Knowl Inf Syst
DOI 10.1007/s10115-012-0561-2

REGULAR PAPER

Model-based probabilistic frequent itemset mining

Thomas Bernecker · Reynold Cheng · David W. Cheung · Hans-Peter Kriegel · Sau Dan Lee · Matthias Renz · Florian Verhein · Liang Wang · Andreas Zuefle

Received: 1 March 2011 / Revised: 30 April 2012 / Accepted: 29 July 2012

Abstract Data uncertainty is inherent in emerging applications such as location-based services, sensor monitoring systems, and data integration. To handle a large amount of imprecise information, uncertain databases have been recently developed. In this paper, we study how to efficiently discover frequent itemsets from large uncertain databases, interpreted under the *Possible World Semantics*. This is technically challenging, since an uncertain database induces an exponential number of possible worlds. To tackle this problem, we propose a novel methods to capture the itemset mining process as a probability distribution function taking two models into account: the Poisson distribution and the normal distribution. These *model-based approaches* extract frequent itemsets with a high degree of accuracy and

T. Bernecker · H.-P. Kriegel · M. Renz · F. Verhein · A. Zuefle
Department of Computer Science, Ludwig-Maximilians-Universität, Munchen, Germany
e-mail: bernecker@dbs.ifi.lmu.de

H.-P. Kriegel
e-mail: kriegel@dbs.ifi.lmu.de

M. Renz
e-mail: renz@dbs.ifi.lmu.de

F. Verhein
e-mail: verhein@dbs.ifi.lmu.de

A. Zuefle
e-mail: zuefle@dbs.ifi.lmu.de

R. Cheng (✉) · D. W. Cheung · S. D. Lee · L. Wang
Department of Computer Science, University of Hong Kong, Pokfulam, Hong Kong
e-mail: ckcheng@cs.hku.hk

D. W. Cheung
e-mail: dcheung@cs.hku.hk

S. D. Lee
e-mail: sdlee@cs.hku.hk

L. Wang
e-mail: lwang@cs.hku.hk

Published online: 21 October 2012

support large databases. We apply our techniques to improve the performance of the algorithms for (1) finding itemsets whose frequentness probabilities are larger than some threshold and (2) mining itemsets with the k highest frequentness probabilities. Our approaches support both tuple and attribute uncertainty models, which are commonly used to represent uncertain databases. Extensive evaluation on real and synthetic datasets shows that our methods are highly accurate and four orders of magnitudes faster than previous approaches. In further theoretical and experimental studies, we give an intuition which model-based approach fits best to different types of data sets.

1 Introduction

In many applications, the underlying databases are uncertain. The locations of users obtained through RFID and GPS systems, for instance, are not precise due to measurement errors [24, 32]. In habitat monitoring systems, data collected from sensors like temperature and humidity are noisy [1]. Customer purchase behaviors, as captured in supermarket basket databases, contain statistical information for predicting what a customer will buy in the future [4,39]. Integration and record linkage tools associate confidence values to the output tuples according to the quality of matching [14]. In structured information extractors, confidence values are appended to rules for extracting patterns from unstructured data [40]. Recently, *uncertain databases* have been proposed to offer a better support for handling imprecise data in these applications [10,14,21,23,30].[1]

In fact, the mining of uncertain data has recently attracted research attention [4]. For example, in [26], efficient clustering algorithms were developed for uncertain objects; in [22] and [41], naïve Bayes and decision tree classifiers designed for uncertain data were studied. Here, we develop scalable algorithms for finding frequent itemsets (i.e., sets of attribute values that appear together frequently in tuples) for uncertain databases. Our algorithms can be applied to two important uncertainty models: *attribute uncertainty* (e.g., Table 1), and *tuple uncertainty*, where every tuple is associated with a probability to indicate whether it exists [13,14,21,30,31].

As an example of uncertain data, consider an online marketplace application (Table 1). Here, the purchase behavior details of customers Jack and Mary are recorded. The value associated with each item represents the chance that a customer may buy that item in the near future. These probability values may be obtained by analyzing the users' browsing histories. For instance, if Jack visited the marketplace ten times in the previous week, out of which *video* products were clicked five times, the marketplace may conclude that Jack has a 50 % chance of buying, or simply being interested in *videos*.

To interpret uncertain databases, the *Possible World Semantics* (or PWS in short) is often used [14]. Conceptually, a database is viewed as a set of deterministic instances (called *possible worlds*), each of which contains a set of tuples. Thus, an uncertain transaction database D generates a set of possible worlds $\mathcal{W}$. Table 2 lists all possible worlds for the database depicted in Table 1. Each world $w_i \in \mathcal{W}$, which consists of a subset of attributes from each transaction, occurs with probability $Pr(w_i)$.

For instance, possible world w_2 consists of two itemsets, $\{food\}$ and $\{clothing, video\}$, for Jack and Mary, respectively. The itemset $\{food\}$ occurs with a probability of $\frac{1}{2}$, since this itemset corresponds to the set of events that Jack purchases food (probability of one), and Jack does not purchase video (probability of $\frac{1}{2}$). Assuming independence

[1] Manuscript received on Mar 01, 2011; revised on Apr 30, 2012; and accepted on Jul 29, 2012.

Table 1 An uncertain database

Customer	Purchase items
Jack	(*video*:1/2),(*food*:1)
Mary	(*clothing*:1),(*video*:1/3); (*book*:2/3)

Table 2 Possible worlds of Table 1

$\mathcal{W}$	Tuples in $\mathcal{W}$	Prob.
w_1	{*food*}; {*clothing*}	1/9
w_2	{*food*}; {*clothing, video*}	1/18
w_3	{*food*}; {*clothing, book*}	2/9
w_4	{*food*}; {*clothing, book, video*}	1/9
w_5	{*food, video*}; {*clothing*}	1/9
w_6	{*food, video*}; {*clothing, video*}	1/18
w_7	{*food, video*}; {*clothing, book*}	2/9
w_8	{*food, video*}; {*clothing, book, video*}	1/9

between these events (i.e., the event that Jack buys food does not impact the probability of Jack buying a video game), the joint probability of these events is $1 \cdot \frac{1}{2} = \frac{1}{2}$. Analogously, the probability of obtaining the itemset {*clothing,video*} for Mary is $1 \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{9}$. As shown in Table 2, the sum of possible world probabilities is one, and the number of possible worlds is exponential in the number of probabilistic items.

Any query evaluation algorithm for an uncertain database has to be correct under PWS. That is, the results produced by the algorithm should be the same as if the query is evaluated on every possible world [14].

Definition 1 (*Possible World Semantics (PWS)*) Let $\mathcal{W}$ be the set of all possible worlds derived from a given uncertain database D and let φ be a query predicate. Under *possible world semantics*, the probability $P(\varphi, D)$ that D satisfies φ is given as the total probability of all worlds that satisfy φ, that is,

$$P(\varphi, D) = \sum_{w \in \mathcal{W}} P(w) \cdot \varphi(w),$$

where $\varphi(w)$ is an indicator function that returns 1 if world w satisfies predicate φ, and zero otherwise, and $P(w)$ is the probability of world w.

Our goal is to discover frequent patterns without expanding D into all possible worlds.

Although PWS is intuitive and useful, querying or mining under this notion is costly. This is due to the fact that an uncertain database has an exponential number of possible worlds. For example, the database in Table 1 has $2^3 = 8$ possible worlds. Performing data mining under PWS can thus be technically challenging. On the other hand, ignoring PWS allows to find efficient solutions, for example using expected support only [11]. However, such approaches do not consider probability distributions, that is, are not able to give any probabilistic guarantees. In fact, an itemset whose expected support is greater than a threshold may have a low probability of having a support greater than this threshold. An example of such a scenario can be found in [39].

The frequent itemsets discovered from uncertain data are naturally probabilistic, in order to reflect the confidence placed on the mining results. Figure 1 shows a *Probabilistic Frequent*

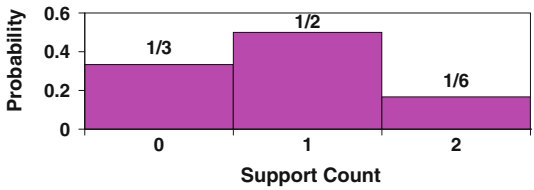


Fig. 1 s-pmf of PFI {*video*} from Table 1

Itemset (or *PFI*) extracted from Table 1. A *PFI* is a set of attribute values that occur frequently with sufficiently high probabilities. In Fig. 1, the *support probability mass function* (or *s-pmf* in short) for the *PFI* {*video*} is shown. This is the pmf for the number of tuples (or *support count*) that contain an itemset. Under PWS, a database induces a set of possible worlds, each giving a (different) support count for a given itemset. Hence, the support of a frequent itemset is described by a pmf. In Fig. 1, if we consider all possible worlds where itemset {*video*} occurs twice, the corresponding probability is $\frac{1}{6}$.

A simple way of finding PFIs is to mine frequent patterns from every possible world and then record the probabilities of the occurrences of these patterns. This is impractical, due to the exponential number of possible worlds. To remedy this, some algorithms have been recently developed to successfully retrieve PFIs without instantiating all possible worlds [39,46]. These algorithms are able to find a PFI in $O(n^2)$ time (where n is the number of tuples contained in the database). However, our experimental results reveal that they require a long time to complete (e.g., with a 300 k real dataset, the dynamic-programming algorithm in [39] needs 30.1 h to complete).

In this paper, we propose an efficient approximate probabilistic frequent itemset mining solution using specific models to capture the frequentness of an itemset. More precisely, we generalize the model-based approach proposed in [43]. The basic idea is to use appropriate standard parametric distributions to approximate the probabilistic support count, that is, the probability distribution (pmf) of the support count, for both attribute- and tuple-uncertain data. The advantage of such parametric distributions is that they can be computed very efficiently from the transaction database while providing quite good approximation of the support pmf. In particular, this paper introduces a generalized model-based frequent itemset mining approach investigating and discussing three alternatives for modeling the support pmf of itemsets. In addition to the Poisson distribution, which has been used for probabilistic frequent itemset mining and probabilistic ranking [20,43], and the expected support [12], we further investigated the normal distribution for probabilistic frequent itemset mining. Based on these models, we show how the cumulative distribution (cdf) of an itemset associated with the support pmf can be computed very efficiently by means of a single scan of the transaction database. In addition, we provide an in-depth analysis of the proposed models on a theoretical as well as experimental level where we focus on approximation quality and characteristics of the underling data. Here, we evaluate and compare the three models in terms of efficiency, effectiveness, and implementation issues. We show that some of the models are very accurate while some models require certain properties of the data set to be satisfied in order to achieve very high accuracy. An interesting observation is that some models allow us to further reduce the runtime introducing specific pruning criteria, these models, however, lack effectiveness compared to others. We show that the generalized *model-based approach* runs in $O(n)$ time and is thus more scalable to large datasets. In fact, our algorithm only needs 9.2 s to find all PFIs, which is four orders of magnitudes faster than solutions that are based on the exact support pmf.

In addition, we demonstrate how the model-based algorithm can work under two semantics of PFI, proposed in [39]: (1) *threshold-based*, where PFIs with probabilities larger than some user-defined threshold are returned; and (2) *rank-based*, where PFIs with the k highest probabilities are returned. As we will show, these algorithms can be adapted to the attribute and tuple uncertainty models. For mining threshold-based PFIs, we demonstrate how to reduce the time scanning the database. For mining rank-based PFIs, we optimize the previous algorithm to improve the performance.

We derive the time and space complexities of our approaches. As our experiments show, model-based algorithms can significantly improve the performance of PFI discovery, with a high degree of accuracy. To summarize, our contributions are:

- A generalized model-based approach to approximately compute PFIs efficiently and effectively;
- In-depth study of three s-pmf approximation models based on standard probability models;
- A more efficient method to verify a threshold-based PFI;
- Techniques to enhance the performance of threshold and rank-based PFI discovery algorithms, for both attribute and tuple uncertainty models; and
- An extensive experimental evaluation of the proposed methods in terms of efficiency and effectiveness performed on real and synthetic datasets.

This paper is organized as follows. In Sect. 2, we review the related work. Section 3 discusses the problem definition. Section 4 describes our model-based support pmf (s-pmf) approximation framework introducing and discussing three models to estimate the s-pmf efficiently and accurately. Then, in Sects. 5 and 6, we present algorithms for discovering threshold- and rank-based PFIs, respectively. Section 7 presents an in-depth experimental evaluation of the proposed s-pmf approximation models in terms of effectiveness and performance results. We conclude in Sect. 8.

2 Related work

Mining frequent itemsets is often regarded as an important first step of deriving association rules [5]. Many efficient algorithms have been proposed to retrieve frequent itemsets, such as Apriori [5] and FP-growth [18]. There are also a lot of adaptations for other transaction database settings, like itemset mining in streaming environments [45]; a survey can be found in [28]. While these algorithms work well for databases with precise and exact values, it is interesting to extend them to support uncertain data. Our algorithms are based on the Apriori algorithm. We are convinced that they can be used by other algorithms (e.g., FP-growth) to support uncertain data. In [33], probabilistic models for query selectivity estimation in binary transaction databases have been investigated. In contrast to our work, transactions are assumed to be certain while the probabilistic models are associated with approximate queries on such data.

Solutions for frequent itemset mining in uncertain transaction databases have been investigated in [3,12,42,44]. In [42], an approach for summarizing frequent itemset patterns based on Markov Random Fields has been proposed. In [3,12,44], efficient frequent pattern mining algorithms based on the expected support counts of the patterns have been developed. However, [39,46] found that the use of expected support may render important patterns missing. Hence, they proposed to compute the probability that a pattern is frequent, and introduced the notion of PFI. In [39], dynamic-programming-based solutions were developed to retrieve

Table 3 Our contributions (marked [√])

Uncertainty	Threshold-PFI	Rank-PFI
Attribute	Exact [39]	Exact [39]
	Approx. [√]	Approx. [√]
Tuple	Exact [38]	
	Approx. (singleton) [46]	Approx. [√]
	Approx. (multiple items) [√]	

PFIs from attribute-uncertain databases, for both threshold- and rank-based PFIs. However, their algorithms have to compute exact probabilities and compute a PFI in $O(n^2)$ time. By using probability models, our algorithms avoid the use of dynamic programming, and can find a PFI much faster (in $O(n)$ time). In [46], approximate algorithms for deriving threshold-based PFIs from tuple-uncertain data streams were developed. While [46] only considered the extraction of singletons (i.e., sets of single items), our solution discovers patterns with more than one item. More recently, [38] developed an exact threshold-based PFI mining algorithm. However, it does not support rank-based PFI discovery. Here, we also study the retrieval of rank-based PFIs from tuple-uncertain data. To the best of our knowledge, this has not been examined before. Table 3 summarizes the major work done in PFI mining.

Other works on the retrieval of frequent patterns from imprecise data include: [9], which studied approximate frequent patterns on noisy data; [27], which examined association rules on fuzzy sets; and [29], which proposed the notion of a "vague association rule". However, none of these solutions are developed on the uncertainty models studied here.

3 Problem definition

In Sect. 3.1, we discuss the uncertainty models used in this paper. Then, we describe the notions of threshold- and rank-based PFIs in Sect. 3.2.

3.1 Attribute and tuple uncertainty

Let V be a set of items. In the *attribute uncertainty model* [10,23,32,39], each attribute value carries some uncertain information. Here, we adopt the following variant [39]: a database D contains n tuples, or transactions. Each transaction t_j is associated with a set of items taken from V. Each item $v \in V$ exists in t_j with an existential probability $Pr(v \in t_j) \in (0, 1]$, which denotes the chance that v belongs to t_j. In Table 1, for instance, the existential probability of *video* in t_{Jack} is $Pr(video_{Jack}) = 1/2$. This model can also be used to describe uncertainty in binary attributes. For instance, the item $video$ can be considered as an attribute, whose value is one, for Jack's tuple, with probability $\frac{1}{2}$, in tuple t_{Jack}.

Under the possible world semantics (PWS), D generates a set of possible worlds $\mathcal{W}$. Table 2 lists all possible worlds for Table 1. Each world $w_i \in \mathcal{W}$, which consists of a subset of attributes from each transaction, occurs with probability $Pr(w_i)$. For example, $Pr(w_2)$ is the product of: (1) the probability that Jack purchases *food* but not *video* (equal to $\frac{1}{2}$) and (2) the probability that Mary buys *clothing* and *video* only (equal to $\frac{1}{9}$). As shown in Table 2, the sum of possible world probabilities is one, and the number of possible worlds is exponentially large. Our goal is to discover frequent patterns without expanding D into possible worlds.

In the *tuple uncertainty model*, each tuple or transaction is associated with a probability value. We assume the following variant [13,31]: each transaction $t_j \in D$ is associated with a

Table 4 Summary of notations

Notation	Description
D	An uncertain database of n tuples
n	The number of tuples contained in D
V	The set of items that appear in D
v	An item, where $v \in V$
t_j	The j-th tuple with a set of items, where $j = 1, \ldots, n$
$\mathcal{W}$	The set of all possible worlds
w_j	A possible world $w_j \in W$
I	An itemset, where $I \subseteq V$
$minsup$	An integer between $(0, n]$
$minprob$	A real value between (0, 1], used in threshold-based PFI
k	An integer greater than zero, used in rank-based PFI
$Pr^I(i)$	Support probability (i.e., probability that I has a support count of i)
$Pr_{freq}(I)$	Frequentness probability of I
p_j^I	$Pr(I \subseteq t_j)$
μ_I	Expected value of X^I
$(\mu_I)_l$	Expected X^I, up to l tuples

set of items and an existential probability $Pr(t_j) \in (0, 1]$, which indicates that t_j exists in D with probability $Pr(t_j)$. Again, the number of possible worlds for this model is exponentially large. Table 4 summarizes the symbols used in this paper.

3.2 Probabilistic frequent itemsets (PFI)

Let $I \subseteq V$ be a set of items, or an *itemset*. The *support* of I, denoted by $s(I)$, is the number of transactions in which I appears in a transaction database [5]. In precise databases, $s(I)$ is a single value. This is no longer true in uncertain databases, because in different possible worlds, $s(I)$ can have different values. Let $S(w_j, I)$ be the support count of I in possible world w_j. Then, the probability that $s(I)$ has a value of i, denoted by $\Pr^I(i)$, is:

$$Pr^I(i) = \sum_{w_j \in \mathcal{W}, S(w_j, I) = i} Pr(w_j) \tag{1}$$

Hence, $Pr^I(i)(i = 1, \ldots, n)$ form a *probability mass function* (or *pmf*) of $s(I)$. We call Pr^I the *support pmf* (or *s-pmf*) of I. In Table 2, for instance, $Pr^{\{video\}}(2) = Pr(w_6) + Pr(w_8) = \frac{1}{6}$, since $s(I) = 2$ in possible worlds w_6 and w_8. Figure 1 shows the s-pmf of $\{video\}$.

Now, let $minsup \in (0, n]$ be an integer. An itemset I is said to be *frequent* if $s(I) \geq minsup$ [5]. For uncertain databases, the *frequentness probability* of I, denoted by $Pr_{freq}(I)$, is the probability that an itemset is frequent [39]. Notice that $Pr_{freq}(I)$ can be expressed as:

$$Pr_{freq}(I) = \sum_{i \geq minsup} Pr^I(i) \tag{2}$$

In Fig. 1, if $minsup = 1$, then $Pr_{freq}(\{video\}) = Pr^{\{video\}}(1) + Pr^{\{video\}}(2) = \frac{2}{3}$.

Using frequentness probabilities, we can determine whether an itemset is frequent. We identify two classes of *Probabilistic Frequent Itemsets* (or *PFI*) below:

- I is a *threshold-based PFI* if its frequentness probability is larger than some threshold [39]. Formally, given a real value $minprob \in (0, 1]$, I is a threshold-based PFI, if $Pr_{freq}(I) \geq minprob$. We call *minprob* the *frequentness probability threshold*.
- I is a *rank-based PFI* if its frequentness probability satisfies some ranking criteria. The *top-k* PFI, proposed in [39], belongs to this class. Given an integer $k > 0$, I is a top-k PFI if $Pr_{freq}(I)$ is at least the k-th highest, among all itemsets. We focus on top-k PFI in this paper.

Before we move on, we would like to mention the following theorem, which was discussed in [39]:

Theorem 1 Anti-Monotonicity: *Let S and I be two itemsets. If $S \subseteq I$, then $Pr_{freq}(S) \geq Pr_{freq}(I)$.*

This theorem will be used in our algorithms.

Next, we derive efficient s-pmf computation methods in Sect. 4. Based on these methods, we present algorithms for retrieving threshold-based and rank-based PFIs in Sects. 5 and 6, respectively.

4 Approximation of S-pmf

From the last section, we can see that the s-pmf $s(I)$ of itemset I plays an important role in determining whether I is a PFI. However, directly computing $s(I)$ (e.g., using the dynamic programming approaches of [39,46]) can be expensive. We now investigate alternative ways of computing $s(I)$. In the following, we study some statistical properties of $s(I)$ and show how to approximate the distribution of $s(I)$ in a computationally efficient way by means of the expected support (cf. Sect. 4.1) and two standard probability distributions: the Poisson distribution (cf. Sect. 4.2) and the normal distribution (cf. Sect. 4.3). In Sect. 4.4, we discuss all three alternatives.

An interesting observation about $s(I)$ is that it is essentially the number of successful *Poisson trials* [37]. To explain, we let X_j^I be a random variable, which is equal to one if I is a subset of the items associated with transaction t_j (i.e., $I \subseteq t_j$), or zero otherwise. Notice that $Pr(I \subseteq t_j)$ can be easily calculated in our uncertainty models:

- For *attribute uncertainty*,

$$Pr(I \subseteq t_j) = \prod_{v \in I} Pr(v \in t_j) \tag{3}$$

- For *tuple uncertainty*,

$$Pr(I \subseteq t_j) = \begin{cases} Pr(t_j) & \text{if } I \subseteq t_j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Given a database of size n, each I is associated with random variables $X_1^I, X_2^I, \ldots, X_n^I$. In both uncertainty models considered in this paper, all tuples are independent. Therefore, these n variables are independent, and they represent n Poisson trials. Moreover, $X^I = \sum_{j=1}^{n} X_j^I$ follows a Poisson binomial distribution.

Next, we observe an important relationship between X^I and $Pr^I(i)$ (i.e., the probability that the support of I is i):

$$Pr^I(i) = Pr(X^I = i) \tag{5}$$

This is simply because X^I is the number of times that I exists in the database. Hence, the s-pmf of I, that is, $Pr^I(i)$, is the pmf of X^I, a Poisson binomial distribution.

Using Eq. 5, we can rewrite Eq. 2, which computes the frequentness probability of I, as:

$$Pr_{freq}(I) = \sum_{i \geq minsup} Pr^I(i) \tag{6}$$

$$= 1 - Pr(X^I \leq minsup - 1) \tag{7}$$

Let $Pr^I_{\leq}(i)$ be the cumulative distribution function (cdf) of X^I, that is,

$$Pr^I_{\leq}(i) = \sum_{j=0..i} Pr^I(j). \tag{8}$$

Therefore, if the cumulative distribution function $Pr^I_{\leq}(i)$ of X^I is known, $Pr_{freq}(I)$ can also be evaluated efficiently:

$$Pr_{freq}(I) = 1 - \sum_{i \leq minsup-1} Pr^I(i) \tag{9}$$

$$= 1 - Pr^I_{\leq}(minsup - 1). \tag{10}$$

Next, we discuss approaches to approximate this cdf, in order to compute $Pr_{freq}(I)$ efficiently.

4.1 Approximation by expected support

A simple and efficient way to evaluate the frequentness of an itemset in an uncertain transaction database is to use the expected support [3,12]. The expected support converges to the exact support when increasing the number of transactions according to the "law of large numbers."

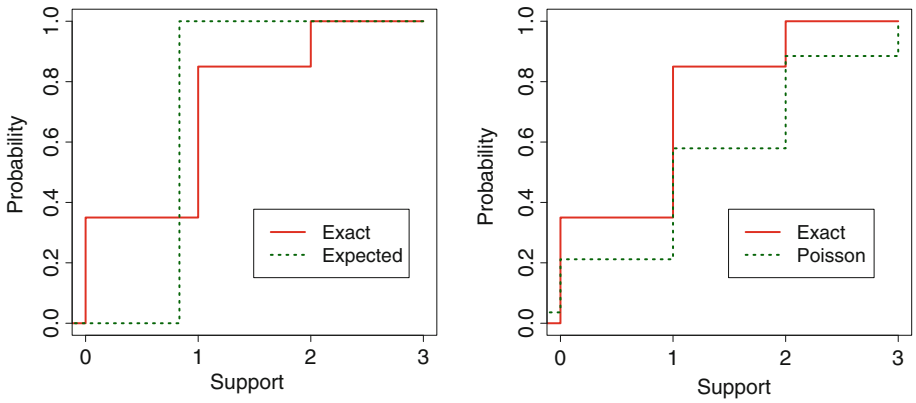
Definition 2 (*Law of Large Numbers*) A "law of large numbers" is one of several theorems expressing the idea that as the number of trials of a random process increases, the percentage difference between the expected and actual values goes to zero. Formally, given a sequence of independent and identically distributed random variables $X_1, \ldots, X_n$, the sample average $\frac{1}{n}\sum_{i=1}^{n} X_i$ converges to the expected value $\mu = \sum_{i=1}^{n} E(X_i)$ for $n \to \infty$. It can also be shown ([17]), that the law of large numbers is applicable for non-identically distributed random variables.

For notational convenience, let p_j^I be $Pr(I \subseteq t_j)$. Since the expectation of a sum is the sum of the expectations, the expected value of X^I, denoted by μ_I, can be computed by:

$$\mu_I = \sum_{j=1}^{n} p_j^I \tag{11}$$

Given the expected support μ_I, we can approximate the cdf $Pr^I_{\leq}(i)$ of X^I as follows:

$$Pr^I_{\leq}(i) = \begin{cases} 0 & \text{if } i < \mu_I \\ 1 & else \end{cases} \tag{12}$$



(a) Expected approximation **(b)** Poisson approximation

Fig. 2 Approximations of the s-pmf of PFI $\{video\}$ from Table 1

According to the above equation, the frequentness probability $Pr_{freq}(I)$ of itemset I is approximated by 1, if μ_I is at least $minsup$ and 0 otherwise. The computation of μ_I can be efficiently done by scanning D once and summing up p_j^I's for all tuples t_j in D. As an example, consider the support of itemset $\{video\}$ in Table 1. Computing $\mu_I = 0.5 + 0.33$ yields the approximated pmf depicted in Fig. 2a. Obviously, the expected support is only a very coarse approximation of the exact support distribution. Important information about the distribution, for example the variance, is lost with this approximation. In fact, we do not know how confident the results are. In the following, we provide a more accurate approximation for $Pr_{\leq}^I(i)$ by taking the (exact) distribution into consideration.

4.2 Poisson Distribution-Based Approximation

A Poisson binomial distribution can be well approximated by a Poisson distribution [8] following the "Poisson law of small numbers".

Definition 3 (*Poisson Law of Small Numbers*) Given a sequence of independent random Bernoulli variables $X_1, \ldots, X_n$, with mean $(\mu)_i$, the density of the sample average $X = \frac{1}{n}\sum_{i=1}^{n} X_i$ is approximately Poisson distributed with $\lambda_X = \frac{1}{n}\sum_{i=1}^{n}(\mu)_i$ if $\max\{P(X_1), \ldots, P(X_n)\}$ tends to zero [19].

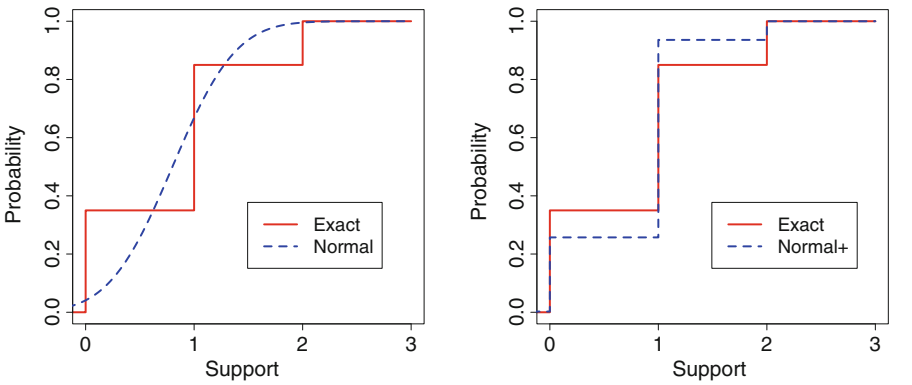
According to this law, Eq. 10 can be written as:

$$Pr_{freq}(I) \approx 1 - F_p(minsup - 1, \mu_I) \tag{13}$$

where F_p is the cdf of the Poisson distribution with mean μ_I, that is, $F_p(minsup - 1, \mu_I) = 1 - \frac{\Gamma(minsup, \mu_I)}{(minsup-1)!}$, where $\Gamma(minsup, \mu_I) = \int_{\mu_I}^{\infty} t^{minsup-1} e^{-t} dt$.

As an example, consider again the support of itemset $\{video\}$ in Table 1. Computing $\mu = \lambda^I = 0.5 + 0.33$ yields the approximated pmf depicted in Fig. 2b. A theoretical analysis of the approximation quality is shown in Appendix 8. The upper bound of the error is small. According to our experimental results, the approximation is quite accurate.

To estimate $Pr_{freq}(I)$, we can first compute μ_I by scanning D once as described above and evaluate $F_p(minsup - 1, \mu_I)$. Then, Eq. 13 is used to approximate $Pr_{freq}(I)$.



(a) Normal distribution-based approximation **(b)** Approximation with continuity correction

Fig. 3 Itemset support distribution approximated with the normal distribution. **a** Normal distribution-based approximation. **b** Approximation with continuity correction

4.3 Normal Distribution-Based Approximation

Provided $|D|$ is large enough which usually holds for transaction databases, X^I converges to the normal distribution with mean μ_I and variance σ_I^2, where

$$\sigma_I^2 = Var(X^I) = \sum_{t_j \in D} Pr(I \subseteq t_j) \cdot (1 - Pr(I \subseteq t_j)) = \sum_{j=1}^{n} p_j^I \cdot \left(1 - p_j^I\right)$$

according to the "Central Limit Theorem".

Definition 4 (*Central Limit Theorem*) Given a sequence of independent random variables $X_1, \ldots, X_n$, with mean μ_i and finite variance ${\sigma_i}^2$, the density of the sample average $X = \frac{1}{n}\sum_{i=1}^{n} X_i$ is approximately normal distributed with $\mu_X = \frac{1}{n}\sum_{i=1}^{n} \mu_i$ and ${\sigma^2}_X = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} {\sigma^2}_i$.

Lemma 1 *The support probability distribution of an itemset I is approximated by the normal distribution with mean μ_I and variance σ_I^2 as defined above. Therefore,*

$$Pr_{freq}(I) \approx 1 - F_n(minsup - 1, \mu_I, \sigma_I^2) \tag{14}$$

where F_n is the cdf of the normal distribution with mean μ_I and variance σ_I^2, that is,

$$F_n(minsup - 1, \mu_I, \sigma_I^2) = \frac{1}{\sigma_I\sqrt{2\pi}} \int_{-\infty}^{minsup-0.5} e^{-\frac{(x-\mu_I)^2}{2\sigma_I^2}} \tag{15}$$

Computing $\mu = 0.5 + 0.33$ and $sigma^2 = 0.5 \cdot 0.5 + 0.33 \cdot 0.67 = 0.471$ yields the approximated pmf depicted in Fig. 3a. The continuity correction which is achieved by running the integral up to $minsup - 0.5$ instead of $minsup - 1$ is an important and common method to compensate the fact that X^I is a discrete distribution approximated by a continuous normal distribution. The effect of the continuity correction is shown in Fig. 3b.

The estimation of $Pr_{freq}(I)$ can be done by first computing μ_I by scanning D once, summing up p_j^I's for all tuples t_j in D and using Eq. 14 to approximate $Pr_{freq}(I)$. For an efficient evaluation of $F_n(minsup-1, \mu_I)$, we use the Abromowitz and Stegun approximation [2] which is necessary because the cumulative normal distribution has no closed-form solution. The result is a very fast parametric test to evaluate the frequentness of an itemset.

While the above method still requires a full scan of the database to evaluate one frequent itemset candidate, threshold- and rank-based PFIs can be found more efficiently.

4.4 Discussion

In this section, we have described three models to approximate the Poisson binomial distribution. Now, we will discuss the advantages and disadvantages of each model theoretically, while in Sect. 7 we will experimentally support the claims made here.

Each of the approximation models is based on a fundamental statistics theorem. In particular, the *Expected* approach exploits the *Law of Large Numbers* [35], the *Normal Approximation* approach exploits the *Central Limit Theorem* [16], and the *Poisson Approximation* approach exploits the *Poisson Law of Small Numbers* [34].

Approximation based on expected support In consideration of the "Law of large numbers," the *Expected* approach requires a large n, that is, a large number of transactions, where the respective itemset is contained with a probability greater than zero. A thorough evaluation of this parameter can be found in the experiments.

Normal distribution-based approximation The rule of thumb for the central limit theorem is, is that for $n \geq 30$, that is, for at least 30 transactions containing the respective itemset with a probability larger than zero, the normal approximation yields good results. This rule of thumb, however, depends on certain circumstances, namely, the probabilities $P(X_i = 1)$ should be close to 0.5. In our experiments, we will evaluate for what settings (e.g., databases size, itemset probabilities) the normal approximation yields good results.

Poisson distribution-based approximation In consideration of the *Poisson Law of Small Numbers*, the Poisson approximation theoretically yields good results, if *all* probabilities $P(X_i)$ are small. This seems to be a harsh assumption, since it forbids any probabilities of one, which are common in real data sets. However, it can be argued that, for large itemsets, the probability may always become small in some applications. In the experiments, we will show how small $\max\{P(X_1), \ldots, P(X_n)\}$ is required to be, in order to achieve good approximations, and how "a few" large probabilities impact the approximation quality. In addition, our experiments aim to give an intuition, in what setting which approximation should be used to achieve the best approximation results.

Computational complexity Each approximation technique requires to compute the expected support $E(X) = \mu = \lambda = \sum_{i=1}^{n} P(X_i)$, which requires a full scan of the database requiring $O(n)$ time and $O(1)$ space. The normal approximation additionally requires to compute the sum of variances, which has the same complexity. This is all that has to be done to compute the parameters of all three approximations. After that, the Expected approach only requires to compare $E(X)$ with $MinSupp$, at a cost of $O(1)$ time. The normal approximation approach in contrast requires to compute the probability that $X > MinSupp$, which requires numeric integration, since the normal distribution it has no closed-form expression. However, there exist very efficient techniques (such as the Abromowitz and Stegun approximation [2]) to quickly evaluate the normal distribution. Regardless, this evaluation is independent of the database size and also requires constant time. The same rationale applies for the Poisson approximation, which does also not have a closed-form solution, but for which there exist

manifold fast approximation techniques. In summary, each of the approximation techniques has a total runtime complexity of $O(n + C_i)$ and a space complexity of $O(1)$. The constant C_i depends on the approximation technique. In the experiments, we will see that the impact of C_i can be neglected in runtime experiments. In summary, each of the proposed approximation techniques runs in O(n) time. These are more scalable methods compared to solutions in [39,46], which evaluate $Pr_{freq}(I)$ in $O(n^2)$ time.

5 Threshold-based PFI mining

Can we quickly determine whether an itemset I is a threshold-based PFI? Answering this question is crucial, since in typical PFI mining algorithms (e.g., [39]), candidate itemsets are first generated, before they are tested on whether they are PFI's. In Sect. 5.1, we develop a simple method of testing whether I is a threshold-based PFI, without computing its frequentness probability. This method is applicable for any s-pmf approximation proposed in Sect. 4. We then enhance this method in Sect. 5.2 by adding pruning techniques that allow to decide whether an itemset must (not) be frequent based on a subset of the database only. Finally, we demonstrate an adaptation of these techniques in an existing PFI mining algorithm, in Sect. 5.3.

5.1 PFI Testing

Given the values of $minsup$ and $minprob$, we need to test whether I is a threshold-based PFI. Therefore, we first need the following definition:

Definition 5 (p-$value$) Let X be a random variable with pmf pmf_X on the domain $\mathcal{R}$. The p-$value$ of X at x denotes the probability that X takes a value less than or equal to x. Formally,

$$p\text{-}value(x, X) = \int_{-\infty}^{x} pmf_X(x).$$

Given the cumulative mass function cmf_X of X, this translates into

$$p\text{-}value(x, X) = cmf_X(x)$$

The computation of the p-value is an inherent method in any statistical program package. For instance, in the statistical computing package R, the p-value of a value x of a normal distribution with mean μ and variance σ^2 is obtained using the function $pnorm(x, \mu, \sigma^2)$. For the Poisson distribution, the function $ppois(x, \mu)$ is used. For the expected approximation, simply return 1 is $\mu > minsup$ and 0 otherwise.

Since $p\text{-}value(minsup, support(I))$ corresponds to the probability that the support of itemset I is equal or less than $minsup$, we can derive the probability $Pr_{freq}(I)$ that the support is at least $minsup$ simply as follows:

$$Pr_{freq}(I) = 1 - (p\text{-}value(minsup, support(I)) + P(support(I) = minsup))$$

which is approximated by

$$1 - (p\text{-}value(minsup - \epsilon, support(I)))$$

where $P(support(I) = minsup)$ is the probability that the support of I is exactly minsup, and ϵ is a very small number (e.g., $\epsilon = 10^{\{-10\}}$).

Now, to decide whether I is a threshold-based PFI, given the values of $minsup$ and $minprob$ we apply the following three steps:

- Compute the parameters μ_I (and σ_I^2 for the normal case).
- Derive the probability $Pr_{freq}(I) = 1 - (p\text{-}value(minsup - \epsilon, support(I))$ given $minsup$ and the respective approximation model support(I).
- if $Pr_{freq}(I) > minprob$, return I as a frequent itemset, otherwise, conclude that I is not a frequent itemset.

Example 1 Consider again the example given in Table 1. Assume that we want to decide whether itemset $\{video\}$ has a support of at least $minsup = 1$ with a probability of at least 60 %. Assume that we want to use the normal approximation model. Therefore, in the first step, we compute the approximation parameters $\mu_{\{video\}} = 0.5+0.33 = 0.83$ and $\sigma^2_{\{video\}} = 0.5 \cdot 0.5+0.33 \cdot 0.67 = 0.47$. Next, we compute $p\text{-}value(minsup, support(\{video\}))$, which corresponds to evaluating the dotted cdf in Fig. 3 at $minsup = 1$, which yields a probability of about 90 %. Since this value is greater than $minprob = 60\%$, we are able to conclude that itemset $\{video\}$ must be frequent for $minsup = 1$ and $minprob = 60\%$.

In the following, we show pruning techniques which are applicable for the expected support model as well as the Poisson approximation model. For the normal approximation, we give a counter-example, why the proposed pruning strategy may yield wrong results.

5.2 Improving the PFI testing process for the poisson approximation

In Step 1 of the last section, D has to be scanned once to obtain μ_I and σ_I^2, for every itemset I. This can be costly if D is large, and if many itemsets need to be tested. For example, in the Apriori algorithm [39], many candidate itemsets are generated first before testing whether they are PFIs. In the following, we will define a monotonicity criterion which can be used to decide if I is frequent by only scanning a subset of the database. In the following, we will show that for the model using expected support and for the Poisson approximation model, the above lemma holds, that is, $(Pr_{freq}(I))_i$ increases monotonically in i. For the normal approximation, however, we will show a counter-example to illustrate that the above property does not hold.

Definition 6 (**n-Monotonicity**) Let $i \in (0, n]$. Let $(X_I)_i$ be an approximation model of the support of itemset I, based on the first i transactions, that is, on the parameters $(\mu_I)_i$ and $(\sigma_I^2)_i$. Also, let $(Pr_{freq}(I))_i$ denote the probability that I has a support of at least minsup, given these parameters. If $(Pr_{freq}(I))_i \geq minprob$, then I is a threshold-based PFI.

Lemma 2 *Property 6 holds for the approximation model using the expected support.*

Proof Let $(X_I)_i$ denote the approximation of the smf of itemset I using expected support. Therefore, $(Pr_{freq}(I))_i$ equals to 1, if $(\mu_I)_i > minsup$ and to 0 otherwise. Now consider the

probability $(Pr_{freq}(I))_{i+1}$. Trivially, if $(Pr_{freq}(I))_i = 0$, then it holds that $(Pr_{freq}(I))_i < (Pr_{freq}(I))_{i+1}$. Otherwise, it holds that $(Pr_{freq}(I))_i = 1$ which implies that $(\mu_I)_i > minsup$. Since

$$(\mu_I)_j = (\mu_I)_i + p_j \geq (\mu_I)_i > minsup$$

it holds that $(Pr_{freq}(I))_{i+1} = 1$.

Thus in all cases $p\text{-}value(minsup, (X_I)_i)$ is non decreasing in i. □

Lemma 3 *Property 6 holds for the Poisson approximation model.*

Proof In the previous proof, we have exploited that $(\mu_I)_i$ increases monotonically in i. To show that $(Pr_{freq}(I))_i$ increases monotonically in i if the Poisson approximation is used, we only have to show the following □

Theorem 2 *$Pr_{freq}(I)$, if approximated by Eq. 13, increases monotonically with μ_I.*

The cdf of a Poisson distribution, $F_p(i, \mu)$, can be written as: $F_p(i, \mu) = \frac{\Gamma(i+1,\mu)}{i!} = \frac{\int_\mu^\infty t^{(i+1)-1}e^{-t}dt}{i!}$

Since $minsup$ is fixed and independent of μ, let us examine the partial derivative w.r.t. μ.

$$\begin{aligned}
\frac{\partial F_p(i,\mu)}{\partial \mu} &= \frac{\partial}{\partial \mu}\left(\frac{\int_\mu^\infty t^{(i+1)-1}e^{-t}\mathrm{d}t}{i!}\right) \\
&= \frac{1}{i!}\frac{\partial}{\partial \mu}\left(\int_\mu^\infty t^i e^{-t}\mathrm{d}t\right) \\
&= \frac{1}{i!}(-\mu_i e^{-\mu}) \\
&= -f_p(i,\mu) \leq 0
\end{aligned}$$

Thus, the cdf of the Poisson distribution $F_p(i, \mu)$ is monotonically decreasing w.r.t. μ, when i is fixed. Consequently, $1 - F_p(i - 1, \mu)$ increases monotonically with μ. Theorem 2 follows immediately by substituting $i = minsup$.

Note that intuitively, Theorem 2 states that the higher value of μ_I, the higher is the chance that I is a PFI.

Lemma 4 *Property 6 does* **not** *hold for the normal approximation model.*

Proof For the normal approximation, we give a counter-example, which illustrates that Property 6 does not hold. □

Example 2 Assume that itemset I is contained in transactions t_1 and t_2 with probabilities of 1 and 0.5, respectively. Assume that $minsup = 1$. Since $p_1 = P(I \in t_1) = 1$, we get $(\mu_I)_1 = 1$ and $(\sigma_I^2)_1 = 1 \cdot 0 = 0$. Computing $(Pr_{freq}(I))_1$ yields 1, since the normal pmf with parameters $\mu = 1$ and $\sigma^2 = 0$ is 1 with a probability of 1 and thus is always greater or equal than $minsup = 1$. Now, consider $(Pr_{freq}(I))_2$: To evaluate this, we use a normal distribution with parameters $(\mu_I)_2 = 1+0.5 = 1.5$ and $(\sigma_I^2)_2 = 1 \cdot 0 = 0+0.5 \cdot 0.5 = 0.25$. Since the normal pmf is unbounded for $\sigma^2 \neq 0$, it is clear that $(Pr_{freq}(I))_2 < 1$, since $p\text{-}value(1, Normal(1.5, 0.25)) > 0$.

Property 6 leads to the following:

Lemma 5 *Let $i \in (0, n]$ and assume that $(Pr_{freq}(I))_i$ is approximated using the expected support model or using Poisson approximation. Then, it holds that if $(Pr_{freq}(I))_i > minprob$ then $Pr_{freq}(I) > minprob$ and I can be returned as an approximate result.*

Proof Notice that $(\mu_I)_i$ monotonically increases with i. If there exists a value of i such that $(\mu_I)_i \geq (\mu)_m$, we must have $\mu_I = (\mu_I)_n \geq (\mu_I)_i \geq (\mu)_m$, implying that I is a PFI. □

Using Lemma 5, a PFI can be verified by scanning only a part of the database.

True hits Let $(\mu_I)_l = \sum_{j=1}^{l} p_j$ and $(\sigma_I^2)_l = \sum_{j=1}^{l} p_j \cdot (1 - p_j)$, where $l \in (0, n]$. Essentially, $(\mu_I)_l$ ($(\sigma_I^2)_l$) is the "partial value" of μ_I (σ_I^2), which is obtained after scanning l tuples. Notice that $(\mu_I)_n = \mu_I$ and $(\sigma_I^2)_n = \sigma_I^2$.

To perform pruning based on the values of $(\mu_I)_l$ and $(\sigma_I^2)_l$, we first define the following property:

Avoiding integration We next show the following. To perform the true hit detection as proposed above, we require to evaluate $(Pr_{freq}(I))_i$ for each $i \in 1, \ldots, n$. In the case of the expected support approximation, this only requires to check if $(\mu_I)_i > minsup$. However, for the Poisson approximation, this requires to evaluate a poisson distribution at minsup. Since the Poisson distribution has no closed-form solution, this requires a large number of numeric integrations, which can be, depending on their accuracy, computationally very expensive. In the following, we show how to perform pruning using the Poisson approximation model, while avoiding the integrations.[2]

Given the values of $minsup$ and $minprob$, we can test whether $(Pr_{freq}(I))_i > minprob$ as follows:

Step 1. Find a real number $(\mu)_m$ satisfying the equation:

$$minprob = 1 - F_{p/n}(minsup - 1, (\mu)_m), \tag{16}$$

where $F_{p/n}$ denotes the cdf of the Poisson distribution. The above equation can be solved efficiently by employing numerical methods, thanks to Theorem 2. This computation has to be performed only once.

Step 2. Use Eq. 11 to compute $(\mu_I)_i$.

Step 3. If $(\mu_I)_i \geq (\mu)_m$, we conclude that I is a PFI. Otherwise, I must not be a PFI.

To understand why this works, first notice that the right side of Eq. 16 is the same as that of Eqs. 13 or 14, an expression of frequentness probability. Essentially, Step 1 finds out the value of $(\mu)_m$ that corresponds to the frequentness probability threshold (i.e., *minprob*). In Steps 2 and 3, if $\mu_I \geq (\mu)_m$, Theorem 2 allows us to deduce that $Pr_{freq}(I) > minprob$. Hence, these steps together can test whether an itemset is a PFI.

In order to verify whether I is a PFI, once $(\mu)_m$ is found, we do not have to evaluate $(Pr_{freq}(I))_i$. Instead, we compute $(\mu_I)_i$ in Step 2, which can be done in $O(1)$ time using $(\mu_I)_{i-1}$. In the next paragraph, we show how the observation made in this paragraph can be used to efficiently detect true drops, that is, itemsets for which we can decide that $\mu_I < (\mu)_m$ by only considering $(\mu_I)_i$.

[2] Recall that true hit detection is not applicable for normal approximation, which is the reason why normal approximation is omitted in this paragraph.

True Drops

Lemma 6 *If I is a threshold-based PFI, then:*

$$(mu_I)_{n-i} \geq (\mu)_m - i \; \forall i \in (0, \lfloor(\mu)_m\rfloor] \tag{17}$$

Proof Let D_l be a set of tuples $\{t_1, \ldots, t_l\}$. Then,

$$\mu_I = \sum_{j=1}^{n} Pr(I \subseteq t_j); (\mu_I)_i = \sum_{j=1}^{i} Pr(I \subseteq t_j)$$

Since $Pr(I \subseteq t_j) \in [0, 1]$, based on the above equations, we have:

$$i \geq \mu_I - (\mu_I)_{n-i} \tag{18}$$

If itemset I is a PFI, then $\mu_I \geq (\mu)_m$. In addition, $(\mu_I)_{n-i} \geq 0$. Therefore,

$$\begin{aligned} & i \geq \mu_I - (\mu_I)_{n-i} \geq (\mu)_m - (\mu_I)_{n-i} \; for \; 0 < i \leq \lfloor(\mu)_m\rfloor \\ \Rightarrow \quad & (\mu_I)_{n-i} \geq (\mu)_m - i \text{ for } 0 < i \leq \lfloor(\mu)_m\rfloor \end{aligned}$$

□

This lemma leads to the following corollary.

Corollary 1 *An itemset I cannot be a PFI if there exists $i \in (0, \lfloor(\mu)_m\rfloor]$ such that:*

$$(\mu_I)_{n-i} < (\mu)_m - i \tag{19}$$

We use an example to illustrate Corollary 1. Suppose that $(\mu)_m = 1.1$ for the database in Table 1. Also, let $I = \{clothing, video\}$. Using Corollary 1, we do not have to scan the whole database. Instead, only the tuple t_{Jack} needs to be read. This is because:

$$(\mu_I)_1 = 0 < 1.1 - 1 = 0.1 \tag{20}$$

Since Eq. 19 is satisfied, we confirm that I is not a PFI without scanning the whole database.

We use the above results to improve the speed of the PFI testing process. Specifically, after a tuple has been scanned, we check whether Lemma 5 is satisfied; if so, we immediately conclude that I is a PFI. After scanning $n - \lfloor(\mu)_m\rfloor$ or more tuples, we examine whether I is not a PFI, by using Corollary 1. These testing procedures continue until the whole database is scanned, yielding μ_I. Then, we execute Step 3 (Sect. 5.1) to test whether I is a PFI.

5.3 Case study: the Apriori algorithm

The testing techniques just mentioned are not associated with any specific threshold-based PFI mining algorithms. Moreover, these methods support both attribute and tuple uncertainty models. Hence, they can be easily adopted by existing algorithms. We now explain how to incorporate our techniques to enhance the Apriori [39] algorithm, an important PFI mining algorithms.

The resulting procedures (Algorithm 1 for Poisson approximation and expected support, and Algorithm 2 for normal approximation) use the "bottom-up" framework of the Apriori: starting from $k = 1$, size-k PFIs (called k-PFIs) are first generated. Then, using Theorem 1, size-$(k + 1)$ candidate itemsets are derived from the k-PFIs, based on which the k-PFIs are found. The process goes on with larger k, until no larger candidate itemsets can be discovered.

Algorithm 1: Apriori-based PFI Mining using the Poisson Approximation or Expected Support

Input: Uncertain database D, $minsup$, $minprob$
Output: All PFI: $F = \{F_1, F_2, \ldots, F_m\}$ // F_k is set of k-PFIs

```
1  begin
2    (μ)_m = MinExpSup(minsup, minprob);
3    (μ)_m = minsup; // Expected only
4    C_1.GenerateSingleItemCandidates(D);
5    k = 1; j = 0;
6    while |C_k| != 0 do
7      for each I ∈ C_k do
8        I.μ = 0;
9      while (++j) ≤ n and |C_k| != 0 do
10       for each I ∈ C_k do
11         I.μ += Pr(I ⊆ t_j);
12         if I.μ ≥ (μ)^m then
13           F_k.push(I);
14           C_k.remove(I);
15         else if j ≥ n − ⌊(μ)_m⌋ then
16           if Pruning(I, (μ)_m, j, n) == true then
17             C_k.remove(I);
18
19     C_{k+1}.GenerateCandidates(F_k);
20     k += 1; j = 0
21   return F;
```

The main difference of Algorithm 1 and Algorithm 2 compared with that of Apriori [39] is that all steps that require frequentness probability computation are replaced by our PFI testing methods. In particular, Algorithm 1 first computes $(\mu)_m$ (Line 2–3) depending on whether the expected support model or the Poisson approximation model is used.

Algorithm 2: Apriori-based PFI Mining using the Normal Approximation

Input: Uncertain database D, $minsup$, $minprob$
Output: All PFI: $F = \{F_1, F_2, \ldots, F_m\}$ // F_k is set of k-PFIs

```
1  begin
2    C_1.GenerateSingleItemCandidates(D);
3    k = 1; j = 0;
4    while |C_k| != 0 do
5      for each I ∈ C_k do
6        I.μ = 0;
7      while (++j) ≤ n and |C_k| != 0 do
8        for each I ∈ C_k do
9          I.μ += Pr(I ⊆ t_j);
10         I.σ^2 += Pr(I ⊆ t_j) · (1 − Pr(I ⊆ t_j))
11       pvalue = F_n(minsup − 1, μ, σ^2) if 1 − pvalue > minprob then
12         F_k.push(I);
13         C_k.remove(I);
14     C_{k+1}.GenerateCandidates(F_k);
15     k += 1; j = 0
16   return F;
```

Then, for each candidate itemset I generated on Line 4 and Line 17, we scan D and compute its $(\mu_I)_i$ (Line 10) and $(\sigma_I^2)_i$. Unless the normal approximation is used, pruning can now be performed: If Lemma 5 is satisfied, then I is put to the result (Lines 11–13). However, if Corollary 1 is satisfied, I is pruned from the candidate itemsets (Lines 14–16). This process goes on until no more candidates itemsets are found.

Complexity. In Algorithm 1, each candidate item needs $O(n)$ time to test whether it is a PFI. This is much faster than the Apriori [39], which verifies a PFI in $O(n^2)$ time. Moreover, since D is scanned once for all k-PFI candidates C_k, at most a total of n tuples is retrieved for each C_k (instead of $|C_k| \cdot n$). The space complexity is $O(|C_k|)$ for each candidate set C_k, in order to maintain μ_I for each candidate.

6 Rank-based PFI mining

Besides mining threshold-based PFIs, the s-pmf approximation methods presented in Section 4 can also facilitate the discovery of rank-based PFIs (i.e., PFIs returned based on their relative rankings). In this section, we investigate an adaptation of our methods for finding top-k PFIs, a kind of rank-based PFIs which orders PFIs according to their frequentness probabilities.

Our solution (Algorithm 3) follows the framework in [39]: A bounded priority queue, Q, is maintained to store candidate itemsets that can be top-k PFIs, in descending order of their frequentness probabilities. Initially, Q has a capacity of k itemsets, and single items with the k highest probabilities are deposited into it. Then, the algorithm iterates itself k times. During each iteration, the first itemset I is popped from Q, which has the highest frequentness probability among those in Q. Based on Theorem 1, I must also rank higher than itemsets that are supersets of I. Hence, I is one of the top-k PFIs. A *generation step* is then performed, which produces candidate itemsets based on I. Next, a *testing step* is carried out, to determine which of these itemsets should be put to Q, with Q's capacity decremented. The top-k PFIs are produced after k iterations.

Algorithm 3: top-k PFI Mining

Input: Uncertain database D, $minsup$, k
Output: Top-k PFI Set: T

```
1  begin
2      C.GenerateSingleItemCandidates(D);
3      C.TestCandidates(D);
4      Q.size = k;
5      Q.UpdateQueue(C);
6      while T.size < k do
7          I = Q.pop();
8          T.push(I);
9          C.GenerateCandidates(I, Q);
10         C.TestCandidates(D);
11         Q.size = Q.size − 1;
12         Q.UpdateQueue(C);
13     return T
```

We now explain how the generation and testing steps in [39] can be redesigned, in Sects. 6.1 and 6.2, respectively.

6.1 Candidate itemset generation

Given an itemset I retrieved from Q, in [39] a candidate itemset is generated from I by unioning I with every single item in V. Hence, the maximum number of candidate itemsets generated is $|V|$, which is the number of single items.

We argue that the number of candidate itemsets can actually be fewer, if the contents of Q are also considered. To illustrate this, suppose Q = *{{abc}, {bcd}, {efg}}*, and I=*{abc}* has the highest frequentness probability. If the generation step of [39] is used, then four candidates are generated for I: *{{abcd}, {abce}, {abcf}, {abcg}}*). However, $Pr_{freq}(\{abce\}) \leq Pr_{freq}(\{bce\})$, according to Theorem 1. Since $\{bce\}$ is not in Q, $\{bce\}$ must also be not a top-k PFI. Using Theorem 1, we can immediately conclude that $\{abce\}$, a superset of $\{bce\}$, cannot be a top-k PFI. Hence, $\{abce\}$ cannot be a top-k PFI candidate. Using similar arguments, $\{abcf\}$ and $\{abcg\}$ do not need to be generated, either. In this example, only $\{abcd\}$ should be a top-k PFI candidate.

Algorithm. Based on this observation, we redesign the generation step (Line 9 of Algorithm 3) as follows: for any itemsets $I' \in Q$, if I and I' contain the same number of items, and there is only one item that differentiates I from I', we generate a candidate itemset $I \cup I'$. This guarantees that $I \cup I'$ contains items from both I and I'.

Since Q has at most k itemsets, the new algorithm generates at most k candidates. In practice, the number of single items ($|V|$) is large compared with k. For example, in the *accidents* Dataset used in our experiments, $|V| = 572$. Hence, our algorithm generates fewer candidates and significantly improves the performance of the solution in [39], as shown in our experimental results.

6.2 Candidate itemset testing

After the set of candidate itemsets, C, is generated, [39] performs testing on them by first computing their exact frequentness probabilities, then comparing with the minimal frequentness probability, Pr_{min}, for all itemsets in Q. Only those in C with probabilities larger than Pr_{min} are added to Q. As explained before, computing frequentness probabilities can be costly. Here, we propose a faster solution based on the results in Sect. 4. The main idea is that instead of ranking the itemsets in Q based on their frequentness probabilities, we order them according to their μ_I values. Also, for each $I' \in C$, instead of computing $Pr_{freq}(I')$, we evaluate $\mu_{I'}$. These values are then compared with $(\mu)_{min}$, the minimum value of μ_I among the itemsets stored in Q. Only candidates whose $\mu_{I'}$s are not smaller than $(\mu)_{min}$ are placed into Q. An itemset I' selected in this way has a good chance to satisfy $Pr_{freq}(I') \geq Pr_{min}$, according to Theorem 2, as well as being a top-k PFI. Hence, by comparing the μ_I values, we avoid computing any frequentness probability values.

Complexity. The details of the above discussions are shown in Algorithm 3. As we can see, $\mu_{I'}$, computed for every itemset $I' \in C$, is used to determine whether I' should be put into Q. During each iteration, the time complexity is $O(n + k)$. The overall complexity of algorithm is $O(kn + k^2)$. This is generally faster than the solution in [39], whose complexity is $O(kn^2 + k|V|)$.

We conclude this section with an example. Suppose we wish to find top-2 PFIs from the dataset in Table 1. There are four single items, and the initial capacity of Q is $k = 2$. Suppose that after computing the μ_Is of single items, $\{food\}$ and $\{clothing\}$ have the highest values. Hence, they are put to Q (Table 5). In the first iteration, $\{food\}$, the PFI with the highest μ_I is returned as the top-1 PFI. Based on our generation step, $\{food, clothing\}$ is the only candidate. However, $\mu_{\{food,\ clothing\}}$ is smaller than $(\mu)_{min}$, which corresponds to the

Table 5 Mining Top-2 PFIs

Iteration	Answer	Priority queue (Q)
0	$\emptyset$	{ {*food*},{*clothing*}}
1	{*food*}	{{*clothing*}}
2	{*clothing*}	$\emptyset$

minimum μ_I among itemsets in Q. Hence, {*food, clothing*} is not put to Q. Moreover, the capacity of Q is reduced to one. Finally, {*clothing*} is popped from Q and returned.

7 Results

We now present the experimental results on two datasets that have been used in recent related work, for example [6,25,43]. The first one, called *accidents*, comes from the Frequent Itemset Mining (FIMI) Dataset Repository.[3] This dataset is obtained from the National Institute of Statistics (NIS) for the region of Flanders (Belgium), for the period of 1991–2000. The data are obtained from the "Belgian Analysis Form for Traffic Accidents", which are filled out by a police officer for each traffic accident occurring on a public road in Belgium. The dataset contains 340,184 accident records, with a total of 572 attribute values. On average, each record has 45 attributes.

We use the first 10k tuples and the first 20 attributes as our default dataset. The default value of $minsup$ is 20% of the database size n.

The second dataset, called *T10I4D100k*, is produced by the IBM data generator.[4] The dataset has a size n of 100k transactions. On average, each transaction has 10 items, and a frequent itemset has four items. Since this dataset is relatively sparse, we set $minsup$ to 1% of n.

For both datasets, we consider both attribute and tuple uncertainty models. For attribute uncertainty, the existential probability of each attribute is drawn from a Gaussian distribution with mean 0.5 and standard deviation 0.125. This same distribution is also used to characterize the existential probability of each tuple, for the tuple uncertainty model. The default values of *minprob* and k are 0.4 and 10, respectively. In the results presented, $minsup$ is shown as a percentage of the dataset size n. Notice that when the values of *minsup* or *minprob* are large, no PFIs can be returned; we do not show the results for these values. Our experiments were carried out on the Windows XP operating system, on a work station with a 2.66-GHz Intel Core 2 Duo processor and 2GB memory. The programs were written in C and compiled with Microsoft Visual Studio 2008.

We first present the results on the real dataset. Sections 7.1 and 7.2 describe the results for mining threshold- and rank-based PFIs, on attribute-uncertain data. We summarize the results for tuple-uncertain data and synthetic data, in Section 7.3.

7.1 Results on threshold-based PFI mining

We now compare the performance of five threshold-based PFI mining algorithms mentioned in this paper: (1) `DP`, the Apriori algorithm used in [39]; (2) `Expected`, the modified

[3] http://fimi.cs.helsinki.fi/.

[4] http://www.almaden.ibm.com/cs/disciplines/iis/.

Table 6 Recall and precision of the approximations

minsup/*n*	0.1	0.2	0.3	0.4	0.5
(a) `Expected` approach.					
Recall	0.99	0.98	1	1	1
Precision	1	1	1	1	1
			Recall & precision vs. *minsup*		
minprob	0.1	0.3	0.5	0.7	0.9
Recall	0.975	0.975	1	1	1
Precision	1	1	1	0.975	0.941
			Recall & precision vs. *minprob*		
n	1k	5k	10k	50k	100k
Recall	0.937	0.986	0.983	1	1
Precision	0.969	1	0.992	1	1
			Recall & precision vs. *n*		
(b) `Normal` approach.					
Recall	1	1	1	1	1
Precision	1	1	1	1	1
			Recall & precision vs. *minsup*		
minprob	0.1	0.3	0.5	0.7	0.9
Recall	1	1	1	1	1
Precision	1	1	1	1	1
			Recall & precision vs. *minprob*		
n	1k	5k	10k	50k	100k
Recall	1	1	1	1	1
Precision	1	1	1	1	1
			Recall & precision vs. *n*		
(c) `Poisson` approach.					
Recall	1	1	1	1	1
Precision	1	0.992	1	1	1
			Recall & precision vs.*minsup*		
minprob	0.1	0.3	0.5	0.7	0.9
Recall	1	1	1	0.983	0.985
Precision	0.986	1	0.985	1	1
			Recall & precision vs. *minprob*		
n	1k	5k	10k	50k	100k
Recall	1	1	1	1	1
Precision	0.989	1	0.992	1	1
			Recall & precision vs. *n*		

Apriori algorithm that uses the expected support only [12]; (3) `Poisson`, the modified Apriori algorithm that uses the Poisson approximation employing the PFI testing method (Sect. 5.1); (4) `Normal`, the modified Apriori algorithm that uses the normal approximation, also employing the PFI testing method (Sect. 5.1); and (5) `MBP`, the algorithm that uses the Poisson approximation utilizing the improved version of the PFI testing method (Sect. 5.2).

(i) Accuracy. Since the model-based approaches `Expected`, `Poisson` and `Normal` each approximate the exact s-pmf, we first examine their respective accuracy with respect to `DP`, which yields PFIs based on exact frequentness probabilities. Here, we use the standard *recall* and *precision* measures [7], which quantify the number of negatives and false positives. Specifically, let `MB`$\in$ {`Expected`, `Poisson`, `Normal`} be one of the model-based approximation approaches, and let F_{DP} be the set of PFIs generated by `DP` and F_{MB} be the set of PFIs produced by `MB`. Then, the recall and the precision of `MB`, relative to `DP`, can be defined as follows:

$$recall = \frac{|F_{DP} \cap F_{MB}|}{|F_{DP}|} \tag{21}$$

$$precision = \frac{|F_{DP} \cap F_{MB}|}{|F_{MB}|} \tag{22}$$

In these formulas, both recall and precision have values between 0 and 1. Also, a higher value reflects a better accuracy.

Table 6(a) to (c) shows the recall and the precision of the `MB` approaches, for a wide range of $minsup$, n, and $minprob$ values. As we can see, the precision and recall values are generally very high. Hence, the PFIs returned by the `MB` approaches are highly similar to those returned by `DP`. In particular, we see that the `Expected` approach generally yields the worst results, having precision and recall values of less than 95 % in some setting. The `Poisson` approach performs significantly better in these experiments. Yet in some settings, the `Poisson` approach reports false hits, while in other settings, it performs false dismissals. The `Normal` approach is most notable in this experiment. In this set of experiments, the `Normal` approach never had a false dismissal, nor did it report a single false hit. This observation also remained true for further experiments in this line, which are not depicted here. In the next set of experiments, we will experimentally investigate the effectivity of the `MB` approach.

Figure 4 shows the exact pmf, as well as the pmfs approximated by the `Poisson` and the `Normal` approach, for a variety of settings. In particular, we scaled both the dataset size and the size of the itemsets whose pmf is to be approximated. Since, in this setting, the probabilities of individual items are uniformly sampled in the [0, 1] interval, and since due to the assumption of item independence, it holds that $P(I) = \prod_{i \in I} P(i)$, a large itemset I implies smaller existence probabilities. In Fig. 4a, it can be seen that for single itemsets (i.e., large probability values), the pmf acquired by the Poisson approximation is too shallow—that is, for support values close to μ_I the exact pmf is underestimated, while for support values far from μ_I, the pmf is overestimated. In Fig. 4d, g it can be observed that this situation does not improve for the Poisson approximation. However, this shortcoming of the Poisson approximation can be explained: Since the Poisson approximation does only have one parameter μ_I, but no variance parameter ${\sigma^2}_I$, it is not able to differ between a set of five transactions with occurrence probabilities of 1, and five million transactions with occurrence probabilities of 10^{-6}, since in both scenarios it holds that $\mu_I = 5 \cdot 1 = 5 \cdot 10^6 \cdot 10^{-6} = 5$. Clearly, the variance is much greater in the later case, and so are the tails of the corresponding exact pmf. Since the Poisson distribution is the distribution of the low probabilities, the Poisson assumes the later case, i.e., a case of very many transactions, each having a very low probability. In contrast, the normal distribution is able to adjust to these different situations by adjusting its ${\sigma^2}_I$ parameter accordingly. Figure 4b, e, h show the same experiments for itemsets consisting of two items, that is, for lower probabilities $I \subseteq t_i$. It can be observed that with lower probabilities, the error of the Poisson approximation decreases,

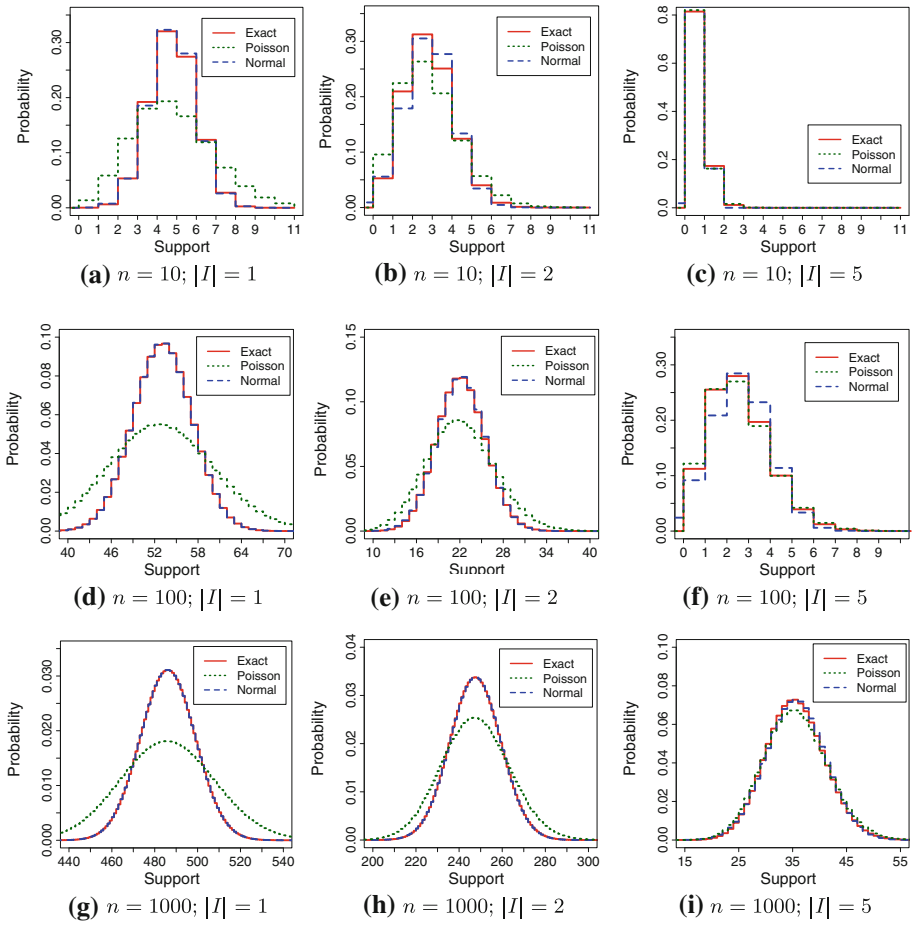
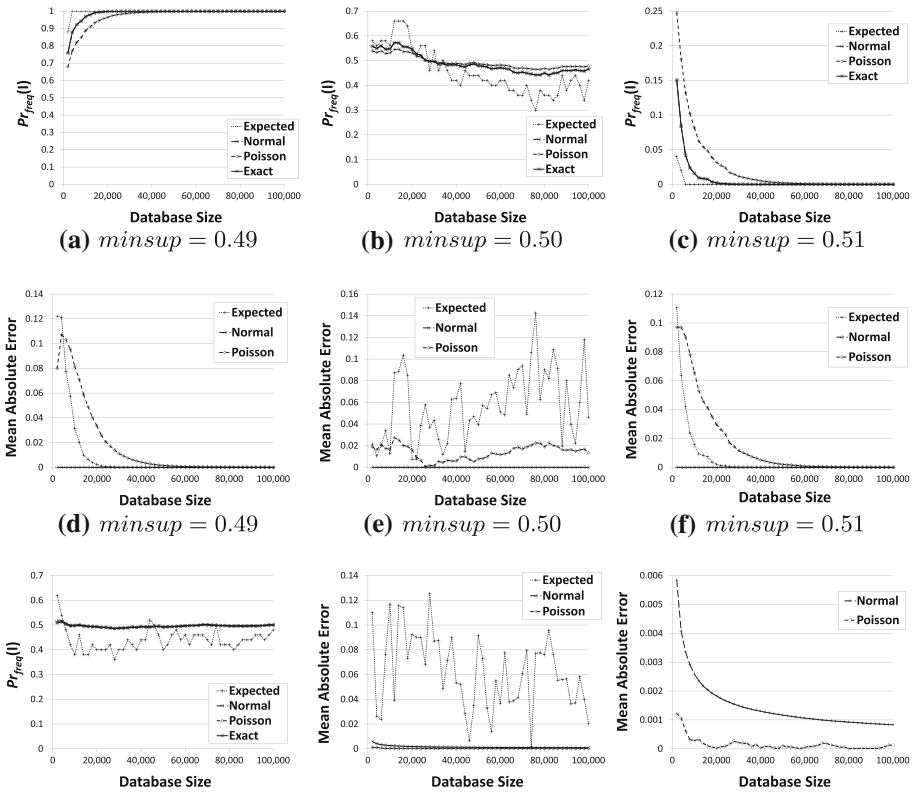


Fig. 4 Illustration of the approximation quality of `Normal` and `Poisson` for various settings. **a** $n = 10$; $|I| = 1$. **b** $n = 10$; $|I| = 2$. **c** $n = 10$; $|I| = 5$. **d** $n = 100$; $|I| = 1$. **e** $n = 100$; $|I| = 2$. **f** $n = 100$; $|I| = 5$. **g** $n = 1,000$; $|I| = 1$. **h** $n = 1,000$; $|I| = 2$. **i** $n = 1,000$; $|I| = 5$

Table 7

		# Itemsets		
n	Model	1	2	5
10	Normal	0.020	0.078	0.180
	Poisson	0.526	0.283	0.077
100	Normal	0.0003	0.0020	0.0134
	Poisson	0.0515	0.0283	0.0052
1,000	Normal	2.39E-6	6.12E-6	0.0004
	Poisson	5.24E-3	2.85E-3	0.0007

while (as we will see later, in Table 7) the quality of the normal approximation decreases. Finally, for itemsets of size 5, the Poisson approximation comes very close to the exact pmf.



(a) $minsup = 0.49$ **(b)** $minsup = 0.50$ **(c)** $minsup = 0.51$

(d) $minsup = 0.49$ **(e)** $minsup = 0.50$ **(f)** $minsup = 0.51$

(g) p-value, $minsup = 0.50$ **(h)** error, $minsup = 0.50$ **(i)** error, $minsup = 0.50$

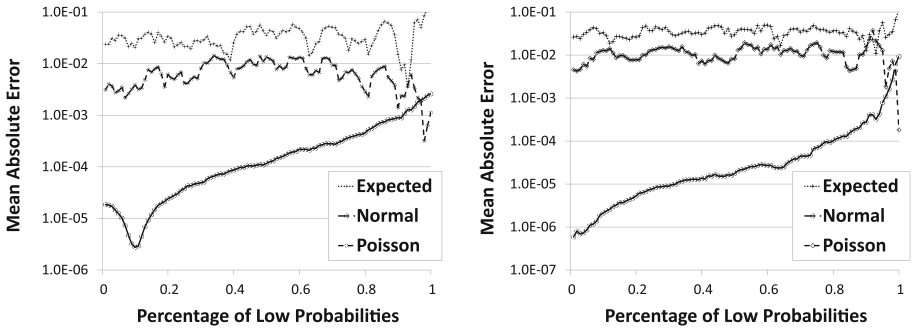
Fig. 5 Accuracy of model-based algorithms versus n. **a** $minsup = 0.49$. **b** $minsup = 0.50$. **c** $minsup = 0.51$. **d** $minsup = 0.49$. **e** $minsup = 0.50$. **f** $minsup = 0.51$. **g** p-value, $minsup = 0.50$. **h** error, minsup=0.50.**i** $error$, $minsup = 0.50$

To gain a better intuition of the approximation errors, we have repeated each of the experiments shown in Fig. 4 one hundred times and measured the total approximation error. That is, we have measured for each approximation $\texttt{DP} \in \{\texttt{Normal}, \texttt{Poisson}\}$ the distance to the exact pmf, computed by the DP approach:

$$D(MP, DP) = \sum_{i=0}^{n} |MP_{pmf} - DP_{pmf}|.$$

The results of this experiment are shown in Table 7. Clearly, the approximation quality of the Normal approach decreases when the size of the itemsets increases (i.e., the probabilities become smaller), while the Poisson approach actually improves. An increase in the database size, is beneficial for both approximation approaches.

The impact of increasing the database size on the individual approximation models has been investigated further in Fig. 5. In this experiment, we use a synthetic itemset where each item is given a random probability uniformly distributed in [0, 1]. In Fig. 5a–c, the average frequentness probability $Pr_{freq}(I)$ that item I is frequent is depicted for all itemsets I consisting of one item. Since all probabilities are uniformly [0, 1] distributed, it is clear that μ_I



(a) $lowP <= 0.10$ **(b)** $lowP <= 0.01$

Fig. 6 Accuracy of the model-based algorithms versus percentage of low probability values. **a** $lowP <= 0.10$. **b** $lowP <= 0.01$

is about $\frac{n}{2}$. Thus, for minsup=0.49, the probability that an item is frequent increases in the database size, for minsup = 0.5, about half of the items are frequent, and for minsup = 0.51, the number of frequent items decreases in the database size. First, it can be observed that the Normal approximation is nearly perfect, since no error between the exact $Pr_{freq}(I)$ and it's Normal approximated value can be observed visually. This is also confirmed by Fig. 5d–f, which show the respective error, that is, the differences between the exact frequentness probability and the approximated frequentness probabilities. In contrast, the Poisson approximation does show a significant error for smaller databases. For minsup = 0.49 and minsup = 0.51, the Poisson approximation is also able to yield very good approximation for database sizes larger than 50.000 while the expected support yields good results for even smaller databases. The reason is that in this case, all the exact frequentness probabilities quickly converge to 1 (0), which can easily be guessed by the expected approach. However, the interesting case is the case where minsup = 0.5, since in this case, the items are expected to have an average frequentness probability of 0.5. However, in this case, the expected approach is forced to guess, since it may only take the values 0 and 1. This explains the large error of the expected approach, which does not decrease in the database size. The Poisson approach performs significantly better than the expected approach, but still the error is relatively high, which matches our previous experiments for large probabilities. To achieve a better setting for the Poisson approximation, we changed the interval from which the item probabilities are sampled, from [0, 1] to [0, 0.1], and repeated the experiment for minsup=0.5. The results are depicted in Fig. 5g, h). It can be observed that in this setting, the Poisson approximation achieves extremely good results, even slightly outperforming the Normal approximation. In Fig. 5i, the expected approach is omitted, to give a better view on the difference of the normal and the Poisson error.

In the previous setting, we have evaluated the performance of the model-based approximation approaches in the case where all items have the same occurrence probabilities. We have seen that for small probabilities, the Poisson approximation works well. Next we will evaluate how many probability values are allowed to be large, in order for the Poisson approximation to still yield good results. In the following experiment, we introduce a new parameter $lowP$, which denotes the fraction of the item probabilities, which are chosen from a smaller interval, such as the interval [0, 0.01], while the remaining $1 - lowP$ fraction of the items has their probabilities chosen from the [0, 1] interval. The result of the experiment evaluating the impact of $lowP$ is given in Fig. 6. It can be seen that the mean absolute approximation error

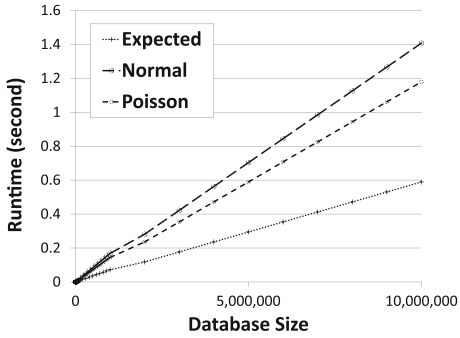
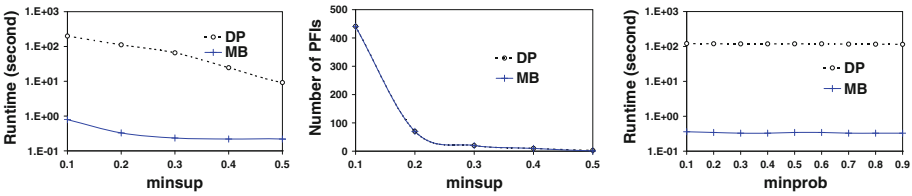


Fig. 7 Performance comparison of model-based approaches



(a) Runtime vs. *minsup* **(b)** $|PFI|$ vs. *minsup* **(c)** Runtime vs. *minprob*

Fig. 8 Threshold-based PFI mining: efficiency of model-based algorithm MB versus dynamic programming DP. **a** Runtime versus $minsup$. **b** $|PFI|$ versus $minsup$. **c** Runtime versus $minprob$

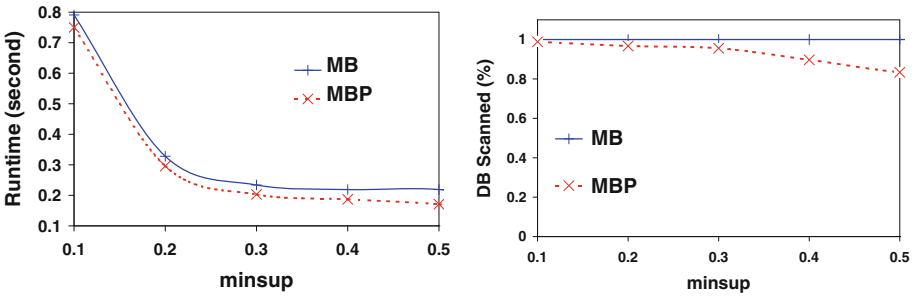
of the Normal approximation increases as the fraction $lowP$ of small probabilities increases. In contrast, the approximation error of the Poisson approximation slightly decreases. However, only for $lowP > 0.98$, the Poisson approximation begins to outperform the Normal approximation. Thus, if the dataset only contains a few probability values that are not small, then the Normal approximation outperforms the Poisson approximation.

While in the previous experiment, the approximation quality has been measured for individual itemsets, we now compare the effectivity of the model-based approaches for the complete Apriori-based algorithm.

(ii) MB vs.DP. Next, we compare the performance (in log scale) of the model-based approaches (MB) and the dynamic programming-based DP. First, we will compare the runtime of the three (MB)s to each other. The result is shown in Fig. 7. It can be seen that the approach using expected support, since it only has to perform a single value comparison ($\mu_I \geq minsup$), is faster than the other model-based approaches by a factor of about two. The normal and the Poisson approximation take about the same time to compute. While in our Java experiments (shown here), the Poisson approximation slightly outperforms the normal approximation, our experiments in *R* (not depicted here) show the opposite situation. In summary, we can say that Poisson and normal approximation take about the same time to evaluate, except for some possibly implementation specific differences.

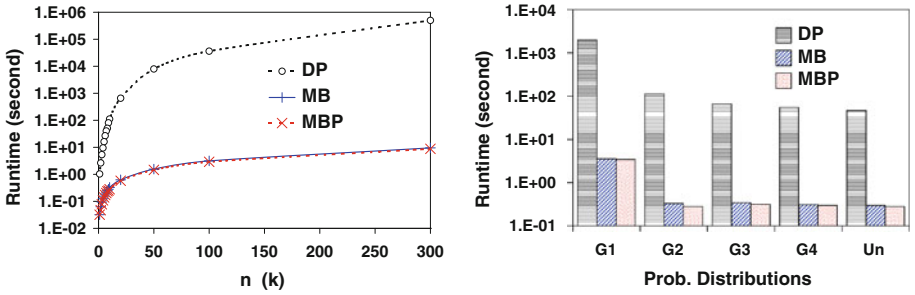
In the following runtime experiments, we will for simplicity only show one graph for the model-based (MB) approaches, which corresponds to the runtime of the normal and Poisson approximation. For the runtime of the expected approach, simply divide the result by two due to the observations made in Fig. 7.

Figure 8a. Observe that MB is about two orders of magnitude faster than DP, over a wide range of *minsup*. This is because MB does not compute exact frequentness probabilities as DP does; instead, MB only computes the μ_I values, which can be obtained faster.



(a) Runtime vs. *minsup* **(b)** DB scanned(%) vs. *minsup*

Fig. 9 Efficiency of MBP versus MB. **a** Runtime versus *minsup*. **b** DB scanned (%) versus *minsup*



(a) Scalability **(b)** Prob. Distributions

Fig. 10 Other results for threshold-based PFIs. **a** Scalability. **b** Prob. Distributions

We also notice that the running times of both algorithms decrease with a higher *minsup*. This is explained by Figure 8b, which shows that the number of PFIs generated by the two algorithms, $|PFI|$, decreases as *minsup* increases. Thus, the time required to compute the frequentness probabilities of these itemsets decreases. We can also see that $|PFI|$ is almost the same for the two algorithms, reflecting that the results returned by MB closely resemble those of DP.

Figure 8c examines the performance of MB and DP (in log scale) over different *minprob* values. Their execution times drop by about 6 % when *minprob* changes from 0.1 to 0.9. We see that MB is faster than DP. For instance, at $minprob = 0.5$, MB needs 0.3 s, while DP requires 118 s, delivering an almost 400-fold performance improvement.

(iii) MB vs. MBP. We then examine the benefit of using the improved PFI testing method (MBP) over the basic one (MB). Figure 9(a) shows that MBP runs faster than MB over different *minsup* values. For instance, when $minsup = 0.5$, MBP has about 25 % of performance improvement. Moreover, as *minsup* increases, the performance gap increases. This can be explained by Fig. 9b, which presents the fraction of the database scanned by the two algorithms. When *minsup* increases, MBP examines a smaller fraction of the database. For instance, at $minsup = 0.5$, MBP scans about 80 % of the database. This reduces the I/O cost and the effort for interpreting the retrieved tuples. Thus, MBP performs better than MB.

(iv) Scalability. Figure 10a examines the scalability of the three algorithms. Both MB and MBP scale well with n. The performance gap between MB/MBP and DP also increases with n. At $n = 20k$, MB and DP need 0.62 and 657.7 s, respectively; at $n = 100k$, MB finishes in

Table 8 Existential probability

	Mean	Standard deviation
G_1	0.8	0.125
G_2(default)	0.5	0.125
G_3	0.5	$\sqrt{1/12} \approx 0.289$
G_4	0.5	0.5
Un	0.5	$\sqrt{1/12} \approx 0.289$

Table 9 Recall and precision of `E-top-k`

n	1k	5k	10k	50k
Recall & Precision	1	1	1	1
	Recall & Precision vs. n			
k	1	10	50	100
Recall & precision	1	1	0.98	0.99
	Recall & precision vs. k			
$minsup/n$	0.1	0.2	0.3	0.4
Recall & precision	1	1	1	1
	Recall & precision vs. $minsup$			

3.1 s while `DP` spends 10 h. Hence, the scalability of our approaches is clearly better than that of `DP`.

(v) Existential probability. We also examine the effect of using different distributions to characterize an attribute's probability, in Fig. 10b. We use Un to denote a uniform distribution and G_i ($i = 1, \ldots, 4$) to represent a Gaussian distribution. The details of these distributions are shown in Table 8.

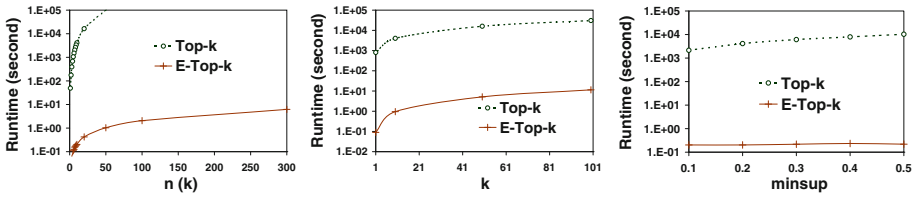
We observe that `MB` and `MBP` perform better than `DP` over different distributions. All algorithms run comparatively slower on G_1. This is because G_1 has high mean (0.8) and low standard deviation (0.125), which generates high existential probability values. As a result, many candidates and PFIs are generated. Also note that G_3 and Un, which have the same mean and standard deviation, yield similar performance. Lastly, we found that the precision and the recall of `MB` and `MBP` over these distributions are the same and are close to 1. Hence, the PFIs retrieved by our methods closely resemble those returned by `DP`.

7.2 Results on rank-based PFI mining

We now compare the first top-k solution proposed in [39] (called `top-k`) and our enhanced algorithm (called `E-top-k`).

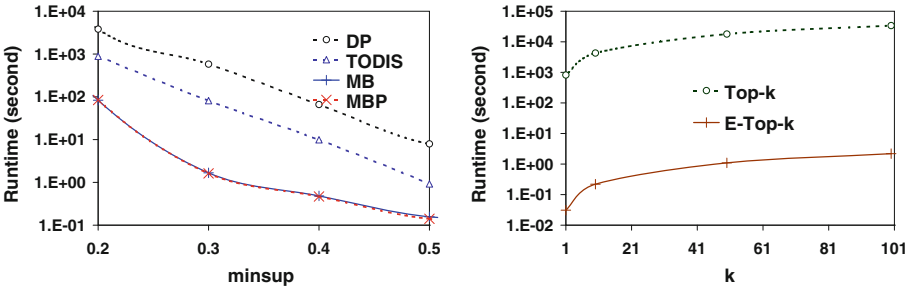
(vi) Accuracy. We measure the recall and precision of `E-top-k` relative to that of `top-k`, by using formulas similar to Eqs. 21 and 22. Table 9 shows the recall and the precision of `E-top-k` for a wide range of n, k and $minsup$ values. We observe that the recall values are equal to the precision values. This is because number of PFIs returned by `top-k` and `E-top-k` are the same. As We can also see the recall and precision values are always higher than 98 %. Hence, `E-top-k` can accurately return top-k PFIs.

(vii) Performance. Figure 11a–c compare the two algorithms under different values of n, k, and $minsup$. Similar to the case of threshold-based PFIs, our solution runs much faster



(a) Runtime vs. n **(b)** Runtime vs. k **(c)** Runtime vs. $minsup$

Fig. 11 Performance of rank-based PFI mining algorithms. **a** Runtime versus n. **b** Runtime versus k. **c** Runtime versus $minsup$



(a) Runtime vs. $minsup$ **(b)** Runtime vs. k

Fig. 12 Results for tuple uncertainty. **a** Runtime versus $minsup$. **b** Runtime versus k

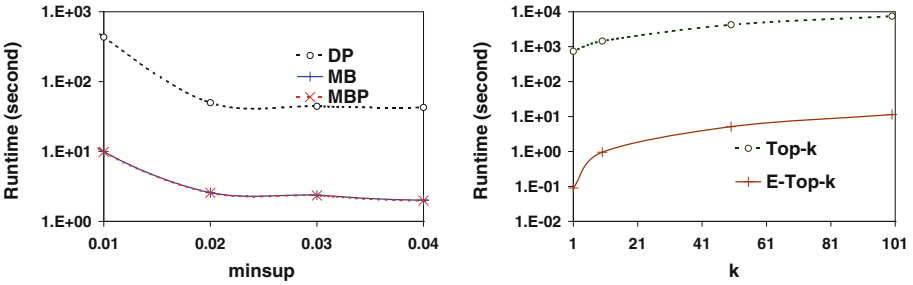
than `top-k`. When $n = 10k$, for example, `E-top-k` finishes in only 0.2 s, giving an almost 20,000-fold improvement over that of `top-k`, which completes in 1.1 hours. In Fig. 11a, we see that the runtime of `top-k` increases faster than that of `E-top-k` with a bigger database. In Fig. 11b, observe that the `E-top-k` is about four orders of magnitude faster than `top-k`.In Fig. 11c, with the increase in $minsup$, `top-k` needs more time, but the runtime of E-top-k only slightly increases. This is because (1) fewer candidates are produced by our generation step and (2) the testing step is significantly improved by using our model-based methods.

7.3 Other experiments

We have also performed experiments on the tuple uncertainty model and the synthetic dataset. Since they are similar to the results presented above, we only describe the most representative ones. For the accuracy aspect, the recall and precision values of approximate results on these datasets are still higher than 98 %. Thus, our approaches can return accurate results.

Tuple uncertainty. We compare the performance of `DP`, `TODIS`, `MB`, and `MBP` in Figure 12(a). `TODIS` [38] is proposed to retrieve threshold-based PFIs from tuple-uncertain databases. It shows that both `MB` and `MBP` perform much better than `DP` and `TODIS`, under different $minsup$ values. For example, when $minsup = 0.3$, `MB` needs 1.6 s, but `DP` and `TODIS` complete in 581 and 81 s, respectively. In Fig. 12b, we also see that `E-top-k` consistently outperforms `top-k` by about two orders of magnitude. This means that our approach, which avoids computing frequentness probabilities, also works well for the tuple uncertainty model.

Synthetic dataset. Finally, we run the experiments on the synthetic dataset. Figure 13a compares the performance of `MB`, `MBP`, and `DP`, for attribute uncertainty model. We found that `MB` and `MBP` still outperform `DP`. Figure 13b tests the performance of `top-k` and `E-top-k`,



(a) Runtime vs. $minsup$ **(b)** Runtime vs. k

Fig. 13 Results for synthetic dataset. **a** Runtime versus $minsup$. **b** Runtime versus k

for tuple uncertainty model. Again, `E-top-k` runs faster than `top-k`, by around two orders of magnitude. Thus, our approach also works well for this dataset.

8 Conclusions

We propose model-based approaches to discover Probabilistic Frequent Itemsets (PFIs) from uncertain databases. Our methods represent the s-pmf of a PFI using probability models, in order to find PFIs quickly. These methods can efficiently extract threshold- and rank-based PFIs. They also support attribute and tuple uncertainty models, which are two common data models. We develop new approximation methods to evaluate frequentness probabilities efficiently. As shown theoretically and experimentally, our algorithms are more efficient and scalable than existing ones. They are also highly accurate, although some models require certain properties of the dataset to be satisfied in order to achieve very high accuracy. We have theoretically and experimentally compared our proposed model-based approaches and shown properties of the data which are required for each model to perform well. To conclude, a summary of the pros and cons of each model is as follows:

Expected support: Easy to implement and efficiently to compute, since the total runtime is about twice as fast as the other approaches, and pruning can be performed. However, this approach lacks effectivity, since this model requires a very large number of transactions containing an itemset I with a probability greater than zero, in order to achieve acceptable results. Except for very small itemsets, which are usually not interesting because they are trivially frequent, this requirement is hardly satisfied. Also, the parameter $minprob$ cannot be integrated into this approach, so that the level of significance of frequent items cannot be determined.

Poisson approximation: Easy to implement and efficiently to compute, since pruning can be performed. However, this model requires that almost all transactions containing I have a very low probability, in order to obtain good approximation results. This requirement is hardly ever met on real data, since such datasets generally contain some transaction containing I with a probability of 1. However, this model is not robust since a few larger probabilities will significantly lower the approximation quality.

Normal approximation: The implementation must make sure to apply continuity correction to acquire the best results. Pruning cannot be performed so that for each itemset, the whole database has to be scanned. The experiments have shown that pruning only increases the total runtime marginally. Regarding the approximation quality, the normal approximation

overall yields by far the best results. Even for a small number of Poisson trials, the Normal approximation yields highly accurate results. On real datasets, it is very difficult to create a scenario where the normal approximation does not yield precision and recall values of one.

In summary, we propose to generally use the Normal approximation, except in very special settings, since the Normal approximation yields the best trade-off between approximation quality, which is nearly always one, and efficiency, which is in $O(n)$ like the other approximation approaches.

In the future, we will examine how the model-based approach can be used to handle other mining problems (e.g., clustering) in uncertain databases. We will also study how other approximation techniques, such as sampling, can be used to mine PFIs.

Acknowledgments This work was supported by the Research Grants Council of Hong Kong (GRF Projects 513508 and 711309E). We would like to thank the anonymous reviewers for their insightful comments.

Appendix A: Quality of the poisson approximation (Sect. 4.2)

In Sect. 4.2, we use the Poisson distribution to approximate the Poisson binomial distribution. Here, we summarize the results of this approximation quality, discussed in [37].

Let $X_1, X_2, \ldots, X_n$ be a set of Poisson trials such that $Pr(X_j = 1) = p_j$ and $X = \sum_{j=1}^{n} X_j$. Then, X follows a Poisson binomial distribution. Suppose $\mu = E[X] = \sum_{j=1}^{n} p_j$. The probability of $X = i$ and $X \leq i$ can be approximated by the probability distribution function (PDF) and the cumulative distribution function (CDF) of the Poisson distribution,

$$Pr(X = i) \approx f(i, \mu) = \frac{\mu^i}{i!} e^{-\mu}$$
$$Pr(X \leq i) \approx F(i, \mu) = \frac{\Gamma(i+1, \mu)}{i!}$$

[37] gives an upper bound of the error of the approximation:

$$|Pr(X \leq i) - F(i, \mu)| \leq (\mu^{-1} \wedge 1) \sum_{j=1}^{n} p_j^2 \tag{23}$$

for $i = 0, 1, 2, \ldots, n$ where $\mu^{-1} \wedge 1 = min(\mu^{-1}, 1)$.

Now, we want to compute a bound on the expression on the right-hand side. Since $\mu = \sum_{j=1}^{n} p_j$,

$$(\mu^{-1} \wedge 1) \sum_{j=1}^{n} p_j^2 = min\left(\frac{1}{\sum_{j=1}^{n} p_j}, 1\right) \sum_{j=1}^{n} p_j^2$$

Obviously the above expression is greater than or equal to 0.

When $0 \leq \sum_{j=1}^{n} p_j \leq 1$,

$$(\mu^{-1} \wedge 1) \sum_{j=1}^{n} p_j^2 = \sum_{j=1}^{n} p_j^2 \leq \sum_{j=1}^{n} p_j \leq 1$$

When $\sum_{j=1}^{n} p_j > 1$,

$$(\mu^{-1} \wedge 1) \sum_{j=1}^{n} p_j^2 = \frac{\sum_{j=1}^{n} p_j^2}{\sum_{j=1}^{n} p_j} \leq \frac{\sum_{j=1}^{n} p_j}{\sum_{j=1}^{n} p_j} = 1$$

So, in either case:

$$0 < (\mu^{-1} \wedge 1) \sum_{i=1}^{n} p_i^2 \leq 1 \tag{24}$$

The upper bound of the error is very small. As also verified by our experiments, the Poisson binomial distribution can be approximated well.

Appendix B: Quality of the normal approximation (Sect. 4.3)

In Sect. 4.3, we use a normal distribution to approximate a Poisson binomial distribution. Here, we will summarize the current state of research in theoretical quality of this approximation. Therefore, let $X_1, \ldots, X_n$ be independent Poisson trials with respective probabilities $p_1, \ldots, p_n$, and let X denote the random variable corresponding to the average $\frac{1}{n} \sum_{i=1}^{n} X_i$ of these random variables. Also, let Y denote the CDF of $\frac{X \cdot \sqrt{(n)}}{\sigma}$, that is, the normalized CDF of X. In [15], the author was able to prove that the maximum error between Y and the CDF Φ of the standard normal distribution is bounded as follows:

$$\sup_x |Y - \Phi| \leq C\Psi,$$

where $\Psi = (\sum_{i=1}^{n} \sigma_i^2)^{\frac{-3}{2}} \cdot \sum_{i=1}^{n} \rho_i$, σ_i^2 is the variance of X_i and ρ_i is the third moment of X_i. C is a constant that was successively lowered from the original estimate of 7.59 ([15]) to the current best estimate of 0.5600 by [36].

References

1. Deshpande A et al (2004) Model-driven data acquisition in sensor networks. In: VLDB
2. Abramowitz and Stegun (1972) Handbook of mathematical functions with formulas, graphs, and mathematical tables. 10 edition
3. Aggarwal C, Li Y, Wang J, Wang J (2009) Frequent pattern mining with uncertain data. In: KDD
4. Aggarwal C, Yu P (2009) A survey of uncertain data algorithms and applications. TKDE 21(5): 609–623
5. Agrawal R, Imieliński T, Swami A (1993) Mining association rules between sets of items in large databases. In: SIGMOD
6. Bernecker T, Kriegel H-P, Renz M, Verhein F, Züfle A (2008) Probabilistic frequent itemset mining in uncertain databases. In: Proceedings of the 15th ACM SIGKDD conference on knowledge discovery and data mining (KDD'09), Paris, France, pp 119–128
7. van Rijsbergen CJ (1979) Information retrieval. Butterworth
8. Cam LL (1960) An approximation theorem for the poisson binomial distribution. Pacific J Math 10: 1181–1197
9. Cheng H, Yu P, Han J (2008) Approximate frequent itemset mining in the presence of random noise. SCKDDM
10. Cheng R, Kalashnikov D, Prabhakar S (2003) Evaluating probabilistic queries over imprecise data. In: SIGMOD
11. Chui CK, Kao B, Hung E (2007) Mining frequent itemsets from uncertain data. In 11th Pacific-Asia conference on advances in knowledge discovery and data mining, PAKDD 2007, Nanjing, China, pp 47–58

12. Chui CK, Kao B, Hung E (2007) Mining frequent itemsets from uncertain data. In: PAKDD
13. Cormode G, Garofalakis M (2007) Sketching probabilistic data streams. In: SIGMOD
14. Dalvi N, Suciu D (2004) Efficient query evaluation on probabilistic databases. In: VLDB
15. Esseen C-G (1942) On the Liapunoff limit of error in the theory of probability. Arkiv för matematik, astronomi och fysik
16. Feller W (1945) The fundamental limit theorems in probability. Bull Amer Math Soc 51: 800–832
17. Feller W (1968) The strong law of large numbers, in an introduction to probability theory and its applications. Wiley, New York
18. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: SIGMOD
19. Hodges J, le Cam L (1959) The poisson approximation to the poisson binomial distribution. The annals of mathematical statistics. Institute of Mathematical Statistics
20. Hua M, Pei J, Zhang W, Lin X (2008) Ranking queries on uncertain data: a probabilistic threshold approach. In: Wang JT-L (ed) Proceedings of the ACM SIGMOD international conference on management of data, SIGMOD 2008, Vancouver, BC, Canada, June 10–12, 2008, pp 673–686. ACM
21. Huang J et al (2009) MayBMS: a probabilistic database management system. In: SIGMOD
22. Ren J, Lee S, Chen X, Kao B, Cheng R, Cheung D (2009) Naive Bayes classification of uncertain data. In: ICDM
23. Jampani R, Perez L, Wu M, Xu F, Jermaine C, Haas P (2008) MCDB: a Monte Carlo approach to managing uncertain data. In: SIGMOD
24. Khoussainova N, Balazinska M, Suciu D (2006) Towards correcting input data errors probabilistically using integrity constraints. In: MobiDE
25. Kozawa Y, Amagasa T, Kitagawa H (2011) Fast frequent itemset mining from uncertain databases using gpgpu. In: DBRank, pp 55–62
26. Kriegel H, Pfeifle M (2005) Density-based clustering of uncertain data. In: KDD
27. Kuok C, Fu A, Wong M (1998) Mining fuzzy association rules in databases. SIGMOD record
28. Liu H, Lin Y, Han J (2011) Methods for mining frequent items in data streams: an overview. Knowl Inf Syst 26(1):1–30
29. Lu A, Ke Y, Cheng J, Ng W (2007) Mining vague association rules. In: DASFAA
30. Mutsuzaki M et al (2007) Trio-one: layering uncertainty and lineage on a conventional dbms. In: CIDR
31. Yiu M, et al. (2009) Efficient evaluation of probabilistic advanced spatial queries on existentially uncertain data. TKDE 21(9): 108–122
32. Sistla P, et al. (1998) Querying the uncertain position of moving objects. In: Temporal databases: research and practice. Springer, New York
33. Pavlov D, Mannila H, Smyth P (2003) Beyond independence: probabilistic models for query approximation on binary transaction data. IEEE Trans Knowl Data Eng 15(6):1409–1421
34. Poisson S (1837) Recherches sur la probilité des Jugements
35. Renze J, Weisstein E. Law of large numbers
36. Shevtsova IG (2010) An improvement of convergence rate estimates in the Lyapunov theorem. Doklady Math 82:862–864
37. Stein C (1986) Approximate computation of expectations. Institute of mathematical statistics lecture notes—monograph series, 7
38. Sun L, Cheng R, Cheung DW, Cheng J (2010) Mining uncertain data with probabilistic guarantees. In: SIGKDD
39. Bernecker T et al (2009) Probabilistic frequent itemset mining in uncertain databases. In: KDD
40. Jayram T et al (2006) Avatar information extraction system. IEEE Data Eng Bull 29(1): 40–48
41. Tsang S, Kao B, Yip KY, Ho W, Lee S (2009) Decision trees for uncertain data. In: ICDE
42. Wang C, Parthasarathy S (2006) Summarizing itemset patterns using probabilistic models. In: Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining, Philadelphia, PA, USA, August 20–23, 2006, pp 730–735
43. Wang L, Cheng R, Lee SD, Cheung DW-L (2010) Accelerating probabilistic frequent itemset mining: a model-based approach. In: CIKM, pp 429–438
44. Weng C-H, Chen Y-L (2010) Mining fuzzy association rules from uncertain data. Knowl Inf Syst 23(2):129–152
45. Yang B, Huang H (2010) Topsil-miner: an efficient algorithm for mining top- k significant itemsets over data streams. Knowl Inf Syst 23(2):225–242
46. Zhang Q, Li F, Yi K (2008) Finding frequent items in probabilistic data. In: SIGMOD

Author Biographies

Thomas Bernecker is an Academic Assistant in the database systems and data mining group of Hans-Peter Kriegel at the Ludwig-Maximilians-Universität München, Germany. His research interests include query processing in uncertain databases, spatio-temporal data mining, and similarity search in spatial, temporal, and multimedia databases.

Reynold Cheng received the B.Eng degree in computer engineering and the MPhil in computer science and information systems from the University of Hong Kong (HKU) in 1998 and 2000, respectively, and the M.Sc. and Ph.D. degrees from the Department of Computer Science, Purdue University, in 2003 and 2005, respectively. He is an associate professor in the Department of Computer Science at HKU. He was the recipient of the 2010 Research Output Prize in the Department of Computer Science of HKU. From 2005 to 2008, he was an assistant professor in the Department of Computing at Hong Kong Polytechnic University, where he received two Performance Awards. He is a member of IEEE, ACM, ACM SIGMOD, and UPE. He has served on the program committees and review panels for leading database conferences and journals. He is a member of the editorial board of Information Systems and DAPD journal. He is also a guest editor for a special issue in TKDE. His research interests include database management, as well as querying and mining of uncertain data.

David W. Cheung received the M.Sc. and Ph.D. degrees in computer science from Simon Fraser University, Canada, in 1985 and 1989, respectively. Since 1994, he has been a faculty member of the Department of Computer Science in The University of Hong Kong. His research interests include database, data mining, database security and privacy. Dr. Cheung was the Program Committee Chairman of PAKDD 2001, Program Co-Chair of PAKDD 2005, Conference Chair of PAKDD 2007 and 2011, Conference Co-Chair of CIKM 2009, and Conference Co-Chair of PAKDD 2011.

Hans-Peter Kriegel is a full-time professor for database systems and data mining in the Department "Institute for Informatics" at the Ludwig-Maximilians-Universität München, Germany, and has served as the department chair or vice chair over the last years. His research interests are in spatial and multimedia database systems, particularly in query processing, performance issues, similarity search, high-dimensional indexing as well as in knowledge discovery and data mining. He has published over 300 refereed conference and journal papers, and he received the "SIGMOD Best Paper Award" 1997 and the "DASFAA Best Paper Award" 2006 together with members of his research team. He is an Associate Editor for The VLDB Journal and ACM TKDD. THe ACM nominated Hans-Peter Kriegel 2009 ACM Fellow for contributing fundamental knowledge to the field, in particular for his contributions to knowledge discovery and data mining, similarity search, spatial data management, and access methods for high-dimensional data.

Sau Dan Lee was a Post-doctoral Fellow at the University of Hong Kong. He received his Ph.D. degree from the University of Freiburg, Germany in 2006 and his M.Phil. and B.Sc. degrees from the University of Hong Kong in 1998 and 1995. He is interested in the research areas of data mining, machine learning, uncertain data management and information management on the WWW. He has also designed and developed backend software systems for e-Business and investment banking.

Matthias Renz is an Assistant Professor in the database systems and data mining group of Hans-Peter Kriegel at the Ludwig-Maximilians-Universität München, Germany. He finished his P.h.D thesis on query processing in spatial and temporal data in winter 2006. His research interests include query processing in uncertain databases, spatio-temporal data mining and similarity search in spatial, temporal, and multimedia databases.

Florian Verhein completed his PhDs in parallel, obtaining one from the Faculty of Engineering and IT at the University of Sydney, Australia and the other from the Faculty of Mathematics, Computer Science and Statistics at Ludwig-Maximilians-Universität, München, Germany. He also holds qualifications in Software Engineering and Business and is a past Microsoft Research Asia Fellow. Florian's research covers various areas in data mining and machine learning, with an emphasis on statistics and algorithms. He currently works at Optiver, a leading market maker and proprietary trading company.

Liang Wang received the B.Eng. degree in computer science from Shanghai Jiaotong University in 2008 and the M.Phil. degree majoring in computer science from the University of Hong Kong in 2011. His research interest is uncertain database, data mining and data management. Now, He is a software engineer at Microsoft Corporation.

Andreas Zuefle is an Academic Assistant in the database systems and data mining group of Hans-Peter Kriegel at the Ludwig-Maximilians-Universität München, Germany. His research interests include query processing in uncertain databases, spatio-temporal data mining and similarity search in spatial, temporal, and multimedia databases.