The VLDB Journal (2008) 17:1121–1141
DOI 10.1007/s00778-007-0054-1

REGULAR PAPER

Power-law relationship and self-similarity in the itemset support distribution: analysis and applications

Kun-Ta Chuang · Jiun-Long Huang · Ming-Syan Chen

Received: 28 February 2006 / Accepted: 28 March 2007 / Published online: 4 July 2007

Abstract In this paper, we identify and explore that the power-law relationship and the self-similar phenomenon appear in the itemset support distribution. The itemset support distribution refers to the distribution of the count of itemsets versus their supports. Exploring the characteristics of these natural phenomena is useful to many applications such as providing the direction of tuning the performance of the frequent-itemset mining. However, due to the explosive number of itemsets, it is prohibitively expensive to retrieve lots of itemsets before we identify the characteristics of the itemset support distribution in targeted data. As such, we also propose a valid and cost-effective algorithm, called algorithm PPL, to extract characteristics of the itemset support distribution. Furthermore, to fully explore the advantages of our discovery, we also propose novel mechanisms with the help of PPL to solve two important problems: (1) determining a subtle parameter for mining approximate frequent itemsets over data streams; and (2) determining the sufficient sample size for mining frequent patterns. As validated in our experimental results, PPL can efficiently and precisely identify the characteristics of the itemset support distribution in various real data. In addition, empirical studies also demonstrate that our mechanisms for those two challenging problems are in orders of magnitude better than previous works, showing the prominent advantage of PPL to be an important preprocessing means for mining applications.

K.-T. Chuang (✉) · M.-S. Chen
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, ROC
e-mail: doug@arbor.ee.ntu.edu.tw

M.-S. Chen
e-mail: mschen@cc.ee.ntu.edu.tw

J.-L. Huang
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC
e-mail: jlhuang@cs.nctu.edu.tw

1 Introduction

The importance of mining frequent itemsets has been recognized in various applications, including web log mining, DNA sequence mining, frequent episodes mining, periodic patterns, to name a few [20]. Due to the data-driven nature of mining algorithms, it is believed in the literature that the parameter tuning of the designed algorithm is usually requested in order to achieve the better result or performance on the targeted applications. It is clear that the deeper knowledge about the characteristics of the targeted data leads to the better execution efficiency and the better interpretation of the mining result. As such, a mechanism to precisely estimate the data characteristics is usually deemed as an important preprocessing means for mining applications.

Recent research advances in the frequent-itemset mining have been in the direction of discovering characteristics of real datasets. For example, the works in [18] and [32] both seek the relationship between different itemset lengths in the targeted dataset. Such relationships can be further utilized to control the mining process [18], or to generate the realistic synthetic datasets for the system parameter tuning [32].

To provide better understanding on real datasets, we in this paper investigate a more important characteristic in real datasets, named the *itemset support distribution*. The *itemset support distribution* refers to the distribution of the count of itemsets versus the itemset support, where an itemset complies with the definition in [1]. Explicitly, we shall study the relationship between the value of support, say 0.01, and the number of itemsets having the support 0.01 in the dataset, as the curve illustrated in Fig. 1. The *itemset support*

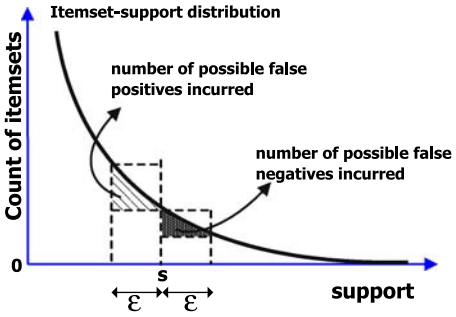


Fig. 1 The illustration of the *itemset support distribution* and the comparison between false positives and false negatives in the frequency-approximation applications

distribution, which is indeed a kind of the probability density function, will state the degree of the cohesion between different items in the dataset. To the best of our knowledge, this fundamental question has not been formally addressed.

Inspired by the power-law relationship observed in many distributions of single words (users, web pages) [6,43], it is important to examine whether the *itemset support distribution* also follows the power-law relationship. From observations on various retail datasets and as validated by our empirical studies later, it is amazingly found that the power-law relationship indeed appears in the *itemset support distribution* and we can characterize that as a Zipf distribution [43].

In addition, we also find the self-similar phenomenon exists in the distribution of itemset supports. *Self-similarity* refers to the phenomenon that a distribution whose appearance is unchanged regardless of the scale at which it is viewed [3,37]. Specifically, our observation shows that the power-law phenomenon, whose existence has been found in the support distribution of 1-items in early works [43], also exists in the support distribution consisting of itemsets with a certain length i ($i \geq 1$). It is a self-similar phenomenon in nature. Importantly, *self-similarity* can provide proper reasons of why the power-law phenomenon also exists in the *itemset support distribution*.

However, to find the parameters characterizing the *itemset support distribution* is more challenging than to find the parameters in the distribution of 1-items, since an extremely large number of itemsets needs to be retrieved. Note that there are at most $m + C_2^m + C_3^m + \cdots + C_{m-1}^m + C_m^m = 2^m - 1$ possible itemsets in a dataset, where m denotes the number of distinct items in the dataset. In practice, the itemset combinational explosion incurs extremely large time and memory consumption, which will drastically decrease the practicability of knowing the characteristics of the *itemset support distribution*. To broaden the applicability of our discovery, we also propose in this paper a valid and cost-effective algorithm, called algorithm PPL (standing for *P*redict the *P*ower-*L*aw relationship), to correctly estimate the parameters of the *itemset support distribution* from a sample dataset while also avoiding the need of generating lots of itemsets. As shown in our empirical studies, algorithm PPL is able to efficiently and precisely extract the characteristics of the power-law relationship. As such, algorithm PPL can be utilized as an efficient pre-processing step for extensive applications of mining frequent patterns.

1.1 Motivating applications

The power-law phenomenon has received a great deal of attention in web mining and data mining research [15]. Nevertheless, this natural phenomenon is not fully explored and utilized in the most important mining application, namely the frequent-pattern mining. We review several popular frequent-pattern mining applications, and comment the advantage of utilizing algorithm PPL as their pre-processing step:

1.1.1 Frequency approximation over data streams

Recent advances in streaming research recognize the importance of frequency approximation [11,25,26,38]. Among them, based on the assumption that the support distribution of single items follows the Zipf distribution, the work in [26] devised algorithm *Space-Saving* to compute top-k single items over time. Algorithm *Space-Saving* has a memory bound equal to $\min(|A|, \left(\frac{1}{\epsilon}\right)^{\frac{1}{\theta}}, \frac{1}{\epsilon})$, where $|A|$ denotes the number of distinct items in the database, ϵ is the error parameter,[1] and θ is the parameter of the Zipf distribution. Clearly, according to our observation, their idea can be further extended to approximately retrieve top-k frequent itemsets, since the *itemset support distribution* also follows the Zipf distribution. In addition, while executing algorithm PPL in advance, the appropriate memory size (depending on the parameter θ which can be identified by PPL) can be assigned prior to the execution of algorithm *Space-Saving*. It enables the efficient implementation. For example, since the memory to maintain itemsets can be previously allocated, we can implement the algorithm by maintaining itemsets in an array-like structure as opposed to a dynamic linked list-like structure, to achieve better efficiency. Note that it is reported that the dynamic data structure in general leads to poor efficiency due to its poor spatial locality and poor temporal locality [19].

1.1.2 False positives versus false negatives

As pointed out in [39], a challenging issue of the frequency approximation over data streams arises from the choice

[1] Specifically, ϵ is used to ensure that no item with true frequency less then $(s - \epsilon)N$ will be output, where s denotes the minimum support and N is the number of tuples in the database.

between false positives and false negatives. Specifically, traditional works of mining approximate frequent patterns, such as [25], are usually false-positive oriented, i.e., they allow identifying non-frequent itemsets as frequent ones. In contrast, the authors in [39] introduce false-negative oriented algorithms, which allow identifying frequent itemsets as non-frequent ones. By simulation, the authors in [39] conclude that false-negative oriented algorithm can lead to small memory consumption as compared to that in false-positive oriented solutions. In practice, our observation in the *itemset support distribution* provides the essential evidence of their studies: *the possible number of inaccurate patterns identified by a false-negative oriented solution (frequent itemsets being identified as non-frequent) is much smaller than the possible number of inaccurate patterns identified by false-positive oriented solutions (non-frequent itemsets being identified as frequent)*. The reason is that the *itemset support distribution* follows the power-law relationship, and the number of small-support itemsets will be larger than the number of high-support itemsets.

Furthermore, the error parameter ϵ in [25] and the parameter δ in [39] control the compromise between the resource consumption and the resulting accuracy, but how to determine these parameters is fully left unsolved to users. Note that users are easy to give desired model accuracy (or a bound of tolerant errors) rather than to give these subtle parameters. Suppose that algorithm PPL is executed as a pre-processing step to approximately estimate the expected number of inaccurate itemsets incurred either as false positives or as false negatives, as illustrated in Fig. 1. As such, a good reference to appropriately determine these error parameters can be provided, because we can estimate the resulting accuracy with respect to any given degree of these error parameters. Algorithm PPL is thus able to provide the better flexibility of striking a compromise between the resource consumption and accuracy in mining approximate frequent patterns over data streams. We will validate the feasibility of this idea in our application studies.

1.1.3 Performance tuning of mining frequent patterns

It is reported that the effectiveness of many heuristic optimizations for mining frequent itemsets usually depends the data characteristics, particularly the sparsity/density of the targeted data [28]. Once the data is long and dense, i.e., the *itemset support distribution* is highly skewed, depth-first approaches such as FP-growth [21] are more powerful than breath-first approaches such as DHP [29]. In contrast, algorithm DHP is more suitable in the presence of sparse data. Hence early identifying the parameters of the *itemset support distribution* (to indicate the level of data sparsity) by algorithm PPL will help users to make proper strategies to achieve the best performance of mining frequent patterns.

1.1.4 Determine the appropriate minimum support

Characterizing the *itemset support distribution* in advance will help the determination of the minimum support. Formally, setting the minimum support is quite subtle, since a small minimum support leads to an extremely large size of frequent itemsets, and in contrast, only a few itemsets are generated when the minimum support is large. In order to obtain a desired mining result, users in general need to tune the minimum support over a wide range, which is time-consuming and is a serious problem for the applicability of mining frequent itemsets. To remedy this, we can execute algorithm PPL as a pre-processing step, which is able to estimate the number of itemsets with support exceeding a specified minimum support. As such, users can easily decide the appropriate minimum support threshold to retrieve the desired number of frequent patterns.[2]

1.1.5 Synthetic data generator

Formally, a synthetic data generator is an important means to perform the sensitivity analysis of a proposed algorithm. Previous works of the association-rule mining usually observe their results by utilizing the synthetic data generator provided in [1]. Recently, the work in [32] seeks the relationship between different itemset lengths to generate more realistic synthetic datasets. Going beyond this, we comment that, a real dataset can be better characterized by giving its *itemset support distribution*. Therefore, the generation of the synthetic transactional datasets by the generators [1,32] can be further modified by considering the characteristics in their *itemset support distributions*.

1.1.6 Determine the sufficient sample size

Random Sampling has been shown to be an effective means to improve the efficiency of mining frequent patterns at the cost of model accuracy [24,34,41]. Clearly, the sample size controls the compromise between mining efficiency and model accuracy. In the literature, *progressive sampling* is the state-of-the-art solution to determine an appropriate sample size [30,31]. Specifically, *progressive sampling* iteratively executes the frequent-pattern mining on samples whose sizes are progressively increased, and the process will be terminated until the mining accuracy is no longer significantly improved. However, such a solution requires multiple executions of

[2] In the literature, solutions to discover top-k frequent patters can obtain frequent patterns without setting the subtle minimum support [7,36]. However, those solutions require to build a complete FP-tree in memory, which is memory-consuming. In general, the most efficient and cost-effective strategy is to determine an appropriate minimum support and then to execute the winner of frequent-pattern mining algorithms, such as algorithms in [5,27,35].

frequent-pattern mining, which is very time-consuming. How to efficiently determine the sufficient sample size for obtaining desired model accuracy remains a challenging issue. To achieve this, we propose an efficient mechanism to determine the sufficient sample size with the help of algorithm PPL. We will realize the idea in this paper, and the details are given in Sect. 4.

1.2 Our contributions

Our contributions in this paper are to solidly study issues related to the power-law relationship in the *itemset support distribution*. More precisely:

(1) We first formalize the problem of the *itemset support distribution* and explore the important phenomenon that the distribution follows the Zipf distribution. In addition, we also find the self-similar phenomenon exists in the distribution, which provides proper answers of why the power-law relationship appears in the *itemset support distribution*.
(2) To broaden the applicability of our discovery, we also present a valid and cost-effective algorithm, called algorithm PPL, to identify characteristics of the *itemset support distribution* without the need of discovering lots of itemsets in advance.
(3) We propose novel mechanisms with the help of PPL to solve two important problems: (a) determining a subtle parameter for mining approximate frequent itemsets over data streams; and (b) determining the sufficient sample size for mining frequent patterns.
(4) We complement our analytical and algorithmic results by a thorough empirical study on real data and demonstrate the prominent advantage of algorithm PPL to be an effective pre-processing means for various mining applications.

This paper is organized as follows. Section 2 formalizes our problem and explores the power-law relationship and self-similarity in the *itemset support distribution*. In Sect. 3, we give the design of algorithm PPL to efficiently and correctly retrieve characteristics of the *itemset support distribution*. The experimental results, including two application studies, are shown in Sect. 4. Finally, this paper concludes with Sect. 5.

2 Power-law and self-similarity

2.1 Review of the power-law relationship and self-similarity

Since the first observation of the power-law relationship in [43], which discovered the frequency of the nth most-frequently-used word in the natural language is approximately inversely proportional to n, the power-law relationship has been successively discovered in many real data, including WWW cache [6], Internet topology [16] and economics,[3] to name a few.[4] Specifically, the power-law relationship can be characterized by several mathematical models, including the well-known Zipf distribution and its variations such as the DGX distribution [4]. Among them, the Zipf distribution is the most widely used form due to its simplicity, as shown by $f_i \propto \left(1/r_i^{\phi}\right)$, where f_i denotes the frequency of words (users, events,...) that are ranked as the r_i^{th} most frequent words (users, events,...) in the dataset, and ϕ is the parameter characterizing the skewness of the distribution. In practice, the Zipf distribution can be further extended to characterize the "count-frequency" relationship, which is stated as $f_i \propto \left(1/c_i^{\phi}\right)$, where f_i is the count of distinct words that appear c_i times in the dataset [4]. Without loss of generality, we discuss the "count-frequency" relationship in the sequel because the "count-frequency" relationship can be deemed as a kind of the probability density function, which is more desirable.

In essence, the Zipf distribution is often demonstrated by scatterplotting the data with the x-axis being $\log(c_i)$ and the y-axis being $\log(f_i)$. The distribution is deemed following the power-law relationship if the points in the log–log plot are close to a single straight line, as shown by

$$\log(f_i) = \theta \log(c_i) + \Omega. \tag{1}$$

Formally, the slope θ and the Y-intercept Ω in Eq. 1 can be estimated by the linear regression:[5]

$$\theta = \frac{\sum_{i=1}^{k} \log(c_i)\log(f_i) - \frac{\left(\sum_{i=1}^{k}\log(c_i)\right)\times\left(\sum_{i=1}^{k}\log(f_i)\right)}{k}}{\sum_{i=1}^{k}\log^2(c_i) - \frac{\left(\sum_{i=1}^{k}\log(c_i)\right)^2}{k}}; \tag{2}$$

$$\Omega = \frac{\sum_{i=1}^{k}\log(f_i)}{k} - \theta \times \frac{\sum_{i=1}^{k}\log(c_i)}{k}, \tag{3}$$

where k denotes the number of points in the log–log plot. Note that the linear regression technique is a method based on the least-square errors. The correlation coefficient (or said

[3] In economics, the power law in slightly different form is known as Pareto's principle or the famous 80–20 rule [23].

[4] See the power-law references in mining applications in [15] and in different domains in http://www.nslij-genetics.org/wli/zipf/.

[5] Other measurements to estimate the parameters of the power-law distribution include the non-linear regression and the maximum likelihood estimation. Among them, the linear regression is the most widely utilized approach due to its feasibility and simplicity.

the *goodness of fit* of the regression line) can be utilized to examine whether those points in the log–log plot exactly lie in the line $\log(f_i) = \theta \log(c_i) + \Omega$ or not [33]. For convenience of discussion, we postpone the formula of the correlation coefficient to Eq. 6 in Sect. 3.3. In addition, for details of the regression technique, which is out of scope for this paper, the reader is asked to follow the pointers in some well-known materials such as [33].

Note that previous observations mostly concentrate on the power-law relationship in the distribution consisting of single events, e.g., single words or single items [6,43]. Naturally, it is important to investigate whether the prevalent power-law relationship also appears in the support distribution of units consisting of a set of words or items. Such cases were first investigated in the computational linguistics literature [14], where the power-law relationship of N-grams had been demonstrated (N-grams denote phrases consisting of N consecutive words). Their studies show that the "count-frequency" relationship of N-grams (with a fixed N) approximately follows the Zipf distribution.

Self-similarity, which has also been observed in various real data including Ethernet traffic and WWW traffic [12,13,15,37], is another natural phenomenon related to the power-law relationship. In general, self-similarity is one type of *fractal*: an object whose appearance is unchanged regardless of the scale at which it is viewed. There are many ways that data can be viewed as self-similar. Among them, a data series is said statistically self-similar if the distribution viewed at varying scaling is the same as that of the original. Followed by the discovery of self-similarity, many phenomena, such as *long-range dependence* and *heavy-tailed distributions*, have also been discovered. Formally, the existence of self-similarity creates new spectrums in various research areas, and the system design with taking self-similarity into account is likely to function in more proper ways. Note that our focus is not on establishing the theoretical model of self-similarity in the *itemset support distribution*; instead we concentrate on examining the existence of self-similarity in the *itemset support distribution* and discussing its consequences. Interested readers can see [3,37] for the theoretical discussion.

2.2 Observations on the power-law relationship in itemset support distribution

In this paper, our first goal is to investigate whether the power-law relationship appears in the distribution of itemset supports in real datasets, where an itemset complies with the definition in [1]. Specifically, let $\mathcal{I} = \{x_1, x_2, \ldots, x_m\}$ be a set of distinct items in the dataset. A set $X \subseteq \mathcal{I}$ with $k = |X|$ is called a k-itemset or simply an itemset. Let the support of an itemset X in the database D be the fraction of transactions in D that contain X.[6] We would like to investigate whether the support distribution of itemsets follows the Zipf distribution, as the form shown by

$$\log(f_i) = \theta \log(s_i) + \Omega, \tag{4}$$

where s_i denotes the support of itemsets and f_i denotes the frequency of itemsets whose supports are s_i. Note that the "support-frequency" relationship in Eq. 4 is physically equivalent to the "count-frequency" relationship. For interest of space, we show the investigation on four retail-like datasets, which are skewed, sparse, and most association-rule algorithms are designed for such types of data [42].

Table 1 Parameters of real datasets

Dataset	I_s	$\|D\|$	$T_{\max}$	T_{avg}
BMS-POS	1,657	515,596	164	6.5
Retail	16,470	88,162	76	10.3
3C_chain	130,108	8,000,000	87	5.4
Book	12,082	100,000	13	2.3

To examine whether the support distribution of itemsets in retail datasets follows the Zipf distribution, four real datasets are investigated in this paper, including two well-known retail benchmark datasets,[7] and two transaction datasets from a 3C chain store and a large book store in Taiwan. Those datasets are summarized in Table 1, where I_s denotes the distinct items in the dataset, $|D|$ denotes the number of transactions, $T_{\max}$ denotes the maximum itemset length and T_{avg} denotes the average itemset length. Furthermore, we execute algorithm FP-growth downloaded from Christian Borgelt's website[8] to obtain itemsets with their supports. Since the number of all itemsets is extremely large (there are $2^{I_s} - 1$ possible itemsets at most), it is very difficult to discover all itemsets in reasonable execution time. For efficiency reasons, we did not retrieve all itemsets in each dataset, but instead retrieve itemsets whose support counts exceed 30, where 30 is a sufficient number in the statistical sense [33].

The observations are shown in Fig. 2, where the curve of the *original support distribution* presents the log–log relationship of the itemset support versus the number of itemsets with the corresponding support (the curve of the *quantized support distribution* will be discussed in the next section). As can be seen, the log–log plot is very Zipf-like, meaning that the power-law relationship indeed appears in the distribu-

[6] The support is considered as the *relative* occurrence frequency. Note that it is defined in some literature as the *absolute* one, i.e., the occurrence frequency in the database.

[7] Downloaded from the website of the ICDM workshop on Frequent Itemset Mining, 2003: http://www.fimi.cs.helsinki.fi/data/.

[8] http://www.fuzzy.cs.uni-magdeburg.de/~borgelt/fpgrowth.html.

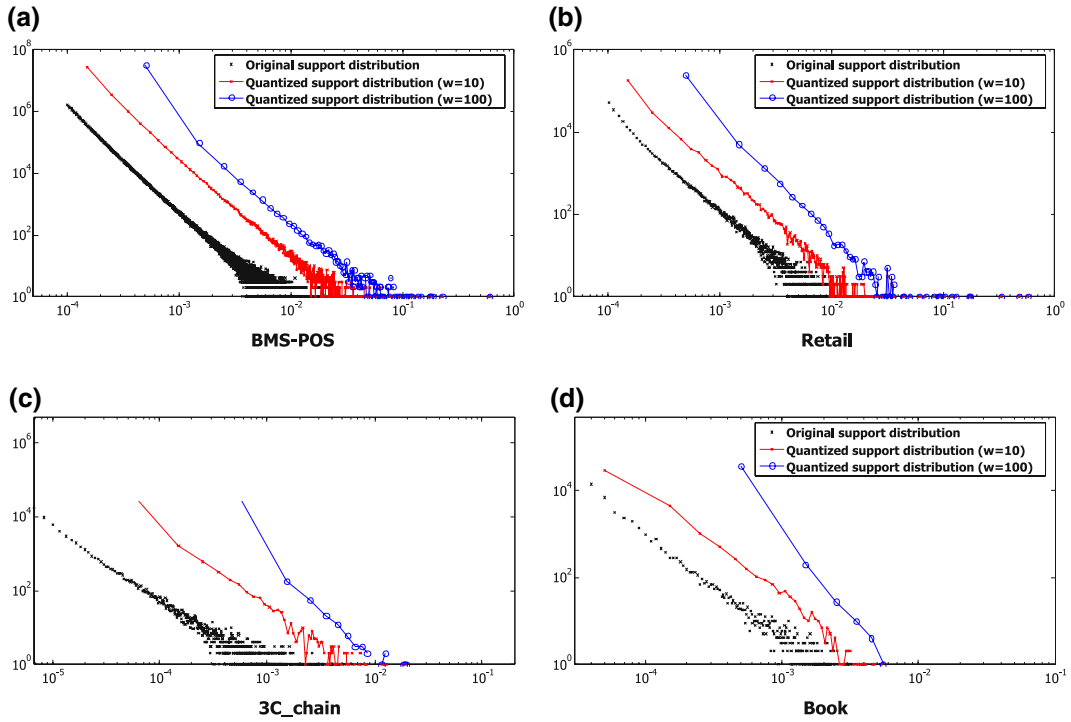


Fig. 2 The support distribution of four real datasets

tion of the itemset support. In addition, the "top-concavity" [9] phenomenon, which is prevalent in the distribution of single items [4,43], is insignificant in the *itemset support distribution*. As such, the Zipf distribution is enough to correctly characterize the power-law relationship in the *itemset support distribution*. We accordingly demonstrate the fact that the power-law relationship appears in the *itemset support distribution*.

2.3 Observations on self-similarity in itemset support distribution

Our observations in Sect. 2.2 are surprising and informative, which show that the power-law relationship exists in the support distribution consisting of all itemsets. Note that previous works mainly reported the existence of the power-law relationship in the support distribution consisting of single events [6,43]. Is there any possible reason to link up previous studies and our investigation? Surprisingly, the self-similar phenomenon can provide a proper answer. We begin the discussion by examining the support distribution consisting of all itemsets with a certain itemset-length. The observations are shown in Fig. 3, where "*i-itemset support distribution*" represents the support distribution consisting of all itemsets with the itemset-length equal to i. We arbitrarily select three itemset-lengths shown in a figure. Clearly, the result in Fig. 3 shows that the support distribution of itemsets with a certain itemset-length also follows the power-law distribution[10] and the support distribution of a larger itemset-length has a sharper slope. The informative result supports a conclusion: *the power-law phenomenon among the supports of single events also appears in the support distribution of their i-combinations for any i*. This is *statistically self-similar* in nature, as analyzed in Remark 1.

Remark 1 Let the database $D = \{t_1, t_2, \ldots, t_d\}$, and $t_i = \{x_{(i,1)}, x_{(i,2)}, \ldots, x_{(i,m_i)}\}$, $1 \leq i \leq d$. Clearly, the tuple t_i contains $C_2^{m_i}$ 2-itemsets, which can be aggregated as $C_1^{m_i}$ 1-items. Accordingly, we have $\sum_{i=1}^{d} C_2^{m_i}$ 2-itemsets in D, which can also be aggregated as $\sum_{i=1}^{d} C_1^{m_i}$ 1-items. It implies that, the distribution of 1-items is an aggregated version with respect to the distribution 2-itemsets. Note that the essence of statistical self-similarity is that the subject shows the same pattern in the distribution, regardless of the scale of the investigation. As such, it can be proved by induction that the *i-itemset support distribution* is an aggregated version with respect to *$(i + j)$-itemset support distribution* for $j \geq 1$. Since *i-itemset support distribution* follows the power-law distribution for any i as shown in Fig. 3, we conclude that

[9] The "top concavity" phenomenon refers to that the top part of the log–log curve tilts vertically (with relatively concave shapes).

[10] Despite the similar purpose of this observation and the observation of N-grams [14], they are inherently different since N-grams are phrases consisting of N *consecutive* words but i-itemsets are itemsets consisting of i *arbitrary-position* items.

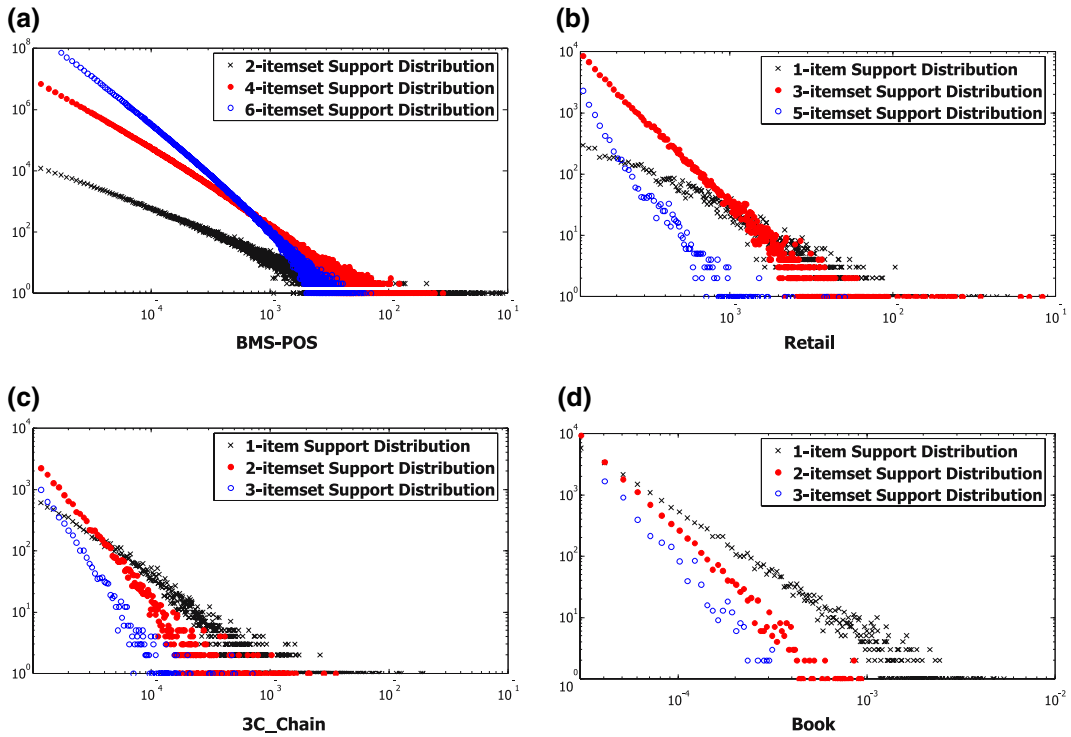


Fig. 3 The support distribution consisting of itemsets with a certain itemset-length

statistical self-similarity exists in the *i-itemset support distribution* for any i.

The existence of self-similarity in the *i-itemset support distribution* is a new discovery. It also provides the evidence for the existence of the power-law phenomenon in the *itemset support distribution*. We further investigate into Fig. 3 and show in Fig. 4 the comparison between the *itemset support distribution* and *i-itemset support distributions* with relatively large itemset-lengths. It is clear to see that, in the log–log plot, the slope of the *i-itemset support distribution* is progressively close to the slope of the *itemset support distribution* as the itemset-length increases. In addition, while the itemset-length exceeding a certain degree, these slopes of the log–log *i-itemset support distributions* are almost the same as that of the log–log *itemset support distribution*. This property leads to Lemma 1 below.

Lemma 1 *Suppose that self-similarity exists in the i-itemset support distribution, which follows the Zipf distribution for any i. The itemset support distribution consisting of all itemsets also follows the Zipf distribution.*

Proof Suppose that f_{ij} denotes the count of i-itemsets whose supports are equal to c_j, where $0 \leq c_j \leq 1$. Since i-*itemset support distribution* follows the Zipf distribution, we have $f_{ij} = s_j^{\theta_i} \times e^{\Omega_i}$ according to Eq. 1, where θ_i and Ω_i denote the slope and the intercept in the log–log *i-itemset support distribution*, respectively. Clearly, the count of itemsets $\widehat{f}_j$ with supports equal to c_j in the *itemset support distribution* is $\widehat{f}_j = \sum_i f_{ij} = \sum_i s_j^{\theta_i} \times e^{\Omega_i}$. As shown in Fig. 4, we know slopes of the log–log *i-itemset support distributions* with relatively large itemset-lengths are almost the same. Let this slope be denoted by $\theta_{\max}$. We have

$$\begin{aligned} \widehat{f}_j &= \sum s_j^{\theta_i} \times e^{\Omega_i} \approx c_j^{\theta_{\max}} \times \left(\sum e^{\Omega_i}\right) \\ \log(\widehat{f}_j) &\approx \theta_{\max} \log(s_j) + \log\left(\sum e^{\Omega_i}\right). \end{aligned} \tag{5}$$

Comparing Eqs. 1 and 5, we can conclude that the *itemset support distribution* follows the Zipf distribution.[11] □

Lemma 1 provides the proper cause why the power-law relationship exists in the *itemset support distribution*. Note that in this section, we show the evidence that the power-law relationship and self-similarity exist in the *itemset support distribution*. Knowing the power-law phenomenon can provide strategies for solving many challenging problems in the frequent-pattern mining which cannot be solved by traditional methods. We will describe these issues in the following sections.

[11] The count of i-itemsets with a small itemset-length is neglected in Eq. 5, since the count of i-itemsets with a relatively large itemset-length is in orders of magnitude larger than that with a small itemset-length.

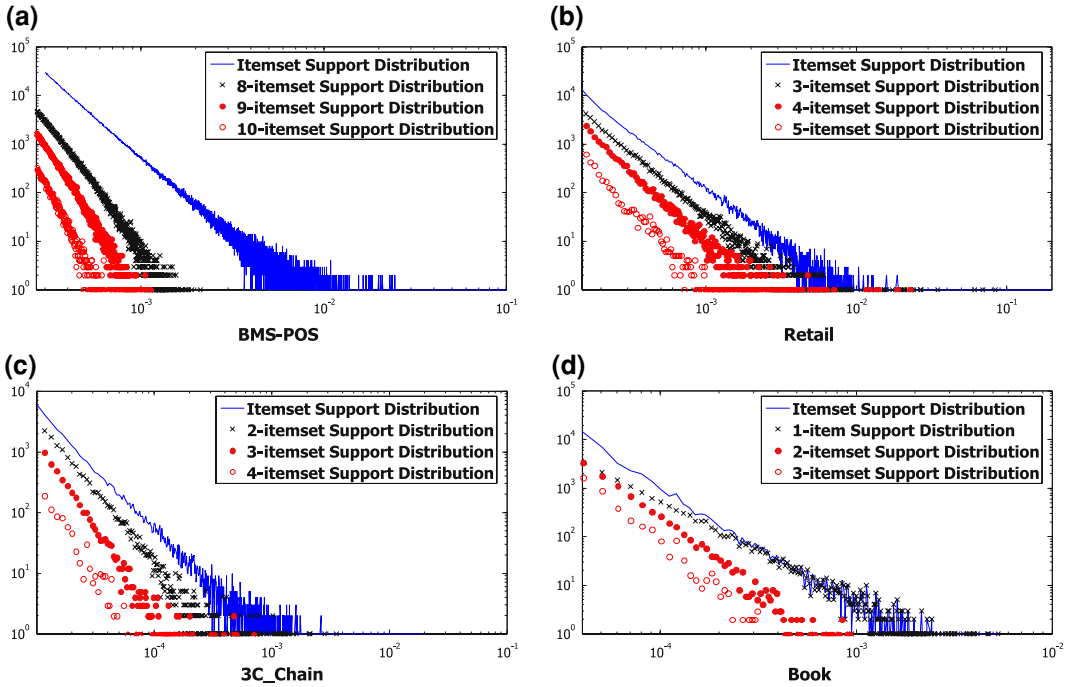


Fig. 4 The comparison between the support distribution and i-itemset support distributions

3 Design of algorithm PPL

As mentioned in Sect. 1.1, recognizing characteristics of the *itemset support distribution* can provide a great benefit to proper mining system designs. However, it is prohibitively expensive to find all itemsets and further estimate the characteristics of the power-law relationship in the *itemset support distribution*, i.e., the slope θ and the Y-intercept Ω in Eqs. 2 and 3. A naive approach to retrieve all itemsets without the support pruning will result in extremely large time and memory consumption (there are $2^m - 1$ possible itemsets at most). To broaden the applicability of our discovery, an efficient approach is required to correctly estimate these parameters.

As a consequence, we propose in this section a valid and cost-effective solution, named PPL, to estimate the parameters of the power-law relationship in the *itemset support distribution*. Formally, time and memory consumption is required if all itemsets with their supports need to be retrieved. To reduce required resources, two conventional methods can be utilized in our case. The first method is to use *sampling* [34]. Another method is to use support pruning techniques [29], where *only the set of high-support itemsets* will be retrieved so as to efficiently discover the parameters of the power-law relationship from the partial set of itemsets. Specifically, to fully utilize the capability of these two approaches, algorithm PPL is decomposed into three phases: (1) sampling; (2) obtaining high-support itemsets; and (3) estimating the parameters of the power-law relationship by the linear regression technique.

However, three challenges will be faced inherently:

(1) *The support distribution obtained in a sample will deviate from the support distribution in the original database*. Note that after sampling, the support of an itemset in the sample may be different from its support in the original dataset [41]. As pointed out in [8,41], the number of itemsets with the support s_j is likely to *increase* after sampling, which can be clarified by the illustration in Fig. 5a. Specifically, according to the *sampling distribution*[12] shown in Fig. 5a, the shadow region is equal to the probability that an itemset with support s_s in the original dataset changes its support to s_j in a sample. Correspondingly, an itemset with support s_j in the original data has a similar probability of changing its support to s_s after sampling. Since the number of itemsets with support s_s is larger than that with support s_j, the moving-in itemsets will be more than the moving-out itemsets with respect to s_j. Clearly, the number of itemsets with support s_j in a sample is thus larger than that in the original dataset. The point is demonstrated in our empirical studies in Fig. 6a, c, where the support distributions obtained in the original dataset and

[12] Suppose that we repeatedly generate a lot of samples of the same sample size. The distribution of the *support* of X among these samples is referred to as the *sampling distribution* of the support of X. Formally, the *sampling distribution* will approximately follow a *normal* distribution with *mean* equal to the support of X in the entire dataset, and its variance depends on the sample size [10].

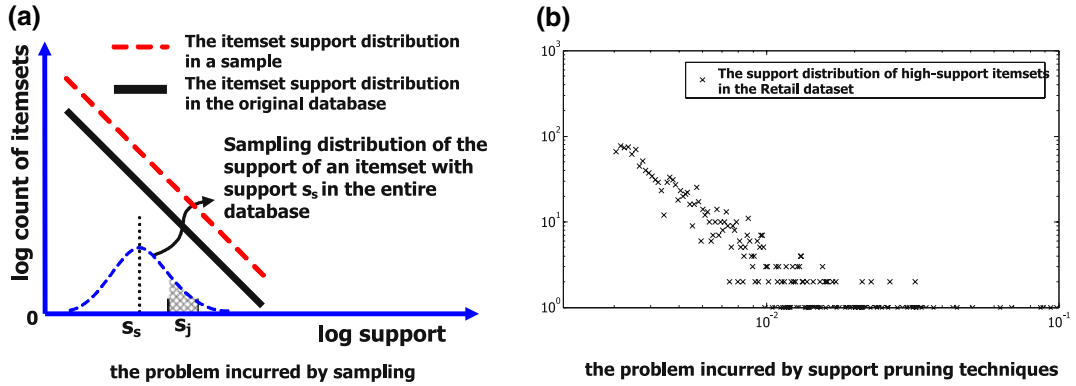


Fig. 5 The problems incurred by two traditional methods in estimating the power-law relationship

in a random sample with 20,000 tuples are included. It can be seen that the support distribution in the sample deviates from that in the original dataset. In practice, due to randomness, we cannot precisely estimate the deviation between the support distribution in a sample and that in the original dataset. As such, utilizing sampling will increase difficulty in correctly predicting the characteristics of the support distribution.

(2) *Without prior knowledge, it is difficult to determine the appropriate minimum support*. Note that same as previous frequent-pattern mining works, we do not know how to determine the subtle minimum support. A large minimum support will result in too few itemsets, which cannot provide sufficient information to correctly estimate the parameters of the Zipf distribution. In contrast, the small minimum support will generate a lot of itemsets, resulting in inefficiency.

(3) *It is difficult to obtain the desired regression line due to the support fluctuation on high-support itemsets*. Consider the observation in Fig. 5b, where a solid straight line represents the regression line over all points with respect to high supports, and the dotted lines show the envelope of the support fluctuation. As can be seen, points with respect to high supports do not exactly follow the Zipf distribution, and the support distribution of these points has the large support fluctuation. Since we only retrieve high-support itemsets, we inevitably face the challenges from the support fluctuation.[13] Without a proper design, the regression line over points with respect to high supports may deviate from the desired regression line.

To overcome those challenges, several novel mechanisms will be devised in algorithm PPL. In the following, we perform step-by-step analysis to discuss the details.

3.1 Phase I: sampling

The goal of Phase I is to select a sample from the original dataset. Note that as mentioned in the first challenge described above, the support distribution in a sample will deviate from that in the original dataset, and the deviation is unpredictable. In fact, this phenomenon can be significantly reduced in the *quantized support distribution*, which will be obtained by the *histogram* technique [22]. Explicitly, all itemsets can be aggregated by means of the traditional equi-width *histogram* and then obtain the *quantized support distribution*. We give the formal definition of the *quantized support distribution* below.

Definition 1 (The quantized support distribution) Given all points (s_i, f_i)in the original support distribution, where f_i denotes the count of itemsets whose supports are s_i. After aggregating those points by means of the equi-width histogram, a set of new points $(\widehat{s_j}, \widehat{f_j})$will be obtained, where $\widehat{s_j}$ denotes the representative value (the default is the median value) of the support range corresponding to the jth bucket in the histogram, and $\widehat{f_j}$ denotes the count of itemsets with supports falling in the jth bucket. The quantized support distribution is the distribution consisting of all points $(\widehat{s_j}, \widehat{f_j})$.

The argument that the *quantized support distribution* is able to significantly reduce the influence of support-deviation follows the observation below:

Observation Note that the *sampling distribution* of the support of an itemset X will approximately follow a *normal* distribution with *mean* equal to the support of X in the entire dataset and its variance depends on the sample size [10]. As shown in Fig. 7a, the shadow region represents the error probability that an itemset with support s_i changes its support

[13] In [6], the slope of the log–log plot is obtained by using the linear regression, excluding the rightmost 100 points to avoid the serious effect of the fluctuation. However, such an approach will fail in our cases since we may only have a few rightmost points, which are summarized from high-support itemsets and may be less than 100 points.

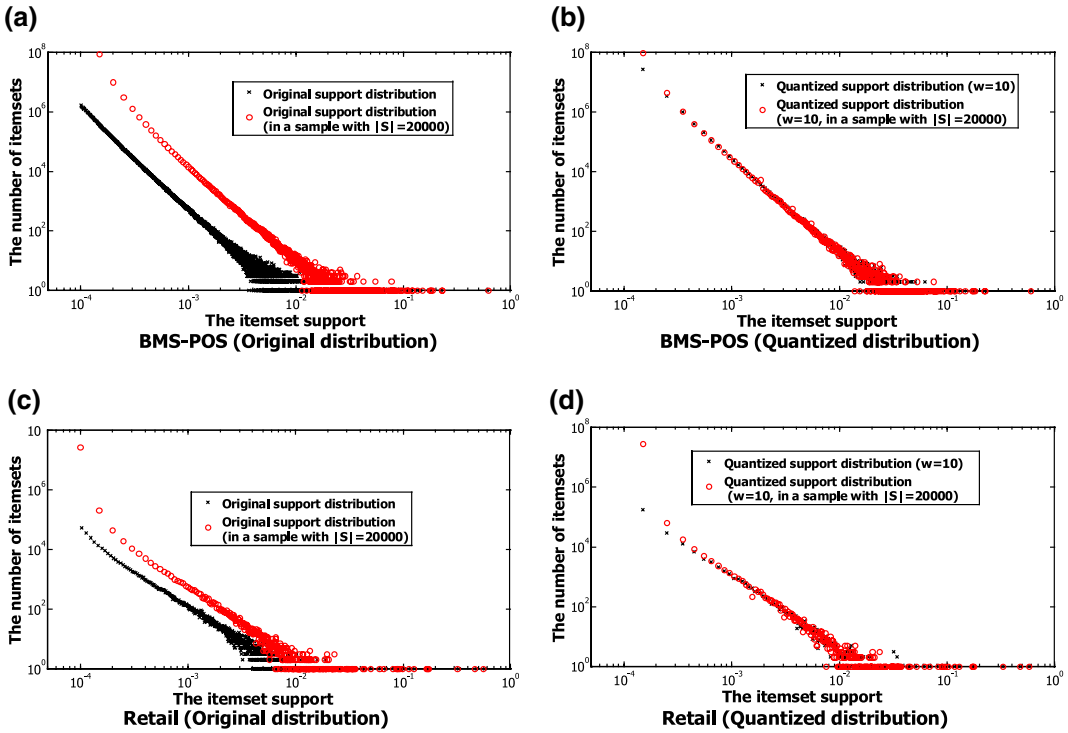
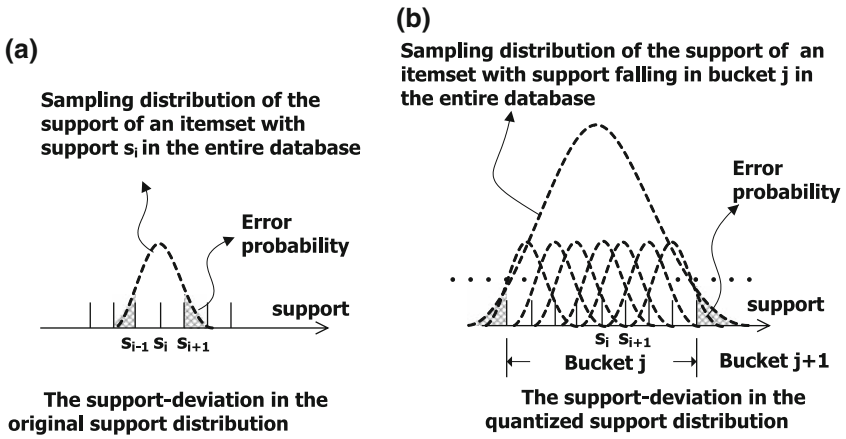


Fig. 6 The original support distribution and the quantized support distribution

Fig. 7 Influence of the support-deviation



after sampling. On the other hand, consider the case of the *quantized support distribution*, as shown in Fig. 7b. The error probability that itemsets with supports in bucket j change to either bucket $j-1$ or bucket $j+1$ after sampling will be relatively small as compared to the error probability illustrated in Fig. 7a. That is, the supports of most itemsets are likely to remain in the same support bucket after sampling. In other words, only itemsets with supports in the margin of a bucket are likely to have the support not falling in the same bucket after sampling. This argument is demonstrated in Fig. 6b, d, where the *quantized support distributions* obtained in the original dataset and in the sample with 20,000 tuples are shown and the parameter w denotes the number of aggregated points. It is clear to see that the *quantized support distribution* in a sample will be close to the *quantized support distribution* in the original dataset.

Following the observation, we comment that the *quantized support distribution* is not sensitive to the support-deviation. That is, *the quantized support distribution in the sample will be close to the quantized support distribution in the entire dataset.* As such, we turn to obtain the *quantized support distribution* in the sample.

However, recalling the observation in Fig. 2, the *quantized support distribution* still deviates from the original support distribution. Importantly, assuming that the original support distribution approximately follows the Zipf distribution,

Lemma 2 below indicates that the *quantized support distribution* also has the same slope as the slope in the original support distribution and has a "predictable" drift of the Y-intercept.

Lemma 2 *Suppose that the itemset support distribution follows the Zipf distribution so that we have* $\log(f_i) \approx \theta \log(s_i) + \Omega$ *.Assuming that there are* w *distinct points in the original support distribution being aggregated as a point in the quantized support distribution, we will have an approximate Zipf distribution as the form*

$$\log(\widehat{f_k}) \approx \theta \log(\widehat{s_k}) + \Omega + \log(w),$$

in the quantized support distribution, where $\widehat{s_k}$ *denotes the representative of the quantized support in the* k^{th} *bucket and* $\widehat{f_k}$ *denotes the count of itemsets whose supports fall in the* k*th bucket. As such, the log–log plot in the quantized support distribution has the slope* θ *and the Y-intercept* $\Omega + \log(w)$*.*

Proof Suppose that $|D|$ is the database size. Let points $(s_{k1}, f_{k1}), (s_{k2}, f_{k2}), \ldots, (s_{kw}, f_{kw})$ be summarized as the kth point $(\widehat{s_k}, \widehat{f_k})$ in the *quantized support distribution*. Note that we have $e^{\Omega} \times s_{ij}^{\theta} \approx f_{ij}$ since it follows the Zipf distribution. Clearly, we have $\widehat{f_k} = \sum_{j=1}^{w} f_{k,j}$, and

$$\widehat{s_k} = \frac{s_{k1} + s_{kw}}{2} = \frac{s_{k1} + \left(s_{k1} + \frac{w}{|D|}\right)}{2} = s_{k1} + \frac{w}{2 \times |D|}.$$

Since $\frac{w}{|D|}$ is in general much small as compared to s_{k1}, we have

$$\frac{s_{kj}^{\theta}}{\widehat{s_k}^{\theta}} = \left(\frac{s_{k1} + \frac{j}{|D|}}{s_{k1} + \frac{w}{2 \times |D|}}\right)^{\theta} \approx 1.$$

Therefore s_{kj}^{θ}, for $1 \le j \le w$, will be approximately equal to $\widehat{s_k}^{\theta}$, which yields that

$$\widehat{f_k} = \sum_{j=1}^{w} f_{kj} \approx e^{\Omega} \times \sum_{j=1}^{w} \widehat{s_k}^{\theta} = e^{\Omega} \times w \times \widehat{s_k}^{\theta}$$

$$\log(\widehat{f_k}) \approx \theta \log(\widehat{s_k}) + \Omega + \log(w). \qquad \square$$

Lemma 2 indicates that the slope θ remains the same in the *quantized support distribution*, and the Y-intercept will be changed to $\Omega + \log(w)$. Figure 2 demonstrates the success of Lemma 2, where we can see that, for high-support points, the slope of the *quantized support distribution* ($w = 10$ or 100) is equal to that of the *quantized support distribution* without sampling. As a result, the side-effect of sampling is overcome.

For ease of reference, we summarize the process to overcome problems incurred by sampling, as shown in Fig. 8:

(1) Obtain the characteristics of the *quantized support distribution* in the sample.
(2) The characteristics of the *quantized support distribution* in the whole dataset are expected equal to that in the sample.
(3) Obtain the characteristics of the original *itemset support distribution* according to Lemma 2.

Once Step 1 is complete, Step 2 and Step 3 can be straightforwardly executed with the mathematical manipulation mentioned above. How to precisely achieve Step 1 will be discussed in Sects. 3.2 and 3.3.

The remaining issue in this phase is, what is the appropriate sample size to obtain the *quantized support distribution* which can be consistent with that in the entire database. Formally, the level of consistency depends on the variance of the *sampling distribution* of the support, and the variance depends on the sample size [34]. A small sample size will lead to a large variance as compared to the variance in a large sample size. As pointed out in previous works of sampling for mining association rules, a sample size either equal to 20,000 [34] or a sample rate equal to 10% [41], can roughly generate a satisfactory set of frequent itemsets. We comment that the sample size 20,000 or 10% is also sufficient to generate the accurate *quantized support distribution* by following several points: (1) The level of complexity to generate the accurate *quantized support distribution* is not higher than to the level of complexity to generate accurate frequent itemsets; (2) in Phase II only high-support itemsets will be generated, whose supports, as indicated in [34], can be easily preserved in samples as compared to supports of low-support itemsets; (3) the difference between counts of itemsets within a certain bucket in a small sample and in the entire dataset is negligible in the log–log scale (note that the characteristics of the power-law relationship is estimated in the log–log scale); (4) the technique devised in Phase III is robust to the difference between *quantized support distributions* in the sample and in the entire dataset.

As simultaneously considering execution efficiency and above points, we therefore set the sample size as 20,000 in default since the sample can be easily executed and maintained in main memory. The discreet users can set the size as 10% of the entire size, as the suggestion in [41]. We will also investigate the issue of the sample size in our empirical studies later.

3.2 Phase II: discover high-support itemsets in the sample

In this phase, the high-support itemsets in the sample will be discovered. Without prior knowledge to determine the appropriate minimum support, we resort to the technique of "*discover top-k itemsets*" [7,36] instead of "discover itemsets with the specified minimum support," where *top-k* itemsets

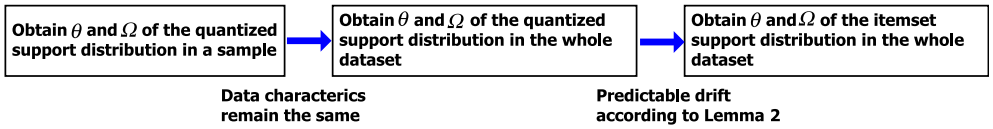


Fig. 8 The flow to overcome problems incurred by sampling

refer to the k most frequent itemsets in the dataset. In practice, the size of k can be easily specified a priori. As will be shown in our experimental results, k equal to 5,000 will suffice to correctly estimate the parameters of the power-law relationship in most cases. As such we set k as 5,000 in default, where top 5,000 itemsets can be efficiently retrieved by the state-of-the-art algorithm for mining *top-k* frequent itemsets. Previous works to discover *top-k* frequent itemsets include [7,36]. Formally, those works shoot for discovering *top-k* itemsets with specified constraints such as discovering *closed* itemsets [7]. Therefore directly extending these solutions to discover *top-k* itemsets (without those specific constraints) will lead to inefficiency since their pruning techniques will be infeasible.

Recent advances in the literature has proposed a new approach, called the MTK algorithm, to identify *top-k* frequent itemsets within an upper constraint of the memory usage [9]. Specifically, MTK is devised as a candidate generate-and-test manner similar to the Apriori algorithm, except the strategy of the candidate generation. With the help of the theory to predict the upper number of the generated candidates [17], MTK is able to decide the set of candidates that will be generated in the next database scan by satisfying two properties (1) a candidate so generated is highly possible to be included in *top-k* frequent itemsets; (2) the total used memory is constrained below a user specified upper memory size.

In fact, applying the MTK algorithm as Phase II of PPL still faces a challenge that multiple database scans may be required, which will lead to execution inefficiency. Fortunately, the input database of the MTK algorithm is not too large to be maintained in the memory since Phase I of PPL has generated a small sample for further use. As such, all operations in Phase II, including the database scan and the candidate generation, can be accomplished in an efficient and memory-resident manner without extra I/O, thus achieving high efficiency.

It is worth commenting that the MTK algorithm is the better approach for our need rather than BOMO [7] and TFP [36] due to an additional reason: as shown in [9], the MTK algorithm outperforms BOMO and TFP while k is not so large ($k < 100{,}000$). Since $k = 5{,}000$ is adequate to provide necessary information for the identification of power-law parameters, it is clear that MTK results in best efficiency in this setting. For interest of space, the details of algorithm MTK are not described in this paper, and interested readers can refer to [9] for the details.

3.3 Phase III: characterize the power-law relationship

The parameters of the Zipf distribution will be estimated in this phase. Suppose that $\{X_1, \ldots, X_k\}$ is the set of *top-k* itemsets which are obtained in Phase II. At the beginning of this phase, we will aggregate these itemsets by means of *histogram* with the support bucket width equal to $\frac{w}{|S|}$, where $|S|$ is the size of the sample dataset and w is the number of distinct and consecutive support counts which will be aggregated into the same bucket. Note that the default of w is 10 in this paper since empirically $w = 10$ is able to preserve the slope of the *itemset support distribution*, as shown in Fig. 5b. As such, *top-k* itemsets is aggregated into a set of points $H_k = \{(\widehat{s}_1, \widehat{f}_1), (\widehat{s}_2, \widehat{f}_2), \ldots, (\widehat{s}_z, \widehat{f}_z)\}$ sorted by $\widehat{s}_i$, where $\widehat{s}_i < \widehat{s}_j$ iff $i < j$. Afterward, we can characterize the power-law relationship by performing the regression analysis on H_k.

However, as pointed out as the third challenge described in the beginning of Sect. 3, directly executing the regression analysis over all points in H_k results in the incorrect estimation due to the support fluctuation on high support itemsets. Therefore a problem arises: "*how to select an appropriate subset of points from H_k to correctly estimate the parameters of the Zipf distribution?*" Recall the observation in Fig. 5b. Points with respect to very high-supports usually do not accurately follow the Zipf distribution. On the other hand, points with respect to low supports usually follow the Zipf distribution. As such, a naive approach can be devised as follows.

3.3.1 Naive approach

It is intuitive to suggest the regression line over a subset of leftmost points in H_k since they are likely to correctly fit the power-law relationship. For example, we may estimate the power-law relationship by performing the regression analysis over the leftmost five points in H_k, i.e., $\{(\widehat{s}_1, \widehat{f}_1), \ldots, (\widehat{s}_5, \widehat{f}_5)\}$. Nevertheless, we did not know how many points are sufficient to obtain the desired regression line. We have to examine all possible regression lines, and then select the one with the best correlation coefficient, since it will have the best power to explain the log–log relationship in the Zipf distribution.

However, such an approach suffers from the problem that the best correlation coefficient does not imply the best fit of the Zipf distribution. In particular, leftmost few points may

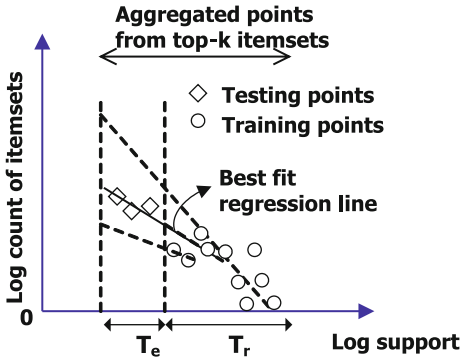


Fig. 9 The illustration of the best fit regression line

result in the best correlation coefficient, but the regression line could be easily bias to outlier points [33]. In addition, the result of the linear regression may be affected by the noise from sampling in Phase I. As such, we further devise a novel solution, which is inspired from the training-and-testing scenario in supervised learning [20], to correctly estimate the parameters of the Zipf distribution from H_k.

3.3.2 Minimizing testing error approach

Let H_k be divided into two distinct subsets, where the first subset, called the testing set T_e, consists of leftmost m points and another subset, called the training set T_r, consists of rightmost $(z-m)$ points. Here z is the number of points in H_k and m is the parameter to adjust the size of the testing set, $m < z$. We have $T_e = \{(\widehat{s}_1, \widehat{f}_1), \ldots, (\widehat{s}_m, \widehat{f}_m)\}$ and $T_r = \{(\widehat{s}_{m+1}, \widehat{f}_{m+1}), \ldots, (\widehat{s}_z, \widehat{f}_z)\}$. Consider the illustration in Fig. 9, where each point in T_e is called a testing point. Our goal is to find the *best fit regression line* from T_r so that all testing points in T_e can well lie in the line. Formally, we give the definition of the *best fit regression line* as follows.

Definition 2 (Best fit regression line) Given the training set T_r and the testing set T_e. The best fit regression line, denoted by $\mathbb{R}_g(\widehat{s}_i) = \widehat{\theta}_g \log(\widehat{s}_i) + \widehat{\Omega}_g$, will satisfy:

(1) $\mathbb{R}_g(\widehat{s}_i) = \widehat{\theta}_g \log(\widehat{s}_i) + \widehat{\Omega}_g$ is the regression line over the leftmost g points in T_r, i.e., $\{(\widehat{s}_{m+1}, \widehat{f}_{m+1}), \ldots, (\widehat{s}_g, \widehat{f}_g)\}$, where $m+1 \le g \le z$.

(2) The correlation coefficient, r_g, over the data points $\{(\widehat{s}_{m+1}, \widehat{f}_{m+1}), \ldots, (\widehat{s}_g, \widehat{f}_g)\}$ is smaller than a predefined threshold δ. Note that r_g is equal to

$$\frac{\sum_{i=m+1}^{g}\sum_{j=m+1}^{g}(\log(\widehat{s}_i)-u_s)\left(\log(\widehat{f}_j)-u_f\right)}{\sqrt{\sum_{i=m+1}^{g}(\log(\widehat{s}_i)-u_s)^2}\sqrt{\sum_{j=m+1}^{g}\left(\log(\widehat{f}_j)-u_f\right)^2}}, \tag{6}$$

where u_s and u_f are the mean of $\log(\widehat{s}_i)$ and $\log(\widehat{f}_j)$, respectively.

(3) $g = \arg\min_u \left\{\sum_{j=1}^{m}\left(\mathbb{R}_u(\widehat{s}_j) - \log(\widehat{f}_j)\right)^2\right\}$, subject to the correlation coefficient $r_f \le \delta$ and $m+1 \le u \le z$.

Ensure: *Best_fit()*:
Input:
Top-k itemsets, X_i, where $i = 1, ..., k$
The number of aggregated support counts, i.e., w
The number of points in T_e, i.e., m
The correlation coefficient threshold δ
Output:
The best fit slope $\widehat{\theta}_g$, and the best fit Y-intercept $\widehat{\Omega}_g$
1. Let $f_i = 0$, where $i = 0,...,\lceil \frac{1}{\Delta} \rceil$; //$\Delta = \frac{w}{|D|}$, the support width of a bucket
2. for $i = 1$ to k; //calculate the count of itemsets in each bucket
3. $f_\alpha = f_\alpha + 1$, where $\alpha = \left\lceil \frac{\sup(X_i)}{\Delta} \right\rceil$ and $\sup(X_i)$ denotes the support of X_i;
4. find Λ, where $f_\Lambda = \min_i \{f_i \neq 0\}$; //the leftmost points in T_e
5. for $t = (\Lambda + m + 1)$ to $\lceil \frac{1}{\Delta} \rceil$ begin
6. $[\Omega_t, \theta_t, r_t]$=linear_reg($(\log(s_j), \log(f_j))$, where $(\Lambda + m + 1) \le j \le t$;
7. $Var_t = \sum_{j=0}^{m}\left[\theta_f \log(s_{\Lambda+j}) + \Omega_t - \log(f_{\Lambda+j})\right]^2$; //the testing error
8. end
9. find g, where $g = \arg\min_u (Var_u)$, subject to $r_u \le \delta$; //Criterion 3 in Def. 2
10. return Ω_g, θ_g with respect to g;

The whole procedure to find the best fit regression line is outlined in Procedure *Best_fit*, where the function linear_reg() will return three parameters, the intercept Ω (see Eq. 3), the slope θ (see Eq. 2) and the *correlation coefficient* r (see Eq. 6). Specifically, the correlation coefficient r_g (a value between -1 and 1) can represent the level how those points are explained by the regression line. The regression line will fit points better when $r_g \to -1$ since without loss of generality, $\widehat{f}_i$ and $\widehat{s}_i$ are negatively correlated. Statistically, it is believed that $r_g \le -0.8$ is sufficient to claim the regression line can explain these points [33]. Thus δ is set as -0.8 in default. Note that Criterion 3 in Definition 2 states that we desire the regression line with the minimum testing error. In addition, algorithm PPL is degenerated into the naive approach if there is no testing point in T_e while simply choosing the regression line with the best correlation coefficient. For comparison purposes, we will also show the result of the naive approach in our experimental results. Note that the best fit regression line will be identified in the partial *quantized support distribution* from *top-k* itemsets discovered in a sample. In light of Lemma 2, the slope and the Y-intercept in the original *itemset support distribution* will be equal to $\widehat{\theta}_g$ and $\widehat{\Omega}_g - \log(w)$, respectively.

We finally summarize the overall flow of algorithm PPL, as shown in Fig. 10: (1) sampling; (2) discover *top-k* frequent

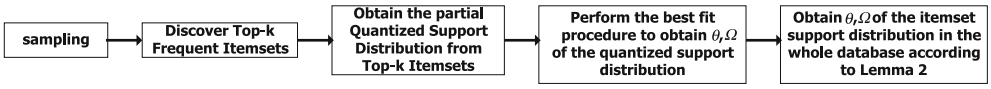


Fig. 10 The flowchart of algorithm PPL

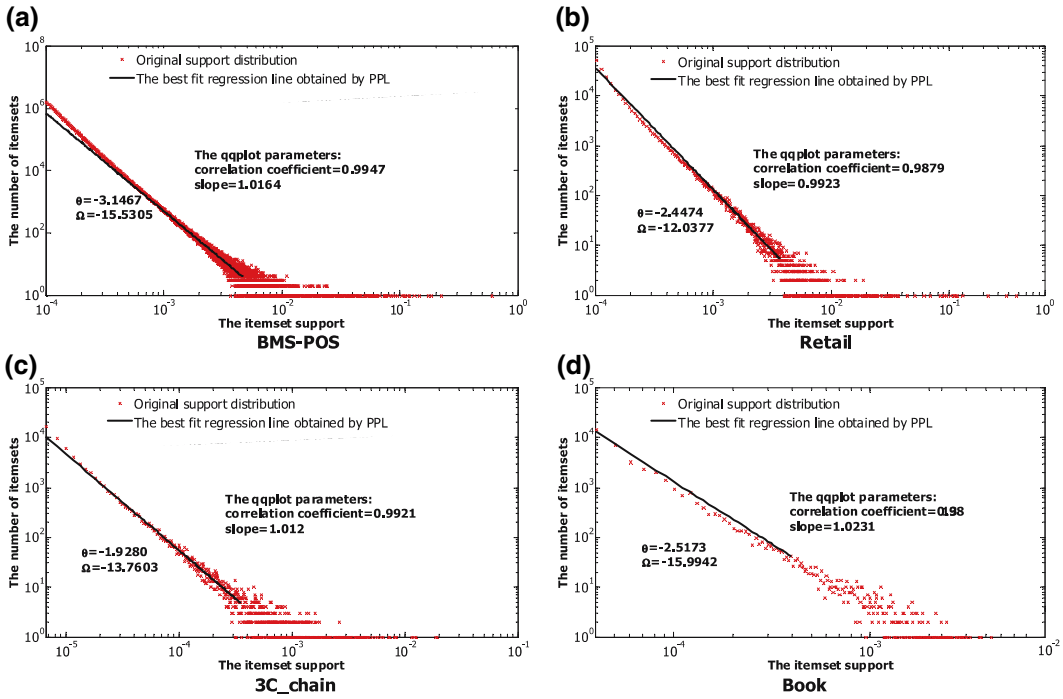


Fig. 11 The results of algorithm PPL

itemsets in the sample; (3) aggregate the support of *top-k* itemsets by means of the equi-width histogram so as to obtain the partial *quantized support distribution*; (4) perform Procedure *Best_fit* to obtain the characteristics of the *quantized support distribution* in the sample; (5) identify the characteristics of the power-law relationship in the *itemset support distribution* in the entire database according to Lemma 2.

4 Experimental studies

In Sect. 4.1, we show the performance studies of PPL to estimate the parameters of the *itemset support distribution*. To demonstrate the applicability of PPL, in Sects. 4.2 and 4.3, we further study two applications out of six applications introduced in Sect. 1. Specifically, in Sect. 4.2 we propose a variant of Lossy-Counting based algorithms [25] with help of PPL for mining approximate frequent itemsets. In Sect. 4.3, we utilize PPL to realize a challenging mining task: determining the sufficient sample size for mining frequent patterns [30]. Note that the reason we select these two applications is that the success of these two applications is fully attributed to the help of the PPL algorithm, whereas the best execution of other four applications needs the integration of other optimization strategies.

4.1 Performance studies of algorithm PPL

The four real skewed datasets described in Table 1 are utilized in our experimental studies. Since the goal to show the support distribution follows the Zipf distribution has been demonstrated in Sect. 2, we in this subsection investigate whether algorithm PPL can efficiently and correctly estimate the parameters of the power-law relationship in the *itemset support distribution*. The simulation is coded by C++ and performed on Windows XP in a 1.7GHz IBM compatible PC with 512MB of memory. The default parameters in the experiments are: (1) $k = 5{,}000$ (*top-k* itemsets); (2) the number of aggregated support counts $w = 10$; (3) the number of points in the training set is equal to 5; (4) the correlation coefficient threshold $\delta = -0.8$; (5) the sample size $|S| = 20{,}000$.

We investigate whether algorithm PPL with the default parameters is able to correctly characterize the power-law relationship in four real datasets. The results are presented in Fig. 11, where the original support distributions and the *best fit regression lines* obtained by algorithm PPL (with their

Dataset	Brute force approach	Algorithm PPL	Efficiency Gain
BMS-POS	632 sec	8 sec	79
Retail	1248 sec	5 sec	249.6
3C_chain	2547 sec	10 sec	254.7
Book	492 sec	3 sec	164

Fig. 12 The execution time of different approaches

slopes θ and Y-intercepts Ω) are shown. Note that the *best fit regression line* is discovered in the quantized support distribution in the sample. As can be seen, the *best fit regression line* can perfectly characterize the Zipf distribution in the four real datasets, showing the effectiveness of PPL.

Furthermore, the execution time is shown in Fig. 12, where the execution time of "Brute force approach" indicates the time to retrieve the original support distribution in Fig. 2 by algorithm FP-growth. Indeed, the brute force approach can correctly determine the parameters of the Zipf distribution by finding most of itemsets, but it will pay for extremely large time and memory consumption. On the other hand, PPL can efficiently estimate the parameters of the power-law relationship by avoiding the expensive process to obtain all itemsets. It is worth mentioning that the efficiency gain in Fig. 12, which is calculated as the execution time of the brute force approach divided by the execution time of algorithm PPL, shows that algorithm PPL is in orders of magnitude faster than the brute force approach. In addition, Fig. 12 also shows that PPL is cost-effective, thus demonstrating the feasibility of PPL to be a pre-processing means for other mining applications.

Same as the experiments in [4], the quantitative analysis of algorithm PPL could be evaluated by the quantile-quantile plot (qqplot), as the one shown in Fig. 13. The qqplot is used to compare the quantiles of two datasets. If the distributions of these two datasets are similar, the qqplot will be linear and the slope will be close to one. As such, we generate a synthetic support distribution according to the parameters estimated by algorithm PPL, and then make a qqplot between the original support distribution and the synthetic support distribution. Afterward, two important factors can be calculated: (1) the slope of the qqplot; (2) the correlation coefficient of points in the qqplot. If these two factors are both close to unity, we can claim that the real distribution and the synthetic distribution are from the same distribution [4], meaning that the regression line can perfectly represent the data distribution.

The qqplots on various correlation coefficient thresholds δ are shown in Fig. 13, where Fig. 13a is the qqplot corresponding to the result of Fig. 11a. We can find that the qqplot in Fig. 13a is close to linear, except points with respect to very low supports and very high supports. Note that Fig. 2 shows that points with respect to high supports in the BMS-POS dataset do not exactly follow the power-law relationship and points with respect to low supports slightly vary from the Zipf distribution, thus causing the deviation of a few points. However, the slope and the correlation coefficient are very close to unity, indicating that the synthetic distribution can almost perfectly fit the real distribution. Furthermore, when we increase the threshold δ, as shown in Fig. 13b, the estimated quality degrades, showing the importance of Criterion 2 in Definition 2. Indeed, a regression line with a low correlation coefficient will lose its effectiveness to estimate the power-law relationship, even though it results in the minimum testing error (Criterion 3 in Definition 2).

For interest of space, other qqplot results of four real datasets are summarized in Fig. 14. At first, we observe results with various numbers of testing points, i.e., $|T_e|$ in Fig. 14. Note that the case $|T_e| = 0$ can be viewed as the naive approach discussed in Sect. 3.3. As can be seen, the naive approach cannot correctly model the distribution since the correlation coefficient and the slope deviate far from unity. On the other hand, $|T_e| = 5$ (default cases) and $|T_e| = 10$ both lead to the desirable result. Moreover, the studies of various δ are also shown, and we can find that $\delta = -0.8$ (default cases) or -0.9 results in the correlation coefficient and the slope both close to one. Note that without loss of generality, the results of $|T_e| = 0$ and $\delta = -0.5$ can be viewed as the case to obtain the regression line over all points from *top-k* itemsets. It can be seen that the regression line over all points loses of its power to explain the real data distribution. The above observations all demonstrate the effectiveness of algorithm PPL.

In addition, with the result of various k, we can conclude that the default $k = 5{,}000$ is sufficient to obtain high quality results. Note that top-5,000 itemsets can be efficiently retrieved in the sample, indicating the efficiency and effectiveness of algorithm PPL. We also investigate the effect of the sample size. Clearly, the result obtained in the sample with the default size 20,000 is close to the result obtained in the large sample with size equal to $0.2 \times |D|$, showing that the resulting quality is not sensitive to the sample size if the sample size is not arbitrarily small. Finally, we observe the result of various w. Note that Lemma 2 may not be valid when w is large. Thus it can be seen that $w = 100$ slightly degrades the estimated quality of algorithm PPL as compared to that in $w = 10$. Since the goal of quantization in this paper is to diminish the side-effect of sampling, we conclude that $w = 10$ is sufficient to achieve this and will give the excellent fit of the *itemset support distribution*.

4.2 Application study: false positive or false negative of frequent itemsets

To better understand the advantage of knowing characteristics of the *itemset support distribution*, we implement the Lossy-Counting based algorithm, denoted by BTS (Buffer-Trie-SetGen), for mining approximate frequent itemsets over

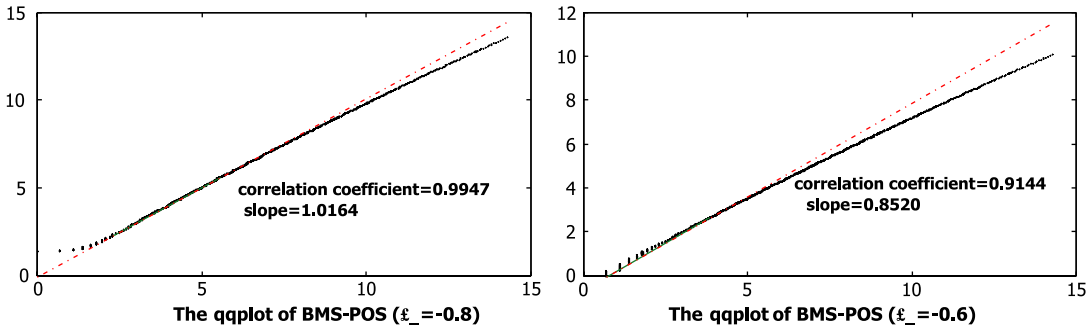


Fig. 13 The qqplot results in BMS-POS with various δ

Fig. 14 The qqplot results of four real datasets

	BMS-POS		Retail		3C_chain		Book	
Variant Parameters	Corr. Coef.	Slope	Corr. Coef.	Slope	Corr. Coef.	Slope	Corr. Coef.	Slope
Default	0.99	1.02	0.99	0.99	0.99	1.01	0.98	1.02
$\|T_e\|$=0 (naive)	0.89	0.73	0.83	0.82	0.91	0.83	0.87	0.93
$\|T_e\|$=10	0.99	0.98	0.98	0.96	0.98	1.01	0.99	1.02
N=-0.5	0.84	0.86	0.91	1.08	0.92	0.94	0.87	1.11
N=-0.9	0.99	1.01	0.99	1.02	0.97	1.06	0.98	1.03
$\|T_e\|$=0; N=-0.5	0.81	1.21	0.77	1.11	0.73	1.18	0.84	1.13
k=10,000	0.98	1.02	0.99	0.97	0.97	1.08	0.98	1.07
k=50,000	0.99	0.99	0.99	1.02	0.98	0.97	0.98	0.94
|S|=10,000	0.93	1.09	0.98	0.97	0.95	1.09	0.98	0.97
|S|=50,000	0.99	0.99	0.99	1.03	0.98	0.99	0.99	1.03
|S|=0.1|D|	0.99	1.03	0.93	1.13	0.99	0.98	0.91	0.94
|S|=0.2|D|	0.99	1.01	0.94	1.04	0.98	0.96	0.94	1.03
w=50	0.98	1.03	0.91	0.94	0.94	1.02	0.96	1.08
w=100	0.93	1.14	0.88	1.13	0.98	1.01	0.92	0.94

data streams [25], and apply algorithm PPL as its pre-processing step.

4.2.1 Background review

BTS is a one-pass algorithm which realizes the ϵ-deficient synopsis of each frequent pattern [25]. That is, BTS guarantees each identified frequent pattern having a support larger than $s - \epsilon$, where s is the specified minimum support and ϵ is an error parameter. Originally, algorithm BTS is *false-positive oriented*: all frequent itemsets in the stream will be retrieved by BTS (100% *recall* rate), but non-frequent itemsets with supports in $[s-\epsilon, s]$ in the stream may be identified as frequent by *BTS* (smaller than 100% *precision* rate). Given a set of true frequent itemsets A and a set of obtained frequent itemsets B, *precision* is defined as $\frac{|A \cap B|}{|B|}$, and another measurement of the quality, i.e., *recall*, is defined as $\frac{|A \cap B|}{|A|}$. For comparison purposes, the work in [39] extends BTS to be a *false-negative oriented* algorithm by deliberately setting the minimum support as $s + \epsilon$. As such, the output contains only those frequent itemsets with support exceeding s but frequent itemsets between s and $s+\epsilon$ may not be included in the output. Indeed, *precision* in the *false-negative* scheme will be equal to 100% while compromising *recall* of the output.

Formally, in works of the ϵ-deficient synopsis, the error parameter ϵ controls the bound of required memory consumption: a larger ϵ leads to smaller memory consumption but incurs a worse frequency approximation; and in contrast, a smaller ϵ results in a better frequency approximation but requires larger memory consumption. Determining a proper decision of ϵ is thus the key to the success of the ϵ-deficient synopsis. Unfortunately, how to decide a proper value of ϵ was fully left unsolved to users in the literature. The default in [25,39] is $\epsilon = 0.1s$, and, however, it is a not proper choice in all situations. Note that users are easy to give desired model accuracy rather than to give the subtle error parameter. As such, a good reference to appropriately determine ϵ can be provided if we execute algorithm PPL in advance:

4.2.2 Suggested enhancement

In practice, while users give the desired *recall* (in the case of false negatives), denoted by r, or the desired *precision* (in the case of false positives), denoted by p, of the result, algorithm PPL can enable the system to automatically determine the proper ϵ. The basic idea is to identify ϵ as the minimum one which satisfies the user desired *recall* and *precision* in such a way that we can achieve desired model accuracy with the smallest memory consumption. Specifically, after performing PPL in an early received substream, two parameters, i.e., the slope θ and the Y-intercept Ω, can be obtained. Accordingly, we can apply the following procedure prior to the execution of algorithm BTS:

Fig. 15 The result of applying algorithm PPL prior to algorithm BTS (in the BMS-POS dataset)

s (%)	Type	Apply PPL	Desired Recall	Desired Precision	α(%)	Obtained Recall	Obtained Precision	Required Memory
0.1	False-Positive	NO	-	-	0.01	1	0.73	53 MB
	False-Positive	YES	1	0.8	0.007	1	0.82	66 MB
	False-Positive	YES	1	0.9	0.002	1	0.91	73 MB
	False-Negative	NO	-	-	0.01	0.96	1	48 MB
	False-Negative	YES	0.9	1	0.014	0.92	1	39 MB
	False-Negative	YES	0.8	1	0.019	0.83	1	33 MB
0.05	False-Positive	NO	-	-	0.005	1	0.68	52 MB
	False-Positive	YES	1	0.8	0.0042	1	0.78	60 MB
	False-Positive	YES	1	0.9	0.0038	1	0.93	71 MB
	False-Negative	NO	-	-	0.005	0.82	1	53 MB
	False-Negative	YES	0.9	1	0.004	0.89	1	57 MB
	False-Negative	YES	0.8	1	0.005	0.82	1	53 MB

(1) Calculate the approximate number of frequent itemsets, f_s, with the given minimum support s, i.e., $f_s = \int_s^1 \left(e^{\Omega} \times x^{\theta}\right) \mathrm{d}x$;
(2) For the false negative case, identify ϵ as $\arg\min_{\alpha} \left\{\frac{f_\alpha}{f_s} \geq r\right\}$, where $f_\alpha = \int_{s+\alpha}^1 \left(e^{\Omega} \times x^{\theta}\right) \mathrm{d}x$. For the false positive case, we will identify ϵ as $\arg\min_{\beta} \left\{\frac{f_\beta}{f_s} \geq p\right\}$, where $f_\beta = \int_{s-\beta}^1 \left(e^{\Omega} \times x^{\theta}\right) \mathrm{d}x$.

4.2.3 Simulation results

We investigate whether algorithm PPL can help algorithm BTS to obtain the desired quality with a proper ϵ. The results are shown in Fig. 15, where the BMS-POS dataset is applied. We specify the desired *recall* equal to one and the desired *precision* equal to 0.8 or 0.9 in the case of false positives. For the case of false negatives, the desired *recall* is specified as 0.8 or 0.9 and the desired *precision* equal to one. For comparison purposes, we also show the result of $\epsilon = 0.1s$, which is the default in [25,39]. In addition, two minimum supports, 0.1 and 0.05, are given, where $s = 0.1$ and $s = 0.05$ leads to 122,449 and 582,752 frequent itemsets in the dataset, respectively. As shown in Fig. 15, it can be seen that the obtained *recall* and *precision* are quite close to the desired one when we apply PPL in advance, indicating the prominent advantage of algorithm PPL for such an application. Although the default $\epsilon = 0.1s$ results in a smaller memory cost in some cases, it indeed pays for an undesired loss of model accuracy. Importantly, we also find that $\epsilon = 0.1s$ may achieve the *recall* rate larger than the desired one in the case of *false-negative* cases, indicating that $\epsilon = 0.1s$ is larger than an appropriate one in such situations. It inevitably leads to large memory consumption. In such cases, algorithm PPL can help to suggest a relatively large ϵ, which achieves the desired *recall* while also leading to smaller memory consumption.

In essence, the applicability of the ϵ-deficient synopsis relies on a proper decision of the error parameter. We in this subsection demonstrate that algorithm PPL can provide a good reference to determine such a subtle parameter, showing its prominent advantage to be a pre-processing means for data mining applications.

4.3 Application study: sufficient sample size for frequent itemsets

To further show the advantages of PPL, we study in this subsection another important problem in frequent-pattern mining: *determining the sufficient sample size for mining frequent itemsets*.

4.3.1 Background review

Recently, research advances in frequent-pattern mining have been in the direction of attempts to reduce the execution complexity so as to discover frequent patterns in extremely large databases [21,34]. Among others, random sampling is one of the most straightforward manners to improve the mining efficiency [24,30,34,41]. However, using random sampling inevitably results in the generation of incorrect frequent patterns, which are not valid with respect to the entire database. It has been reported that a larger sample size leads to better consistency between the result obtained in the sample and that in the original dataset, but incurs larger time consumption; in contrast, a small sample size leads to smaller time consumption, but incurs the loss of model accuracy obtained [41]. How to determine an appropriate sample size is thus the key to the success of the sampling technique.

Traditionally, *Chernoff bounds* can provide a way to determine an appropriate sample size. However, the suggested sample size is conservative and is usually too large for mining frequent patterns [41]. In the literature, *progressive sampling* is another solution to determine the appropriate sample size [30]. Specifically, *progressive sampling* iteratively executes the frequent-pattern mining on random samples whose sizes are progressively increased, and the process will be terminated until the mining accuracy is no longer significantly improved. Finally, satisfactory model accuracy can be obtained without a prohibitively large sample size. However, the targeted application may be iteratively executed on many samples with variant sample sizes. It is also time-consuming. The practicability of such an approach thus needs further justification.

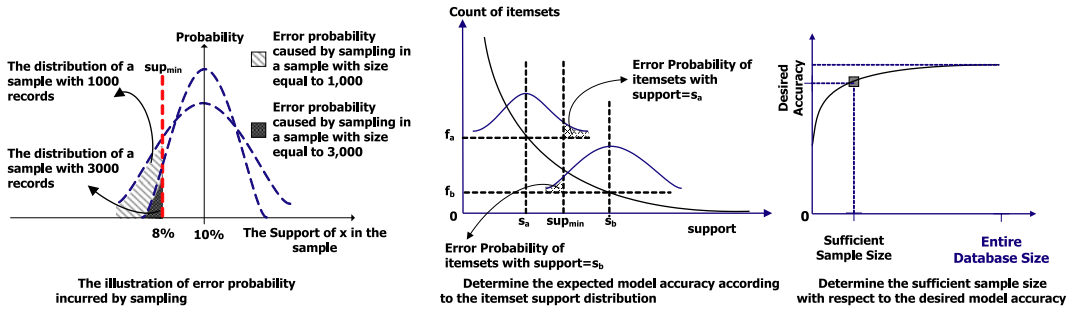


Fig. 16 Determining the sufficient sample size

4.3.2 Suggested enhancement

To provide better flexibility and efficiency to determine an appropriate sample size, we devise in this subsection an innovation approach with the help of PPL. The basic concept can be best understood by the illustration in Fig. 16, where Fig. 16a depicts the *sampling distribution* of the support of an itemset. Formally, the *sampling distribution* will approximately follow a *normal* distribution[14] with *mean* equal to the support of x in the entire dataset, and its deviation is equal to

$$\sigma_{x,N} = \sqrt{\frac{s_x \times (1 - s_x)}{N}}, \tag{7}$$

where N denotes the sample size and s_x denotes the support of x in the entire dataset [10]. Consider the example in Fig. 16a, where the support of the itemset x in the entire dataset is equal to 10% and the minimum support, denoted by sup_{min}, is specified as 8%. Clearly, a larger sample size leads to a smaller deviation (according to Eq. 7), and thus the corresponding error probability of incorrectly identifying the itemset type is smaller. Without loss of generality, the error probability of an itemset with support s_x can be approximated by

$$\xi(s_x, N) = \begin{cases} \int_0^{\text{sup}_{\text{min}}} \frac{1}{\sqrt{2\pi}\sigma_{x,N}} g(y)\mathrm{d}y, & \text{if } s_x \geq \text{sup}_{\text{min}} \\ \int_{\text{sup}_{\text{min}}}^1 \frac{1}{\sqrt{2\pi}\sigma_{x,N}} g(y)\mathrm{d}y, & \text{if } s_x < \text{sup}_{\text{min}} \end{cases}, \tag{8}$$

where $g(y) = exp\left(\frac{-(y-s_x)^2}{2\sigma_{x,N}}\right)$. Clearly, users are easy to give desired model accuracy rather than to directly decide a sample size. As such, we propose another aspect of determining a sufficient sample size: "*given user desired model accuracy, i.e., recall and precision, the sufficient sample size is the minimum sample size which can achieve desired model accuracy.*" This idea can be realized with the help of PPL. Specifically, we can determine the expected *recall* rate $E_r(N)$ and *precision* rate $E_p(N)$ in a sample size N according to the following equations:

$$E_r(N) = \frac{E\left[F_{f\Rightarrow f}(N)\right]}{F_f};$$

$$E_p(N) = \frac{E\left[F_{f\Rightarrow f}(N)\right]}{E\left[F_{f\Rightarrow f}(N)\right] + E\left[F_{nf\Rightarrow f}(N)\right]},$$

where F_f denotes the number of frequent itemsets in the entire dataset, $F_{f\Rightarrow f}(N)$ denotes the number of itemsets which are frequent both in the entire dataset and in the sample, $F_{nf\Rightarrow f}(N)$ denotes the number of itemsets which are non-frequent in the entire dataset but are identified as frequent in the sample.

After we execute algorithm PPL, and obtain the slope θ and the Y-intercept Ω of the *itemset support distribution*, we can further determine the expected model accuracy obtained in each sample size by considering the illustration in Fig. 16b. Clearly, the expectation of the *recall* rate can be rewritten as

$$E_r(N) = \frac{\int_{\text{sup}_{\text{min}}}^1 \left(e^{\Omega} \times s_x^{\theta}\right) \times [1 - \xi(s_x, N)]\,\mathrm{d}s_x}{\int_{\text{sup}_{\text{min}}}^1 \left(e^{\Omega} \times s_x^{\theta}\right)\mathrm{d}s_x},$$

where $\left(e^{\Omega} \times s_x^{\theta}\right)$ represents the number of itemsets with support s_x, and $[1 - \xi(s_x, N)]$ denotes the probability that an itemset with support s_x in the entire dataset can be correctly identified as a frequent itemset ($s_x \geq \text{sup}_{\text{min}}$). Similarly, the expectation of the *precision* rate can be rewritten as

$$E_p(N) = \frac{\int_{\text{sup}_{\text{min}}}^1 \left(e^{\Omega} \times s_x^{\theta}\right) \times [1 - \xi(s_x, N)]\,\mathrm{d}s_x}{G(N)},$$

[14] In essence, the *sampling distribution* follows a binomial distribution. However, as the sample size is large, the distribution can be approximated by a normal distribution or a Poisson distribution. The details can be found in [10]. For simplicity, we convey our idea without distinguishing the variants. It is a matter of implementation to precisely calculate the probability.

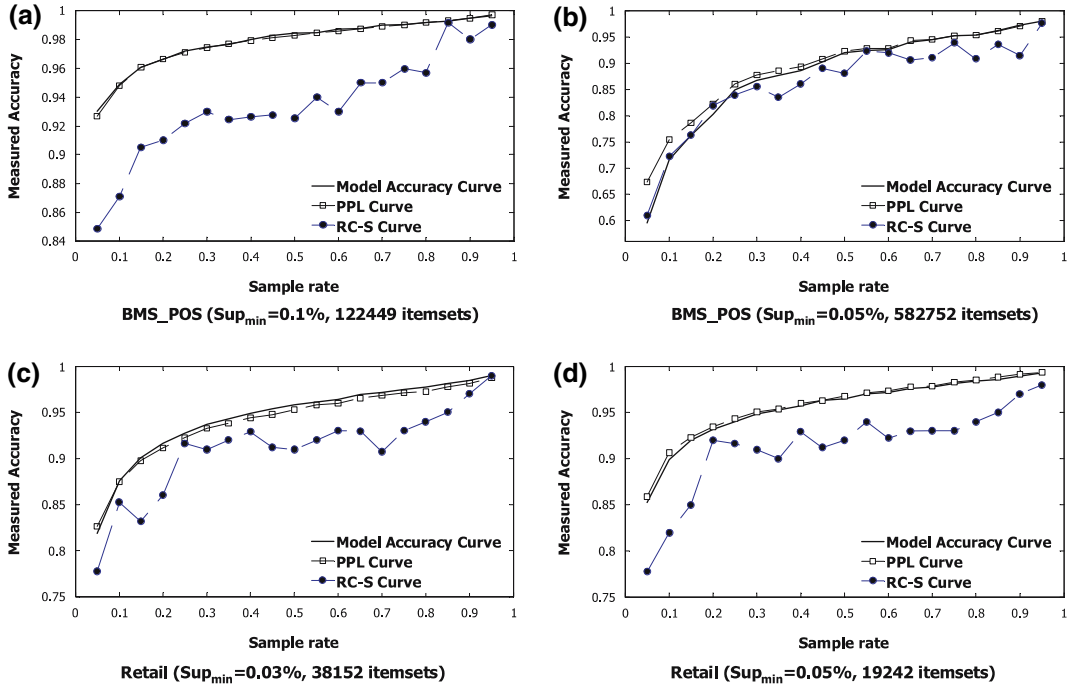


Fig. 17 The estimated model accuracy curves

where $G(N) = \left\{\int_{\sup_{\min}}^{1} \left(e^{\Omega} \times s_x^{\theta}\right) \times [1 - \xi(s_x, N)]\, \mathrm{d}s_x\right\} + \left\{\int_{0}^{\sup_{\min}} \left(e^{\Omega} \times s_y^{\theta}\right) \times \xi(s_y, N) \mathrm{d}s_y\right\}$.

For simplicity, we utilize the well-known metric *F-Score*, which is calculated as the harmonic mean of *precision* and *recall,* to simultaneously consider *recall* and *precision* [2]. Formally, the expected *F-Score* of a sample size N is defined as

$$E_F(N) = \frac{2 \times E_p(N) \times E_r(N)}{E_p(N) + E_r(N)}, \tag{9}$$

where $0 \leq E_F(N) \leq 1$. With the help of PPL, we can quickly determine the expected *F-Score* of each sample size and plot the curve of $E_F(N)$ versus the sample size N, as the illustrative curve shown in Fig. 16c. In essence, the curve is monotonically increasing as the sample size increases. As such, the sufficient sample size can be determined as the minimum sample size to achieve the desired model accuracy.

We summarize the procedure to determine the sufficient sample size: (1) execute PPL, and obtain the slope θ and the Y-intercept Ω of the *itemset support distribution*; (2) progressively increase the sample size and calculate the expected *F-Score* according to Eq. 9; and (3) determine the sample size as the minimum sample size which can achieve the desired model accuracy according to the result obtained in Step 2.

4.3.3 Simulation results

We assess the efficiency and the effectiveness of the proposed method to suggest the sufficient sample size (without ambiguity, we still call the proposed method as PPL in the sequel). For comparison purposes, the state-of-the-art progressive sampling algorithm, RC-S [30], is also evaluated. To give the best credit of algorithm RC-S, algorithm *Eclat* [40] is applied as the frequent-pattern mining method according to the suggestion in [30]. Furthermore, in all experiments, the user-specified parameter α of algorithm RC-S is set to one, which is consistent with its default value addressed in [30]. For fair comparison, we use an arithmetic sampling schedule to estimate model accuracy in different sample rates: $\{0.05, 0.1, 0.15, \ldots, 0.95\}$. In addition, model accuracy of a sample size is calculated as the average *F-Scores* over 50 runs on samples with this size.

In Fig. 17, the "Model Accuracy Curve" represents the corresponding curve of the accurate model accuracy versus the sample size, as the curve illustrated in Fig. 16c. In addition, the model accuracy curves estimated by PPL (denoted by "PPL Curve" in Fig. 17) and RC-S (denoted by "RC-S Curve" in Fig. 17) are also plotted. Importantly, the excellent approximation of the model accuracy curve can be obtained by *PPL* in different datasets; and in contrast, the RC-S is unstable and cannot correctly estimate the model accuracy curve. Since RC-S estimates model accuracy of a sample size only on a sample with the size (for efficiency reasons), it is clear that the effectiveness of RC-S is solely affected by

Fig. 18 The estimation effectiveness and execution time of PPL and RC-S

Data	Minimum Support	Desired Accuracy	The Sufficient Sample Rate	The Sample Rate (suggested by PPL)	PPL Execution Time (sec)	The Sample Rate (suggested by RC-S)	RC-S Execution Time (sec)
POS	0.1%	0.8	5%	5%	9	5%	124
		0.9	10%	10%	9	15%	163
	0.05%	0.8	20%	20%	9	20%	234
		0.9	40%	40%	9	55%	582
Retail	0.03%	0.8	5%	5%	7	5%	32
		0.9	15%	15%	7	25%	75
	0.05%	0.8	5%	5%	7	10%	23
		0.9	10%	10%	7	20%	67
3C_Chain	0.001%	0.8	5%	5%	11	5%	9
		0.9	15%	15%	11	25%	183
	0.0005%	0.8	15%	15%	11	25%	204
		0.9	20%	20%	11	30%	287
Book	0.010%	0.8	5%	5%	5	5%	3
		0.9	5%	5%	5	10%	13
	0.0050%	0.8	20%	20%	5	25%	49
		0.9	30%	30%	5	35%	88

randomness, which makes samples with the same size deviate from each other. It causes the unstable estimation of model accuracy that can be obtained. On the other hand, since PPL helps to formalize this problem in the statistical sense, it can precisely estimate the expected model accuracy obtained in each sample size without facing the challenge incurred by *randomness*, showing the prominent advantage of PPL in this application.

In addition, we show in Fig. 18 the sufficient sample sizes that are suggested by PPL and RC-S while users give desired model accuracy. The corresponding execution time required by these two algorithms is also shown. Two common user desired *F-Scores*, i.e., 0.8 and 0.9, are both given. Confirming the observation in Fig. 17 that PPL can perfectly estimate the model accuracy curve, we can see that the sufficient sample rates suggested by PPL in each case is almost equal to the accurately sufficient sample rate, which is manually checked from the accurate model accuracy curve. In addition, the execution time required by PPL is much shorter than that required by RC-S. It is because that RC-S needs to iteratively execute algorithm *Eclat* on samples with progressively increasing sizes until the desired model accuracy is achieved. It is very time-consuming, particularly when either the minimum support is small or the desired accuracy is high. Note that RC-S may lead to execution time shorter than that of PPL when the desired accuracy is small (e.g., cases of the sufficient sample rate equal to 5%). However, in general cases, RC-S cannot efficiently and stably estimate the correct sample size. On the other hand, the execution time of PPL is steady in a dataset regardless of the corresponding minimum support and the desired accuracy. Clearly, both considering effectiveness and efficiency, PPL is the winner in this application, thus showing its advantages to be an excellent pre-processing means for frequent-pattern mining.

5 Conclusions

In this paper, we demonstrated that the power-law relationship and the self-similar phenomenon appear in the distribution of itemset supports in real datasets. Discovering these nature phenomena is useful to many mining applications. We also proposed algorithm PPL to efficiently identify characteristics of the power-law relationship in the *itemset support distribution*. As shown in the experimental results, algorithm PPL is able to efficiently extract the characteristics of the power-law relationship with high accuracy. To fully explore the advantages of our discovery, we have also solved two challenging problems with the help of PPL. Empirical studies demonstrated that our solutions are in orders of magnitude better than previous works, showing the prominent advantage of PPL to be an important pre-processing means for mining applications.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. of VLDB (1994)
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison–Wesley, Reading (1999)
3. Beran, J.: Statistics for long-memory processes. Monographs on Statistics and Applied Probability. Chapman & Hall, London (1994)
4. Bi, Z., Faloutsos, C., Korn, F.: The "DGX" Distribution for Mining Massive, Skewed Data. In: Proc. of ACM SIGKDD (2000)
5. Borgelt, C.: Efficient implementations of apriori and eclat. In: Proc. of Workshop on Frequent Itemset Mining Implementations (2004)
6. Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web caching and zipf-like distributions: evidence and implications. In: Proc. of IEEE INFOCOM (1999)
7. Cheung, Y.L., Fu, A.W.: Mining Association Rules without Support Threshold: with and without Item Constraints. In: TKDE (2004)
8. Chuang, K.-T., Chen, M.-S., Yang, W.-C.: Progressive sampling for association rules based on sampling error estimation. In: Proc. of PAKDD (2005)
9. Chuang, K.-T., Huang, J.-L., Chen, M.-S.: Mining Top-k Frequent Patterns in the Presence of the Memory Constraint. In: Technical Report, under submission. A short version is published in Proc. of ACM CIKM (2005)
10. Cochran, W.G.: Sampling Techniques. Wiley, London (1977)
11. Cormode, G., Muthukrishnan, S.: Summarizing and mining skewed data streams. In: Proc. of SIAM SDM (2005)

12. Crovella, M.E., Bestavros, A.: Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. In: Proc. of ACM SIGMETRICS (1996)
13. Dill, S., Kumar, R., McCurley, K., Rajagopalan, S., Sivakumar, D., Tomkins, A.: Self-similarity in the web. In: Proc. of VLDB (2001)
14. Egghe, L.: The distribution of n-grams. Scientometrics (2000)
15. Faloutsos, C.: Next Generation Data Mining Tools: Power Laws and Self-similarity for Graphs, Streams and Traditional Data. ECML (2003)
16. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the internet topology. In: Proc. of ACM SIGCOMM (1999)
17. Geerts, F., Goethals, B., Bussche, J.V.D.: Tight upper bounds on the number of candidate patterns. ACM Trans. Database Syst. (2005)
18. Geerts, F., Goethals, B., Bussche, J.V.D.: A tight upper bound on the number of candidate patterns. In: Proc. of IEEE ICDM (2001)
19. Ghoting, A., Buehrer, G., Parthasarathy, S., Y.Chen, Kim, D., Nguyen, A., Dubey, P.: Cache-conscious frequent pattern mining on a modern processor. In: Proc. of VLDB (2005)
20. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2000)
21. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proc. of ACM SIGMOD (2000)
22. Ioannidis, Y.: The history of histograms. In: Proc. of VLDB (2003)
23. Koch, R.: The 80/20 Principle: The Secret of Achieving More With Less. Nicholas Brealey Publishing, London (1998)
24. Lee, S.D., David Cheung, W.-L., Kao, B.: Is sampling useful in data mining? A case in the maintenance of discovered association rules. DMKD **2**(3), 233–262 (1998)
25. Manku, G.S., Motwani, R.: Approximate frequency counts over streaming data. In: Proc. of VLDB (2002)
26. Metwally, A., Agrawal, D., Abbadi, A.E.: Efficient computation of frequent and top-k elements in data streams. In: Proc. of ICDT (2005)
27. Orlando, S., Lucchese, C., Palmerini, P., Perego, R., Silvestri, F.: kDCI: a Multi-Strategy Algorithm for Mining Frequent Sets. In: Proc. of Workshop on Frequent Itemset Mining Implementations (2004)
28. Orlando, S., Palmerini, P., Perego, R., Silvestri, F.: Adaptive and resource-aware mining of frequent sets. In: Proc. of IEEE ICDM (2002)
29. Park, J.-S., Chen, M.-S., Yu, P.S.: An effective hash based algorithm for mining association rules. In: Proc. of ACM SIGMOD (1995)
30. Parthasarathy, S.: Efficient progressive sampling for association rules. In: Proc. of IEEE ICDM (2002)
31. Provost, F., Jensen, D., Oates, T.: Efficient progressive sampling. In: Proc. of ACM SIGKDD (1999)
32. Ramesh, G., Maniatty, W.A., Zaki, M.J.: Feasible itemset distributions in data mining: Theory and application. In: Proc. of ACM PODS (2003)
33. Rice, J.A.: Mathematical statistics and data analysis. Duxbury Press, North Scituate (1995)
34. Toivonen, H.: Sampling large databases for association rules. In: Proc. of VLDB (1996)
35. Uno, T., Asai, T., Uchida, Y., Arimura, H.: Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In: Proc. of Workshop on Frequent Itemset Mining Implementations (2004)
36. Wang, J., Han, J., Lu, Y., Tzvetkov, P.: TFP: An Efficient Algorithm for Mining Top-K Frequent Closed Itemsets. In: TKDE (2005)
37. Willinger, W., Taqqu, M.S., Leland, W.E., Wilson, D.V.: Self-similarity in high-speed packet traffic: analysis and modelling of ethernet traffic measurements. Stat. Sci. **10**(1) (1995)
38. Wong, R.C.-W., Fu, A.W.: Mining top-k itemsets over a sliding window based on zipfian distribution. In: Proc. of SIAM SDM (2005)
39. Yu, J.X., Chong, Z., Lu, H., Zhou, A.: False positive or false negative: Mining frequent itemsets from high speed transactional data streams. In: Proc. of VLDB (2004)
40. Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New algorithms for fast discovery of association rules. In: Proc. of ACM SIGKDD (1997)
41. Zaki, M.J., Parthasarathy, S., Wei, I., Ogihara, M.: Evaluation of sampling for data mining of association rules. In: Int. Workshop on Research Issues in Data Engineering (1997)
42. Zheng, Z., Kohavi, R., Mason, L.: Real world performance of association rule algorithms. In: Proc. of SIGKDD (2001)
43. Zipf, G.K.: Human Behavior and the Principle of Least Effort. Addison–Wesley, Reading (1949)