	0	0	0	...	0	1
.	.	.	.	.		.
.	.	.	.	.		.
.	.	.	.	.		.
tm-1	1	0	0	...	1	3
tm	0	0	1	...	1	2
freq(i)	99	201	7	..	411	50614

Figure 1: Representation of an example database as a sequence of transactions (to the left) and the incidence matrix with the resulting frequencies of the single items (to the right).

is produced by adding an additional item to the k-itemset. In a database with n unique items, for each k-itemset there exist exactly $n - k$ different 1-extensions.

3.1 A simple stochastic baseline model for single items

A suitable stochastic item occurrence model for the baseline frequencies needs to describe the occurrence of items with different usage frequencies in a robust and mathematically tractable way. For the model we consider the occurrence of items $I = \{i_1, i_2, \ldots, i_n\}$ in a database with a fixed number of transactions. An example database is depicted in figure 1. For the example we use $t = 20,000$ transactions and $n = 500$ items. To the left we see a graphical representation of the database as a sequence of transactions over time. The transactions contain items depicted by the bars at the intersections of transactions and items. The typical representation used for data mining is the incidence matrix with the items as columns and the transactions as rows in figure 1 to the right. The row sums represent the size of the transactions and the column sums are the frequencies of the items in the database. The total sum represents the number of incidences (item occurrences) in the database. Dividing the number of incidences by the number of transactions gives the average transaction size of $50,614/20,000 = 2.531$ and dividing the number of incidences by the number of items gives the average item frequency of $50,614/500 = 101.228$.

In the following we will model the baseline for the distribution of the items' frequency counts (marked by the shaded area in figure 1). For the baseline we suppose that each item occurs in the database following an independent Poisson process with an individual latent rate λ. Therefore, the frequency for each item in

the database is a value drawn from the Poisson distribution with its latent rate.

We assume that the individual rates are randomly drawn from a suitable distribution defined by the continuous random variable Λ. Then the probability distribution of R, a random variable which gives the number of times an arbitrarily chosen item occurs in the database, is

$$Pr[R = r] = \int_0^\infty \frac{e^{-\lambda}\lambda^r}{r!} dG_\Lambda(\lambda),\ r = 0, 1, 2, ...,\ \lambda > 0 \tag{5}$$

which is a Poisson mixture model. It results from the continuous mixture of Poisson distributions with rates following the mixing distribution G_Λ.

Heterogeneity in the occurrence frequencies between items is accounted for by the form of the mixing distribution of the latent rates. A commonly used and very flexible mixing distribution is the Gamma distribution with the density function

$$g_\Lambda(\lambda) = \frac{e^{-\lambda/a}\lambda^{k-1}}{a^k\Gamma(k)},\ a > 0,\ k > 0 \tag{6}$$

where a and k are the scaling and the shape parameters.

Integrating equation 5 with equation 6 as the mixing distribution is known to result in the negative binomial (NB) distribution (see, e.g. Johnson et al. [15]) with the probability distribution

$$Pr[R = r] = (1 + a)^{-k} \frac{\Gamma(k + r)}{\Gamma(r + 1)\Gamma(k)} \left(\frac{a}{1 + a}\right)^r,\ r = 0, 1, 2, ... \tag{7}$$

This distribution gives the probability that we see arbitrarily chosen items with a frequency of $r = 0, 1, 2, ...$ in the database. The mean of the distribution, the average frequency of the items in the database, is given by $m = a/k$. $Pr[R = 0]$ represents the proportion of available items which never occurred during the time the database was recorded.

Once the parameters k and a are known, the probabilities of finding items with a frequency of r in the database can be efficiently computed by calculating the probability of the zero class by $Pr[R = 0] = (1 + a)^{-k}$ and then using the recursive relationship

$$Pr[R = r + 1] = \frac{k + r}{r + 1} \frac{a}{1 + a} Pr[R = r] \tag{8}$$

(see [15, p. 213]).

It has to be noted that the negative binomial distribution has some interesting properties besides the fact that its variance always exceeds its mean: The geometric distribution, which represents the discrete form of the memoryless exponen-

tial distribution, is a special case of the NB distribution with $k = 1$. For $k \to 0$ the zero-truncated NB distribution converges into the logarithmic series distribution and for $k \to \infty$ the NB distribution converges into the Poisson distribution. This versatility and the fact that the NB distribution has a long and heavy tail makes it very useful for describing all kinds of count data.

Although, the NB model (often also called Gamma-Poisson model) simplifies reality considerably with its assumed Poisson processes and the Gamma mixing distribution, it is widely applied for accident statistics, birth-and-death processes, economics, library circulation, market research, medicine, and military applications [15, pp. 223–224]. Especially interesting for this paper is the extensive application of the NB model for repeat buying behavior as a mixture of product purchases by consumers with heterogeneous purchase rates [12]. This application demonstrates how powerful the NB model is to analyze panel data which have a very similar structure as transaction data recorded by point-of-sale scanners, Web severs, and data recorded for various other applications. Burrell [8] used the NB model to predict the circulation of books in a library and argues that despite its shortcomings in terms of perfectly fitting empirical data, the simple model provides predictions of future circulation with an accuracy which is adequate for general management requirements. Interesting here is the fact that the model is not applied to purchases as in the repeat buying application but to borrowing books which can be seen as the acquisition of information very similar to requesting Web pages from a Web server. More recently, Lee et al. [18] used the NB model to describe the visit frequency of the Yahoo! portal Web site as a mixture of visits by individuals.

3.2 Fitting the model to transaction data sets

The parameters of the NB distribution can be estimated by several methods including the method of moments, maximum likelihood and others [15, pp. 214–220]. Particularly simple is the method of moments where $\tilde{k} = \bar{x}^2/(s^2 - \bar{x})$ and $\tilde{a} = \bar{x}/\tilde{k}$ can be directly computed from the observed mean $\bar{x}$ and variance s^2. However, with empirical data we face two problems: (a) as reported for other applications of the NB model, there exist items with a too high frequency to be covered by the Gamma mixing distribution used in the model, and (b) the zero-class (available items which never occurred in the database) are not observable.

Expected outliers with a too high frequency will distort the mean and the variance of the observed data. For robust estimates we can trim a percentage of the

	WebView-1	POS	Artif-1
Transactions	59,602	515,597	100,000
Avg. trans. size	2.5	6.5	10.1
Median trans. size	1	4	10
Distinct items	497	1,657	844

Table 1: Characteristics of the data sets.

items with the highest frequencies.

A way to obtain the missing zero-class is to subtract the number of observed items from the total number of items which were available at the time the data set was recorded. The number of available items can be obtained from the provider of the data set. After trimming outliers and calculating the zero-class the method of moments can be used to estimate the parameters.

If the total number of available items is unknown, the size of the zero-class has to be estimated together with the parameters of the NB distribution. The standard procedure for this type of estimation problem is the *Expectation Maximization* algorithm [10], which iteratively estimates missing data values using the observed data and intermediate values of the parameters, and then uses the estimated data and the observed data to update the parameters for the next iteration. The procedure stops when the parameters stabilize.

To demonstrate that the model can be used to describe use the two e-commerce data sets *WebView-1* and *POS* provided by Blue Martini Software for the KDD Cup 2000 [16] and an artificial data set, *Artif-1*. WebView-1 contains several months of clickstream data for an e-commerce Web site where each transaction consists of the product detail page views during a session. POS is a point-of-sale data set containing several years of data. Artif-1 is better known as *T10I4D100K*, a widely used artificial data set generated by the procedure described in Agrawal and Srikant [4].

Table 1 contains the basic characteristics of the data sets. The data sets differ in the number of items and the average number of items per transactions. The real-world data sets show that their median transaction size is considerably smaller than their mean which indicates that the distribution of transaction lengths is skewed with many very short transactions and some very long transactions. The artificial data set does not show this property with a mean almost equal to its median. For a comparison of the data sets' properties and their impact on the effectiveness of different association rule mining algorithms see Zheng et al. [31]. The number of distinct items of the artificial data set differs slightly from the set used by Zheng et al. [31]. Although the data sets were generated using the same

	WebView-1	POS	Artif-1
Observed items	342	1,153	843
Trimmed items	9	29	0
Estim. zero-class	6	2,430	4
Used items ($\tilde{n}$)	339	3,554	847
Item occurrences	33,802	87,864	202,325
$\bar{x}$	99.711	24.723	238.873
s^2	11,879.543	9,630.206	59,213.381
$\tilde{k}$	0.844	0.064	0.968
$\tilde{a}$	118.141	386.297	242.265
χ^2 p-value	0.540	0.101	0.914

Table 2: The fitted NB models using samples of 20,000 transactions.

implementation of the procedure in [4] by the Quest team at IBM Almaden[1], there are minimal variations due to differences in the random number generators and the used seeds.

Before we estimated the model parameters with the Expectation Maximization algorithm, we discarded the first 10,000 transactions for WebView-1 since a preliminary data screening showed that the average transaction size and the number of items used in these transactions is more volatile and significantly smaller than for the rest of the data set. This might indicate that at the beginning of the database there were still major changes made to the Web shop (e.g., reorganizing the Web site, or adding and removing promotional items). POS does not show such an effect. For robust estimates trimming 2.5% of the items with the highest frequency proved appropriate for the two real-world data sets. The synthetic data set does not contain outliers and therefore no trimming was necessary.

In table 2 we summarize the results of the fitting procedure for samples of size 20,000 transactions from the three data sets. To check whether the model provides a useful approximation for the data we used the χ^2-goodness-of-fit test. As recommended for the test we combined classes so that in no class the expected count is below 5 and used a statistical package to calculated the p-values. For all data sets we found high p-values ($\gg 0.05$) which indicates that no significant difference between the data and the corresponding models were found.

To evaluate the stability of the models we also estimated the parameters for samples of different sizes. In the stationary case, where the model applies, we expect that the parameter k is independent of sample size while the parameter a depends linearly on the sample size. This can be simply explained by the fact that if we observe the Poisson process for each item twice as long, we will end up

[1]The generator is available at http://www.almaden.ibm.com/software/quest/Resources/

	sample size	$\tilde{k}$	$\tilde{a}$	$\tilde{a}/size$	$\tilde{n}$
WebView-1	5,000	0.882	33.503	0.0067	322
WebView-1	10,000	0.933	58.274	0.0058	325
WebView-1	20,000	0.844	118.140	0.0059	339
WebView-1	40,000	0.868	218.635	0.0055	395
POS	5,000	0.038	87.605	0.0175	4,887
POS	10,000	0.060	178.200	0.0178	3,666
POS	20,000	0.064	386.300	0.0193	3,554
POS	40,000	0.064	651.406	0.0163	3,552
Artif-1	5,000	0.956	62.134	0.0124	849
Artif-1	10,000	0.975	123.313	0.0123	845
Artif-1	20,000	0.968	242.265	0.0121	847
Artif-1	40,000	0.967	493.692	0.0123	846

Table 3: Estimates for the NB-model using samples of different sizes.

with all latent parameters λ being double. For the Gamma mixing distribution this means that the scaling parameter a must be double as high. Therefore, a divided by the size of the sample should be constant.

Table 3 gives the parameter estimates and the estimated total number of items n (observed items + estimated zero class) for the used databases with sample sizes $5,000$ to $40,000$ transactions. The estimates for the parameters k and $a/size$, and the number of items n are generally constant over different sizes of the same data set. Minor variations and outliers for the real-world data sets are caused by instabilities in the data set. The stability of the estimates shown in table 3 enables us to use model parameters estimated from one sample for another sample of the same database even if the samples differ in size. A possible application is that, even if frequently new transactions are added to the database, the model can be applied to these transactions without the need to re-estimate the model every time.

Applied to associations, equation 7 in the section above gives the probability distribution of observing single items (1-itemsets) with a frequency of r. Therefore, the expected number of frequent 1-itemsets with a minimum support of σ^{freq} is given by

$$nPr[R \geq \sigma^{freq}],$$

where n is the number of available items. In figure 2 we show for the data sets WebView-1, POS and Artif-1 the frequent 1-itemsets predicted by the fitted models (solid line) and the actual number (dashed line) by a varying minimum support constraint. For easier comparison we use relative support for the plots. In all three plots we can see how the models fit the skewed support distributions.

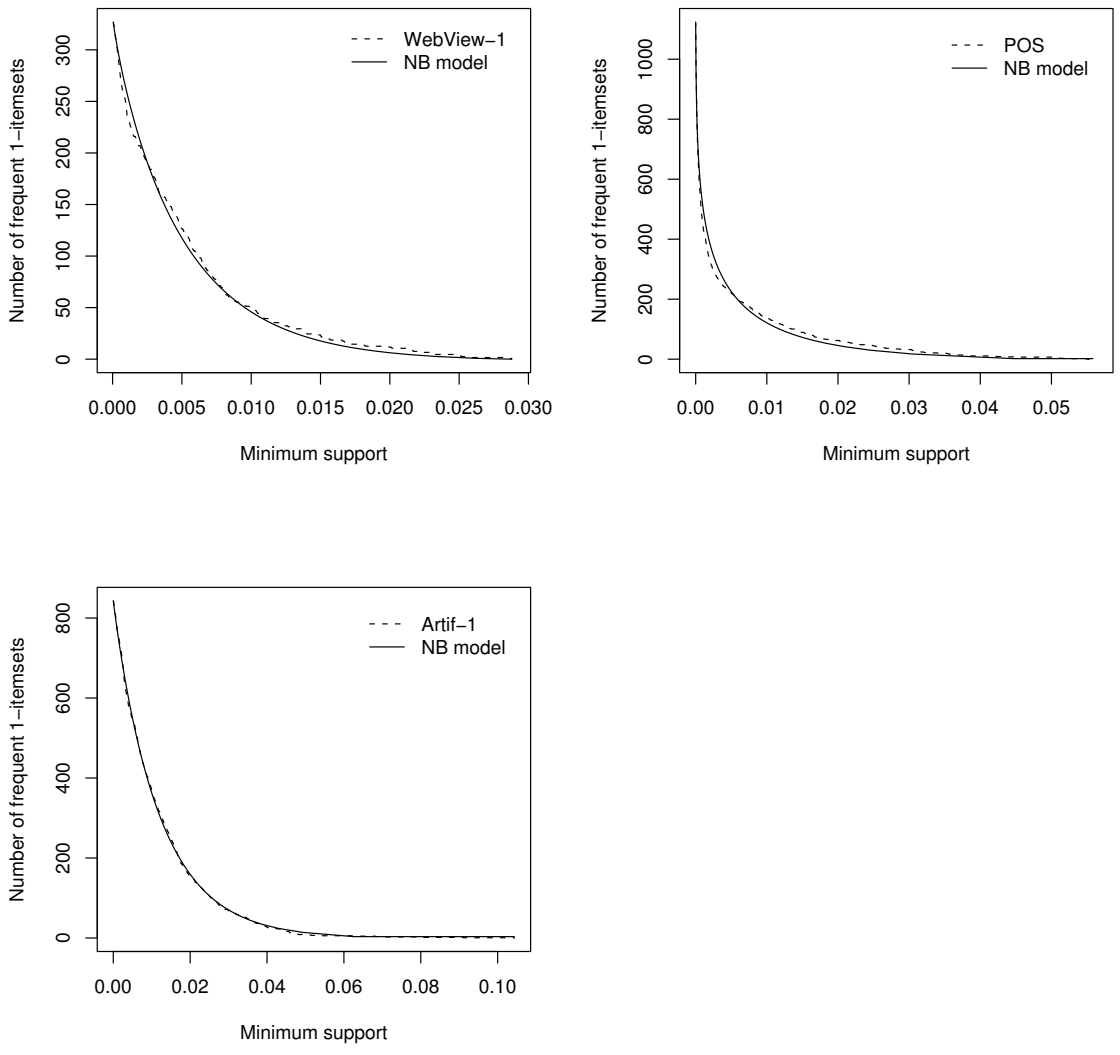


Figure 2: Actual versus predicted number of frequent items by minimum support.

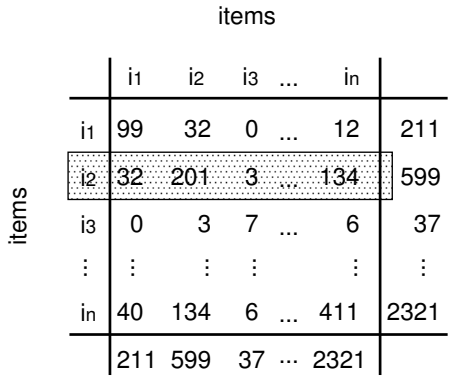


Figure 3: A $n \times n$ matrix for counting 2-itemsets in the database.

3.3 Extending the baseline model to k-itemset

After only considering 1-itemsets, we show how the model developed above can be extended to provide a baseline for the distribution of support over all possible 1-extensions of an itemset. We start with 2-itemsets before we generalize to itemsets of arbitrary length. Figure 3 shows an example of the co-occurrence frequencies of all items (occurrence of 2-itemsets) in transactions organized as an $n \times n$ matrix. The matrix is symmetrical around the main diagonal which contains the count frequencies of the individual items $\text{freq}(i_1), \text{freq}(i_2), \ldots, \text{freq}(i_n)$. By adding the count values for each row or for each column we get the number of incidences in all transactions which contain the respective item.

To build the model for all 1-extensions of item i_2, we only need the information in the shaded area in figure 3 which contains the frequency counts for all 1-extensions of i_2 plus $\text{freq}(i_2)$ in cell $(2, 2)$. Clearly, these counts are only affected by transactions which contain item i_2. For the baseline, all items are assumed to occur independently in the database. If we select all transactions which contain item i_2, we get a sample of size $\text{freq}(i_2) = 201$ from the database which is random in respect to all items but i_2. Thus, under the independence assumption, the co-occurrence counts in the sample follow again Poisson processes. Following the model in section 3.1 we can obtain a new random variable R_{i_2} which models the occurrences of an arbitrarily chosen 1-extensions of i_2. Since the rates of the processes and therefore also the scale parameter of the model depend on the number of transactions, the rates need to be rescaled for the smaller size of the sample which will be explained in the following generalization for itemsets of arbitrary length.

Instead of the 1-extensions of a single item, we analyze now the count frequencies for all possible 1-extensions of an already known pattern l which is an

itemset of arbitrary length. All 1-extensions of l can be generated by joining the pattern with all possible single items $c \in I \setminus l$. We call the items c candidate items since each item generates together with l a 1-extension which is a candidate for a new pattern. For counting the occurrence of all 1-extensions of l we only need to consider the transactions which contain the pattern l. Since all candidate items, for which the assumptions of the baseline hold, must be independent from the items in l, the considered transactions represent a sample of size $\text{freq}(l)$ which is random in respect to the candidate items. Therefore, we would expect that the number of candidate items with frequency r in the sample follows the model and has a NB distribution. More precisely, as the baseline we expect the counts for the 1-extensions to be modeled by a random variable R_l with the probability distribution

$$Pr[R_l = r] = (1 + a_l)^{-k} \frac{\Gamma(k + r)}{\Gamma(r + 1)\Gamma(k)} \left(\frac{a_l}{1 + a_l} \right)^r \; for \; r = 0, 1, 2, ... \tag{9}$$

The distribution's shape parameter k is not affected by sample size and we can use the estimate $\tilde{k}$ from the database. The parameter a of the NB distribution is linearly dependent on the sample size since it represents the scale parameter of the mixing Gamma distribution which controls the latent item usage rates. We have to rescale $\tilde{a}$ for the sample size to obtain an estimate for a_l.

To rescale a we could use the proportion of the transactions in the sample relative to the size of the database which was used to estimate $\tilde{a}$. In section 3.2 above we showed that for estimating the parameter for different sample sizes gives a stable value for $\tilde{a}$ per transaction. A problem with applying transaction-based rescaling is that the more items we include in l, the smaller the number of remaining items in the transactions gets. This would reduce the effective transaction length and the estimated model would not be applicable any more. Therefore, we will ignore the concept of transactions for the following and treat the data set as a series of incidences. For the model this is unproblematic since the mixture model never used the information that items occur together in transactions. Now we can rescale a by the proportion of incidences in the sample relative to the total number of incidences in the database from which we estimated the parameter. We rescale the parameter for the baseline that models the frequency of items $c \in I \setminus l$ in the transactions which contain l in two steps:

1. We calculate $\tilde{a}'$, the parameter per incidence, by dividing the parameter ob-

tained from the database by the total number of incidences in the database.

$$\tilde{a}' = \frac{\tilde{a}}{\sum_{\{t|t\in\mathcal{D}\}} |t|} \tag{10}$$

2. We rescale the parameter for pattern l by multiplying $\tilde{a}'$ with the number of incidences in the sample (transactions which contain l) excluding the occurrences of the items in l.

$$\tilde{a}_l = \tilde{a}' \sum_{\{t|t\in\mathcal{D}\wedge t\supset l\}} |t \setminus l| \tag{11}$$

   The items in l have to be excluded since we only consider the candidate items.

For item i_2 in the example in figure 3 the rescaled parameter can be easily calculated from the sum of incidences for the item (599) in the $n \times n$ matrix together with the the sum of incidences $(50,614)$ in the total incidence matrix in figure 1 (in section 3.1) by $\tilde{a}' = \tilde{a}/50614$ and $\tilde{a}_{i_2} = \tilde{a}' \cdot 599$.

3.4 Deriving a model-based frequency constraint for itemsets

The NB distribution with the parameters rescaled for pattern l provides a baseline for the frequency distribution of the candidate items in the transactions that contain l, i.e., the number of different itemsets $l \cup \{c\}$ with $c \in I \setminus l$ we would expect per support count, even if all items were independent. If some item candidates are related to the items in l, the transactions that contain l cannot be considered a random sample for these items. These items will have a higher frequency in the sample than expected by the model.

To find a set L of non-random patterns (with the item candidates with a too high co-occurrence frequency) we need to identify a frequency threshold σ_l^{freq} where accepting item candidates with a frequency count $r \geq \sigma_l^{freq}$ separates associated items best from items which co-occur often by pure chance. For this task we have to define a quality measure on the set of patterns L which we accept. *Precision* is a possible quality measure which is widely used in information retrieval and by the machine learning community [17].

Definition 4 (Precision). *Let L be a set of 1-extensions of l which are generated by joining the pattern l with all candidate items $c \in I \setminus l$ and which co-occurrence with l in*

at least ρ transactions. For the set L which we define precision as

$$\text{precision}_l(\rho) = \frac{\sum_{r=\rho}^{r_{max}} n_{obs}(r) - (n - |l|)Pr[R_l \geq \rho]}{\sum_{r=\rho}^{r_{max}} n_{obs}(r)} = 1 - \frac{(n - |l|)Pr[R_l \geq \rho]}{\sum_{r=\rho}^{r_{max}} n_{obs}(r)}, \tag{12}$$

where $n_{obs}(r)$ is the observed number of candidate items which have a co-occurrence frequency of r with pattern l, and r_{max} is the highest observed co-occurrence.

Precision measures the proportion of predicted positive cases that are correct. In equation 12 the number of predicted positive cases is the number of patterns $n_{obs}(r)$ with a co-occurrence frequency $r \geq \rho$. As the estimate for the number of correct positive cases we subtract the number of random itemsets predicted by the NB model.

The predicted error rate for the set of accepted patterns L is given by $1 -$ precision. A suitable selection criterion for a count threshold is to allow only a percentage of falsely accepted patterns. For example, if for an application we need all rules with the antecedent l and a single item as the consequent (which can be accomplished by finding all patterns which include l plus an additional item) and the maximum number of acceptable spurious rules is 5% we can use the minimum precision threshold $\pi = 0.95$.

The smallest possible frequency threshold which satisfies a set minimum precision threshold π can be found by

$$\sigma_l^{freq} = \text{argmin}_\rho\{\text{precision}_l(\rho) \geq \pi\}.$$

The set of the chosen candidate items for l is then

$$C_l = \{c | c \in I \setminus l \wedge \text{freq}(l \cup \{c\}) \geq \sigma_l^{freq}\}$$

and the set of accepted patterns is

$$L = \{l' | l' = l \cup \{c\} \wedge c \in C_l\}.$$

Table 4 contains an example for the model-based frequency constraint using data from the WebView-1 database. We analyze the 1-extensions of pattern $l = \{10311, 12571, 12575\}$ at a minimum precision threshold of 95%. The estimates for n, k and a are taken from table 2 in section 3.2. And parameter a is rescaled to $a_l = 1.164$ using formulas 10 and 11. The column n_{obs} contains the number of items with a co-occurrence frequency of r with pattern l. The value at $r = 0$ is in parentheses since it is not directly observable but was calculated as the difference between the estimated number of total available candidate

r	n_{obs}	n_{model}	precision
0	(183)	176.71178	-
1	81	80.21957	-
2	48	39.78173	-
3	13	20.28450	-
4	6	10.48480	-
5	0	5.46345	-
6	1	2.86219	-
7	0	1.50516	-
8	1	0.79378	-
9	0	0.41955	-
10	0	0.22214	0.92108
11	2	0.11779	0.95811
12	1	0.06253	0.96661
13	1	0.03323	0.97632
14	1	0.01767	0.98109
15	0	0.00941	0.97986
16	0	0.00501	0.98927
17	0	0.00267	0.99428
18	1	0.00305	0.99695

Table 4: Finding the optimal support threshold at $\pi = 0.95$.

items for l (calculated by $n - |l|$) and the observed items (sum of observations with $r > 0$). The column n_{model} contains the predicted frequencies calculated by $(n - |l|)Pr[R_l \geq r]$. To find the threshold σ_l^{freq} the precision function in formula 12 is evaluated starting with $\rho = r_{max}$ and ρ is reduced till we get a precision value which is below the minimum precision threshold of 95%. Then the found threshold is the last value for ρ which produced a precision above the threshold (in the example at $\sigma_l^{freq} = 11$). After the threshold is found, there is no need to evaluate the rest of the precision function with $r < 10$. All item candidates with a co-occurrence frequency greater than the found threshold are selected. In the example in table 4 this gives 6 items with $r \geq 11$. The set of chosen candidates for the example is $C_l = \{32213, 12703, 12487, 12875, 12483, 34893\}$.

For the way support thresholds are chosen, here there exists an interesting connection to confidence.

Theorem 1. *Choosing a minimum support threshold σ_l for all 1-extensions of itemset l (all itemsets $l \cup \{c\}$ where $c \in I \setminus l$) is equivalent to choosing a minimum confidence threshold for all rules which can be constructed with l as the antecedent and one of the items c as the consequents, i.e., the constraints*

$$\text{supp}(l \cup \{c\}) \geq \sigma_l \Leftrightarrow \text{conf}(l \Rightarrow \{c\}) \geq \gamma_l$$

are equivalent.

Proof. *Since* $\text{conf}(l \Rightarrow \{c\})$ *is defined as* $\text{supp}(l \cup \{c\})/\text{supp}(l)$ *we get* $\text{supp}(l \cup \{c\}) \geq \gamma_l \cdot \text{supp}(l)$. *Since* $\text{supp}(l)$ *is a positive constant for all rules with the same antecedent, we get the equality* $\gamma_l = \sigma_l/\text{supp}(l)$. □

As an example, suppose a database contains 20,000 transactions, the pattern l is contained in 1600 ($supp(l) = 1600/20,000 = 0.08$) and we choose to require that each candidate item c must have a co-occurrence frequency of at least 1200 with pattern l. This means that the itemsets $l \cup \{c\}$ need a minimum support of $\sigma_l = 1200/20,000 = 0.06$ and that all rules $l \Rightarrow \{c\}$ which can be constructed for the supported itemsets will have at least a confidence of $\gamma_l = 0.06/0.08 = 0.75$.

So, the presented approach of choosing an individual minimum support for all 1-extensions of an itemset can also be interpreted as choosing a single minimum confidence for the set of rules with the same antecedent and with a single item as the consequent.

The aim of the model-based frequency constraint is to find as many non-spurious patterns given a precision threshold as possible. After developing the model-based frequency constraint for one pattern we now extend the view to the whole itemset lattice and formally introduce the concept of *NB-frequent itemsets*.

Definition 5 (NB-frequency). *A k-itemset* l' *with* $k > 1$ *is a NB-frequent itemset if, and only if, at least a fraction* θ *(at least one) of its subsets* $\{l | l \in l' \setminus \{c\} \wedge c \in l'\}$ *are NB-frequent itemsets and satisfy* $\text{freq}(l \cup \{c\}) \geq \sigma_l^{freq}$. *The support thresholds are individually chosen for each pattern* l *by* $\sigma_l^{freq} = \text{argmin}_\rho\{\text{precision}_l(\rho) \geq \pi\}$ *where* π *is a user-specified minimum precision threshold and* $\text{precision}(\cdot)$ *is the function defined above in equation 12. All itemsets of size* 1 *are per definition NB-frequent.*

This definition clearly shows that NB-frequency in general is not downward closed since only a fraction θ of the $(k-1)$-subsets of a NB-frequent set of size k are required to be also NB-frequent. Only the special case with $\theta = 1$ offers downward closure, but since the definition of NB-frequency is recursive, we can only determine if an itemset is NB-frequent if we first evaluate all subsets. However, the definition provides two important properties of NB-frequent itemsets:

1. A k-itemset can only be NB-frequent if at least one $(k-1)$-subset of the itemset is NB-frequent.

2. A proportion θ of the $(k-1)$-subsets has to be NB-frequent in order for a k-itemset to be NB-frequent.

The first property enables us to build algorithms which find all NB-frequent itemsets in a bottom-up search (expanding from 1-itemsets). The second property enables us to prune the search space where the magnitude of pruning depends on the parameter θ.

Conceptually, mining NB-frequent itemsets with the extreme values 0 and 1 for θ is similar to Omiecinski's any-confidence and all-confidence [21] which are based on the confidence measure. Above, in theorem 1, we showed that the minimum supports σ_l chosen for NB-frequent itemsets $l \cup \{c\}$ are equivalent to choosing minimums on confidence $\gamma_l = \sigma_l / \text{supp}(l)$ for the rules $l \Rightarrow \{c\}$. An itemset passes a threshold on any-confidence if at least one rule can be constructed from the itemset which has a confidence value greater or equal of the threshold. This is similar to mining NB-frequent itemsets with $\theta = 0$ where one combination for which $\text{conf}(l \Rightarrow \{c\}) \geq \gamma_l$ suffices to generate a pattern. For all-confidence all rules which can be constructed from a pattern must have a confidence greater or equal than a threshold. This is similar to mining NB-frequent itemsets with $\theta = 1$ where we require for all combination that $\text{conf}(l \Rightarrow \{c\}) \geq \gamma_l$. Note, that in contrast to all- and any-confidence we do not use a single threshold for mining NB-frequent itemsets, but an individual threshold is chosen by the model for each pattern l.

4 Mining algorithms using the model-based frequency constraint

In this section we develop two algorithms to mine NB-frequent itemset. One algorithm uses a breadth-first and the other a depth-first search strategy.

Both algorithms use a function which implements the candidate item selection mechanism of the model-based frequency constraint. We present the pseudocode for this NB-Select function in figure 4. The function is called for one pattern l and gets count information, characteristics of the data set, and the user-specified precision threshold and returns a set of selected candidate items. In lines 1 and 2 the maximal observed count r_{max} and the rescale factor $r_{rescale}$ for the parameter a are calculated. In line 3 the NB-Select function creates a histogram of occurrence frequencies from the data structure $\mathcal{L}$. In lines 4 to 6 the smallest count threshold is found which still satisfies the precision constraint. Calculating the probability for the model can be implemented efficiently using the recursive relationship presented in section 3.1. Finally, in line 7 the set of selected candidate items is

function NB-Select$(l, \mathcal{L}, n, \tilde{k}, \tilde{a}', \pi)$:

l = the pattern for which candidate items are selected
$\mathcal{L}$ = a data structure which holds count information for items which co-occur with pattern l; we use tuples $\langle c, c.count \rangle$, where c represents a candidate item and $c.count$ the count.
n = the total number of available items
$\tilde{k}$ and $\tilde{a}'$ = estimated parameters for the data set
π = user-specified precision threshold

1. $r_{max} = max(c.count)$ in $\mathcal{L}$
2. $r_{rescale} = sum(c.count)$ in $\mathcal{L}$
3. **for** each tuple $\langle c, c.count \rangle \in \mathcal{L}$ **do** $n_{obs}[c.count]{+}{+}$
4. **do**
5. precision $= 1 - \frac{(n-|l|)P[R_l \geq \rho | k=\tilde{k}, a=r_{rescale} \cdot \tilde{a}']}{\sum_{r=\rho}^{r_{max}} n_{obs}[r]}$
6. **while** (precision $\geq \pi \wedge (\rho{-}{-}) > 0$)
7. **return** $\{c | \langle c, c.count \rangle \in \mathcal{L} \wedge c.count > \rho\}$

Figure 4: Pseudocode for choosing candidate items for pattern l using a minimum precision constraint.

returned. The data structure $\mathcal{L}$, which holds count information for potential candidate items, can be efficiently implemented as e.g., a vector of counters.

4.1 A breadth-first search algorithm

The first search algorithm, called NB-BFS, uses a breadth-first search strategy and follows the outline of the well known *Apriori* algorithm [4]. Figure 5 contains the pseudocode of the algorithm. The input for the algorithm are the database, a set of all items, the estimated characteristics of the data set, the user-specified precision threshold and the required fraction of NB-frequent subsets. The result is the set of all NB-frequent itemsets. Note, some available items will not be present in the database (occur zero times) and therefore I can contain more distinct items than occur in $\mathcal{D}$.

The algorithm starts with initializing L_1, the NB-frequent patterns of length one, with the set of all available items. Per definition all 1-itemsets are NB-frequent which makes conceptually sense since a single item cannot occur independently of itself. The loop in lines 2 to 18 performs a level-wise search for patterns starting with level $k = 2$ and stops when for a level no new patterns are found. In line 3 we create for each already found pattern of size $k - 1$ an empty list (in data structure $\mathcal{L}$) which will store the counts for all possible candidate items. Lines 4 to 12 are used to count the co-occurrences of the patterns with

potential candidates. Line 5 makes sure that for each transaction only patterns are considered that are present in the transaction which speeds up the counting process considerably (see implementation of the subset function of the Apriori algorithm [4]). After counting for a level is completed, for each pattern the NB-Select function is called to select the candidate items which satisfy the precision constraint (line 14). Since the algorithm does not use the count information for a pattern after processing NB-Select, the data structure $\mathcal{L}$ can be removed from memory (line 16). This data structures can alternatively be stored in external memory, if the count information is needed for later filtering, e.g., for applying other measures of interest that can be calculated from support values. In line 17 new patterns are generated by the function NB-Gen-BFS using the old patterns, the selected candidate items and the parameter θ. The NB-Gen-BFS will be discussed in its own section below. Finally, after no longer patterns are found the algorithm returns all found patterns.

4.2 A depth-first search algorithm

Figure 6 contains the pseudocode for NB-DFS, an algorithm to mine associations which applies a depth-first search strategy using recursion. The algorithm is similar to *DepthProject* an algorithm to efficiently find long maximal itemsets [1].

In contrast to NB-BFS, the algorithm gets a known pattern l and a conditional database $\mathcal{D}_l$ (a sub-database which only contains transactions which contain the pattern l) to mine patterns which are supersets of l. NB-DFS also uses a data structure $\mathcal{L}$ to store the count information for the candidate items obtained by scanning all transactions in the conditional database (lines 1 to 7 in table 6). In line 8 the candidate items for l are selected using the NB-Select function. Line 10 is used to generate new patterns with the NB-Gen-DFS function which will be discussed together with parameter θ in the section below. In lines 11 to 14 for each new pattern a conditional databases is built and NB-DFS is called recursively. Finally, the algorithm returns the set of all found patterns.

To mine all patterns starting with all 1-itemsets, NB-DFS is called with NB-DFS$(\emptyset, \mathcal{D}, n, \tilde{k}, \tilde{a}', \pi, \theta)$ which uses line 9 to make sure that all items present in the database are checked for building patterns.

4.3 Pattern generation and pruning

Pattern generation for the model-based mining algorithms has a similar function as candidate generation in support-based algorithms since it also controls what

algorithm NB-BFS$(\mathcal{D}, I, \tilde{k}, \tilde{a}', \pi, \theta)$:

$\mathcal{D}$ = the database
I = a set of all available items in the database
$\tilde{k}$ and $\tilde{a}'$ = estimated characteristics of the database
π = user-specified precision threshold
θ = user-specified required fraction of NB-frequent subsets
L_i = the NB-frequent itemsets of size i
$\mathcal{L}(l)$ = data structure for the count information for all patterns l
$\mathcal{C}(l)$ = data structure for selected candidate items for all patterns l

1. $L_1 = I$
2. **for** $(k = 2; L_{k-1} \neq \emptyset; k{+}{+}\,)$ **do begin**
3. **forall** $l \in L_{k-1}$ **do** create an empty set $\mathcal{L}(l) = \emptyset$
4. **forall** transactions $t \in \mathcal{D}$ **do begin**
5. $L_t = \{l | l \in L_{k-1} \wedge l \subseteq t\}$ // count only patterns contained in t
6. **forall** $l \in L_t$ **do begin**
7. **forall** $c \in t \setminus l$ **do begin**
8. **if** no tuple exists for item candidate c **then** add $\langle c, 1\rangle$ to set $\mathcal{L}(l)$
9. **else** $c.count{+}{+}$ for tuple $\langle c, c.count\rangle$ in set $\mathcal{L}(l)$
10. **end**
11. **end**
12. **end**
13. **forall** $l \in L_{k-1}$ **do begin**
14. $\mathcal{C}(l) =$ NB-Select$(l, \mathcal{L}(l), n, \tilde{k}, \tilde{a}', \pi)$ // select candidate items
15. **end**
16. delete $\mathcal{L}$
17. $L_k =$ NB-Gen-BFS$(L_{k-1}, \mathcal{C}, \theta)$ // generate new patterns
18. **end**
19. **return** $\bigcup_k L_k$

Figure 5: Pseudocode for a level-wise search algorithm using the model-based constraint.

algorithm NB-DFS$(l, \mathcal{D}_l, n, \tilde{k}, \tilde{a}', \pi, \theta)$:

l = a pattern
$\mathcal{D}_l$ = a conditional database only containing transactions which include l
n = the number of all available items in the database
$\tilde{k}$ and $\tilde{a}'$ = estimated characteristics of the database
π = user-specified precision threshold
θ = user-specified required fraction of NB-frequent subsets
L' = the set of all so far found NB-frequent itemsets
$\mathcal{L}$ = data structure for the count information
C = a set of selected candidate items

1. $\mathcal{L} = \emptyset$;
2. **forall** transactions $t \in \mathcal{D}_l$ **do begin**
3. **forall** $c \in t \setminus l$ **do begin**
4. **if** no tuple exists for c **then** add $\langle c, 1 \rangle$ to set $\mathcal{L}$
5. **else** $c.count{+}{+}$ for tuple $\langle c, c.count \rangle$ in set $\mathcal{L}$
6. **end**
7. **end**
8. **if** $l \neq \emptyset$ **then** $C =$ NB-Select$(l, \mathcal{L}, n, \tilde{k}, \tilde{a}', \pi)$ // select candidate items
9. **else** $C = \{c | \langle c, c.count \rangle \in \mathcal{L}\}$ // initial run
10. delete $\mathcal{L}$
11. $L =$ NB-Gen-DFS(l, C, θ) // generate new patterns
12. **forall** $l' \in L$ **do begin**
13. $\mathcal{D}_{l'} = \{t | t \in \mathcal{D}_l \wedge l' \subseteq t\}$ // create a conditional database
14. $L' = L' \cup$ NB-DFS$(l', \mathcal{D}_{l'}, n, \tilde{k}, \tilde{a}', \pi, \theta)$
15. **end**
16. **return** L'

Figure 6: Pseudocode for a recursive depth first search algorithm using the model-based constraint.

parts of the search space is pruned. A suitable candidate generation strategy is important for the performance of mining algorithms.

For Apriori-like support-based algorithms prior to pattern generation the occurrences of some candidate itemsets are counted and the support threshold is used to discard infrequent itemsets. The rest are frequent itemsets (patterns) which are used to generate new candidates for counting. In this process candidates are produced by joining the frequent itemsets and then using the downward closure property of support to prune candidates which cannot be frequent since not all subsets are frequent.

For finding NB-frequent itemsets the process is similar. First, candidates are counted, but instead of the simple support threshold, the NB-Select function is used for each pattern to decide which candidate items are accepted as non-random. Then the NB-Gen function (which will be developed later in this section) is used to combine existing NB-frequent patterns with their selected candidate items to form new NB-frequent patterns which are also used as the starting points for counting new candidate items.

As already discussed, NB-frequency does not possess the downward closure property which allows pruning in the same way as support. However, the definition of NB-frequency provides us with a way to prune the search space. From the definition we know, that in order for a pattern of length k to be NB-frequent at least a proportion θ of its $(k-1)$-subset have to be NB-frequent. Each k-itemset has k subsets of size $k-1$. Therefore, we can prune the search space by deleting all k-itemsets for which we found less than $\theta \cdot k$ NB-frequent $(k-1)$-subset.

In figures 7 and 8 we present the pseudocode for pattern generation functions for the two search algorithms. The pattern generation function for the breadth first search algorithm in figure 7 follows largely the candidate generation process of the Apriori algorithm. First, candidate itemsets are generated by joining the patterns L_{k-1} from the last level with their selected candidate items stored in $\mathcal{C}$. In addition we count for each candidate itemset by how many patterns of length $k-1$ it is generated. Then, in the prune step this counts are used to delete candidate itemsets which were not produced a sufficient number of times in the join step. The remaining candidate itemsets are NB-frequent patterns and are returned.

For the depth-first search algorithm the pattern generation function is called for each pattern individually. However, for the generation of a new NB-frequent pattern l' of size k we need the information of how many different NB-frequent patterns of size $k-1$ produce l' together with one of their selected candidate items. We also need to make sure that no part of the lattice is traversed more

function NB-Gen-BFS$(L_{k-1}, \mathcal{C}, \theta)$:

L_i = NB-frequent patterns of size i
$\mathcal{C}(l)$ = data structure holding candidate items chosen by NB-Select for pattern l
θ = user-specified parameter
l' = a candidate itemset; $l'.count$ holds the number of NB-frequent patterns l which produce l' by joining them with their selected candidate items in $\mathcal{C}(l)$

1. **forall** $l \in L_{k-1}$ **do begin**
2. $L = \{l \cup \{c\} | c \in \mathcal{C}(l)\}$ // join patterns with candidate items
3. **forall** $l' \in L$ **do** $l'.count{+}{+}$ // count number of combinations
4. $L_k = L_k \cup L$
5. **end**
6. **forall** $l' \in L_k$ **do begin**
7. **if** $l'.count < \theta \cdot |l'|$ **then** delete l' from L_k // prune step
8. **end**
9. **return** L_k

Figure 7: Pseudocode for the pattern generation function for NB-BFS.

function NB-Gen-DFS(l, C, θ):

l = a NB-frequent pattern
C = the set of candidate items chosen by NB-Select for pattern l
θ = user-specified parameter
$\mathcal{R}(l')$ = a global repository to keep track of traversed itemsets ($\mathcal{R}(l').done$) and the number of NB-frequent patterns l which already produced l' ($\mathcal{R}(l').count$); if for l' no entry exists in $\mathcal{R}$ a new entry with $\mathcal{R}(l').count = 0$ and $\mathcal{R}(l').done = false$ is created

1. $L = \{l \cup \{c\} | c \in C\}$ // join pattern with candidate items
2. **forall** $l' \in L$ **do begin**
3. **if** $\mathcal{R}(l').done == true$ **then** delete l' from L
4. **else begin**
5. $\mathcal{R}(l').count{+}{+}$ // count number of combinations
6. if $\mathcal{R}(l').count < \theta |l'|$ **then** delete l' from L // prune step
7. **else** $\mathcal{R}(l').done = true$
8. **end**
9. **end**
10. **return** L

Figure 8: Pseudocode for the pattern generation function for NB-DFS.

than once. Other depth-first mining algorithms (e.g., FP-Growth or DepthProject) solve this problem by using special representations of the database (frequent pattern tree structures [13] or a lexicographic tree [1]) which ensure that no part of the search space is traversed more than once. However, these techniques build on the fact that the algorithms mine frequent itemsets using the downward closed support constraint. To enforce the fraction θ of NB-frequent subsets and to ensure that itemsets in the lattice are only traversed once by NB-DFS we use an approach similar to Borgelt's implementation of closed and maximal item set filtering in his implementation of the *Eclat* algorithm [6]. A global repository $\mathcal{R}$ is used to keep track of the number of times a candidate itemset was already generated and of the itemsets which were already traversed. Figure 8 contains the pseudocode for the generation function. The function generates new NB-frequent patterns for a single pattern and its set of candidate items produced by NB-Select. In line 1 candidate itemsets are generated by joining the pattern with its candidate items. In lines 2 to 6 the count for each candidate itemset in the repository is updated and all itemsets for which we did not see enough combinations by now or which where already traversed are deleted. All remaining itemsets are new NB-frequent patterns. In line 5 these patterns are marked as done in the repository. Finally, the set of generated patterns is returned.

4.4 Comparison of the two algorithms

Compared to the level-wise breadth first search algorithm (NB-BFS), the depth first algorithm (NB-DFS) uses significantly more passes over the database. However, every time only a conditional database is scanned. This conditional database only contains the transactions which include the pattern which is currently expanded. As the pattern grows larger, the conditional database gets smaller and smaller. And if the original database is too large to fit into main memory, a conditional databases will fit into memory after the pattern grew in size which makes the subsequent scans very fast. Conditional databases can be easily implemented as lists of transaction ids.

The main advantage of NB-DFS is that it uses considerably less memory that BFS. The memory usage of BFS is dominated by the data structure $\mathcal{L}$ which holds the counters. NB-BFS needs to store $n|L_{k-1}|$ counters for level k where $|L_{k-1}|$ is the number of discovered patterns at level $k-1$. Since for low values of π and θ the number of patterns found for small k can get extremely high, NB-BFS is for such settings intractable.

NB-DFS always only counts the co-occurrences for one pattern at a time and therefore only needs to store n counters in $\mathcal{L}$. In addition NB-DFS needs to store the repository $\mathcal{R}$ which holds a counter and a boolean for each evaluated itemset. For low parameter values the size of this data structure grows fast, but in the pattern generation process, once a pattern is accepted, the counter for the pattern is not necessary any more and the memory can be released (this function could be added to line 7 in table 8). In this case, the repository would need one bit per itemset and only counters for the itemsets on the border which is currently explored. The number of these itemsets is relatively small compared to the total number of explored itemsets.

5 Experimental results

In this section we analyze the properties and the effectiveness of mining NB-frequent itemsets. We implemented NB-Select, the depth-first algorithm NB-DFS and the pattern generation function NB-Gen-DFS. With this implementation we will analyze how many patterns are found, the distribution of pattern sizes and error rates for different parameter settings.

To compare the performance of NB-frequent itemsets with existing methods we use frequent itemsets and itemsets generated using all-confidence as benchmarks. We chose frequent itemsets since a single support value represents the standard in mining association rules. All-confidence was chosen because its promising properties and its conceptual similarity with mining NB-frequent itemsets with $\theta = 1$.

5.1 Investigation of the pattern generation behavior

First, we examine how the number of patterns generated by the model-based algorithm depends on the precision parameter π. We use three different settings for the pattern generation function. We use the least and most restrictive settings with $\theta = 0$ and $\theta = 1$, and as the third setting we use $\theta = 0.5$ which requires that at least half of the possible $(k-1)$-subsets are NB-frequent in order to generate a pattern. Generally, we vary the parameter π for NB-Select between 0.5 and 0.999. However, since combinatorial explosion limits the range of practicable settings, depending on the data set and the parameter θ, some values of π are omitted.

In the plots in figure 9 we report the influence of the different settings on the three data sets already used in this paper (20,000 from WebView-1, POS and

π	**WebView-1**	**POS**	**Artif-1**
0.999	0.60	4.26	14.38
0.99	0.73	5.15	16.54
0.95	1.00	6.74	19.37
0.9	1.70	13.73	19.71
0.8	4.22	39.21	21.46
0.7	6.97	86.12	22.28

Table 5: CPU-time in seconds to mine 20,000 transactions of the data sets using $\theta = 0.5$.

Artif-1). In the plots to the left we see that by reducing π the number of accepted (NB-frequent) patterns increases fast for $\theta = 0$ while it only grows slowly for the most restrictive setting $\theta = 1$. At $\theta = 0.5$ the number of accepted patterns grows at a rate somewhere in between the two extreme settings. Although, for the extreme settings all three data sets react similar, for $\theta = 0.5$ there is a clear difference visible between the real-world data sets and the used artificial data set. While the growth rate for the real-world data sets is closer to $\theta = 0$, the growth rate for the the artificial data set is closer $\theta = 1$. For the artificial data set we also find already a relatively high number of accepted patterns at π near to one (clearly visible for $\theta = 0$ and $\theta = 0.5$), a characteristic which the real-world data sets do not show.

To analyze the influence of the growth of accepted patterns with parameter π on the execution time needed by the algorithm, we also recorded the CPU time[2] for the algorithm. The results for the setting $\theta = 0.5$ and the three data sets is given in table 5. The time needed to find the NB-frequent itemsets depends on the number of items and the structure of the data sets. For example, the data set WebView-1 has compared to the other two data sets fewer items and is extremely sparse with very short transactions (on average only 2.5 items). Therefore, the algorithm needs to search less itemsets and takes less time. Within each data set the time for different settings of the parameter π depends on how much of the search space can be pruned. Since pruning and the number of accepted itemsets is inversely related for finding NB-frequent itemsets, the needed time grows linearly with the number of accepted itemsets (compare left-hand side plots in figure 9). As for other algorithms, execution time is roughly linear in the number of transactions.

Next, we analyze the size of the accepted patterns. For comparison we gen-

[2]We used a single processor Pentium 4 machine running RedHat Linux release 9. The algorithm was implemented in JAVA and compiled using the gnu ahead-of-time compiler gcj version 3.2.2. CPU time was taken using the time command and we added user and system time.

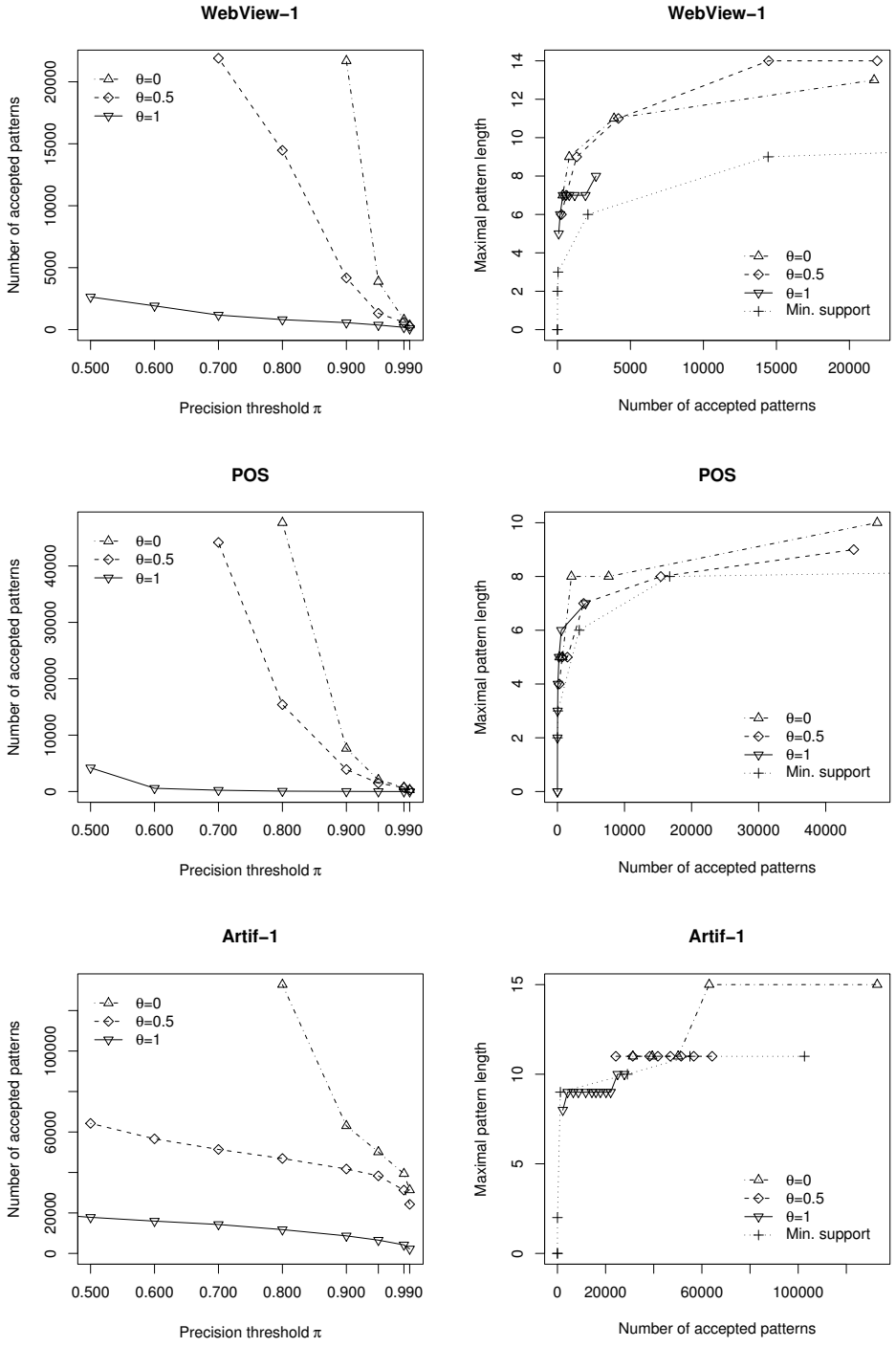


Figure 9: Comparison of different settings for the pattern generation on three data sets.

erated also frequent itemsets using the implementations of Apriori and Eclat by Christian Borgelt[3]. We varied the minimum support threshold σ between 0.1 and 0.0005 (experimentally we found the following values to work the best: 0.1, 0.05, 0.01, 0.005, 0.001, and 0.0005). In the plots to the right in figure 9 we show the maximal pattern size by the number of accepted (NB-frequent or frequent) patterns for the data sets and the settings used in the plot to the left. Naturally, the maximal size grows for all settings with the number of accepted pattern which in turn grows with a decreasing precision threshold π or minimum support σ. For the real-world data sets the model-based constraint tends to produce longer patterns for the same number of accepted patterns than minimum support. For the artificial data a clear difference is only visible for the setting $\theta = 0$.

The longer maximal pattern size for the model-based constraint is caused by NB-Select's way of selecting an individual support constraint for all 1-extensions of a pattern. To analyze this behavior we look at the minimum supports required by NB-Select for the data set WebView-1 at $\pi = 0.95$ and $\theta = 0.5$. In figure 10 we use a box-and-whisker plot to represent the distributions of the minimum support thresholds required by NB-Select for different pattern sizes. In the plot the lines in the boxes represent the median required minimum supports, the box spans from the lower to the upper quartile of the values, and the whiskers extend from the minimum to the maximum. The plot shows that the median of the required support falls with pattern size.

Seno and Karyptis [26] already proposed to reduce the required support threshold with itemset size to improve the chances of finding longer maximal frequent itemsets without being buried in millions of shorter frequent itemsets. The authors suggested to use instead of a fixed minimum support a minimum support function which decreases with itemset size. Seno and Karyptis used in their example in [26] a linear function together with an absolute minimum, however, the optimal choice of a support function and its parameters is still a research question. In contrast to their approach, there is no need to specify such a function for the model-based frequency constraint. NB-Select automatically adjusts support as needed. Furthermore, NB-Select does not use a single threshold per itemset size but it selects individual thresholds for all 1-extensions of a pattern.

Figure 10 shows that the median of the accepted support thresholds chosen by the NB-model falls from 0.0011 at length $k = 2$ to 0.0002 at $k = 9$ at an almost constant rate of 22% per increase of the itemset size by 1. Reducing support by a constant rate seems more intuitive than using a linear support function of the

[3] Available at http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html

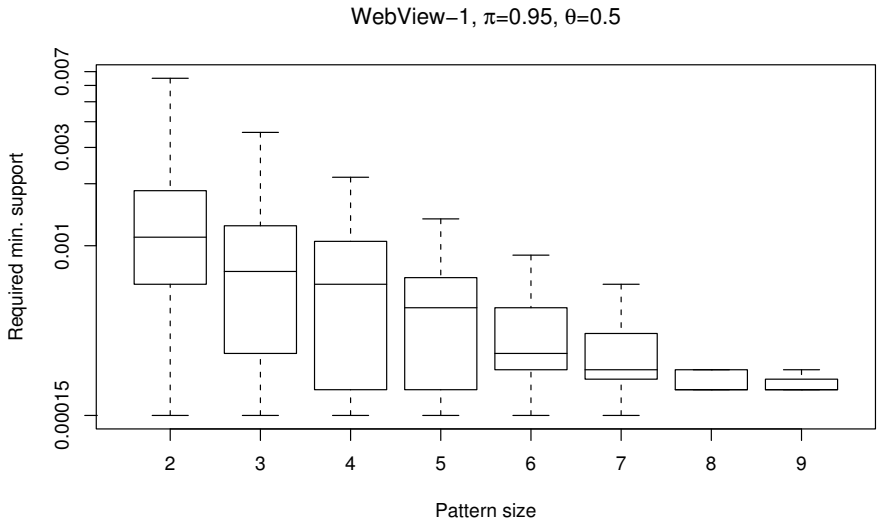


Figure 10: Histogram of the number of antecedents by the minimum support selected by NB-Select.

itemset size. For unrelated items, to give all k-itemsets the same chance to be selected, support would have to be chosen as a function of the form $\sigma_k = \sigma^{mk}$ with $m = 1$. However, since we are not interested in finding itemsets with unrelated items we would expect the optimal value for m in the range $[0, 1]$. For WebView-1 and the settings $\pi = 0.95$ and $\theta = 0.5$ NB-Select automatically chose for the individual support values a m around 0.22.

5.2 Effectiveness of pattern discovery

After we studied the behavior of the model-based algorithm and its ability to accept longer patterns already at lower support, we need to evaluate if these additional discovered itemset represent non-spurious patterns in the database. For this evaluation we need to know what true patterns exist in the data set and then compare how effective the algorithms are in discovering these patterns. Since for most real-world data sets the underlying patterns are unknown, we resort to artificial data sets, where the generation process is known and can be completely controlled.

To generate artificial data sets we use the generator developed by Agrawal and Srikant [4]. To understand the generation process and how to control the characteristics of the generated data sets, we first discuss the way the generator produces transactions (see [4] for more details).

The generator first generates a list of $|L|$ different patterns (called maximal potentially frequent itemsets; in the following we use the abbreviation MPFIs). Only these patterns will be used to build the transactions. The size of each of these itemsets is chosen from a Poisson distribution with a mean equal to the parameter $|I|$, the average size of the MPFIs. The first MPFI is generated by randomly choosing items from the pool of available items of size N. For each following MPFI some items are reused from the previous MPFI and the rest is chosen again randomly. The proportion of reused items is decided using an exponentially distributed random variable with unit mean times the *correlation level* (this is the implementation, however in [4] the authors state that correlation level is the mean of the exponential distribution). The correlation level models the idea that frequent itemsets found in real-world data sets often share some items. By default it is set to 0.5 and the authors of the generator report in [4] that changing this parameter has not much impact on the performance results of their experiments. Each MPFI has an weight assigned to it which is randomly chosen from an exponential distribution with a mean of 1.

After all MPFIs are generated, the algorithm starts to generate transactions. For each transaction a target transaction length is taken from a Poisson distribution with a mean of $|T|$, the average transaction length. Then a MPFI is randomly selected where the probability for selecting a MPFI is proportional to its weight. Before adding the MPFI's items to the transaction, items are dropped till a uniformly distributed random number between 0 and 1 is less than the *corruption level.* This models the fact that not all items of a MPFI are present in the same transaction. The corruption level is chosen for each itemset from a normal distribution with a mean of 0.5 and a variance of 0.1 (the original values used in [4]). Additional MPFIs are selected, corrupted and added to the transaction till the transaction is full. The MPFI that would increase the transaction size over the target size has a 50% chance that the items are still included in the transaction and a 50% chance that its items are used to start a new transaction. With this procedure the needed number of transaction $|D|$ is generated. Note, since only items included in a MPFI can be used in the transactions and since not always all available items are chosen to be part of a MPFI, the number of items in the transactions is normally smaller than the parameter N.

To evaluate the effectiveness of pattern discovery we need to know all MPFIs. Therefore, we adapted the code of the generator so that all MPFIs are reported (in the original version only the MPFIs with the highest weights were reported). Using this modified generator we generated two artificial data sets with the pa-

Parameter	Description	Artif-1	Artif-2
$\|D\|$	Number of transactions	100,000	100,000
$\|T\|$	Avg. size of transactions	10	10
$\|I\|$	Avg. size of maximal potentially frequent itemsets (MPFIs)	4	2
$\|L\|$	Number of MPFIs	2,000	4,000
N	Number of items	1,000	1,000

Table 6: Parameters for the artificial data sets used in the experiments.

rameter settings presented in table 6.

The first data set, Artif-1, represents the standard data set T10I4D100K used in [4] and in many other papers. For this data set only 2,000 MPFIs (different patterns) are used with an average size of 4. Itemsets (corrupted MPFIs) of size 1 in the data set can be interpreted as noise which makes finding patterns (MPFIs of size 2+ which can be used to generate rules) harder. We can estimate the percentage of noise in the data set since we know the generation process. For the generation, the items in chosen MPFIs are dropped using a series of Bernoulli trials with a probability p around 0.5 (the corruption level) which is stopped after the first failure. The probability for dropping no item is $1 - p$, therefore 50% of all MPFIs are added uncorrupted to transactions. The probability of dropping one item is $p(1 - p)$, the probability of dropping two items is $p^2(1 - p)$, and so forth. In general, the probability of dropping n items is $p^n(1 - p)$. Together with the Poisson size distribution with a mean of 4, we can calculate the percentage of incidences in the data set which originate from either a MPFI of size one or from a MPFI which is corrupted enough so that only a single item is added to a transaction. After tabulating the probabilities for each MPFI size, weighing them using the Poisson distributed sizes, and removing all resulting itemsets of size zero (which are ignored in the data set generation process since they do not produce an incidence), we obtain a noise of only 3.76% of the incidences in the data set. This is an extremely low value and means that the task of finding itemsets in a noisy database degenerates to the task of separating transactions into parts of patterns.

To make the data mining task more difficult, we need to generate a data set with more noise. The first way which comes to mind is to increase the corruption level. However, since a higher corruption level also affects MPFIs of size one and reduces their chance of being added to a transaction, the increase of noise is only small. For example, increasing the corruption level from 0.5 to 0.75 means

that on average only 25% of the MPFIs of size one chosen to be included in a transaction are actually included after corrupting each MPFI. Therefore, the noise level only increases from 3.76% to 5.57%. A more effective way to increase noise is to decrease the average size of the MPFIs. For the data set Artif-2 we reduce the average size of the MPFIs to 2. This results in a noise level of 17.00% of the incidences in the data set. A side effect of reducing the average MPFI size is that the chance of producing longer patterns is smaller. To work against this effect, we double the number if the MPFIs to $4,000$. This again make the mining task harder since there are now more patterns to mine within the same number of transactions and incidences.

We took again for each data set the first $20,000$ transactions for mining patterns and additionally, to analyze if performance is influenced by the data set size, we also use sets of size $5,000$ and $80,000$ taken from Artif-2. For the model-based algorithm we estimated the parameters of the model and then generated patterns with the settings $\theta = 0$, $\theta = 0.5$ and $\theta = 1$. For minimum precision we used values between 0.999 and 0.1 (0.999, 0.99, 0.95, 0.9, 0.8 and in 0.1 steps down to 0.1). For $\theta = 1$ we used all values, for $\theta = 0.5$ we only used values ≥ 0.5, and for $\theta = 0$ we used values ≥ 0.8.

For comparison we used again Borgelt's implementation of the Eclat algorithm to find frequent itemsets at minimum support levels between 0.1 and 0.0005 (0.01, 0.005, 0.004, 0.003, 0.002, 0.0015, 0.0013, 0.001, 0.0007, and 0.0005). As a second benchmark we also generated itemsets using all-confidence [21]. We varied the threshold on all-confidence between 0.01 and 0.6 (0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05, 0.04, 0.03, 0.02, 0.01). The used minimum support levels and all-confidence thresholds were found after some experimentation to cover a wide area of the possible true positives/false positives combinations for the data sets.

To compare the ability to discover the patterns (MPFIs and their subsets) which were used to generate the artificial data sets, we counted the *true positives* (itemsets discovered by the algorithm which were used in the data set generation process) and *false positives* (discovered itemsets which were not used in the data set generation process). This information together with the *total number of true positives*, all patterns used to generate a data set, is used to visually inspect the algorithms' performances over the range of their parameter spaces.

In figures 11 and 12 we report performance using two plots per data set or data set size. The first plot is a precision/recall plot known form information retrieval [25] which is also common in machine learning [17]. Precision is defined

as

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{13}$$

and recall is defined as

$$\text{recall} = \frac{\text{true positives}}{\text{total number of true positives}}. \tag{14}$$

The markers in the precision/recall plots (plots to the left in figures 11 and 12) represent the results for the algorithm using different values for π and three settings for θ. Markers for the same setting of θ are connected by lines since all points on these lines can be reached by a combination of the two settings used for the markers next to them. The top right corner of each precision/recall plot represents the optimal combination where all patterns are discovered (recall $= 1$) and no false pattern is selected (precision $= 1$). Curves that are closer to the top right corner represent better retrieval performance. For comparison we added to the plots the precision/recall curves for minimum support and all-confidence.

The precision/recall plots in figures 11 and 12 show that with $\theta = 1$ and $\pi \geq 0.5$ reachable recall is comparably low, typically smaller than 0.5, while precision is high. On the data sets with 20,000 transactions it shows similar performance as all-confidence. However, it outperforms all-confidence considerably on the small data set (Artif-2 with 25,000 transactions) while it is outperformed by all-confidence on the larger data set (Artif-2 with 80,000 transactions). This observation suggests that the additional knowledge of the structure of the data is more helpful if only little data is available.

With $\theta = 0$, where pattern generation only requires one direct NB-frequent subset, the algorithm reaches higher recall but precision deteriorates considerably with increased recall. The performance is generally better than minimum support and all-confidence. Only for settings with very low values for π, precision degrades so strongly that its performance is worse than minimum support and all-confidence. This effect can be seen for $\pi = 0.5$ on data set Artif-2 with 80,000 transactions in figure 12.

The model-based algorithm with $\theta = 0.5$ performs overall the best with higher recall while loosing less precision. Its performance clearly beats minimum support, all-confidence, and the model based algorithm with settings $\theta = 0$ and $\theta = 1$ on all data sets.

A weakness of precision/recall plots and many other other ways to measure accuracy is that they are only valid for comparison under the assumption of uni-

form misclassification cost, i.e., the error cost for false positives and false negatives are equal. For some applications missing a pattern (not detecting a fraudulent transaction) has higher cost than a false alarm. For other application a false alarm (stocking a wrong item combination in a retail store) incurs higher cost. A representation that does not depend on uniform misclassification cost is the *Receiver Operator Characteristics graph* (ROC graph) developed for signal detection (see [28]). ROC graphs are used in machine learning to compare classifier accuracy [24] independently of class distribution (proportion of true positives to true negatives) and the distribution of misclassification costs. A ROC graph is a plot with the *false positive rate* on the x-axis and the *true positive rate* on the y-axis and presents the possible error tradeoffs for each classifier. The ideal classifier would produce a point to the top left corner of the plot. If a classifier can be parameterized the points obtained using different parameters can be connected by a line called a *ROC curve.* If all points of one classifier are superior to the points of another classifier, the first classifier dominates the latter one. This means that for all possible cost and class distributions the first classifier can produce better results.

In our case the mining algorithms are seen as different classifiers which classify each possible itemset as either a positive (the discovered patterns) or a negative (the rest). In figures 11 and 12 we present the ROC graphs for the 2 data sets (plots to the right). To see the trade-off between true positives and false positives we show absolute numbers instead of rates in the plots (the interpretation is the same as with rates). For the used data sets, the model-based frequency constraint with $\theta = 0.5$ dominates all other settings as well as minimum support and all-confidence. As for the precision/recall plots, the advantage of the model-based constraint over minimum support increases with noise and its advantage over all-confidence increases with a reduced number of available transactions.

Summarizing the performance on all data sets, it can be said that the model-based constraint and all-confidence are less affected by the increase in noise (from Artif-1 to Artif-2 with 20,000 transactions) than minimum support. The decrease in data set size to 5,000 transactions affects all-confidence while it has considerably less effect on the model-based constraint and on minimum support. For an increasing data set size (see Artif-2 with 80,000 transactions) and for the model-based algorithm at a set π, recall increases while at the same time precision decreases. This happens because with more available data NB-Select's predictions for precision get closer to the real values.

In table 7 we summarize the actual precision of the mined associations with $\theta = 0.5$ at different settings for the precision threshold. The close agreement

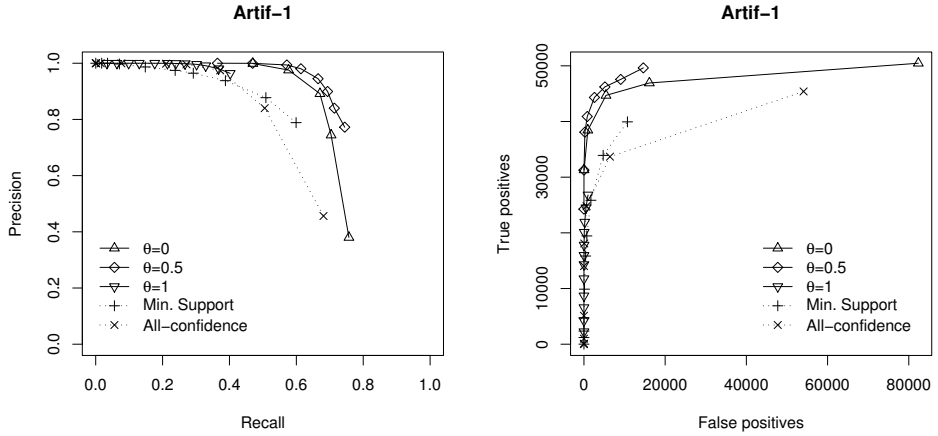


Figure 11: Performance of the algorithm on Artif-1 using 20,000 transactions.

π	precision
0.999	1.0000000
0.990	0.9997855
0.950	0.9704649
0.900	0.8859766
0.800	0.7848500
0.700	0.7003764
0.600	0.5931635
0.500	0.4546763

Table 7: Comparison of the set precision threshold π and the actual precision of the mined associations for $\theta = 0.5$ on data set Artif-2 with 80,000 transactions.

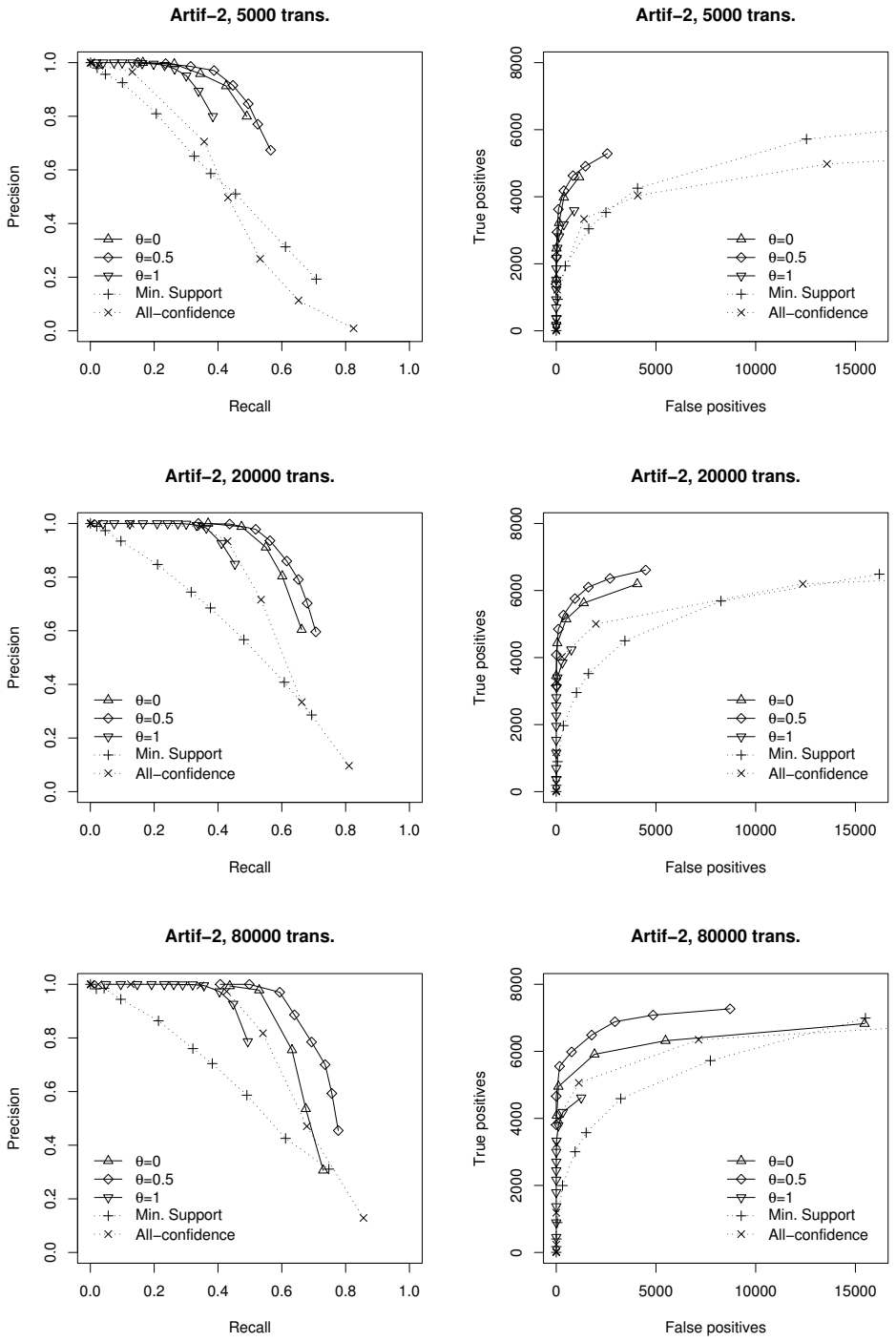


Figure 12: Performance of the algorithm on Artif-2 using different data set sizes.

	precision	recall
$\theta = 1$	15.94%	1.54%
$\theta = 0.5$	6.86%	8.32%
$\theta = 0$	16.88%	14.09%
Min. support	30.65%	23.37%
All-confidence	79.49%	17.31%

Table 8: Relative differences of precision and recall between the results from data sets Artif-1 and Artif-2 (both with 20,000 transactions).

between the columns indicates that, with enough available data, the set precision threshold gets close to the precision of the set of mined associations. This is an important property of the model-based constraint since it makes the precision parameter easier to understand and set for the person who applies data mining. While suitable thresholds on measures as support and all-confidence are normally found for each data set by experimentation, the precision threshold can be set with a maximal acceptable error rate for an application in mind.

Furthermore, the precision/recall plots in figures 11 and 12 (with 20,000 transactions) show that the results of the model-based constraint (especially for $\theta = 0.5$) is less dependent of the structure and noise in the data set. To quantify this finding, we calculate the relative differences between the resulting precision and recall values for each parameter setting of each algorithm. In table 8 we present the average of the relative differences per algorithm. While precision differs for support between the two datasets on average by about 30%, all-confidence exhibits an average difference of nearly 80%. The model-based algorithm only differs by less than 20%, and with $\theta = 0.5$ the precision difference is only about 7%. This suggests that setting an average value for π (e.g., 0.9) will produce reasonable results independently of the data set. The user only needs to resort to experimentation with different settings for the parameter if she needs to optimize the results.

6 Conclusion

The contribution of this paper is that we presented a model-based alternative to using a single, user-specified minimum support threshold for mining associations in transaction data. We extended the simple and robust stochastic mixture model (the NB model) to develop a baseline model for incidence counts (co-occurrences of items) in the database. We used this baseline model together with a precision threshold to find an individual frequency constraint (a support threshold) for all

1-extensions of an itemset.

Based on this model-based frequency constraint, we developed the notion of NB-frequent itemsets. We presented two mining algorithms to find all NB-frequent itemsets in a database. The first algorithm, NB-BFS, is an Apriori-like algorithm with a breadth-first search strategy. The second algorithm, NB-DFS, is similar to the DepthProject and employs a recursive depth-first search strategy. Although, the definition of NB-frequency, which is based on individual frequency constraints, does not provide the important downward closure property of support, we showed how the search space can be pruned adequately.

Experiments showed that the model-based frequency constraint automatically reduces the average needed frequency (support) with growing pattern size. Compared with support it tends to be more selective for shorter rules while still accepting longer rules with lower support. This property reduces the problem of being buried in a great number of short patterns when using a relatively low support threshold in order to find longer patterns.

Further experiments on artificial data sets indicate that the model-based constraint is significantly more effective in finding non-spurious patterns, especially in noisy data sets or where only a relatively small database is available.

The experiments also show that the precision parameter of the model-based algorithm depends less than support or any-confidence on the data set. This is a big advantage which reduces the need for time-consuming experimentation with different parameter settings for each new data set.

Another advantage of the model-based algorithm is that the precision threshold focuses on the predicted precision of the mined set of associations and, therefore, has a direct interpretation for the application of the results. This makes communicating and choosing a suitable setting easier.

Finally, it has to be noted that the model-based constraint developed in this paper can only be used for databases which are generated by a process similar to the developed baseline model. The developed baseline model is robust and reasonable for most transaction data (e.g., point-of-sale data). For other types of data, different baseline models need to be developed which can then be incorporated in mining algorithms following the outline of this paper.

Acknowledgments

I wish to thank Blue Martini Software for contributing the KDD Cup 2000 data, Ramakrishnan Srikant from the IBM Almaden Research Center for making the

code for the synthetic transaction data generator available, and to Christian Borgelt for the free implementations of Apriori and Eclat.

Also I want to thank Andreas Geyer-Schulz and Kurt Hornik for the long discussions on modeling transaction data.

References

[1] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. Depth first generation of long patterns. pages 108–118, 2000.

[2] C. C. Aggarwal and P. S. Yu. A new framework for itemset generation. In *PODS 98, Symposium on Principles of Database Systems*, pages 18–24, Seattle, WA, USA, 1998.

[3] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington D.C., May 1993.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499, Santiago, Chile, Sept 1994.

[5] Z. Bi, C. Faloutsos, and F. Korn. The "DGX" distribution for mining massive, skewed data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD01)*, pages 17–26, 2001.

[6] C. Borgelt. Efficient implementations of apriori and eclat. In B. Goethals and M. J. Zaki, editors, *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, Melbourne, FL, USA, November 2003.

[7] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 255–264, Tucson, Arizona, USA, May 1997.

[8] Q. L. Burrell. Using the gamma-poisson model to predict library circulations. *Journal of the American Society for Information Science*, 41(3):164–170, 1990.

[9] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):64–78, 2001.

[10] 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, Series B (Methodological)*, 39:1–38, 1977.

[11] W. DuMouchel and D. Pregibon. Empirical bayes screening for multi-item associations. In F. Provost and R. Srikant, editors, *Proceedings of the ACM SIGKDD Intentional Conference on Knowledge Discovery in Databases & Data Mining (KDD01)*, pages 67–76. ACM Press, 2001.

[12] A. S. C. Ehrenberg. *Repeat-Buying: Facts, Theory and Application*. Charles Griffin & Company Ltd., London, 1988.

[13] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation. *Data Mining and Knowledge Discovery*, 8:53–87, 2004.

[14] B. A. Huberman, P. L. T. Pirolli, J. E. Pitkow, and R. M. Lukose. Strong regularities in World Wide Web surfing. *Science*, 280(5360):95–97, 1998.

[15] N. L. Johnson, S. Kotz, and A. W. Kemp. *Univariate Discrete Distributions*. John Wiley & Sons, New York, 2nd edition, 1993.

[16] R. Kohavi, C. Brodley, B. Frasca, L. Mason, and Z. Zheng. KDD-Cup 2000 organizers' report: Peeling the onion. *SIGKDD Explorations*, 2(2):86–98, 2000.

[17] R. Kohavi and F. Provost. Glossary of terms. *Machine Learning*, 30(2–3):271–274, 1988.

[18] S. Lee, F. Zufryden, and X. Dreze. Modeling consumer visit frequency on the internet. In *34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 7*, 2001.

[19] B. Liu, W. Hsu, and Y. Ma. Mining association rules with multiple minimum supports. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-99)*, pages 337–341, 1999.

[20] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In U. M. Fayyad and R. Uthurusamy, editors, *AAAI Workshop on Knowledge Discovery in Databases (KDD-94)*, pages 181–192, Seattle, Washington, 1994. AAAI Press.

[21] E. R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):57–69, Jan/Feb 2003.

[22] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Lecture Notes In Computer Science archive Proceeding of the 7th International Conference on Database Theory, Lecture Notes In Computer Science (LNCS)*. Springer, 1999.

[23] J. Pei, J. Han, and L. V. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proceedings of the 17th International Conference on Data Engineering, April 02 - 06, 2001, Heidelberg, Germany*, 2001.

[24] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In D. Heckerman, H. Mannila, and D. Pregibon, editors, *The Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pages 43–48, Newport Beach, CA, August 1997. AAAI Press.

[25] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[26] M. Seno and G. Karypis. Lpminer: An algorithm for finding frequent itemsets using length decreasing support constraint. In N. Cercone, T. Y. Lin, and X. Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, pages 505–512. IEEE Computer Society, 2001.

[27] C. Silverstein, S. Brin, and R. Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2:39–68, 1998.

[28] C. van Rijsbergen. *Information retrieval*. Butterworth, London, 1979.

[29] H. Xiong, P.-N. Tan, and V. Kumar. Mining strong affinity association patterns in data sets with skewed support distribution. In B. Goethals and M. J. Zaki, editors, *Proceedings of the IEEE International Conference on Data Mining, November 19 - 22, 2003, Melbourne, Florida*, November 2003.

[30] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. Technical Report 651, Computer Science Department, University of Rochester, Rochester, NY 14627, July 1997.

[31] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In F. Provost and R. Srikant, editors, *Proceedings of the ACM SIGKDD Intentional Conference on Knowledge Discovery in Databases & Data Mining (KDD01)*, pages 401–406. ACM Press, 2001.