On Exploring the Power-Law Relationship in the Itemset Support Distribution

Kun-Ta Chuang and Jiun-Long Huang and Ming-Syan Chen
Department of Electrical Engineering
National Taiwan University
E-mail: {doug,jlhuang}@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw

Abstract. We identify and explore in this paper an important phenomenon that the power-law relationship appears in the distribution of itemset supports. Characterizing such a relationship will benefit many applications such as providing the direction of tuning the performance of the frequent-itemset mining. Nevertheless, due to the explosive number of itemsets, it will be prohibitively expensive to retrieve characteristics of the power-law relationship in the distribution of itemset supports. As such, we also propose in this paper a valid and cost-effective algorithm, called algorithm *PPL*, to extract characteristics of the distribution without the need of discovering all itemsets in advance. Experimental results demonstrate that algorithm *PPL* is able to efficiently extract the characteristics of the power-law relationship with high accuracy. In addition, while applying algorithm *PPL* prior to discover approximate frequent itemsets, a desired result can be effectively obtained, showing its prominent advantage of being an important pre-processing means for mining applications.

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 [11]. 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 on the targeted applications. It is beyond dispute that the deeper knowledge about the characteristics of your data will lead 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 pre-processing means for mining applications.

Recent research advances in frequent-itemset mining algorithms are thus in the direction of discovering characteristics of real datasets. For example, the works in [9] and [18] both seek the relationship between different itemset lengths in the targeted dataset. Such relationships can be further utilized to control the mining process [9], or to generate the realistic synthetic datasets for the system parameter tuning [18].

To provide better understanding on real datasets, we in this paper investigate the 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 Figure 1. The *itemset support 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.

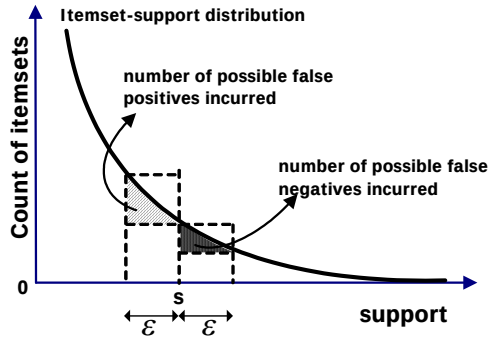


Fig. 1. The illustration of the *itemset support distribution* and the comparison between false positives and false negatives in the frequency-approximation applications.

Inspired by the power-law relationship observed in many distributions of single words (users, web pages) [3][26], 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 also appears in the *itemset support distribution* and we can characterize that by the Zipf distribution [26].

However, to find the parameters characterizing the *itemset support distribution* will be more challenging than to find the parameters in the distribution of single items since all itemsets need to be retrieved. The extremely large time and memory consumption cannot be avoided due to the itemset combinational explosion. Note that the costly process will drastically decrease the practicability of knowing the characteristics of the *itemset support distribution*. To remedy this, 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 avoiding the need of generating all itemsets. As shown in our empirical studies, algorithm *PPL* is able to efficiently and precisely extract the characteristics of the power-law relationship. Hence algorithm *PPL* can be utilized as an excellent pre-processing step for extensive applications of mining frequent patterns.

1.1 Motiving Applications

We in the following review several applications which attract a lot of research attention, and comment the advantage of utilizing algorithm *PPL* as their pre-processing step.

Frequency approximation over data streams: Recent advances in streaming research recognize the importance of frequency approximation [7][14][15][22]. Among them, as the assumption that the support distribution of single items following the Zipf distribution, the work in [15] devised algorithm *Space-Saving*, to compute top-k single items with the 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. In fact, while executing algorithm *PPL* in advance, algorithm *Space-Saving* can be further extended to approximately retrieve top-k frequent itemsets over data streams, and the appropriate memory size, depending on the parameter θ identified by *PPL*, can be assigned prior to the execution of algorithm *Space-Saving*. It will benefit the efficient implementation. For example, since the memory to maintain itemsets can be previously allocated, we can implement

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

the algorithm by maintaining itemsets in an array-like structure as opposed to the dynamic linked list-like structure, to pursue the higher execution efficiency (note that as pointed out in [10], the dynamic data structure will in general lead to poor efficiency due to its poor spatial locality and poor temporal locality).

False positives versus false negatives: As pointed out in [23], a challenging issue of the frequency approximation over data streams arises from the choice between false positives or false negatives. Specifically, the work in [14] allows false positives, i.e., allowing to identify non-frequent itemsets as frequent ones. For reducing the memory consumption, the authors of [23] suggest to allow false negatives, i.e., allowing to miss some frequent itemsets without incurring any false positive. In practice, the result of our observation in the *itemset support distribution* will support the argument that allowing false positives will incur more inaccurate itemsets (non-frequent itemsets being identified as frequent) as compared to that incurred by allowing false negatives (frequent itemsets being identified as non-frequent). It results from 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. In addition, while knowing the parameters of the power-law relationship in the *itemset support distribution* in advance, we will be able to approximately estimate the expected number of inaccurate itemsets incurred either as false positives or as false negatives, as illustrated in Figure 1. Thus the error parameter ϵ in the work of [14] or the parameter δ in the work of [23] (in light of the Chernoff bound, δ can be used to derive the value of ϵ) can be appropriately determined (not just being fully left unsolved to users) to balance the resulting accuracy and the memory consumption.

Performance tuning of mining frequent patterns: As pointed out in [16], the effectiveness of many heuristic optimizations for mining frequent itemsets usually relies on the data characteristics, particularly depending on the sparsity/density of the targeted data. Once the data is long and dense, i.e., the *itemset support distribution* is highly skewed, depth-first approaches such as FP-growth [12] will be more powerful than breath-first approaches such as DHP [17]. Oppositely, algorithm DHP will be 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 benefit the strategy decision for the best performance of mining frequent patterns.

Determine the appropriate minimum support: Another application of knowing the *itemset support distribution* in advance lies in the help to the determination of the minimum support. As we know, setting the minimum support is quite subtle since a small minimum support may lead to an extremely large size of frequent itemsets. Oppositely, only a few itemsets may be generated when the minimum support is large. In order to obtain a desired result, users in general need to tune the minimum support over a wide range, which is very time-consuming and indeed is a serious problem for the applicability of mining frequent itemsets. While executing algorithm *PPL* as a pre-processing step, we can estimate the number of itemsets with support exceeding a specified minimum supports, which will help users to decide the appropriate minimum support.

Synthetic data generator: Formally, using synthetic data is the common way to perform the sensitivity analysis of a proposed algorithm, and previous works of the association-rule mining usually observe their results by utilizing the synthetic data generator provided in [1]. Recently, the work in [18] seeks the relationship between different itemset lengths in the targeted dataset, to generate more realistic synthetic datasets. Going beyond this, we comment that, a real dataset is able to be better characterized by also giving its *itemset support distribution*. Therefore, for better capturing the characteristics of the real datasets for the targeted application, the generation of the synthetic transactional datasets by the generators

[1][18] can be further improved by considering the characteristics in their *itemset support distributions*.

1.2 Our Contributions

Our contributions is 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.

(2) We present a valid and cost-effective algorithm, called algorithm *PPL*, to identify characteristics of the *itemset support distribution* without the need of discovering all itemsets in advance.

(3) 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 important pre-processing means for mining applications.

We then individually present those issues in the following sections.

2 Identify the Power-Law Relationship in the Itemset Support Distribution

2.1 Review of the Power-Law Relationship

Since the first observation of the power-law relationship in [26], which discovered the frequency of the n^{th} 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 world data, including WWW characteristics, Internet topology, to name a few[2]. 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 [2]. 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 [2]. Without loss of generality, we will 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 will be deemed following the power-law relationship if the points in the log-log plot are close to a single straight line, as shown by

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

In particular, the slope θ and the Y-intercept Ω in Eq. 1 can be estimated by the linear regression[3]:

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

[2] See http://www.nslij-genetics.org/wli/zipf/ for the power-law references from different domains.

[3] 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.

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

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[4] (or said 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 [19]. Due to space limitations, we only describe the Zipf distribution here. For details of the regression technique, which will be out of scope for this paper, the reader is asked to follow the pointers in some well-known materials such as [19].

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 [3][26]. 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 [8], 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) follows the Zipf distribution.

2.2 Observations on the 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 $\mathcal{I} = \{x_1, x_2, ..., x_m\}$ 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[5]. We would like to investigate whether the support distribution of itemsets follows the Zipf distribution, as the form shown by

$$\log(f_i) = \theta \log(s_i) + \Omega, \quad (4)$$

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 since the "count" presents the absolute support count. For interest of space, we in this paper concentrate on the investigation of retail datasets, which are skewed and sparse, and most association-rule discovery algorithms were designed for such types of data [25] (interested readers can find observations on other types of real datasets in http://arbor.ee.ntu.edu.tw/~doug/paper/PPL/index.html).

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

Table 1: Parameters of real datasets

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

[4] For convenience of discussion, we will postpone the formula of the correlation coefficient to Eq. 5 in Section 3.3.

[5] 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.

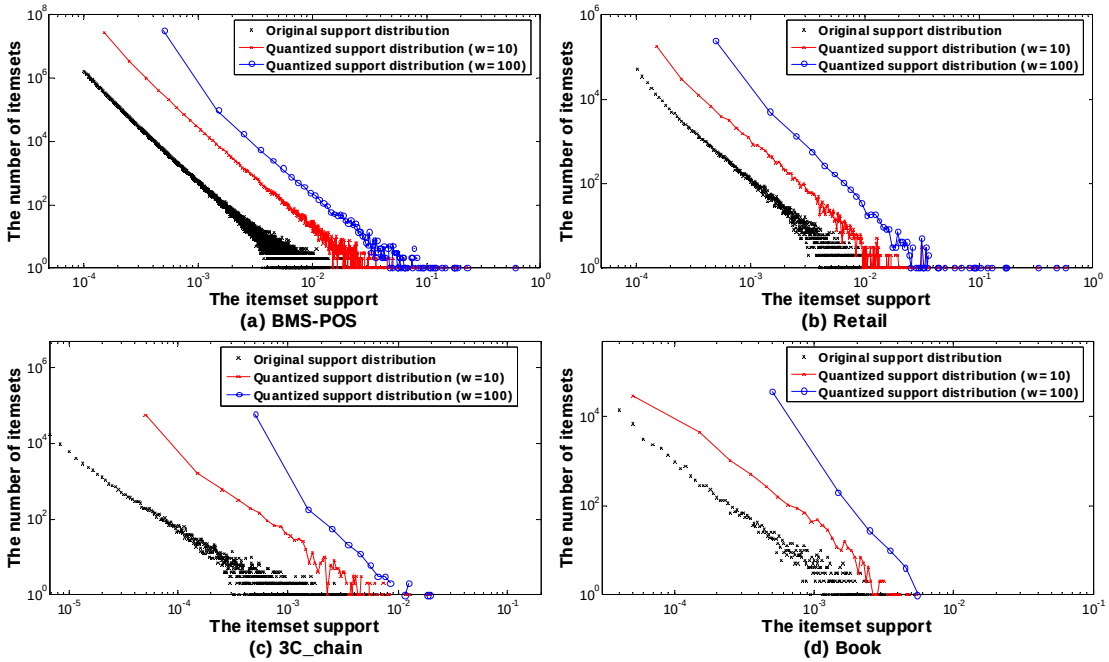


Fig. 2. The support distribution of four real datasets.

benchmark datasets[6], 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[7] 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 a reasonable execution time. For efficiency reasons, we did not retrieve all itemsets in the BMS-POS and the 3C_chain datasets, but instead retrieve itemsets whose support counts exceed 30, where 30 is a sufficient number in the statistical sense [19].

The observations are shown in Figure 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 distribution of the itemset support. In addition, the "top-concavity"[8] phenomenon, which is prevalent in the distribution of single items [2][26], is insignificant in the distribution of itemset supports. 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.

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

[7] The URL is http://fuzzy.cs.uni-magdeburg.de/~borgelt/fpgrowth.html.

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

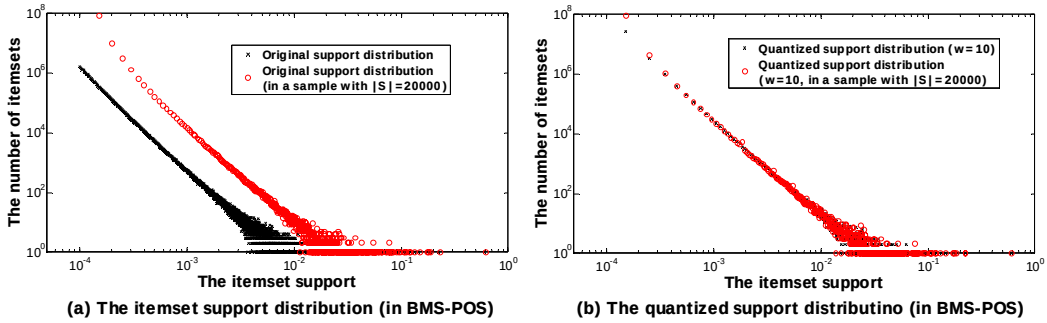


Fig. 3. The support distribution after sampling (the BMS-POS dataset).

3 Design of Algorithm PPL

As mentioned above, recognizing characteristics of the support distribution will benefit the proper mining system design. However, although we have demonstrated in Section 2 that the power-law relationship appears in the support distribution, it is prohibitively expensive to find all itemsets and further estimate the characteristics of the Zipf distribution, i.e., the slope θ and the Y-intercept Ω in Eqs. 2 and 3. The extremely large time consumption results from the expensive process to retrieve all itemsets without the support pruning. An efficient approach is still demanded to correctly estimate those parameters.

As a consequence, we propose in this paper a valid and cost-effective solution, named *PPL* (standing for **P**redict the **P**ower-**L**aw relationship), to estimate the parameters of the power-law relationship in the *itemset support distribution*. Since the time consumption is dominated by the process of retrieving all itemsets in the large database, algorithm *PPL* utilizes two approaches to improve the efficiency. The first one is to utilize *sampling* techniques to retrieve itemsets [20]. The other approach is to retrieve *only the set of high-support itemsets* with the help of the support pruning techniques [17] 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 devised as a three-phases approach: (1) sampling; (2) obtaining high-support itemsets; (3) estimating the parameters of the power-law relationship by the linear regression from the high-support itemsets discovered in the sample.

However, while pursuing the efficiency, algorithm *PPL* will face three challenges:

(1) *The support distribution obtained in a sample will deviate from the support distribution in the original database*. Note that after sampling, the supports of many low-support itemsets will become higher, and vice versa [24]. Unfortunately, as pointed out in [24], the number of itemsets with a specified support is likely to *increase* after sampling due to the large amount of transfers from low-support itemsets (the number of low-support itemsets is large than the number of high-support itemsets). As shown in our empirical studies in Figure 3(a), where the support distributions obtained in the original dataset and in a random sample with 20,000 tuples are included, it can be apparently observed that the support distribution in the sample deviates from that in the original dataset. Indeed, due to randomness, we cannot estimate the deviation between the support distribution in a sample and that in the original dataset. Thus after sampling, it will be difficult to correctly predict the characteristics of the support distribution in the original dataset.

(2) *It is very difficult to determine the appropriate minimum support without prior knowledge*. Note that *PPL* will only discover high-support itemsets. However, we will not know how to determine the subtle minimum support. A large minimum support will result in too

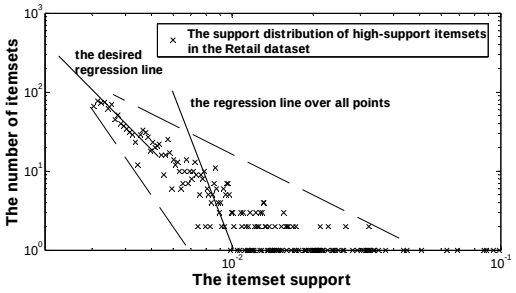


Fig. 4. Estimating the power-law relationship from high-support itemsets.

few itemsets to provide the sufficient information to correctly estimate the parameters of the Zipf distribution. Oppositely, the small minimum support will generate a lot of itemsets, thus resulting in inefficiency.

(3) *It is difficult to obtain the desired regression line due to the support fluctuation on high-support itemsets*. Note that the Zipf distribution can be characterized by the regression line. However, consider the observation in Figure 4, where a solid straight lines 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. It will incur the large least-square errors, and the regression line over points with respect to high supports may deviate from the desired regression line[9].

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 [13]. 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 s_i denotes the support of itemsets and 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 j^{th} bucket of the histogram, and $\widehat{f_j}$ denotes the count of itemsets with supports falling in the j^{th} 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 reduce the influence of support-deviation follows the observation below:

Observation: Suppose that we repeatedly generate a lot of samples of the same sample size. The distribution of the *support* of X among these samples, i.e., the *sampling distribution* of

[9] In [3], 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 the rightmost 100 points which are summarized from high support itemsets

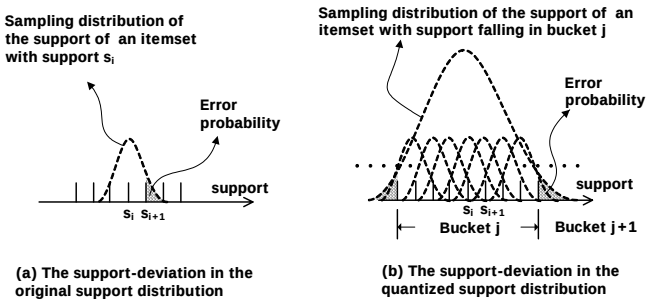


Fig. 5. Influence of the support-deviation.

the support of X, will approximately follow a *normal* distribution with *mean* equal to the support of X in the entire dataset [6]. In addition, the variance of the *sampling distribution* depends on the sample size [20]. As shown in Figure 5(a), the *sampling distribution* of an itemset with support equal to s_i in the entire database indicates that the support will be likely larger than s_i with the probability equal to the shadow region. As a result, a percentage of itemsets in the sample will have supports inconsistent with the corresponding support in the entire dataset. Accordingly, the *itemset support distribution* after sampling will deviate from the *itemset support distribution* in the entire dataset. This argument is demonstrated in Figure 3(a).

On the other hand, consider the case of the *quantized support distribution.* As shown in Figure 5(b), the error probability, i.e., the probability of itemsets with supports in bucket j changing to bucket $j+1$ after sampling, will be relatively small as compared to the error probability illustrated in Figure 5(a). The reason lies in that 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 Figure 3(b), 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* will be insensitive to the support-deviation, meaning that *the quantized support distribution in the sample will be close to the quantized support distribution in the entire dataset.* As a result, we will aim to obtain the *quantized support distribution* in the sample.

Another problem, as shown in Figure 2, is that the *quantized support distribution* still deviates from the original support distribution. Importantly, assuming that the original support distribution approximately follows the Zipf distribution, Proposition 1 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.

Proposition 1: *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}

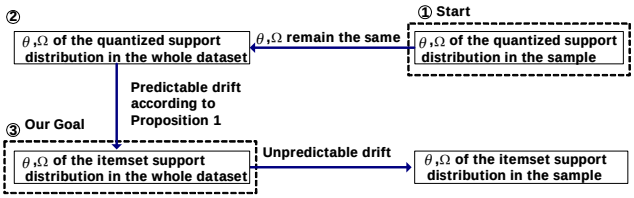


Fig. 6. The flow to overcome problems incurred by sampling.

bucket. As such, the log-log plot in the quantized support distribution has the slop θ and the Y-intercept $\Omega + \log(w)$.

Proof: Note that we have $e^{\Omega} \times s_{i,j}^{\theta} \approx f_{i,j}$ for the point ($s_{i,j}$,$f_{i,j}$) in the original support distribution since it follows the Zipf distribution. Suppose that $|D|$ is the database size. Let points ($s_{k,1}$,$f_{k,1}$), ($s_{k,2}$,$f_{k,2}$), ..., ($s_{k,w}$,$f_{k,w}$) be summarized as the k^{th} point ($\widehat{s}_k$,$\widehat{f}_k$) in the *quantized support distribution*. We have $\widehat{f}_k = \sum_{j=1}^{w} f_{k,j}$, and

$$\widehat{s}_k = \frac{s_{k,1} + s_{k,w}}{2} = \frac{s_{k,1} + \left(s_{k,1} + \frac{w}{|D|}\right)}{2} = s_{k,1} + \frac{w}{2 \times |D|}.$$

Since $\frac{w}{|D|}$ is in general much weak as compared to $s_{k,1}$, we have

$$\frac{s_{k,j}^{\theta}}{\widehat{s}_k^{\theta}} = \left(\frac{s_{k,1} + \frac{j}{|D|}}{s_{k,1} + \frac{w}{2\times|D|}}\right)^{\theta} \approx 1.$$

Therefore $s_{k,j}^{\theta}$, for $1 \leq j \leq w$, will be approximately equal to $\widehat{s}_k^{\theta}$, which yields that

$$\widehat{f}_k = \sum_{j=1}^{w} f_{k,j} \approx e^{\Omega} \times \sum_{j=1}^{w} \widehat{s}_k^{\theta} = e^{\Omega} \times w \times \widehat{s}_k^{\theta},$$
$$\log(\widehat{f}_k) \approx \theta \log(\widehat{s}_k) + \Omega + \log(w)\,. \blacksquare$$

Proposition 1 indicates that the slope θ remains in the *quantized support distribution*, and the Y-intercept will be changed to $\Omega + \log(w)$. Figure 2 demonstrates Proposition 1, 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.

Based on the foregoing, the process to overcome problems incurred by sampling, as shown in Figure 6, will be summarized as:
(1) Obtain the characteristics of the *quantized support distribution* in the sample.
(2) The characteristics of the *quantized support distribution* in the whole dataset is expected equal to that in the sample.
(3) In light of Proposition 1, obtain the characteristics of the original *itemset support distribution*.

Note that while step 1 is completed, steps 2 and 3 can be straightforwardly executed with the mathematical manipulation mentioned above. How to precisely achieve step 1 will be discussed in Section 3.2 and Section 3.3.

The remaining issue in this phase is, what is the appropriate sample size to obtain the *quantized support distribution* in the sample which is 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 relies on the sample size [20]. 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 equal to 20,000 [20] or a

sample rate equal to 10% [24], will be sufficient to generate the accurate set of frequent itemsets. We argue that the sample size 20,000 or 10% is also sufficient to generate the accurate *quantized support distribution* by following several points: (1) the complexity to generate the accurate *quantized support distribution* is analogous to the complexity to generate accurate frequent itemsets; (2) in Phase II only high-support itemsets will be generated, whose supports, as indicated in [20], can be easily preserved in samples as compared to supports of low-support itemsets; (3) the discrepancy between counts of itemsets within a bucket in the sample and in the entire dataset will be unapparent in the log-log scale (the characteristics of the power-law relationship is estimated in the log-log scale); (4) the technique in Phase III is specifically designed to be robust to the inconsistency 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 [24]. 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*" [4][21] instead of "discover itemsets with the specified minimum support," where *top-k* itemsets 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 [4][21]. Formally, those works shoot for discovering *top-k* itemsets with specified constraints such as discovering *closed* itemsets [4]. 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. In view of this, we propose an efficient algorithm, called algorithm *MTK*, to efficiently retrieve *top-k* itemsets without such constraints while also guaranteeing the bound of the memory consumption. For interest of space, the details of algorithm *MTK* are not described in this paper, and interested readers can refer to [5][10] 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 $\{X_1$, ..., $X_k\}$ 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 Figure 2. As such, *top-k* itemsets will be aggregated into a set of points $H_k = \{\left(\widehat{s}_1, \widehat{f}_1\right), \left(\widehat{s}_2, \widehat{f}_2\right), ..., \left(\widehat{s}_z, \widehat{f}_z\right)\}$ sorted by $\widehat{s}_i$, where $\widehat{s}_i < \widehat{s}_j$ iff $i < j$. We therefore can characterize the power-law relationship by performing the regression analysis over the partial *quantized support distribution* which are summarized from *top-k* itemsets discovered in the sample.

[10] This technical report can be downloaded in http://arbor.ee.ntu.edu.tw/~doug/paper/PPL/mtk.pdf

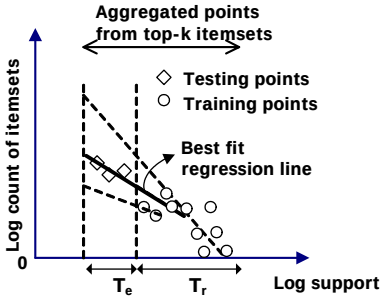


Fig. 7. The illustration of the best fit regression line.

However, as pointed out as the third challenge described in the beginning of Section 3, directly executing the regression analysis over all points in H_k will result in the incorrect estimation due to the support fluctuation on high support itemsets. Therefore the 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 Figure 4. Points with respect to very high-supports usually do not accurately follow the Zipf distribution. On the other hand, without loss of generality, points with respect to low supports usually follow the Zipf distribution. As such, one may intuitively claim a naive approach as follows.

Naive Approach: It is intuitive to suggest the regression line over first several points in H_k since they are sufficient to correctly fit the power-law relationship. For example, we may estimate the power-law relationship by performing the regression analysis over the first five points in H_k, i.e., $\{\left(\widehat{s}_1, \widehat{f}_1\right), ..., \left(\widehat{s}_5, \widehat{f}_5\right)\}$. Nevertheless, we indeed did not know how many points is sufficient to obtain the desired regression line. Thus 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, sometimes few points will result in the best correlation coefficient, but the regression line could be bias to outlier points [19]. In addition, sampling in Phase I may incur noise, which will also affect the result of the linear regression. As a result, we devise a novel solution, which is inspired from the training and testing scenario in supervised learning [11], to correctly estimate the parameters of the Zipf distribution from H_k.

Minimizing Testing Error Approach: Suppose that H_k is divided into two distinct and consecutive subsets of points, i.e., the training set T_r and the testing set T_e, where $T_e = \left\{\left(\widehat{s}_1, \widehat{f}_1\right), ..., \left(\widehat{s}_m, \widehat{f}_m\right)\right\}$ and $T_r = \left\{\left(\widehat{s}_{m+1}, \widehat{f}_{m+1}\right), ..., \left(\widehat{s}_z, \widehat{f}_z\right)\right\}$. Here m is the parameter to adjust the size of the testing set and $m < z$. Consider the illustration in Figure 7, 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* in the following.

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 first g points in T_r, i.e., $\left\{\left(\widehat{s}_{m+1}, \widehat{f}_{m+1}\right), ..., \left(\widehat{s}_g, \widehat{f}_g\right)\right\}$, where $m + 1 \leq g \leq z$.

Procedure: *Best_fit*

Input:

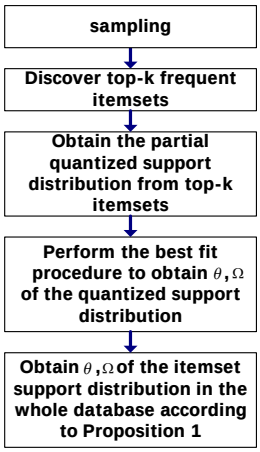
Top-k itemsets, X_i, where $i = 1, ..., k$.
The number of aggregated support counts w.
The number of points in T_e, i.e., $|T_e|$.
The correlation coefficient threshold δ.

Output:

The best fit slope $\widehat{\theta}_g$, and the best fit Y-intercept $\widehat{\Omega}_g$

1. Let $f_i = 0$, where $i = 0, ..., \lceil \frac{1}{\Delta} \rceil$;
2. **for** $i = 1$ to k;
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 $f_\Lambda = \min_i\{f_i \neq 0\}$;
5. **for** $f = (\Lambda + |T_e|{+}1)$ to $\lceil \frac{1}{\Delta} \rceil$ **begin**
6. $[\Omega_f, \theta_f, r_f]$=linear_reg($(\log(s_m), \log(f_m))$, where $(\Lambda + |T_e|{+}1) \leq m \leq f$;
7. $Var_f = \sum_{j=1}^{m} \left[\theta_f \log(s_j) - \log(f_j) + \Omega_f \right]^2$;
8. **end**
9. find g, where $g = \arg\min_u(Var_u)$, subject to $r_u \leq \delta$;
10. **return** Ω_g, θ_g with respect to g;

(a) Procedure Best_fit



(b) Overall flow of algorithm PPL

Fig. 8. The implementation of algorithm *PPL*.

(2) The correlation coefficient, r_g, over the data points $\left\{ \left(\widehat{s}_{m+1}, \widehat{f}_{m+1} \right), ..., \left(\widehat{s}_g, \widehat{f}_g \right) \right\}$ is smaller than a pre-defined threshold δ. Note that,

$$r_g = \frac{\sum_{i=m+1}^{g} \sum_{j=m+1}^{g} \left(\log(\widehat{s}_i) - u_s \right) \left(\log(\widehat{f}_j) - u_f \right)}{\sqrt{\sum_{i=m+1}^{g} \left(\log(\widehat{s}_i) - u_s \right)^2} \sqrt{\sum_{j=m+1}^{g} \left(\log(\widehat{f}_j) - u_f \right)^2}}, \tag{5}$$

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 $r_f \leq \delta$ and $m + 1 \leq u \leq z$.

The whole procedure to find the best fit regression line is outlined in Procedure *Best_fit* in Figure 8(a), 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. 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 $r_g \leq -0.8$ is sufficient to claim the regression line can explain these points [19]. 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. It is worth mentioning that, algorithm *PPL* will degenerate to the naive approach if there is no testing point in T_e and simply choose 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 discovered in the *quantized support distribution* generated from top-k itemsets in the sample. In light of Proposition 1, 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 Figure 8(b): (1) sampling; (2) discover *top-k* frequent 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*

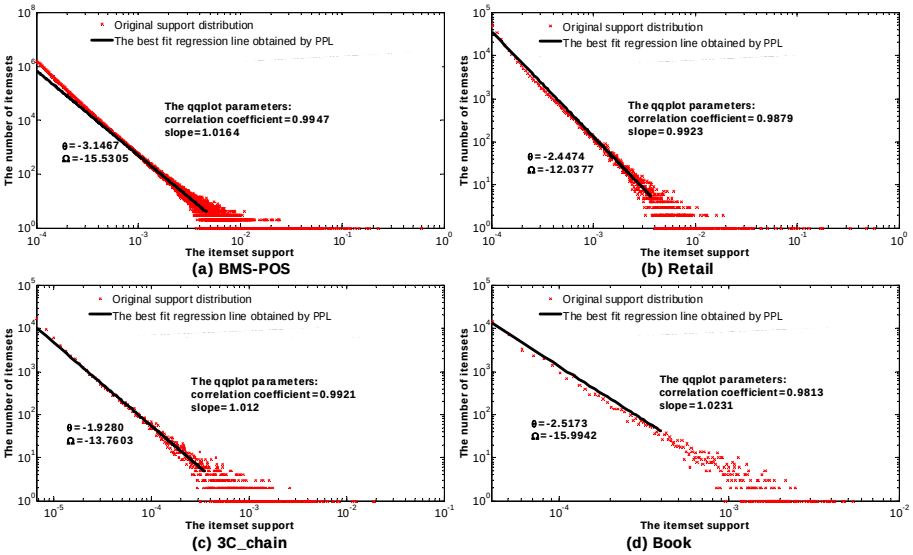


Fig. 9. The results of algorithm *PPL*.

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 Proposition 1.

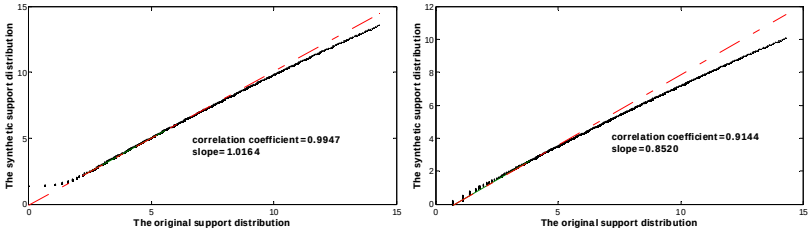
4 Experimental Studies

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 Section 2, we in this section 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 $|T_e| = 5$; (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 Figure 9(a)~Figure 9(d), where the original support distributions and the *best fit regression lines* obtained by algorithm *PPL* (with their 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 Figure 11, where the execution time of "Brute force approach" indicates the time to retrieve the original support distribution in Figure 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 the extremely large time 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.



(a) The qqplot of BMS-POS (δ =-0.8) (b) The qqplot of BMS-POS (δ =-0.6)

Fig. 10. The qqplot results in BMS-POS with various δ.

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. 11. The execution time of different approaches.

It is worth mentioning that the efficiency gain in Figure 11, 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.

Same as the experiments in [2], the quantitative analysis of algorithm *PPL* will be evaluated by the traditional method of quantile-quantile plot (qqplot), as the one shown in Figure 10. 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 both are close to one, we can claim that the real distribution and the synthetic distribution are from the same distribution [2], meaning that the regression line can perfectly represent the data distribution.

The qqplots on various correlation coefficient thresholds δ are shown in Figure 10, where Figure 10(a) is the qqplot corresponding to the result of Figure 9(a). We can find that the qqplot in Figure 10(a) is close to linear, except points with respect to very low supports and very high supports. Note that points with respect to high supports have been observed not exactly following the power-law relationship and points with respect to low supports in the BMS-POS dataset upwardly 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 mostly correctly fit the real distribution. Furthermore, when we increase the threshold δ, as shown in Figure 10(b), the estimated quality degrades, showing the importance of the criterion 2 of the *best fit regression line*. Indeed, a regression line with the low correlation coefficient loses its effectiveness to estimate the power-law relationship, even though it satisfies criterion 3, i.e., having the minimum testing error.

Due to space limitations, other qqplot results of four real datasets are summarized in Figure 12. At first, we observe results with various $|T_e|$. Note that the case $|T_e| = 0$ can be deemed as the naive approach discussed in Section 3.3. As can be seen, the naive approach cannot correctly model the distribution since the correlation coefficient and the slope deviate a lot 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 will result in the correlation coefficient and the slope close to one. Note that without loss of generality, the results of $|T_e| = 0$ and $\delta = -0.5$ can be

	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
δ=-0.5	0.84	0.86	0.91	1.08	0.92	0.94	0.87	1.11
δ=-0.9	0.99	1.01	0.99	1.02	0.97	1.06	0.98	1.03
$\|T_e\|$=0; δ=-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

Fig. 12. The qqplot results of four real datasets.

deemed 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-5000 itemsets can be efficiently retrieved in the sample, indicating the efficiency and effectiveness of algorithm *PPL*. We also investigate the influence 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 insensitive to the sample size if the sample size is not arbitrarily small. Finally, we observe the result of various w. Note that Proposition 1 will not hold when w is large. Thus it can be seen that $w = 100$ slightly degrades the estimated quality of algorithm *PPL*. Since the goal of *histogram* 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 Case 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 as *BTS* (Buffer-Trie-SetGen), for mining approximate frequent itemsets over data streams [14], and apply algorithm *PPL* as its pre-processing step.

Background Review: The one-pass algorithm, *BTS*, utilizes the concept of the ϵ-deficient synopsis to approximately maintain possible frequent itemsets, which contains a parameter ϵ to control the incorrect frequent itemsets incurred, as illustrated in Figure 1. Originally, algorithm *BTS* preserves the *recall* of the output but sacrifices the *precision* by allowing false positives, which yields that non-frequent itemsets whose supports fall in $s - \epsilon$ and s will be identified as frequent by *BTS*, where s denotes the minimum support. Given a set of true frequent itemsets A and a set of obtained frequent itemsets B, the *precision* is defined as $\frac{|A \cap B|}{|B|}$, and another measurement of the quality, i.e., the *recall*, is defined as $\frac{|A \cap B|}{|A|}$. For comparison purposes, the work in [23] extends *BTS* to be a negative-positive oriented algorithm by deliberately setting the minimum support as $s + \epsilon$. As such, the output will contain only those frequent itemsets with support exceeding s but frequent itemsets between s and $s + \epsilon$ may be not included in the output. Indeed, the *precision* in the negative positive scheme will be equal to one while compromising the *recall* of the output.

As pointed out in [23], the memory consumption (to store the potential frequent itemsets) and the resulting quality of algorithm *BTS* relies on the value of ϵ. However, the decision of ϵ remains unsolved (the default in [14] is $\epsilon = 0.1s$, which is not proper in all situations).

s (%)	Type	Apply PPL	Desired Recall	Desired Precision	ϵ (%)	Obtained Recall	Obtained Precision
0.1	False-Positive	NO	-	-	0.01	1	0.73
	False-Positive	YES	1	0.8	0.008	1	0.82
	False-Positive	YES	1	0.9	0.002	1	0.91
	False-Negative	NO	-	-	0.01	0.96	1
	False-Negative	YES	0.9	1	0.014	0.92	1
	False-Negative	YES	0.8	1	0.019	0.83	1
0.05	False-Positive	NO	-	-	0.005	1	0.68
	False-Positive	YES	1	0.8	0.0042	1	0.78
	False-Positive	YES	1	0.9	0.0038	1	0.93
	False-Negative	NO	-	-	0.005	0.82	1
	False-Negative	YES	0.9	1	0.004	0.89	1
	False-Negative	YES	0.8	1	0.005	0.82	1

Fig. 13. The result of applying algorithm *PPL* prior to algorithm *BTS* (in the BMS-POS dataset).

Formally, a large ϵ may lead to a small memory consumption but have the acute accuracy loss (either the *recall* or the *precision*). In contrast, a small ϵ will result in a better model accuracy at the cost of the memory consumption. It is indeed difficult to determine the proper decision of ϵ by users.

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 have the desired model accuracy and the smallest memory consumption with respect to the resulting accuracy. Specifically, after executing algorithm *PPL* to obtain the slope θ and the Y-intercept Ω, the itemset support distribution, as illustrated in Figure 1, can be plotted. Accordingly, we can perform the following procedure prior to the execution of algorithm *BTS*:

(1) Calculate the approximate number of frequent itemsets, f_s, with the given minimum support s, i.e., $f_s = \int_s^1 \left(e^{\Omega} \times x^{\theta}\right) dx$;

(2) If the false negative is permitted, identify ϵ as $\arg\min_{\alpha} \left\{ \frac{f_{\alpha}}{f_s} \geq r \right\}$, where $f_{\alpha} = \int_{s+\alpha}^1 \left(e^{\Omega} \times x^{\theta}\right) dx$. If the false positive is permitted, identify ϵ as $\arg\min_{\beta} \left\{ \frac{f_{\beta}}{f_s} \geq p \right\}$, where $f_{\beta} = \int_{s-\beta}^1 \left(e^{\Omega} \times x^{\theta}\right) dx$.

Simulation Results: We then investigate whether algorithm *PPL* can help algorithm *BTS* to obtain the desired quality. The results are shown in Figure 13, 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 [14][23]. 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, respectively. As shown in Figure 13, it can be seen that the obtained *recall* and *precision* is quite close to the desired one while we utilize algorithm *PPL* to estimate the characteristics of the *itemset support distribution*, indicating the prominent advantage of algorithm *PPL*. On the other hand, the default $\epsilon = 0.1s$ indeed results in an undesired loss of the quality in some cases. Interestingly, we can find that $\epsilon = 0.1s$ may obtain the *recall* larger than the desired one in the case of false negatives, which will inevitably lead to the large memory consumption. In such cases, algorithm *PPL* can help to suggest a relatively large ϵ, which can obtain the desired *recall* while also leading to the smaller memory consumption. In this experiment, we demonstrate the applicability of algorithm *PPL* to be a prominent pre-processing means for data mining applications.

5 Conclusions

In this paper, we demonstrated that the power-law relationship appears in the distribution of itemset supports in the real datasets. Discovering such a relationship is useful for many applications. To avoid the costly process of retrieving all itemsets, we proposed algorithm *PPL* to efficiently extract characteristics of the power-law relationship. As shown in the experimental results, algorithm *PPL* is able to efficiently extract the characteristics of the power-law relationship with high accuracy. In addition, while applying algorithm *PPL* prior to discover approximate frequent itemsets, a subtle parameter can be appropriately determined, showing its prominent advantage of being an important pre-processing means for mining applications.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of VLDB*, 1994.
2. Z. Bi, C. Faloutsos, and F. Korn. The "DGX" Distribution for Mining Massive, Skewed Data. In *Proc. of SIGKDD*, 2000.
3. L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proc. of IEEE INFOCOM*, 1999.
4. Y.L. Cheung and A.W. Fu. Mining Association Rules without Support Threshold: with and without Item Constraints. In *TKDE*, 2004.
5. K.-T. Chuang and M.-S. Chen. Efficient Top-k Frequent Pattern Discovery for Large Databases in the Presence of the Memory Constraint. *Technical Report*, 2005.
6. W. G. Cochran. *Sampling Techniques*. John Wiley and Sons, 1977.
7. G. Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. In *Proc. of SDM*, 2005.
8. L. Egghe. The distribution of n-grams. *Scientometrics*, 2000.
9. F. Geerts, B. Goethals, and J. V. d. Bussche. A tight upper bound on the number of candidate patterns. In *Proc. of IEEE ICDM*, 2001.
10. A. Ghoting, G. Buehrer, S. Parthasarathy, Y.Chen D. Kim, A. Nguyen, and P. Dubey. Cache-conscious frequent pattern mining on a modern processor. In *Proc. of VLDB*, 2005.
11. J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.
12. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. of ACM SIGMOD*, 2000.
13. Y. Ioannidis. The history of histograms. In *Proc. of VLDB*, 2003.
14. G. S. Manku and R. Motwani. Approximate frequency counts over streaming data. In *Proc. of VLDB*, 2002.
15. A. Metwally, D. Agrawal, and A. E. Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proc. of ICDT*, 2005.
16. S. Orlando, P. Palmerini, R. Perego, and F. Silvestri. Adaptive and resource-aware mining of frequent sets. In *Proc. of ICDM*, 2002.
17. J.-S. Park, M.-S. Chen, and P. S. Yu. An effective hash based algorithm for mining association rules. In *Proc. of SIGMOD*, 1995.
18. G. Ramesh, W. A. Maniatty, and M. J. Zaki. Feasible itemset distributions in data mining: Theory and application. In *Proc. of ACM PODS*, 2003.
19. J. A. Rice. *Mathematical statistics and data analysis*. Duxbury Press, 1995.
20. H. Toivonen. Sampling large databases for association rules. In *Proc. of VLDB*, 1996.
21. J. Wang, J. Han, Y. Lu, and P. Tzvetkov. TFP: An Efficient Algorithm for Mining Top-K Frequent Closed Itemsets. In *TKDE*, 2005.
22. R. C.-W. Wong and A.W. Fu. Mining top-k itemsets over a sliding window based on zipfian distribution. In *Proc. of SIAM SDM*, 2005.
23. J. X. Yu, Z. Chong, H. Lu, and A. Zhou. False positive or false negative: Mining frequent itemsets from high speed transactional data streams. In *Proc. of VLDB*, 2004.
24. M.J. Zaki, S. Parthasarathy, Wei Li, and M. Ogihara. Evaluation of sampling for data mining of association rules. In *Int. Workshop on Research Issues in Data Engineering*, 1997.
25. Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proc. of SIGKDD*, 2001.
26. G.K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley Press, 1949.