# Summarizing Itemset Patterns: A Profile-Based Approach*

Xifeng Yan Hong Cheng Jiawei Han Dong Xin

Department of Computer Science
University of Illinois at Urbana-Champaign
{xyan, hcheng3, hanj, dongxin}@uiuc.edu

## ABSTRACT

Frequent-pattern mining has been studied extensively on scalable methods for mining various kinds of patterns including itemsets, sequences, and graphs. However, the bottleneck of frequent-pattern mining is not at the efficiency but at the interpretability, due to the huge number of patterns generated by the mining process.

In this paper, we examine how to summarize a collection of itemset patterns using only $K$ representatives, a small number of patterns that a user can handle easily. The $K$ representatives should not only cover most of the frequent patterns but also approximate their supports. A generative model is built to extract and profile these representatives, under which the supports of the patterns can be easily recovered without consulting the original dataset. Based on the restoration error, we propose a quality measure function to determine the optimal value of parameter $K$. Polynomial time algorithms are developed together with several optimization heuristics for efficiency improvement. Empirical studies indicate that we can obtain compact summarization in real datasets.

**Categories and Subject Descriptors:** H.2.8 [Database Management]: Database Applications - Data Mining

**General Terms:** Algorithms

**Keywords:** frequent pattern, summarization, probabilistic model

## 1. INTRODUCTION

Mining frequent patterns is an important data mining problem with broad applications, including association rule mining, indexing, classification, and clustering (see e.g., [2, 26, 25, 8, 14]). Recent studies on frequent-pattern mining have seen significant performance improvements on efficient identification of various kinds of patterns, e.g., itemsets, sequences, and graphs ([2, 3, 13, 7]). Patterns with other interesting measures were also examined extensively (see e.g., [10, 5, 19, 24, 18]). However, the major challenge of frequent-pattern mining is not at the efficiency but at the *interpretability*: the huge number of frequent patterns makes the patterns themselves difficult to explore, thus hampering the individual and global analysis of discovered patterns.

There are two sources leading to the interpretability issue. First, the rigid definition of frequent patterns often generates a large number of redundant patterns, most of which are slightly different. A pattern is frequent if and only if it occurs in at least $\sigma$ fraction of a dataset. According to this definition, any subset of a frequent itemset is frequent. This downward closure property leads to an explosive number of frequent patterns. For example, a frequent itemset with $n$ items may generate $2^n$ sub-itemsets, all of which are frequent. The introduction of closed frequent itemsets [19] and maximal frequent itemsets [10, 5] can partially alleviate this redundancy problem. A frequent pattern is *closed* if and only if a super-pattern with the same support does not exist. A frequent pattern is *maximal* if and only if it does not have a frequent super-pattern. Unfortunately, for any pattern $\alpha$, as long as there is a small disturbance on the transactions containing $\alpha$, it may generate hundreds of subpatterns with different supports. We term the original pattern, $\alpha$, a *master pattern* and its deviated subpatterns, *derivative patterns*. Intuitively, it is more interesting to examine the master patterns, rather than the derivative patterns. Unfortunately, there is no clear boundary between master patterns and their derivatives.

Secondly, as long as the number of discovered patterns is beyond tens or hundreds, it becomes difficult for a user to examine them directly. A user-friendly program should present the top-$k$ distinct patterns first and arrange the remaining patterns in a tree structure so that a user can start quickly from a small set of representative patterns. The patterns delivered by the existing top-$k$ mining algorithms such as [11] are the most frequent closed itemsets, but not distinct ones. Users often prefer distinct patterns with little overlap for interactive exploration.

In this paper, we intend to solve the pattern interpretability issue by summarizing patterns using $K$ representatives. There are three subproblems around this summarization task: *what is the format of these representatives? how can we find these representatives? and what is the measure of their quality?*

*This work was supported in part by the U.S. National Science Foundation NSF IIS-02-09199 and IIS-03-08215. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.


*KDD'05,* August 21–24, 2005, Chicago, Illinois, USA.

A number of proposals have been made to construct a *concise* and *lossless representation* of frequent patterns. For example, Pasquier et al. [19] introduced the concept of closed frequent patterns, and Calders et al. [6] proposed mining all non-derivable frequent itemsets. These kinds of patterns are concise in the sense that all of the frequent patterns can be derived from them. Unfortunately, the number of patterns generated in these two approaches is still too large to handle.

Researchers have also developed *lossy compression* methods to summarize frequent patterns: maximal patterns by Gunopulos [10], top-$k$ patterns by Han et al. [11], error-tolerant patterns by Yang et al. [27], and condensed pattern bases by Pei et al. [21]. Nevertheless, the discrimination between patterns is not emphasized in these studies. Generally speaking, a user may not only be interested in a small set of patterns, but also patterns that are significantly different. A recent approach proposed by Afrati et al. [1] uses $K$ itemsets to cover a collection of frequent itemsets. Their solution is interesting but leaves the support integration issue open: It is unknown how to cover the support information in a summarization. In this paper, we investigate this issue and further advance the summarization concept.

Given a set $\mathcal{I}$ of items $o_1, \ldots, o_d$ and a transaction dataset $D = \{t_1, \ldots, t_n\}$, where each transaction is a subset of $\mathcal{I}$. The pattern collection $F$ is a set of patterns $\alpha_1, \ldots, \alpha_m$, $\alpha_i \subseteq \mathcal{I}$. We are interested in partitioning the pattern set into $K$ groups such that the similarity within each group is maximized and the similarity between the groups is minimized. Frequent patterns are distinguished from each other not only because they have different composition, but also because they have different supports. Suppose $\alpha_i$ and $\alpha_j$ exhibit strong similarity on these two criteria. It is likely that $\alpha_i$ and $\alpha_j$ can be merged to one pattern, $\alpha_i \cup \alpha_j$. This is the intuition behind merging two patterns (or clustering them in the same group). We develop a generative model $\mathbf{M}$ to measure these two similarity criteria simultaneously. Based on this model, we are able to evaluate the quality of such merging by measuring the probability that $\alpha_i$ and $\alpha_j$ are generated by $\mathbf{M}$.

Using the above generative model, we can arrange all of the patterns in a tree structure using a hierarchical agglomerative clustering method, where patterns with the highest similarity are grouped together first. In this hierarchical tree, a user can flexibly explore patterns with different summarization granularity. Our methods can successfully compress thousands of frequent patterns into hundreds or even tens of distinct patterns.

Our major contributions are outlined as follows.

1. We propose a statistical model which is good not only at summarizing patterns, but also at integrating their supports. After compressing numerous patterns to $K$ representatives, our methods are able to recover these patterns and their supports from the $K$ representatives.

2. A principled similarity measure based on Kullback-Leibler divergence is developed to group highly correlated patterns together. We show that the summarization algorithm based on this measure can complete in polynomial time.

3. We use $K$ representatives to summarize the pattern set and devise a mechanism to estimate the support of frequent patterns from these $K$ representatives only. In addition, the estimation error is used to evaluate the summarization quality. We monitor quality changes over different $K$s in order to determine the optimal number of representatives for a given pattern set. To the best of our knowledge, ours is the first algorithm that can guide the selection of $K$, thus eliminating the obstacle for the applicability of pattern summarization.

4. Empirical studies indicate that the method can build very compact pattern summarization in many real data sets. For example, on a typical `mushroom` dataset[1], the method can summarize thousands of frequent patterns accurately using around 30 patterns.

The rest of the paper is organized as follows. Section 2 introduces the concept of pattern profile for similar frequent patterns. The details of the similarity measure, the quality evaluation function, as well as the summarization algorithms are introduced in Section 3. We report our experimental results in Section 4, discuss related work in Section 5, and conclude our study in Section 6.

## 2. PATTERN PROFILE

Let $\mathcal{I}$ be a set of items $o_1, o_2, \ldots, o_d$. A subset of $\mathcal{I}$ is called an *itemset*. A transaction dataset is a collection of itemsets, $D = \{t_1, \ldots, t_n\}$, where $t_i \subseteq \mathcal{I}$. For any itemset $\alpha$, we write the transactions that contain $\alpha$ as $D_\alpha = \{t_i | \alpha \subseteq t_i \text{ and } t_i \in D\}$.

Definition 1 (Frequent Itemset). *For a transaction dataset $D$, an itemset $\alpha$ is* frequent *if $\frac{|D_\alpha|}{|D|} \geq \sigma$, where $\frac{|D_\alpha|}{|D|}$ is called the* support *of $\alpha$ in $D$, written $s(\alpha)$, and $\sigma$ is the* minimum support threshold, *$0 \leq \sigma \leq 1$.*

Frequent itemsets have the Apriori property: any subset of a frequent itemset is frequent [2]. Since the number of subsets of a large frequent itemset is explosive, it is more efficient to mine closed frequent itemsets only.

Definition 2 (Closed Frequent Itemset). *A frequent itemset $\alpha$ is* closed *if there does not exist an itemset $\beta$ such that $\alpha \subseteq \beta$ and $D_\alpha = D_\beta$.*

Figure 1 shows a sample dataset, where the first column represents the transactions and the second the number of transactions. For example, 50 transactions have only items $a$, $c$, and $d$; and 100 transactions have only items $b$, $c$, and $d$. There are $1,150$ transactions in $D_1$. If we set the minimum support at 40%, itemset $\langle abcd \rangle$ is frequent, and so are its sub-itemsets. There are 15 frequent itemsets, among which 4 are closed. As one can see, the number of closed frequent itemsets is much less than that of frequent itemsets.

| *transaction* | *number* |
|---|---|
| acd | 50 |
| bcd | 100 |
| abcd | 1000 |

**Figure 1:** $D_1$

[1] http://fimi.cs.helsinki.fi/data/mushroom.dat

Note that in Definition 1, for a transaction that contributes to the support of a pattern, it must contain the entire pattern. The rigid definition of closed frequent patterns causes a severe problem. A small disturbance within the transactions may result in hundreds of subpatterns that could have different supports. There exists significant pattern redundancy since many patterns are actually derived from the same pattern. On the other hand, a pattern could be missed if its support is below $\sigma$ while its derivative subpatterns pass the threshold. In this situation, it is necessary to assemble small correlated subpatterns together to recover it. If we relax the exact matching criterion in the support definition and allow one item missing in a pattern, $\langle abcd \rangle$ becomes the only pattern in $D_1$.

Inspired by this example, we find that it is very important to group similar itemsets together to eliminate redundant patterns. We can merge the itemsets in each group to a master pattern. *A master pattern is the union of all the itemsets within a group.* Before we formalize the format of pattern summarization, let us first examine what a good summarization means.

Suppose $\alpha$ and $\beta$ can be grouped together to form a master pattern, $\alpha \cup \beta$. That means the supports of $\alpha$, $\beta$, and $\alpha \cup \beta$ should not be significantly different from each other. Or more fundamentally, $D_\alpha$ and $D_\beta$ should be highly correlated so that the transactions containing $\alpha$ will likely contain $\beta$, and vice versa.

| *transaction* | *number* |
|---|---|
| a | 50 |
| bcd | 100 |
| abcd | 1000 |

**Figure 2:** $D_2$

Although the support similarity is one of the major criteria in determining whether we should summarize two patterns into one, there is another measure that determines how good a summarization is. Let us compare $D_1$ with another dataset $D_2$ shown in Figure 2. For these two datasets, we can summarize all the subpatterns of $\langle abcd \rangle$ as $\langle abcd \rangle$ because their supports are very close to each other. However, the quality of these two summarizations is different. If we allow approximate matching between a pattern and a transaction, pattern $\langle abcd \rangle$ is more likely contained by transactions "abc" in $D_1$ (one item missing) than by transactions "a" in $D_2$ (three items missing). That means, the summarization of $D_1$ as $\langle abcd \rangle$ has better quality. This intuition will be clarified when we explain our pattern profile model below.

According to the above discussion, we propose using a probability profile to describe a representative, instead of using an itemset only. The profile has a probability distribution on items.

Definition 3 (Bernoulli Distribution Vector). *Let $\mathcal{I} = \{o_1, \ldots, o_d\}$ be a set of items, and $x_i$ be a boolean random variable indicating the selection of $o_i$. $\mathbf{p}(\mathbf{x}) = [p(x_1), \ldots, p(x_d)]$ is a Bernoulli distribution vector over d dimensions, where $x_1, \ldots,$ and $x_d$ are independent boolean random variables.*

Suppose patterns $\alpha_1, \alpha_2, \ldots, \alpha_l$ are grouped together to form a master pattern $\alpha_1 \cup \alpha_2 \cup \ldots \cup \alpha_l$. We can estimate the distribution vector that generates the dataset $D' = \bigcup_{i=1}^{l} D_{\alpha_i}$.

$$P(D'|\theta) = \prod_{t_j \in D'} \prod_{i=1}^{d} p(x_i = t_j^i), \tag{1}$$

where $t_j^i$ is the value of $x_i$ in the $j_{th}$ transaction and $\theta$ is a set of probability $\{p(x_i)\}$. When $t_j^i = 1$, it means that the $j_{th}$ transaction has item $o_i$.

According to maximum likelihood estimation (MLE), the "best" generative model should maximize the log likelihood $\mathrm{L}(\theta|D') = \log P(D'|\theta)$, which leads to

$$\frac{\partial \mathrm{L}(\theta|D')}{\partial \theta} = 0 \tag{2}$$

The well-known result is

$$p(x_i = 1) = \frac{\sum_{t_j \in D'} t_j^i}{|D'|}. \tag{3}$$

That is, $p(x_i = 1)$ is the relative frequency of item $o_i$ in $D'$.

We use a probability distribution vector derived from MLE to describe the item distribution in a set of transactions. Here we formulate the concept of pattern profile.

Definition 4 (Pattern Profile). *Let $\alpha_1, \alpha_2, \ldots, \alpha_l$ be a set of patterns and $D' = \bigcup_i D_{\alpha_i}$. A profile $\mathbf{M}$ over $\alpha_1, \alpha_2, \ldots,$ and $\alpha_l$ is a triple $\langle \mathbf{p}, \phi, \rho \rangle$. $\mathbf{p}$ is a probability distribution vector learned through Eq. (3). Pattern $\phi = \cup_i \alpha_i$ is taken as the master pattern of $\alpha_1, \ldots, \alpha_l$. $\rho = \frac{|D'|}{|D|}$ is regarded as the support of the profile, also written as $s(\mathbf{M})$.*

| | a | b | c | d |
|---|---|---|---|---|
| $D_1$ | 0.91 | 0.96 | 1.0 | 1.0 |
| $D_2$ | 0.91 | 0.96 | 0.96 | 0.96 |

**Table 1: Pattern Profile**

If we want to build a pattern profile for $\langle abcd \rangle$ in $D_1$ and $D_2$, we can derive the distribution vectors for the sample datasets in Figures 1 and 2. The resulting profiles are shown in Table 1. For example, $p(a) = \frac{50+1000}{50+100+1000} = 0.91$. Table 1 demonstrates that the profile derived from $D_1$ has higher quality since it is much closer to a *perfect profile*, a profile with $p(x_i = 1) = 1.0$ for $o_i \in \phi$. In addition, without accessing the original dataset, we can conclude that $\langle bcd \rangle$ in $D_1$ is more frequent than the other size-3 subpatterns of $\langle abcd \rangle$. Pattern profile actually provides more information than the master pattern itself; it encodes the distribution of subpatterns. The key difference between our profile model and the model proposed by Afrati et al. [1] is that our profile model can accommodate the patterns themselves as well as their supports.

The support of a pattern in a dataset $D$ can be regarded as the average probability of observing a pattern from a transaction,

$$p(\alpha) \quad = \quad \sum_{t \in D} p(\alpha|t) * p(t),$$

where $p(t) = \frac{1}{|D|}$ and $p(\alpha|t) = 1$ if $\alpha \subseteq t$, 0 otherwise.

In our profile model, we can regard the probability of observing a pattern as the probability that the pattern is generated by its corresponding profile times the probability of observing this profile from a transaction,

$$p(\alpha|t) \sim p(\alpha|\mathbf{M}) * p(\mathbf{M}|t), \tag{4}$$

where we assume the conditional independence $p(\alpha|\mathbf{M}, t) = p(\alpha|\mathbf{M})$. According to this model, we can estimate the support for a given pattern $\alpha$ from the profile it belongs to.

DEFINITION 5 (ESTIMATED SUPPORT). *Let $\mathbf{M}$ be a profile over a set of patterns $\{\alpha_1, \alpha_2, \ldots, \alpha_l\}$. The estimated support of $\alpha_k$ is written as $\hat{s}(\alpha_k)$,*

$$\hat{s}(\alpha_k) = s(\mathbf{M}) \times \prod_{o_i \in \alpha_k} p(x_i = 1), \tag{5}$$

*where $s(\mathbf{M}) = \frac{|D_{\alpha_1} \cup \ldots \cup D_{\alpha_l}|}{|D|}$, $\mathbf{p}$ is the distribution vector of $\mathbf{M}$ and $x_i$ is the boolean random variable indicating the selection of item $o_i$ in pattern $\alpha_k$.*

Surprisingly, the calculation of an estimated support is only involved with $d+1$ real values: the $d$-dimensional distribution vector of a profile and the number of transactions that support the profile. This result becomes one of the most distinguishing features in our summarization model. It means that we can use very limited information in a profile to recover the supports of a rather large set of patterns.

## 3. PATTERN SUMMARIZATION

Our pattern profile model shows how to represent a set of patterns in a compact way and how to recover their supports without accessing the original dataset. However, the problem of selecting a set of similar patterns for summarization is not yet solved. We formalize this summarization problem as follows.

DEFINITION 6 (PATTERN SUMMARIZATION). *Given a set of patterns $F = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ that are mined from a database $D = \{t_1, t_2, \ldots, t_n\}$, pattern summarization is to find $K$ pattern profiles based on the pattern set $F$.*

A potential solution to the summarization problem is to group frequent patterns into several clusters such that the similarity within clusters is maximized and the similarity between clusters is minimized. Once the clustering is done, we can calculate a profile for each cluster.

We can construct a specific profile for each pattern that only contains that pattern itself. Using this representation, we can measure the distance between two patterns based on the divergence between their profiles. The distance between two patterns should reflect the correlation between the transactions that support these two patterns. Namely, if two patterns $\alpha$ and $\beta$ are correlated, $D_\alpha$ and $D_\beta$ likely have large overlap; and the non-overlapping parts exhibit high similarity. Several measures are available to fulfill this requirement. A well-known one is the Kullback-Leibler divergence between the distribution vectors in the profiles of $\alpha$ ($\mathbf{M}_\alpha$) and $\beta$ ($\mathbf{M}_\beta$),

$$\mathrm{KL}(\mathbf{p}||\mathbf{q}) = \sum_{i=1}^{d} \sum_{x_i \in \{0,1\}} p(x_i) \log \frac{p(x_i)}{q(x_i)}, \tag{6}$$

where $\mathbf{p}$ is the distribution vector of $\mathbf{M}_\alpha$ and $\mathbf{q}$ is the distribution vector of $\mathbf{M}_\beta$.

When $p(x_i)$ and $q(x_i)$ have zero probability, $\mathrm{KL}(\mathbf{p}||\mathbf{q}) = \infty$. In order to avoid this situation, we smooth the probability of $p(x_i)$ (and $q(x_i)$) with a background prior,

$$p'(x_i) = \lambda u + (1-\lambda) p(x_i),$$

where $\lambda$ is a smoothing parameter, $0 < \lambda < 1$, and $u$ could be the background distribution of item $o_i$.

When $\mathrm{KL}(\mathbf{p}||\mathbf{q})$ is small, it means that the two distribution vectors $\mathbf{p}$ and $\mathbf{q}$ are similar, and vice versa. This could be justified by Taylor's formula with remainder,

$$\begin{aligned}\mathrm{KL}(p(x_i)||q(x_i)) &= \theta log\frac{\theta}{\eta} + (1-\theta) log\frac{1-\theta}{1-\eta} \\ &= \theta - \eta + \theta o(\frac{\theta}{\eta} - 1) + \eta - \theta + (1-\theta) o(\frac{1-\theta}{1-\eta} - 1) \\ &= \theta o(\frac{\theta-\eta}{\eta}) + (1-\theta) o(\frac{\eta-\theta}{1-\eta}),\end{aligned}$$

where $\theta = p(x_i = 1)$ and $\eta = q(x_i = 1)$. When $0 < \theta < 1$ and $0 < \eta < 1$, it implies that

$$\mathrm{KL}(p(x_i)||q(x_i)) < \epsilon \Leftrightarrow p(x_i) \sim q(x_i).$$

Note that for any $o_i \in \alpha$, $p(x_i = 1) = 1$ and for any $o_i \in \beta$, $q(x_i = 1) = 1$ (after smoothing, both of them are close to 1, but not equal to 1). When two distribution vectors $\mathbf{p}$ and $\mathbf{q}$ of patterns $\alpha$ and $\beta$ are similar, transactions containing $\alpha$ will likely contain $\beta$ too, and vice versa. It indicates that these two patterns are strongly correlated, i.e., $p(x_i) \sim q(x_i) \Leftrightarrow D_\alpha \sim D_\beta$. Likely, patterns $\alpha$ and $\beta$ are derivative patterns of the same pattern $\alpha \cup \beta$. Therefore, KL-divergence can serve as a distance measure for the pattern summarization task. This conclusion can further be justified by the connection of KL-divergence with the following generative model.

Given several profiles $\{\mathbf{M}_\alpha\}$, and a pattern $\beta$, we have to decide how likely $\beta$ is generated by a profile $\mathbf{M}_\alpha$, or how likely $D_\beta$ is generated by $\mathbf{M}_\alpha$. If $D_\beta$ is generated by $\mathbf{M}_\alpha$, then we cannot tell the difference between $D_\beta$ and the transactions $D_\alpha$ covered by $\mathbf{M}_\alpha$. In this case, we can put patterns $\alpha$ and $\beta$ in one cluster. The best profile in $\{\mathbf{M}_\alpha\}$ should maximize

$$P(D_\beta|\mathbf{M}_\alpha) = \prod_{t_j \in D_\beta} \prod_{i=1}^{d} p(x_i = t_j^i),$$

where $\mathbf{p}$ is the distribution vector of $\mathbf{M}_\alpha$. It is equivalent to maximizing $\mathrm{L}(\mathbf{M}_\alpha) = \log P(D_\beta|\mathbf{M}_\alpha)$. Let $\mathbf{q}$ be the distribution vector of pattern $\beta$.

$$\begin{aligned}\frac{\mathrm{L}(\mathbf{M}_\alpha)}{n} &= \frac{1}{n} \sum_{t_j \in D_\beta} \sum_{i=1}^{d} \log p(x_i = t_j^i) \\ &= \sum_{i=1}^{d} \frac{n_i}{n} \log p(x_i = 1) + \frac{n - n_i}{n} \log p(x_i = 0) \\ &= \sum_{i=1}^{d} \sum_{x_i \in \{0,1\}} q(x_i) \log p(x_i), \end{aligned} \tag{7}$$

where $n = |D_\beta|$ and $n_i$ is the number of transactions (in $D_\beta$) having item $o_i$. We may add the entropy of $\mathbf{q}$,

$$\mathrm{H}(\mathbf{q}(\mathbf{x})) = -\sum_{i=1}^{d} \sum_{x_i \in \{0,1\}} q(x_i) \log q(x_i),$$

on the left and right side of Eq. (7). Hence,

$$\frac{\mathrm{L}(\mathbf{M}_\alpha)}{n} + \mathrm{H}(\mathbf{q}(\mathbf{x})) = \sum_{i=1}^{d} q(x_i=1)\log\frac{p(x_i=1)}{q(x_i=1)} + q(x_i=0)\log\frac{p(x_i=0)}{q(x_i=0)} = -\mathrm{KL}(\mathbf{q}(\mathbf{x})||\mathbf{p}(\mathbf{x})) \quad (8)$$

According to Eq. (8), the best profile maximizing $P(D_\beta|\mathbf{M}_\alpha)$ is the profile that minimizes $\mathrm{KL}(\mathbf{q}(\mathbf{x})||\mathbf{p}(\mathbf{x}))$. That means that KL-divergence can measure how likely a pattern is generated from a profile, indicating that it is a reasonable distance measure for grouping profiles.

In the rest of this section, we will introduce our summarization methods based on hierarchical clustering and K-means clustering, as well as the potential optimization heuristics. We are not going to explore the details of clustering algorithms, since essentially any clustering algorithm based on KL-divergence can be used. We will focus on the practical issues raised by pattern summarization.

## 3.1 Hierarchical Agglomerative Clustering

Hierarchical clustering produces a dendrogram where two clusters are merged together at each level. The dendrogram allows a user to explore the pattern space in a top-down manner and provides a global view of patterns.

**Algorithm 1** Pattern Summarization: Hierarchical Clustering

Input: Transaction dataset $D$,
Pattern set $F = \{\alpha_1, \ldots, \alpha_m\}$,
Number of representatives $K$,
Output: A set of pattern profiles $M_{C_1}, \ldots, M_{C_K}$.

1: initialize $k = m$ clusters, each of which has one pattern;
2: compute the pairwise KL divergence among $C_1, \ldots, C_k$, $d_{ij} = \mathrm{KL}(\mathbf{M}_{C_i}||\mathbf{M}_{C_j})$;
3: **while** $(k > K)$
4: select $d_{st}$ such that $s, t = \mathrm{argmin}_{i,j} d_{ij}$;
5: merge clusters $C_s$ and $C_t$ to a new cluster $C$;
6: $D_C = D_{C_s} \cup D_{C_t}$;
7: $\mathcal{I}_C = \mathcal{I}_{C_s} \cup \mathcal{I}_{C_t}$;
8: update the profile of $C$ over $D_C$ by Eq. (3);
9: calculate the KL-divergence between $C$ and the remaining clusters;
10: $k = k - 1$;
11: **return**;

Algorithm 1 outlines the pattern summarization process using a hierarchical clustering approach. At the beginning, it calculates the KL-divergence between any pair of patterns. In the following iterations, it repeats Lines 3-10 by selecting the two clusters which have the smallest KL-divergence (Line 4) and merging them into one cluster. The iteration procedure terminates when the total number of clusters becomes $K$.

A cluster $C$ is defined as a collection of patterns, $C \subseteq F$. The set of transactions that contain a pattern in $C$ is written as $D_C = \cup_\alpha D_\alpha, \alpha \in C$ and the master pattern in $C$ is written as $\mathcal{I}_C = \cup_\alpha \alpha, \alpha \in C$. The newly merged cluster inherits the transactions that support the original clusters and the patterns that are owned by the original clusters. It has a newly built profile over the merged transactions (Line 8).

The profiling construction has to scan the dataset once, thus taking $O(nd)$ for each merge operation, where $n$ is the number of transactions in $D$, and $d$ is the size of the global itemset $\mathcal{I}$. The initial KL-divergence construction (Line 2) takes $O(m^2 d)$, where $m$ is the number of patterns. For each cluster $C_i$, we can maintain a distance list between $C_i$ and other clusters and sort them in increasing order. Whenever a new cluster $C$ is generated, the two merged clusters are deleted from the distance lists in time $O(m)$. A new distance list is created for $C$ and sorted in time $O(m\log m)$. Note that we need not insert the distance from $C$ in the existing distance lists. The minimum KL-divergence can be found by checking the first element in $O(m)$ distance lists. Therefore, hierarchical clustering itself can be done in $O(m^2 \log m)$. Hence, Algorithm 1 can finish in $O(m^2 \log m + m^2 d + mnd)$.

When the dataset is very large, it may be expensive to recompute the profile by scanning the whole dataset. Fortunately, it is effective to profile a new cluster through sampling based on Hoeffding bound [12].

## 3.2 K-means Clustering

One major computation cost in hierarchical clustering is the pair-wise KL-divergence calculation. In each iteration, Algorithm 1 has to calculate $O(m)$ times of KL-divergence. In total, Algorithm 1 has to do $O(m^2)$ KL-divergence computation. One approach to eliminate the quadratic cost is to adopt K-means clustering. Using K-means, we can achieve very fast clustering for a large number of patterns.

**Algorithm 2** Pattern Summarization: K-means

Input: Transaction dataset $D$,
Pattern set $F = \{\alpha_1, \ldots, \alpha_m\}$,
Number of representatives $K$,
Output: A set of pattern profiles $M_{C_1}, \ldots, M_{C_k}$.

1: randomly select $K$ patterns as the initial clusters;
2: **for** each pattern $\alpha$ **do**
assign its membership to the cluster that has the smallest KL-divergence $\mathrm{KL}(\mathbf{M}_\alpha||\mathbf{M}_{C_j})$;
3: update the profiles of newly formed clusters by Eq. (3);
4: repeat Lines 2-3 until small change in $\mathbf{M}_{C_1}, \ldots \mathbf{M}_{C_K}$ or the summarization quality does not increase;
5: **return**;

Algorithm 2 outlines the major steps of the K-means algorithm. In the initial step, Algorithm 2 randomly selects $K$ patterns as the initial cluster centers. In the following iterations, it reassigns patterns to clusters according to the KL-divergence criterion. The profiles of newly formed clusters are then updated (Line 3 Algorithm 2). This procedure will terminate until there is only small change in $\mathbf{M}_{C_1}, \ldots,$ and $\mathbf{M}_{C_K}$ or it meets other stop conditions, e.g., the summarization quality does not increase any more (see Section 3.4).

Conceptually, Algorithm 2 is similar to distributional clustering [4] and divisive clustering [9]. The first difference is that we treat each dimension as a separate distribution while other approaches put all dimensions together and create a multinomial model. The second difference is that our qual-

ity evaluation function (see Section 3.4) is different from the mutual information loss function proposed by Dhillon et al. [9]. The third difference is that distributional clustering or divisive clustering mixes the profiles directly by assigning equal weight to each instance, while we prefer to recompute the profiles through the original dataset. In the experiment section, we will illustrate the performance difference between these two strategies.

The time complexity of Algorithm 2 is $O((mkd + knd)r)$, where $m$ is the number of patterns, $k$ is the number of clusters, $d$ is the number of distinct items, $n$ is the number of transactions, and $r$ is the number of iterations. Generally K-means clustering can complete clustering faster than hierarchical clustering. However, it cannot provide a hierarchical clustering tree for pattern navigation. Another drawback of K-means is that its output is highly related with the seed selection (Line 1, Algorithm 2), which is undesirable in some applications.

## 3.3 Optimization Heuristics

We develop two optimization heuristics to speed up the clustering process.

### 3.3.1 Closed Itemsets vs. Frequent Itemsets

We can start the clustering process either from closed frequent itemsets or from frequent itemsets. The following lemma shows that either way generates the same result.

LEMMA 1. *Given patterns $\alpha$ and $\beta$, if $\alpha \subseteq \beta$ and their supports are equal, then* $\mathrm{KL}(\mathbf{M}_\beta||\mathbf{M}_\alpha)$=$\mathrm{KL}(\mathbf{M}_\alpha||\mathbf{M}_\beta) = 0$.

**Proof.** If $\alpha \subseteq \beta$ and $s(\alpha) = s(\beta)$, then $D_\alpha = D_\beta$, which leads to the above lemma. ∎

Any frequent itemset must have a corresponding closed frequent itemset. According to Lemma 1, their profiles have zero KL-divergence, indicating that the clustering based on frequent itemsets will be the same as the clustering based on closed frequent itemsets. Therefore, we can summarize closed frequent itemsets instead of frequent itemsets. Since the number of closed frequent itemsets is usually less than that of frequent itemsets, summarizing closed itemsets can significantly reduce the number of patterns involved in clustering, thus improving efficiency.

### 3.3.2 Approximate Profiles

Another issue is whether we should rebuild the profile from scratch as Algorithm 1 (Line 8) and Algorithm 2 (Line 3) do. The profile updating dominates the computation in both algorithms since it has to access the original dataset. A potential solution is to mix two profiles directly without checking the original dataset in Algorithm 1,

$$\mathbf{p}(\mathbf{x}) = \frac{|C_s|}{|C_s| + |C_t|}\mathbf{p}_s(\mathbf{x}) + \frac{|C_t|}{|C_s| + |C_t|}\mathbf{p}_t(\mathbf{x}), \qquad (9)$$

or weigh each pattern's profile equally in Algorithm 2,

$$\mathbf{p}(\mathbf{x}) = \frac{1}{|C|}\sum_{\alpha \in C}\mathbf{p}_\alpha(\mathbf{x}). \qquad (10)$$

This approximation can significantly improve clustering efficiency. However, it may affect the summarization quality since the mixed profile may no longer reflect the real distribution. Traditional clustering algorithms usually assume the instances are sampled independently (i.i.d). However, in our case, the i.i.d. assumption for frequent patterns is not valid. Most of the frequent patterns are not independent at all. It may be incorrect to have an equal weight for each pattern or weigh two clusters according to their sizes.

## 3.4 Quality Evaluation

One way to evaluate the quality of a profile is to calculate the probability of generating its master pattern from its profile, $p(\phi) = \prod_{o_i \in \phi} p(x_i = 1)$. The closer $p(\phi)$ to 1, the better the quality. Because the master patterns from different profiles have different sizes, it is pretty hard to combine multiple $p(\phi)$s to give a global quality assessment. We are going to examine an additional measure in this section.

Through Eq. (5) in Section 2, we show that the support of a pattern covered by one profile can be estimated through its distribution vector directly. A high quality profile should have this estimation as close as possible to the real support. When we apply this measure to a set of profiles, we encounter an ambiguity issue since one pattern may have different estimated supports according to different profiles. Suppose we are given minimum information: a set of profiles, each of which is a triple ⟨distribution vector, master pattern, support⟩. The information about which pattern belongs to which profile is not given. For any pattern $\alpha$, it could be a subset of several master patterns. In this situation, we may get multiple support estimations for $\alpha$. Which one should we select? A simple strategy is to select the maximum one,

$$\hat{s}(\alpha_k) = \max_{\mathbf{M}} s(\mathbf{M}) \times \prod_{o_i \in \alpha_k} p_{\mathbf{M}}(x_i = 1). \qquad (11)$$

This strategy is consistent with the support recovery for a frequent pattern given a set of closed frequent patterns. Let $F$ be a set of closed frequent patterns. For any frequent pattern $\alpha$, its support is the same as the maximum support of its super-pattern $\beta$, $\alpha \subseteq \beta$ and $\beta \in F$.

DEFINITION 7 (RESTORATION ERROR). *Given a set of profiles $\mathbf{M}_1, \ldots, \mathbf{M}_K$ and a testing pattern set $T = \{\alpha_1, \alpha_2, \ldots, \alpha_l\}$, the quality of a pattern summarization can be evaluated by the following average relative error, called restoration error,*

$$J = \frac{1}{|T|}\sum_{\alpha_k \in T}\frac{|s(\alpha_k) - \hat{s}(\alpha_k)|}{s(\alpha_k)}. \qquad (12)$$

Restoration error measures the average relative error between the estimated support of a pattern and its real support. If this measure is small enough, it means that the estimated support of a pattern is quite close to its real support. A profile with small restoration error can provide very accurate support estimation.

The measure in the above definition is determined by the testing pattern set. We may choose the original patterns (which have been summarized into $K$ master patterns) as the testing case. We can also assess the quality over the itemsets that are estimated to be frequent, i.e.,

$$J_c = \frac{1}{|T'|}\sum_{\alpha_k \in T'}\frac{|\hat{s}(\alpha_k) - s(\alpha_k)|}{\hat{s}(\alpha_k)}, \qquad (13)$$

where $T'$ is the collection of the itemsets generated by the master patterns in profiles and $\hat{s}(\alpha_k) \geq \sigma$.

The measure $J$ tests "frequent patterns", some of which may be estimated as "infrequent", while $J_c$ tests "estimated

frequent patterns", some of which are actually "infrequent". Therefore, these two measures are complementary to each other. As long as $J_c$ is relatively small, we can obtain the support of a generated itemset with high accuracy and determine whether it is frequent or not. The following lemma shows that if we summarize closed frequent itemsets using $K$ profiles, the subsets generated by the master patterns in these $K$ profiles will cover all of the frequent itemsets.

LEMMA 2. *Let $\{\mathbf{M}_1, \ldots, \mathbf{M}_K\}$ be a set of profiles learned over a collection of closed frequent itemsets $\{\alpha_1, \ldots, \alpha_m\}$ using hierarchical clustering or K-means clustering. For any frequent itemset $\pi$, there must exist a profile $\mathbf{M}_k$ such that $\pi \subseteq \phi_k$, where $\phi_k$ is the master itemset of $\mathbf{M}_k$.*

**Proof.** For any frequent itemset $\pi$, there exists a closed frequent itemset $\alpha_i$ such that $\pi \subseteq \alpha_i$. $\alpha_i$ must belong to one cluster, say $M_k$. Hence, $\alpha_i \subseteq \phi_k$, where $\phi_k$ is the master itemset of $\mathbf{M}_k$. Therefore, $\pi \subseteq \phi_k$. ∎

## 3.5 Optimal Number of Profiles

The summarization quality is related to the setting of $K$, i.e., the number of profiles. A smaller $K$ is always preferred. Nevertheless, when the summarization is too coarse, it may not provide any valuable information. In order to determine the optimal number of profiles, we can apply a constraint on the summarization result. For example, for any profile $\mathbf{M} = (\mathbf{p}, \phi, \rho)$, we require $p(x_i) \geq 0.9$ for any $i$ such that $o_i \in \phi$. The optimal number is the smallest $K$ that does not violate this constraint. In this section, we examine the summarization quality change to determine the optimal value of $K$.

When two distribution vectors $\mathbf{p}$ and $\mathbf{q}$ calculated from patterns $\alpha$ and $\beta$ are close to each other, transactions containing $\alpha$ will likely contain $\beta$ too, and vice versa. Thus, $D_\alpha$ is similar to $D_\beta$ and $D_\alpha \cup D_\beta$ is similar to both $D_\alpha$ and $D_\beta$. Let $\mathbf{r}$ be the probability distribution vector over $D_\alpha \cup D_\beta$. When $\mathbf{p}$ and $\mathbf{q}$ are close, the mixture $\mathbf{r}$ will be close to them too. Therefore, the support estimation of any pattern according to Eq. (11) will not change much when we merge $\alpha$ and $\beta$. It implies that the merge of two similar profiles will not significantly change the summarization quality.

On the other hand, if a clustering algorithm has to merge two profiles $\mathbf{M}_\alpha$ and $\mathbf{M}_\beta$ that have a large KL-divergence, it may dramatically change the estimated support of a given pattern. Therefore, when we gradually decrease the value of $K$, we will observe the deterioration of the summarization quality. By checking the derivative of the quality over $K$,

$$\frac{\partial J}{\partial K},$$

we can find the optimal value of $K$ practically: If $J$ increases suddenly from $K^*$ to $K^* - 1$, $K^*$ is likely to be a good choice for the optimal number of profiles.

Figure 3 shows the summarization quality along the number of profiles for a real dataset, Mushroom. The support threshold is set at 25%. We use hierarchical clustering to summarize this pattern set. The curve indicates that there are three optimal candidates to choose: 30, 53, and 76. A user can select one of them for examination based on their need on the summarization quality. Figure 4 shows the quality change along the number of profiles. The derivative of $J$ clearly indicates three huge quality changes.

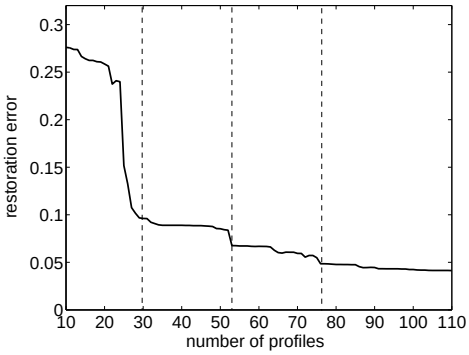


**Figure 3: Mushroom 25%: $J$**

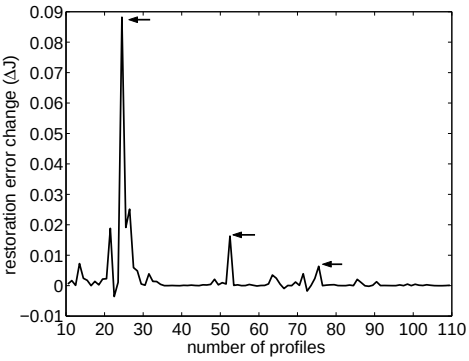


**Figure 4: Mushroom 25%: $\triangle J$**

# 4. EMPIRICAL STUDY

In this section, we provide the empirical evaluation for the effectiveness of our summarization algorithm. We use two kinds of datasets in our experiments: three real datasets and a series of synthetic datasets. The clustering algorithms are implemented in Visual C++. All of our experiments are performed on a 3.2GHZ, 1GB-memory, Intel PC running Windows XP.

## 4.1 Real Datasets

**Mushroom.** The first dataset, Mushroom, is available in the machine learning repository of UC Irvine. We obtained a variant from FIMI repository. This dataset consists of 8124 hypothetical mushroom samples with 119 distinct features. Each sample has 23 features. A support threshold of 25% was used to collect 688 closed frequent patterns (5545[2] frequent itemsets).

Figure 5 shows the average restoration error over the closed frequent patterns ($J$) and the average restoration error over the frequent itemsets generated by the resulting profiles ($J_c$). The two restoration errors $J$ and $J_c$ are quite close. This indicates that we can use the $K$ representative profiles to properly estimate the supports of the original closed patterns as well as the supports of the patterns not in the original set but derivable from the profiles. We also examined the standard deviation of the two restoration errors and found they are pretty close to $J$ or $J_c$. From this aspect, our sum-

[2]The number of frequent itemsets is slightly different from the reported [1]; our results are verified by several public softwares in FIMI repository, http://fimi.cs.helsinki.fi/src.

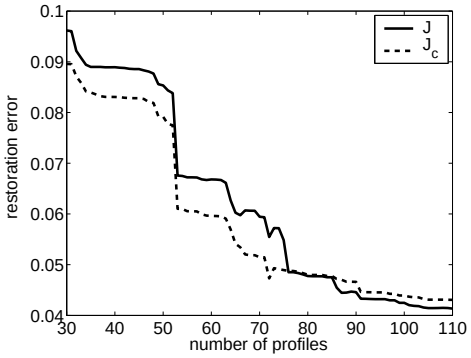


**Figure 5: Mushroom: Hierarchical Clustering**

marization method is stable and accurate in the restoration of the patterns and their supports.

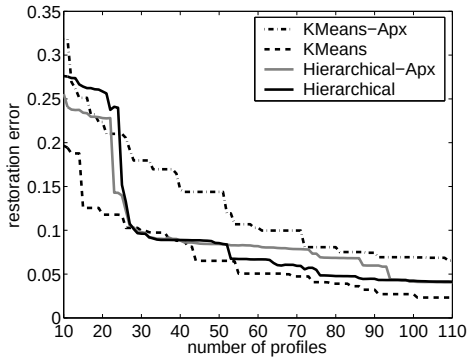


**Figure 6: Mushroom**

Figure 6 shows the average restoration error ($J$) over hierarchical clustering and K-means clustering with or without applying the profile approximation heuristics. In KMeans-Apx and Hierarchical-Apx, we do not go back to the dataset to rebuild the cluster profilers. Instead, we directly interpolate the existing profiles as described in Section 3.3.2. Overall, the 688 closed patterns can be successfully summarized into 30 profiles with good quality – the average restoration error at 30 profiles is less than 0.1. In other words, the error of estimating the support for a pattern is less than 10% of that pattern's real support, and even as low as 5% when we summarize them into 53 profiles or over.

**BMS-Webview1.** The second dataset, BMS-Webview1, is a web click-stream dataset from a web retailer company: Gazelle.com. The data was provided by Blue Martini Software [16]. In this experiment, we set the minimum support threshold at 0.1% and got 4,195 closed itemsets. BMS-Webview1 is completely different from the Mushroom dataset. It consists of many small frequent itemsets over a large set of items (itemsets of size 1–3 make up 84.55% of the total 4,195 patterns versus 14.10% for Mushroom), which makes the summarization more difficult.

Figure 7 shows the average restoration error over hierarchical clustering and K-means clustering with or without applying the profile approximation heuristics. As shown in the figure, when we use the profile approximation heuristics in K-means, the summarization quality is much worse than that of building the profiles from scratch. The restoration

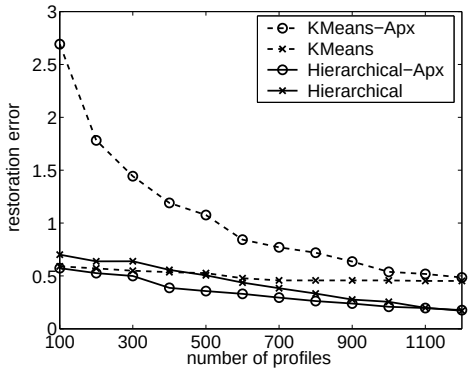


**Figure 7: BMS-Webview1**

at $K = 100$ is 259% for K-means with the profile approximation while it is around 60% for hierarchical clustering or K-means without profile approximation.

Overall, the summarization quality for BMS-Webview1 patterns is worse than that of Mushroom. When we use 1,200 profiles to summarize the patterns, the restoration error is 17%. The difference is due to the underlying distribution of patterns. There is much redundancy between patterns in Mushroom. By examining the 688 closed patterns of Mushroom, we can easily identify "rough" groups of patterns, where patterns in each group look very similar in composition and differ in support by only a very small number. Intuitively, these patterns can be summarized accurately. In BMS-Webview1, patterns are much shorter and sparser. Apparently, it is not good to put two itemsets into one cluster while they have very little overlap in composition. Such pattern distribution can be also explained from the data characteristics, i.e., the click-stream dataset usually contains random and short user access sessions, where the pattern explosive problem is not as serious as in dense datasets like Mushroom.

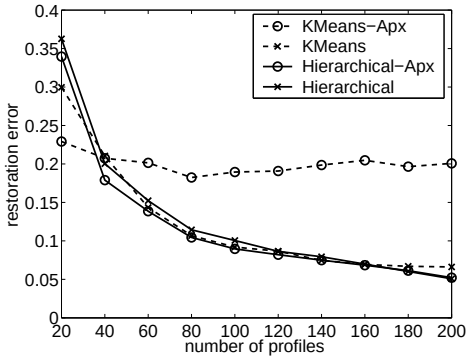


**Figure 8: Replace**

**Replace.** The third dataset, Replace, is a program trace dataset collected from the "replace" program, one of *Siemens Programs*, which are widely used in software engineering research [15]. We recorded the program call and transition information of 4,395 correct executions. Each type of program calls and transitions is taken as one item. Overall, there are 66 kinds of calls and transitions. The frequent patterns mined in this dataset may reveal the normal pro-

gram execution structures, which can be compared with the abnormal executions for bug isolation.

We set the minimum support threshold at 3% in this experiment and obtained 4,315 closed frequent execution structures. Figure 8 shows the average restoration error over hierarchical clustering and K-means clustering.

All methods except K-means clustering with profile approximation achieve good summarization quality. The quality of the three methods is quite close and the restoration error is about 6% when we use 200 clusters. The curves indicate that there are two optimal $K$ values to choose: 40 and 80, since the error decreases significantly at these points compared with their neighboring points. For K-means with profile approximation, we further examine the summarization quality. Though the average restoration error does not decrease as we increase the number of profiles, the standard deviation of the error does lower – the standard deviation is 21.37% at $K = 20$ versus 7.39% at $K = 200$. It means that the summarization quality improves as we use more profiles.

## 4.2 Synthetic Datasets

In this experiment, we want to study how the underlying distribution in patterns can affect the summarization quality. We used a series of synthetic datasets with different distributions to test it. The synthetic data generator is provided by IBM and is available at http://www.almaden.ibm.com/software/ quest/Resources/index.shtml. Users can specify parameters like the number of transactions, the number of distinct items, the average number of items in a transaction, etc., to generate various kinds of data.

We generated seven transaction datasets, where we vary the number of items to control the distribution of patterns in these datasets. Each dataset has 10,000 transactions, each of which has an average of 20 items. The number of distinct items in each dataset varies from 40, 60, up to 160. Since it may not be fair to compare the result using a fixed support threshold in these datasets, we intentionally obtained the top-500 frequent closed patterns from each dataset and summarize them into 50 and 100 profiles using hierarchical clustering. Figure 9 shows the average restoration error $J$.

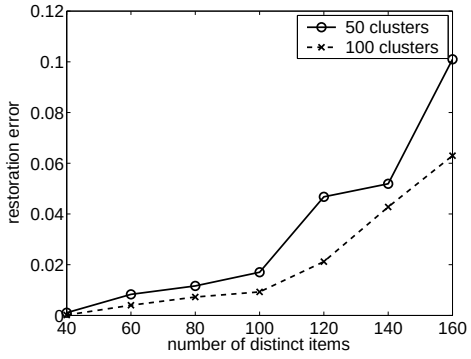


**Figure 9: Synthetic Data: Hierarchical Clustering**

As shown in Figure 9, the summarization quality deteriorates as the number of distinct items in the datasets increases. When the number of items is small, the dataset has a dense distribution. There exists much redundancy between patterns. So, they can be summarized with small restoration errors; while for a dataset with a large number of items, the patterns are sparsely distributed. Thus, it is harder to summarize them with reasonably good quality. This experiment shows that the summarization quality has close relation with the patterns themselves. This result is also observed in the previous real datasets. Dense data with high redundancy can be summarized with good quality while sparse patterns with little overlap cannot be grouped together very well.

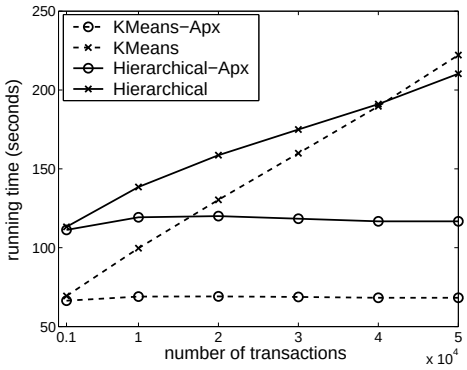


**Figure 10: Synthetic Data: Efficiency**

We also tested the running time of our pattern summarization methods over six synthetic datasets by varying the number of transactions from 1,000, 10,000 up to 50,000. A set of about 1,100 closed patterns is obtained from each dataset using a minimum support of 10%. We tested hierarchical clustering and K-means clustering with or without applying the profile approximation heuristics over these datasets. Figure 10 shows the running time. The running time of Hierarchical-Apx and KMeans-Apx does not change with the transaction number because we simply interpolate the profiles as described in Section 3.3.2. The running time of hierarchical clustering and K-means clustering without using profile approximation heuristics increases linearly with the number of transactions. This figure shows that profile approximation can really improve the efficiency.

# 5. RELATED WORK

Lossless methods have been proposed to reduce the output size of frequent itemset patterns. Pasquier et al. [19] developed the concept of closed frequent patterns, and Calders et al. [6] proposed mining non-derivable frequent itemsets. These kinds of patterns are concise in the sense that all of the frequent patterns can be derived from these representations. Lossy compression methods were also developed in parallel: maximal patterns by Gunopulos [10], error-tolerant patterns by Yang et al. [27] and Pei et al. [22], and Top-k patterns by Han et al. [11]. These methods can reduce the pattern set size further. For example, in maximal pattern mining, all of the frequent subpatterns are removed so that the resulting pattern set is very compact. Besides lossless and lossy methods, other concepts like support envelopes [23] were also proposed to explore association patterns.

Our pattern approximation model is also related to the probabilistic models developed by Pavlov et al. [20] for query approximations, where frequent patterns and their supports are used to estimate query selectivity. Mielikäinen and Mannila [17] proposed an approximation solution based

on ordering patterns. The closest work to our study is a novel pattern approximation approach proposed by Afrati et al. [1], which uses $k$ frequent (or border) itemsets to cover a collection of frequent itemsets. Their result can be regarded as a generalization of maximal frequent itemsets. In [1] Afrati et al. mentioned the support integration issue: It is unknown how to integrate the support information with the approximation. In this paper, we solved this problem, thus advancing the summarization concept. Interestingly, the $K$ representatives mined by our approach can be regarded as a generalization of closed frequent itemsets.

# 6. CONCLUSIONS

We have examined how to summarize a collection of itemset patterns using only $K$ representatives. The summarization will solve the interpretability issue caused by the huge number of frequent patterns. Surprisingly, our profile model is able to recover frequent patterns as well as their supports, thus answering the support integration issue raised by Afrati et al. [1]. We also solved the problem of determining the optimal value of $K$ by monitoring the change of the support restoration error. Empirical studies indicate that we can obtain very compact summarization in real datasets. Our approach belongs to a post-mining process; we are working on algorithms that can directly apply our profiling model to the mining process.

# 7. REFERENCES


[1] F. Afrati, A. Gionis, and H. Mannila. Approximating a collection of frequent sets. In *Proc. of 2004 ACM Int. Conf. on Knowledge Discovery in Databases (KDD'04)*, pages 12 – 19, 2004.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of 1993 Int. Conf. on Management of Data (SIGMOD'93)*, pages 207–216, 1993.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of 1995 Int. Conf. on Data Engineering (ICDE'95)*, pages 3–14, 1995.

[4] L. Baker and A. McCallum. Distributional clustering of words for text classification. In *Proc. of 1998 ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR'98)*, pages 96–103, 1998.

[5] R. Bayardo. Efficiently mining long patterns from databases. In *Proc. of 1998 Int. Conf. on Management of Data (SIGMOD'98)*, pages 85–93, 1998.

[6] T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *Proc. of 2002 European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD'02)*, pages 74–85, 2002.

[7] L. Dehaspe, H. Toivonen, and R. King. Finding frequent substructures in chemical compounds. In *Proc. of 1998 Int. Conf. on Knowledge Discovery and Data Mining (KDD'98)*, pages 30–36, 1998.

[8] M. Deshpande, M. Kuramochi, and G. Karypis. Frequent sub-structure-based approaches for classifying chemical compounds. In *Proc. of 2003 Int. Conf. on Data Mining (ICDM'03)*, pages 35–42, 2003.

[9] I. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *J. of Machine Learning Research*, 3:1265–1287, 2003.

[10] D. Gunopulos, H. Mannila, R. Khardon, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. In *Proc. 1997 ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'97)*, pages 209 – 216, 1997.

[11] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *Proc. of 2002 Int. Conf. on Data Mining (ICDM'02)*, pages 211–218, 2002.

[12] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. American Statistical Associations*, 58:13–30, 1963.

[13] L. Holder, D. Cook, and S. Djoko. Substructure discovery in the subdue system. In *Proc. AAAI94 Workshop on Knowledge Discovery in Databases (KDD94)*, page 169180, 1994.

[14] J. Huan, W. Wang, D. Bandyopadhyay, J. Snoeyink, J. Prins, and A. Tropsha. Mining spatial motifs from protein structure graphs. In *Proc. of 8th Ann. Int. Conf. on Research in Computational Molecular Biology (RECOMB'04)*, pages 308–315, 2004.

[15] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand. Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria. In *Proc. of 16th Int. Conf. on Software engineering (ICSE'94)*, pages 191–200, 1994.

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

[17] T. Mielikäinen and H. Mannila. The pattern ordering problem. In *Prof. 7th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'03)*, pages 327–338, 2003.

[18] E. Omiecinski. Alternative interest measures for mining associations. *IEEE Trans. Knowledge and Data Engineering*, 15:57–69, 2003.

[19] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. of 7th Int. Conf. on Database Theory (ICDT'99)*, pages 398–416, 1999.

[20] D. Pavlov, H. Mannila, and P. Smyth. Beyond independence: Probabilistic models for query approximation on binary transaction data. *IEEE Trans. Knowledge and Data Engineering*, 15:1409–1421, 2003.

[21] J. Pei, G. Dong, W. Zou, and J. Han. On computing condensed frequent pattern bases. In *Proc. 2002 Int. Conf. on Data Mining (ICDM'02)*, pages 378–385, 2002.

[22] J. Pei, A. Tung, and J. Han. Fault-tolerant frequent pattern mining: Problems and challenges. In *Proc. of 2001 ACM Int. Workshop Data Mining and Knowledge Discovery (DMKD'01)*, pages 7–12, 2001.

[23] M. Steinbach, P. Tan, and V. Kumar. Support envelopes: a technique for exploring the structure of association patterns. In *Proc. of 2002 ACM Int. Conf. on Knowledge Discovery in Databases (KDD'04)*, pages 296 – 305, 2004.

[24] P. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proc. of 2002 ACM Int. Conf. on Knowledge Discovery in Databases (KDD'02)*, pages 32–41, 2002.

[25] K. Wang, C. Xu, and B. Liu. Clustering transactions using large items. In *Proc. of 8th Int. Conf. on Information and Knowledge Management (CIKM'99)*, pages 483 – 490, 1999.

[26] X. Yan, P. Yu, and J. Han. Graph indexing: A frequent structure-based approach. In *Proc. of 2004 ACM Int. Conf. on Management of Data (SIGMOD'04)*, pages 335–346, 2004.

[27] C. Yang, U. Fayyad, and P. S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proc. of 2001 ACM Int. Conf. on Knowledge Discovery in Databases (KDD'01)*, pages 194–203, 2001.