Summarization and Visualization of Multi-Level and Multi-Dimensional Itemsets

Summarization and Visualization of Multi-Level and Multi-Dimensional Itemsets

Matteo Francia, Matteo Golfarelli, Stefano Rizzi*

DISI — University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

Abstract

Frequent itemset (FI) mining aims at discovering relevant patterns from sets of transactions. In this work we focus on multi-level and multi-dimensional data, which provide a rich description of subjects through multiple features each at different levels of detail. Summarization of FIs has been only marginally addressed so far with specific reference to multi-level and multi-dimensional FIs. In this paper we fill this gap by proposing SUSHI, a framework for summarizing and visually exploring multi-level and multi-dimensional FIs. Specifically, SUSHI is based on (i) a similarity function for FIs which takes into account both their extensional (support-based) and intensional (feature-based) natures; (ii) theoretical results concerning antimonotonicity of support and similarity in multi-level settings, which allow us to propose an efficient clustering algorithm to generate hierarchical summaries; and (iii) two integrated approaches to summary visualization and exploration: a graph-based one, which highlights inter-cluster relationships, and a tree-based one, which emphasizes the relationships between the representative of each cluster and the other FIs in that cluster. SUSHI is evaluated using both a real and a synthetic dataset in terms of effectiveness, efficiency, and understandability of the summary, with reference to three different strategies for choosing cluster representatives. Overall, SUSHI shows to outperform previous approaches and to be a valuable tool to

*Corresponding author

Email address: m.francia@unibo.it, matteo.golfarelli@unibo.it, stefano.rizzi@unibo.it (Stefano Rizzi)

February 12, 2020

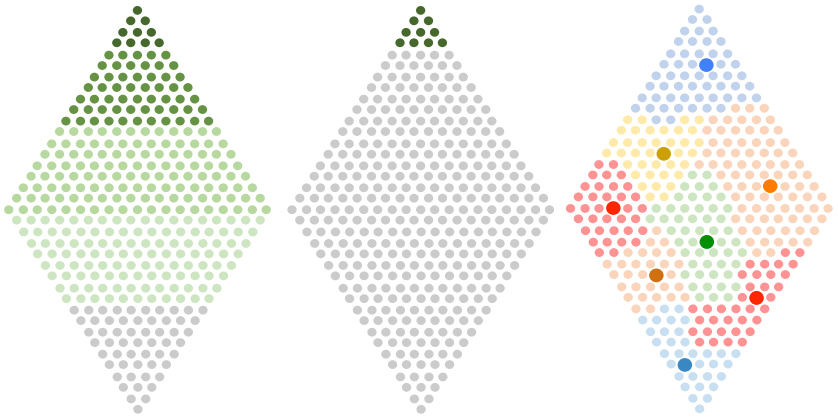
expedite the analysis of FIs. Besides, one of the three strategies for choosing cluster representatives shows to be the most effective one.
Keywords: Frequent itemset mining, Hierarchical clustering, Itemset summarization, Itemset visualization

---

1. Introduction

Frequent itemset (FI) mining is an unsupervised mining technique used in descriptive analytics to uncover frequent correlations in (possibly huge) transactional datasets [2]. In its initial formulation for market basket analysis, items correspond to products and a transaction corresponds to a set of products bought together by a customer; a set of items that appear together in many transactions is said to be frequent. More recently FI mining has been applied to other contexts, where items are not necessarily homogeneous (as is the case for plain products) but, aimed at providing a richer description of subjects and events, they are modeled as multi-dimensional and multi-level objects. In a *multi-dimensional* item [22, 9] an event is described by multiple features (e.g., product and customer age); an item is *multi-level* [22, 5] if it can be described using a hierarchy with different levels of details (e.g., products and product types). The mining of multi-dimensional and multi-level items is now well understood and extends traditional FI mining with the possibility of aggregating infrequent specific itemsets into frequent generic itemsets [22, 5].

The exponential nature of FIs —n items may lead to 2^n FIs— makes it difficult for analysts to explore their information content. Figure 1 gives a qualitative picture of the containment lattice for itemsets [22]; itemsets in the upper part of the lattice appear more frequently together in transactions. To effectively explore the lattice, data analysts normally define a frequency threshold (or even a frequency range) to cut irrelevant itemsets. Setting a low threshold (Figure 1(a)) gives a broad picture of the correlations, but the FIs selected are too many for a well-focused analysis. On the other hand, setting a high threshold (Figure 1(b)) gives a strong focus on the most frequent FIs but may lead to missing



(a) (b) (c)

Figure 1: Itemset lattice, with dark itemsets being more frequent: FI mining with a low frequency threshold (a) and a high frequency threshold (b); summarization, with big dots corresponding to cluster representatives

interesting correlations. Besides, in both cases, the FIs selected are very similar to each other, so most of them do not really give useful information to the analyst. In [48], the authors highlight the need for *summaries* to overcome these problems. Summarization selects a minimal subset of representative FIs that describe the entire population while maximizing the diversity of these representatives (Figure 1(c)). Some approaches to summarization have been proposed in the past (e.g., [9, 35]), however most of them do not consider the multi-level and multi-dimensional natures of FIs and do not provide user-friendly ways to explore summaries.

In this paper we propose *SUSHI* (*SUmmarization and viSualization of Hierarchical Itemsets*), a comprehensive framework for analyzing multi-level and multi-dimensional FIs based on original techniques for summarization and visual exploration. In SUSHI, summarization operates in synergy with FI mining to empower and simplify analysis: indeed, it makes the approach more robust with reference to the frequency threshold and enables both specific and general patterns to be discovered. On top of that, visual exploration unveils and highlights hidden information, supports the process of understanding and decision making, and allows analysts to focus their attention on what is important.

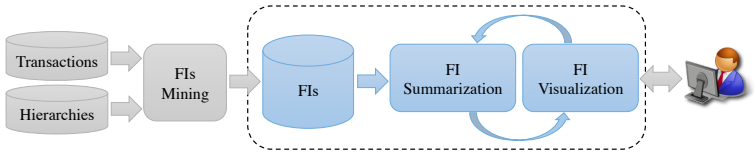


Figure 2: A functional architecture of SUSHI

Note that some visual representations for FIs have been previously proposed (e.g., [27, 7]); however, none of them enables an in-depth and multi-resolution navigation of summaries.

Our work is inspired by a compelling case study focused on customer profiling starting from the tracking of daily GPS trajectories. Each transaction describes a mall customer by means of several features (e.g., where she lives, where she works, how much she earns), and each feature is described at different, hierarchically-organized levels of details (e.g., she lives close to the Whitney Museum, which is located in the Greenwich Village district, which belongs to Manhattan). In this context, a FI describes the profile of a group of people sharing the same features/behavior. Multi-level and multi-dimensional FIs enable a rich representation of the behavior of customer groups; however, the huge number of mined FIs encourages the adoption of effective summarization and visualization techniques to provide decision makers with useful information for marketing and advertising.

As shown in Figure 2, SUSHI works independently of the algorithm applied for generating the FIs taken in input (e.g., Apriori [2], FP-Growth [23]). The summarization and the visualization components work jointly to show analysts the relevant information at multiple levels of detail; analysts iteratively create and visualize new summaries that better meet their needs by adjusting a set of parameters, and navigate them to get insights over summarized data.

In a preliminary paper [17] we have proposed an extension of the definition of containment to multi-level itemsets and a similarity function for FIs which takes into account both their extensional (support-based) and intensional (feature-based) natures. Relying on that background, in this work we offer the following

original contributions:

(i) a clustering algorithm that uses this similarity function to create a hierarchical summary to shrink a set of FIs down to a set of relevant patterns; to define the representative FI of each cluster, we discuss different strategies;

(ii) theoretical results concerning antimonotonicity of support and similarity in multi-level settings; these results allow the reduction of the search space for the clustering algorithm;

(iii) an evaluation of the search space and of the computational complexity of the clustering algorithm;

(iv) two integrated approaches to summary visualization and exploration: a graph-based one, which highlights inter-cluster relationships, and a tree-based one, which emphasizes the relationships between the representative of each cluster and the other FIs in that cluster;

(v) a comprehensive set of tests for evaluating the efficiency and effectiveness of our approach against others in the literature; we also made some tests with data science students to evaluate how well SUSHI enhances the analysts' capabilities in discovering relevant patterns.

The remainder of the paper is organized as follows. After discussing the related approaches to summarization and visualization in Section 2, in Section 3 we provide a formal background for multi-level and multi-dimensional itemsets. Then, in Section 4 we propose a definition of itemset similarity. Section 5 describes our approach to building summaries of FIs, while Section 6 explains how we visualize them. Finally, Section 7 presents the results of the experimental tests and Section 8 concludes the paper.

2. Related work

2.1. Mining and summarization of FIs

FI mining has been applied to multiple fields and types of items [21]. Items can be either binary (e.g., product purchases in market basket analysis) or categorical (e.g., IP addresses in a network table). Besides, they can be homogeneous (items from a single dimension at the same abstraction level, e.g., products) or multi-dimensional (items taken from multiple dimensions, e.g., products and customers), and they can be multi-level (items described at different levels of abstraction, e.g., products and product categories).

Since the application of the Apriori algorithm in market basket analysis [2], scholars defined many performant algorithms to extract FIs [30]; among them, FP-Growth [23] and very recently CBPM [14] and GMiner [10]. FI mining is currently witnessing a plethora of declinations [30], among them: high-utility itemsets (i.e., the extraction of FIs whose utility is higher than a given score [33]); colossal itemset mining (i.e., the extraction of FIs in datasets with few transactions containing even millions of items [40]); and frequent trajectory mining (i.e., the extraction of common routes in urban areas [46]). Given the possibility to mine FIs at different granularities [22], external knowledge can be used to generalize infrequent (specific) itemsets into frequent (generic) itemset [5]. For instance, two distinct products *milk* and *beer* can be grouped together into a generic FI only knowing that they both have type *beverage* [5]. While these algorithms address the mining of either flat or multi-dimensional and multi-level itemsets, they do not provide any form of summaries.

Since the number of FIs is exponential [2], summarization is necessary to support an effective data analysis [3]. Itemset summarization enables an effective explorations of large datasets by replacing groups of similar FIs with their representatives. Classical approaches to summarization reduce the exploration space by limiting the retrieved FIs to maximal-covered [20], closed-covered [34], and δ-covered [39] FIs. However, these approaches do not consider the exponential cardinality of the summary, its pertinence with respect to the exploration focus,

and the diversity of the FIs retrieved in the summary, which are indeed well-known driving criteria for data condensation [8]. To overcome these limitations, among the summarization techniques that have been well surveyed in [3], two approaches are well-known. In BUS [9], summarization exploits the definition of *itemset containment* (i.e., a FI can be summarized into another if the latter is a superset of the former) to uncover clusters of FIs by the greedy optimization of *compaction gain* and *information loss* metrics. BUS has been extended to MBUS [24] by the retrieval of *interesting* and *intelligible* itemset representatives. Further summarization techniques have been achieved by generalizing similar FIs belonging to sets [41, 1] or hyperrecatangles [42]; by computing an optimal number of probability vectors for cluster partitions [44, 35]; or by minimizing the restoration error [25]. In [49], the authors propose a summarization suitable only for high-utility itemsets; when itemsets are considered with equal utility, the summarization criterion boils down to the itemset containment used in [9, 24]. Overall, these contributions consider neither multi-dimensional nor multi-level FIs, nor they provide summary visualization and navigation. A schematic comparison between SUSHI and the summarization approaches for FIs is shown in Table 1.

As representative FIs are elected in place of a group of similar FIs, a key issue in summarization lies in the definition of itemset similarity. Similarity has been defined in terms of itemset intersection (e.g., [9, 24]) and support (i.e., the frequency of two itemsets within the dataset [43, 29]). However, these approaches *do not* exploit multi-level knowledge to aggregate FIs in intensionally-coherent groups.

From the computational point of view, summarization (seen as the problem of finding the minimal set approximating a given collection) is a NP-hard problem [1]. In SUSHI we adopt a greedy strategy based on a similarity function accounting for both the extensional and intensional natures of FIs. We also exploit the mathematical properties of our similarity function (specifically, its antimonotonicity) to sensibly prune the algorithm search space.

With respect to the above-mentioned contributions, in SUSHI we (i) intro-

Table 1: Approaches to FI summarization classified by similarity function, representative cluster element, clustering type (E=Exclusive, O=Overlapping, T=Total, P=Partial), multi-dimensional (MD) and multi-level (ML) data, and presence of summary visualization

Appr.	**Similarity function**	**Representative**	**Type**	**MD**	**ML**	**Vis**
[1]	Maximum coverage	FI	E, P	✗	✗	✗
[44]	Kullback-Leibler divergence	Probab. profile	E, T	✗	✗	✗
[41]	Maximum coverage	FI	O, T	✗	✗	✗
[9]	Compaction, inf. loss	FI	E, T	✗	✗	✗
[24]	Compaction, inf. loss, interestingness	FI	E, T	✗	✗	✗
[25]	Restoration error	FI	E, T	✗	✗	✗
[35]	Compressible profile	Probab. profile	E, T	✗	✗	✗
[42]	Coverage cost function	Hyperrectangle	E, T	✗	✗	✗
[29]	Support	FI	E, T	✗	✗	✗
SUSHI	Support, relevance	FI	E, T	✓	✓	✓

duce a novel similarity function based on the concepts of *relevance* and itemset containment, properly extended to deal with multi-dimensional and multi-level itemsets; (ii) introduce multiple strategies to shrink the exponential number of FIs to a meaningful *hierarchical* summary; (iii) provide a theoretical foundation to exploit the antimonotonicity of our similarity function to prune the algorithmic space; and (iv) propose an end-to-end framework supporting the navigation of the produced hierarchical summary. Finally, our contribution differs from approaches such as top-k (e.g., [50]) and minimum description length (i.e., compressing a dataset by exploiting regularities among data as in [31]), as our goal is not only to reduce and summarize the FIs but also to ensure an in-depth summary exploration. Furthermore, while a research branch close to FI mining is association rule mining, the summarization of association rules (e.g., [13]) relies on metrics (e.g., confidence, lift) that cannot be mapped to FIs.

2.2. Visual exploration

To be useful, a summary should give data analysts an overview of salient information and enable in-depth explorations of FI groups. This goal is not achievable by just displaying a plain list of FIs, nor by arranging all the available FIs in space. Visual techniques have high potential impact on dataset exploration [26], and should not require complex skills from decision makers.

Scholars have addressed the curse of FI and association rule cardinality by providing visualizations based on polylines [27], circular graph layout [7], treemaps [38], and parallel coordinates reflecting the item taxonomy [45]. The analyst is also included in the loop to visually mine FIs and association rules [28] by providing a visual representation of the algorithm search space and by enabling an interactive pruning. However, with increasing numbers of FIs, the proposed visualizations hardly scale up to human-understandable visualizations, and prevent analysts from perceiving itemset similarity at first sight.

In SUSHI, consistently with the graph visualization proposed in [16] and abiding by the "overview first, zoom and filter, then details-on-demand" mantra [37], we complement summarization with a hierarchical visualization enabling user-friendly and multi-resolution explorations of summaries.

3. Formal background

In this work we deal with multi-level itemsets, so we start by defining concept hierarchies as in classic multi-dimensional modeling [19].

Definition 1 (Hierarchy). *A* hierarchy H *is defined by (i) a set* L_H *of categorical* levels*, (ii) for each level* $l \in L_H$*, a* domain $Dom(l)$ *including a set of* values*, (iii) a* roll-up *partial order* $\succeq_H$ *of* L_H*, and (iv) a* part-of *partial order* $\geq_H$ *of* $\bigcup_{l \in L_H} Dom(l)$*. Exactly one level* $dim(H) \in L_H$*, called* dimension*, is such that* $dim(H) \succeq_H l$ *for each other* $l \in L_H$*. The part-of partial order is such that, for each couple of levels* l *and* l' *such that* $l \succeq_H l'$ *and for each value* $v \in Dom(l)$*, there is exactly one value* $v' \in l'$ *such that* $v \geq_H v'$*.*

Definition 1 allows information to be provided at a level coarser than the one of dimensions. In practice, hierarchies can be incomplete, i.e., some levels may be missing in the part-of partial order. In this case, the techniques proposed in [19] can be used to balance hierarchies by filling the missing values.

The itemsets we consider are also multi-dimensional, i.e., they describe events along different features (e.g., worksIn). Each feature corresponds to a

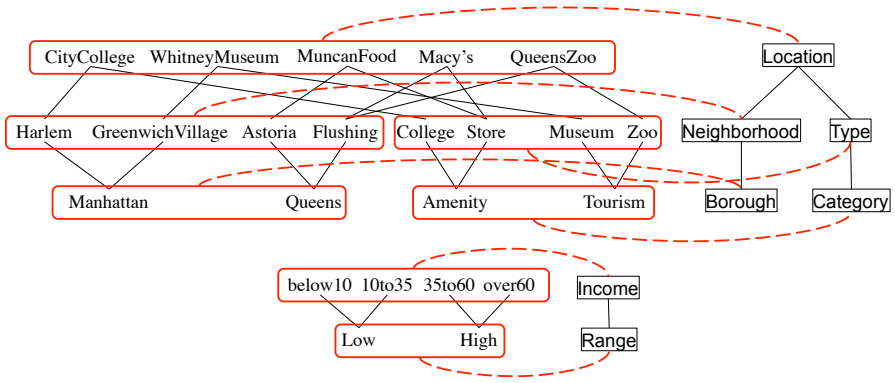


Figure 3: Hierarchies (right) and values (left) for the Profiling domain schema; in red, the level-domain mappings

hierarchy (e.g., Location) and defines the semantics carried by an item at one specific level of that hierarchy.

Definition 2 (Domain Schema). *A* domain schema *is a triple* $\mathcal{D} = (\mathcal{H}, \mathcal{E}, \mu)$ *where* $\mathcal{H}$ *is a set of hierarchies,* $\mathcal{E}$ *is a set of* features*, and* μ *is a function mapping each feature onto one hierarchy.*

Example 1. *Our working example relies on the* Profiling *domain schema, which describes the customers who regularly visit a mall and includes two hierarchies as shown in Figure 3. One hierarchy is rooted in the* Location *dimension and has two branches that describe locations from the geographical point of view and based on their characteristics, respectively. The roll-up partial order states, for instance, that* Neighborhood $\succeq_{\text{Location}}$ Borough*; the part-of partial order states that* Harlem $\geq_{\text{Location}}$ Manhattan*. The second hierarchy is rooted in the* Income *dimension and classifies incomes based on their ranges. The features of* Profiling *are* worksIn*,* frequents*, and* earns*; specifically* $\mu(\text{worksIn}) = \mu(\text{frequents}) = \text{Location}$*,* $\mu(\text{earns}) = \text{Income}$*.*

We are now ready to define an item as a couple of a feature and a value taken from the domain of one of the levels of the corresponding hierarchy. Itemsets are non-redundant sets of items, i.e., two items in an itemset cannot be defined on values related in the part-of partial order (e.g., GreenwichVillage and

Manhattan). Finally, transactions are itemsets whose items are all defined on dimension values, i.e., at the top level of a hierarchy (e.g., Macy's).

Definition 3 (Itemset and Transaction). *Given domain schema* $\mathcal{D} = (\mathcal{H}, \mathcal{E}, \mu)$, *an* item *of* $\mathcal{D}$ *is a couple* $i = (f, v)$ *where* $f \in \mathcal{E}$, $v \in Dom(l)$, *and* l *is a level of hierarchy* $\mu(f)$. *An* itemset I *of* $\mathcal{D}$ *is a set of distinct items of* $\mathcal{D}$ *where, for each* $i, i' \in I$ *such that* $i = (f, v)$ *and* $i' = (f, v')$, *it is* $v \not\geq_{\mu(f)} v'$ *and* $v' \not\geq_{\mu(f)} v$. *We call a* transaction *an itemset only including items defined over dimensions of* $\mathcal{H}$.

Example 2. *With respect to the* Profiling *multi-dimensional and multi-level domain, examples of an itemset* I *and a transaction* T *are:*

$$I = \{(\mathsf{worksIn}, \text{Harlem}), (\mathsf{frequents}, \text{Museum}), (\mathsf{frequents}, \text{Macy's}), (\mathsf{earns}, \text{High})\}$$
$$T = \{(\mathsf{worksIn}, \text{CityCollege}), (\mathsf{frequents}, \text{WhitneyMuseum}),$$
$$(\mathsf{frequents}, \text{Macy's}), (\mathsf{earns}, 35\text{to}60)\}$$

Note that, for semantic reasons, some features are actually disjunctive, meaning that they cannot realistically take two or more values at the same time; for instance, this is the case for earns. Others are not; for instance, a person can frequent several places. In practice it is not necessary to model this property of features, since real (correct) transactions will always have at most one single value for a disjunctive feature, so the same will hold for all FIs as well.

4. Working with itemsets

Working with multi-dimensional and multi-level items requires classic definitions related to FI mining to be extended. We start by defining a notion of *containment* between itemsets that generalizes set containment taking hierarchies into account; based on itemset containment we can then define the *support* of itemsets, which in turn is necessary to select FIs out of the set of all itemsets.

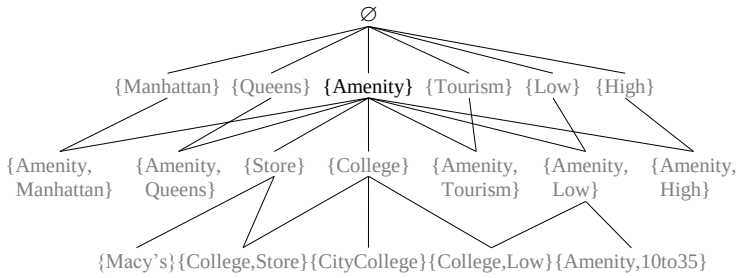


Figure 4: A portion of the containment lattice for the Profiling; itemsets in gray are partially expanded

Definition 4 ((Multi-Level) Itemset Containment). *Given two itemsets I and I', we say that I is* contained *in I' (denoted $I \sqsubseteq I'$) if for each item $i \in I$, $i = (f, v)$, there is an item $i' \in I'$, $i' = (f, v')$ such that $v' \geq_{\mu(f)} v$. Given a set $\mathcal{F}$ of itemsets, an itemset $I \in \mathcal{F}$ is* directly contained in $I' \in \mathcal{F}$ with reference to $\mathcal{F}$ *(denoted $I \dot{\sqsubseteq}_{\mathcal{F}} I'$) if there is no other itemset $I'' \in \mathcal{F}$ such that $I \sqsubseteq I'' \sqsubseteq I'$.*

Let $\mathcal{I}$ denote the set of all items of a schema domain and $2^{\mathcal{I}}$ denote the set of all itemsets as in Definition 3. The containment relationship is reflexive, antisymmetric, and transitive, and for each pair of itemsets in $\mathcal{I}$ there are a least upper bound and a greatest lower bound; so relationship $\sqsubseteq$ induces a lattice on $2^{\mathcal{I}}$, whose top and bottom elements are the empty itemset and $\mathcal{I}$, respectively. Given two itemsets I and I', we denote with $lub(I, I')$ and $glb(I, I')$ their least upper bound (i.e., the least element in $2^{\mathcal{I}}$ that is greater than or equal to both I and I') and greatest lower bound (i.e., the greatest element in $2^{\mathcal{I}}$ that is less than or equal to both I and I') in the lattice.

Example 3. *Figure 4 shows a portion of the containment lattice for the* Profiling *domain schema; for simplicity we focus on features* frequents *and* earns *and denote items by their value only. For instance, for* frequents *it is* $\{\text{Amenity}\} \sqsubseteq \{\text{College}, \text{Store}\}$ *and* $\{\text{Amenity}\} \dot{\sqsubseteq}_{2^{\mathcal{I}}} \{\text{College}\}$. *Intuitively,* $\{\text{Store}\}$ *and* $\{\text{College}\}$ *arise by describing* $\{\text{Amenity}\}$ *at a higher level of detail;* $\{\text{Amenity}, \text{Manhattan}\}$ *arises from a branch of the* Location *hierarchy; and* $\{\text{Amenity}, \text{Low}\}$ *arises by*

extending {Amenity} *with the additional item* {Low}.

We say that transaction T *supports* itemset I if $I \sqsubseteq T$. For instance, in Example 2, T supports I. Given a set of transactions $\mathcal{T}$, the set of transactions that support I is denoted by $\mathcal{T}_I \subseteq \mathcal{T}$. At this stage we can introduce a numerical property of itemsets, their *support*.

Definition 5 (Itemset Support). *Given itemset I, its support $sup(I)$ within a set of transactions $\mathcal{T}$ is defined as $sup(I) = |\mathcal{T}_I|/|\mathcal{T}|$.*

An itemset I is called *frequent* if its support is greater or equal to a given threshold. Since the containment relationship induces a lattice on the set $2^{\mathcal{I}}$ of all possible itemsets, it also induces a partial order over the set $\mathcal{F} \subseteq 2^{\mathcal{I}}$ of FIs. Thus, from now on we will say that $\mathcal{F}$ is a partially ordered set (POS).

A valuable property of the support for non-hierarchical itemsets is antimonotonicity along the containment relationship: for each I, I' such that $I \subseteq I'$ it clearly is $sup(I) \geq sup(I')$ [2]. Remarkably, this property also holds for our hierarchical itemsets along the multidimensional containment relationship introduced in Definition 4, as proved by the following theorem.

Theorem 1 ((Multi-Level) Antimonotonicity of Support). *Given two itemsets I and I' such that $I \sqsubseteq I'$, for all sets of transactions it is $\mathcal{T}_{I'} \subseteq \mathcal{T}_I$ and $sup(I) \geq sup(I')$.*

Proof: To prove that $\mathcal{T}_{I'} \subseteq \mathcal{T}_I$ it is sufficient to prove that for each transaction T that supports I', T also supports I (for Definition 5). For Definition 3, if T supports I' then for each item $i' \in I'$, $i' = (f, v')$, there is at least one item $i_0 \in T$, $i_0 = (f, v_0)$, such that $v_0 \geq_{\mu(f)} v'$. But since $I \sqsubseteq I'$, we know by Definition 4 that for each item $i \in I$, $i = (f, v)$, there is an item $i' \in I'$, $i' = (f, v')$ such that $v' \geq_{\mu(f)} v$. The part-of partial order $\geq_{\mu(f)}$ is transitive, so $v_0 \geq_{\mu(f)} v$; then we obtain that for each $i \in I$ there is at least one item $i_0 \in T$ such that $v_0 \geq_{\mu(f)} v'$, in other words that T supports I. From here, it immediately follows that $sup(I) \geq sup(I')$. □

Note that, due to this result, the transactions supporting I' also support I.

In Section 5 we will compute summaries using clustering, which clearly requires the definition of a similarity function to measure how "close" two FIs are. To this end, we start by introducing *feature-based* similarity, which increases with the number of features (i.e., semantics) shared by two FIs. Clearly, if two FIs include two distinct groups of transactions, feature-based similarity is not meaningful, so it is associated with a *support-based* similarity which increases with the percentage of transactions supporting both FIs. There is no obvious correlation between these two aspects of similarity; for instance, support-based similarity can be low even if feature-based similarity is high when non-shared features are rare and supported by a small fraction of transactions.

In a multi-level and multi-dimensional domain, defining feature-based similarity only by counting the common items between two FIs would be reductive; in fact, the informative value carried by these items in terms of level of detail should be considered as well. For instance, knowing that a person frequents Macy's is more relevant than knowing that she frequents a generic amenity. Intuitively, an FI is more *relevant* than another if it includes a larger number of distinct features; in turn, the relevance of a feature increases with the level of detail at which it is expressed.

Definition 6 (Itemset Relevance). *Given itemset I, its* relevance *is*

$$rel(I) = \sum_{f \in Feat(I)} \left(rel(f) + \sum_{l \in Lev_f(I)} rel(l) \right)$$

where $Feat(I)$ is the set of distinct features of the items in I, $Lev_f(I)$ is the set of levels of the values coupled with feature f in the items of I, $rel(f)$ is the relevance of f, and $rel(l)$ is the relevance of level l.

Assuming that higher levels of detail carry more informative content and relevance, means assuming that $rel(l) \geq rel(l')$ if $l \succeq_{\mu(f)} l'$. As a consequence, given any two itemsets such that $I \sqsubseteq I'$, we always have $rel(I') \geq rel(I)$.

We can now define the similarity between two itemsets as a linear combination of a support-based and a feature-based similarity.

Definition 7 (Itemset Similarity). *Given a set of transactions $\mathcal{T}$, a POS of FIs $\mathcal{F}$, two FIs I and I' supported by $\mathcal{T}$, and a coefficient $\lambda \in [0..1]$, the* similarity *of I and I' is defined as $sim(I, I') = \lambda sim_{sup}(I, I') + (1 - \lambda) sim_{fea}(I, I')$, where*

$$sim_{sup}(I, I') = \begin{cases} \frac{sup(glb(I,I'))}{sup(I)+sup(I')-sup(glb(I,I'))}, \text{ if } glb(I, I') \in \mathcal{F} \\ 0, \text{ otherwise} \end{cases} \quad (1)$$

$$sim_{fea}(I, I') = \begin{cases} \frac{rel(lub(I,I'))}{rel(glb(I,I'))}, \text{ if } lub(I, I'), glb(I, I') \in \mathcal{F} \\ 0, \text{ otherwise} \end{cases} \quad (2)$$

Both sim_{sup} and sim_{fea} range in [0..1]; they can be explained as follows:

- sim_{sup} is the ratio between the number of transactions that support both I and I' and the number of transactions that support either I or I'. The higher the portion of transactions supporting both I and I', the higher sim_{sup} as requested by the the support-based principle.

- sim_{fea} is the ratio between the relevances of the $lub()$ and $glb()$ of the two FIs. When the features belonging to the two FIs I, I' are dissimilar, (i) the relevance of the $lub(I, I')$ will be low since most of the features in I, I' will be dropped or their level will be less detailed; (ii) the relevance of the $glb(I, I')$ will be high since most of the features in I' must be added to those in I. This behavior satisfies the feature-based principle.

Clearly, since the *lub* and *glb* operators are commutative, it is always $sim(I, I') = sim(I', I)$.

Example 4. *With reference to the hierarchies of Figure 3, let the POS of FIs $\mathcal{F}$ be the one shown in Figure 5, restricted to features* frequents *and* earns. *We assume that (i) for each feature f it is $rel(f) = 1$, and (ii) relevance increases*

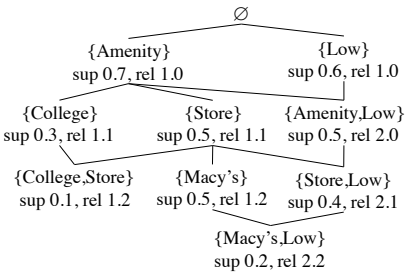


Figure 5: The POS $\mathcal{F}$ of FIs for Examples 4, 5, and 7 (each FI is annotated with its support and relevance)

by 0.1 for each level of detail. Given FIs $I = \{(\mathsf{frequents}, \text{Store})\}$ *and* $I' = \{(\mathsf{frequents}, \text{Amenity}), (\mathsf{earns}, \text{Low})\}$*, it is* $lub(I, I') = \{(\mathsf{frequents}, \text{Amenity})\}$*,* $glb(I, I') = \{(\mathsf{frequents}, \text{Store}), (\mathsf{earns}, \text{Low})\}$*, and* $sim(I, I') = 0.57$*.*

To decrease the complexity of summary creation, in Section 5 we will leverage on the properties of itemset similarity and containment. First of all, Theorem 2 proves that itemset similarity is antimonotonic.

Theorem 2 (Antimonotonicity of Itemset Similarity). *Given three FIs* I*,* I'*, and* I'' *such that* $I \sqsubseteq I' \sqsubseteq I''$*, for all sets of transactions and all* $\lambda \in [0..1]$ *it is* $sim(I, I') \geq sim(I, I'')$*.*

Proof: Support-based similarity is antimonotonic along the containment relationship. This can be easily proved by considering that, in Equation 1, since $I \sqsubseteq I' \sqsubseteq I''$ it is $glb(I, I') = I'$ and $glb(I, I'') = I''$, and by recalling that function *sup* is antimonotonic. Similarly, feature-based similarity is antimonotonic along the containment relationship. This can be easily proved by considering that, in Equation 2, since $I \sqsubseteq I' \sqsubseteq I''$ it is $lub(I, I') = lub(I, I'') = I$, and by recalling that $rel(I') \leq rel(I'')$ by assumption. But then, *sim* is antimonotonic for any λ in that it is a linear combination of antimonotonic functions. □

The next property, formalized by Theorem 3, states that direct itemset containment (see Definition 4) always entails higher similarity, i.e., in case an itemset I' is directly contained in an itemset I''', the similarity between I' and I''' is

greater than the similarity of I' to any other itemset I'' that does not directly contain I'. To prove this result we rely on the following lemma.

Lemma 1. *Given a POS of FIs $\mathcal{F}$ and three FIs $I', I'', I''' \in \mathcal{F}$ such that $I''' = glb(I', I'')$, $I' \not\sqsubseteq I''$, $I'' \not\sqsubseteq I'$, for all sets of transactions and all $\lambda \in [0..1]$ it is $sim(I', I''') \geq sim(I', I'')$ and $sim(I'', I''') \geq sim(I', I'')$.*

Proof: We prove the thesis by contradiction, assuming that $sim(I', I'') > sim(I', I''')$. We consider sim_{sup} first:

$$\frac{sup(glb(I', I''))}{sup(I') + sup(I'') - sup(glb(I', I''))} > \frac{sup(glb(I', I'''))}{sup(I') + sup(I''') - sup(glb(I', I'''))}$$

From here, recalling that $glb(I', I''') = I'''$, we get

$$\frac{sup(I''')}{sup(I') + sup(I'') - sup(I''')} > \frac{sup(I''')}{sup(I')}$$

Since $sup(I'') \geq sup(I''')$ because of Theorem 1, this inequality admits no solution. This also holds for the relevance component of the similarity function, sim_{fea}. Let $I = lub(I', I'')$:

$$\frac{rel(lub(I', I''))}{rel(glb(I', I''))} > \frac{rel(lub(I', I'''))}{rel(glb(I', I'''))} \Leftrightarrow \frac{rel(I)}{rel(I''')} > \frac{rel(I')}{rel(I''')}$$

But since $I \sqsubseteq I'$, for Definition 6 we have $rel(I') \geq rel(I)$, then the inequality admits no solution and we have a contradiction. From the two contradictions it follows that $sim(I', I''') \geq sim(I', I'')$ and $sim(I'', I''') \geq sim(I', I'')$. □

Theorem 3. *Given a POS of FIs $\mathcal{F}$ and any two FIs $I', I'' \in \mathcal{F}$, for each $I''' \in \mathcal{F}$ such that $I' \dot{\sqsubseteq}_{\mathcal{F}} I'''$, it is $sim(I', I'') \leq sim(I', I''')$.*

Proof: The proof comes by applying Theorem 2 to the result of Lemma 1. □

This theorem allows to sensibly reduce the summarization space in Section 5, narrowing the number of cluster comparisons down to the pairs of clusters whose representatives are directly contained into one another (i.e., it prevents an all-against-all comparison).

5. Summarizing frequent itemsets

Since SUSHI supports interactive exploration and navigation, it gives summaries a hierarchical organization to let them be analyzed at different levels of detail. To create summaries we adopt an agglomerative hierarchical clustering technique, which progressively merges couples of clusters starting from singletons until one single cluster is obtained. The result of hierarchical clustering is commonly represented using a dendrogram, i.e., a binary tree structure containing a k-block set partition for each $1 \leq k \leq n$, where n is the number of objects to be clustered [11]. In our context, due to the constraints we pose on mergeability, in some cases it is impossible to merge all the FIs into one single cluster, so the dendrogram will actually not be a tree but a forest of trees.

5.1. Summaries and representatives

Definition 8 (H-Summary and Summary). *Given a POS of FIs $\mathcal{F}$, a* hierarchical summary *(briefly,* h-summary*) of $\mathcal{F}$ is a (directed) forest HS where each node is a cluster $c \subseteq \mathcal{F}$ and each leaf is a singleton cluster. Each node c is labeled with the similarity $sim(c)$ of the representatives of the two clusters that were merged to obtain c (conventionally, $sim(c) = 1$ for singleton clusters). A* summary *S of HS is any set of clusters in HS that define a (complete and disjoint) partition of $\mathcal{F}$.*

Among all possible summaries for HS, the one including all and only the roots of HS is denoted as $min(HS)$ and called *minimum summary* (see Figure 6).

Note that, while traditional dendrograms are typically cut at a given similarity level to produce a clustering, in our definition a summary can include clusters at different similarity levels. This is to allow analysts to interactively choose which cluster they want to analyze in more detail during visualization (see Section 6).

In a summary, each cluster is represented by a single FI. To define the representative $rep(c)$ of cluster c, we provide three different strategies:

- *Top*: the representative is the most general FI in c, $rep(c) \sqsubseteq I, \forall I \in c$

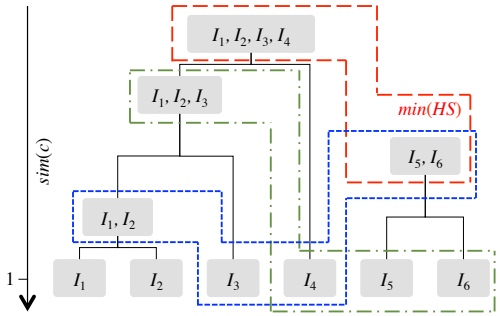


Figure 6: A simple h-summary with three possible summaries, including the minimum one

- *Bottom*: the representative is the most specific FI in c, $I \sqsubseteq rep(c), \forall I \in c$
- *Medoid*: the representative is the cluster "center", $rep(c) = med(c)$

The Top strategy is the one most commonly used in the literature [9], while Bottom is related to the notion of *maximal FI* (i.e., a FI for which none of its immediate supersets is frequent [20]). As confirmed by both our tests and assessment with real users (see Section 7), Top often lacks in properly characterizing the clusters. The reason for this is that, as the cluster cohesion decreases, one or more features appearing in some of the cluster FIs may disappear from the representative, which leads to very low relevance. Conversely, when the Bottom representative is adopted, all the features appearing in at least one FI of the cluster are included. Obviously, both Top and Bottom are *extreme* representatives of clusters, thus they often do not show high similarity with the other FIs in the the cluster. To overcome this problem, we alternatively propose the Medoid strategy; to the best of our knowledge, no previous approach relies on medoid representatives for FIs. A medoid is defined as follows:

Definition 9 (Medoid). *Given a summary HS and a cluster $c \in HS$, a FI $\overline{I} \in c$ is a* candidate medoid *of c if, for each other FI $I^* \in c$, it is $sim(\overline{I}, I^*) > 0$ and $\sum_{I \in c} sim(\overline{I}, I) \geq \sum_{I \in c} sim(I^*, I)$. If c has at least one candidate medoid, then the* medoid *of c, denoted $med(c)$, is the one with the highest relevance and, at the same relevance, with the highest support.*

Given two clusters c', c'', computing the medoid of $c' \cup c''$ requires all the

pairwise similarities between the FIs to be estimated. This entails the computation of a quadratic number of similarities in the cardinality of $c' \cup c''$ (each FI against all the others), which can be unfeasible for large datasets:

$$med(c) = argmax_{\bar{I} \in c' \cup c''} \sum_{I \in c' \cup c''} sim(\bar{I}, I)$$

To cut this complexity down to linear, while merging two clusters we can approximate the optimal medoid based on the cluster representatives:

$$\begin{aligned} med^*(c', c'') &= argmax_{\bar{I} \in c' \cup c''} sim(\bar{I}, rep(c')) \cdot |c'| + sim(\bar{I}, rep(c'')) \cdot |c''| \\ &\simeq argmax_{\bar{I} \in c' \cup c''} \sum_{I \in c'} sim(\bar{I}, I) + \sum_{I \in c''} sim(\bar{I}, I) = med(c' \cup c'') \end{aligned}$$

Thanks to the last step, for each candidate medoid $\bar{I}$ we can replace the sum of the similarities between $\bar{I}$ and all other FIs with the similarity between $\bar{I}$ and the two representative medoids.

Depending on the strategy adopted for picking cluster representatives, different mergeability constraints arise. Indeed, two clusters can actually be merged only if the resulting cluster is well-formed, i.e., if it has a representative itself. Note that, for a singleton cluster $c = \{I\}$, it is $rep(c) = I$ for all strategies.

Definition 10 (Mergeability). *Two clusters c and c' are mergeable (denoted $c \leftrightarrow c'$) if: $rep(c) \sqsubseteq rep(c')$ or $rep(c') \sqsubseteq rep(c)$, in case of Top or Bottom strategies; the union of c and c' has a medoid, in case of Medoid strategy.*

Example 5. *With reference to the POS of FIs $\mathcal{F}$ in Figure 5, let the following clusters be given: $c_1 = \{\{\text{College}\}, \{\text{College}, \text{Store}\}\}$, $c_2 = \{\{\text{Amenity}\}\}$, $c_3 = \{\{\text{Low}\}\}$. According to the Top strategy, it is $rep(c_1) = \{\text{College}\}$; c_1 can be merged with c_2 but not with c_3 since $\{\text{College}\} \not\sqsubseteq \{\text{Low}\}$ and $\{\text{Low}\} \not\sqsubseteq \{\text{College}\}$. According to the Bottom strategy, it is $rep(c_1) = \{\text{College}, \text{Store}\}$; c_1 can be merged with c_2 but not with c_3 since $\{\text{College}, \text{Store}\} \not\sqsubseteq \{\text{Low}\}$ and $\{\text{Low}\} \not\sqsubseteq \{\text{College}, \text{Store}\}$. According to the Medoid strategy, it is $rep(c_1) = \{\text{College}, \text{Store}\}$; c_1 can be merged with c_2 but not with c_3 since $glb(\{\text{College}, \text{Store}\},$*

$\{\text{Low}\}) \notin \mathcal{F}$, *hence,* $sim(\{\text{College}, \text{Store}\}, \{\text{Low}\}) = 0$ *and* $c_1 \cup c_3$ *has no candidate medoids.*

5.2. Search space for multi-dimensional and multi-level FIs

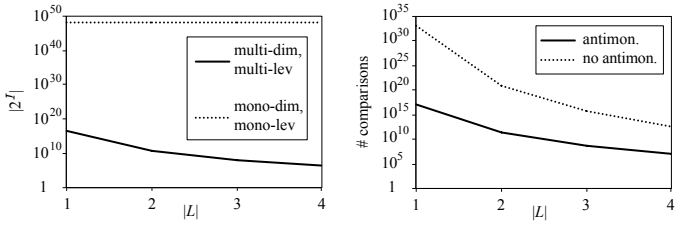
Hierarchical clustering on a set $\mathcal{F}$ of FIs entails $|\mathcal{F}|(|\mathcal{F}| - 1)/2$ comparisons between FIs; so, reasoning about the cardinality of $\mathcal{F}$ is crucial. The actual cardinality of $\mathcal{F}$ clearly depends on the number of transactions in the dataset and on the frequency threshold. To discuss the impact of the multi-dimensional and multi-level setting on $|\mathcal{F}|$ in general terms, we consider the worst possible case, in which $\mathcal{F} = 2^{\mathcal{I}}$ because there is a transaction for each possible itemset and the threshold is 1 (all itemsets are frequent).

When working with mono-dimensional and mono-level items, the total number of possible (non-empty) itemsets is $|2^{\mathcal{I}}| = 2^n - 1$, where $n = |\mathcal{I}|$ is the total number of items. In the multi-dimensional case these n items are not homogeneous, i.e., they are picked from the domains of multiple dimensions. If ν is the cardinality of these domains, which we assume for simplicity to be constant, the number of possible itemsets cuts down to

$$|2^{\mathcal{I}}| = \sum_{k=1}^{n/\nu} \binom{n/\nu}{k} \cdot \nu^k = (\nu + 1)^{n/\nu} - 1 \tag{3}$$

where n/ν is the number of dimensions (assuming for simplicity that all features are disjunctive). Finally, in the multi-level case, some meta-knowledge of part-of relationships between items is available and expressed in the form of hierarchies as in Definition 2, so some more combinations of items become unfeasible: indeed, we recall from Definition 3 that there cannot be any part-of relationship between the values of the items included in an itemset (e.g., $\{(\mathsf{worksIn}, \text{Harlem}), (\mathsf{worksIn}, \text{Manhattan})\}$ is not an itemset). In this case, the number of possible itemsets is further reduced to

$$|2^{\mathcal{I}}| = \sum_{k=1}^{n/(\nu|L|)} \binom{n/(\nu|L|)}{k} \cdot |L|^k \cdot \nu^k = (\nu|L| + 1)^{n/(\nu|L|)} - 1 \tag{4}$$



(a) (b)

Figure 7: (a) Search space measured as the number of itemsets in function of the number of levels; (b) number of comparisons at the first iteration in function of the number of levels

where $|L|$ is the number of levels in each hierarchy (assumed for simplicity to be the same for all hierarchies) and $n/(\nu|L|) = |\mathcal{H}|$ is the number of hierarchies. Clearly, if $|L| = 1$ (flat hierarchies), Formula 4 boils down to Formula 3.

Figure 7(a) shows how the number of itemsets computed with Formula 4 changes when the number of levels $|L|$ changes from 1 (mono-level items) to 4 (multi-level items), with $n = 160$ and $\nu = 10$. As a reference, also the number of itemsets in the case of mono-dimensional and mono-level items is plotted. Clearly, when dimensions are structured in hierarchies the number of FIs is significantly smaller than in the "flat" case.

Even if $|\mathcal{F}|$ is significantly smaller for multi-level items than for mono-level ones, executing all $|\mathcal{F}|(|\mathcal{F}|-1)/2$ comparison would be expensive. Fortunately, this is not the case in our approach due to Theorem 3 that allows to compare only the couples of clusters whose representatives are directly contained into one another with reference to the set of all clusters representatives. The most expensive iteration for agglomerative hierarchical clustering algorithm is the first one, when all clusters are singleton so there are $|\mathcal{F}|$ clusters. Assuming for simplicity that all hierarchies are linear (i.e., no diamond-like hierarchies are present), an itemset including k items has exactly k fathers in the containment lattice. The total number of comparisons required at this stage is:

$$\sharp \text{ comparisons} = \sum_{k=1}^{n/(\nu|L|)} \binom{n/(\nu|L|)}{k} \cdot |L|^k \cdot \nu^k \cdot k = n(\nu|L|+1)^{n/(\nu|L|)-1}$$

Figure 7(b) plots this formula for increasing numbers of hierarchy levels, distinguishing whether the antimonotonicity of the similarity function is considered or not. Besides the reduction of the number of comparisons due to antimonotonicity exploitation, the plot, as in Figure 7(a), also emphasizes how multi-level containment further reduces the number of comparisons. In the subsequent iterations of the clustering algorithm, the number of comparisons varies depending on the actual shape of the hierarchies and on the strategy chosen for cluster representatives, but it is always smaller than the first one since (i) the number of clusters decreases at each iteration, and (ii) not all pairs of clusters can actually be merged due to the mergeability constraints.

Example 6. *Given* $n = 120$*,* $l = 3$*,* $v = 10$ *and a null support threshold (i.e., all itemsets are frequent) it is* $|2^{\mathcal{I}}| = |\mathcal{F}| = 923520$*. The number of comparisons is* $36 \cdot 10^5$*, that is less than the number of required comparisons required by mono-dimensional and mono-level itemset clustering,* $|\mathcal{F}|(|\mathcal{F}|-1)/2 = 43 \cdot 10^{10}$*.* □

5.3. Building h-summaries

In this section we explain how h-summaries can be efficiently computed by agglomerative clustering. Considering the drastic reduction in the number of useful comparisons enabled by the antimonotonicity of the similarity function and by mergeability constraints, the efficiency of the agglomerative clustering algorithm can be greatly improved by storing the set of *useful comparisons* at each iteration within a graph $UC = (C, M)$, where C is the current set of clusters and M stores an arc for each couple of clusters whose comparison is actually useful. Keeping the arcs in M sorted by decreasing similarity values ensures that the most similar couple of clusters can be found in constant time.

The pseudocode for summarizing FIs is shown in Algorithm 1. HS stores the h-summary being built by agglomerative hierarchical clustering. Graph $UC = (C, M)$ keeps track of cluster mergeability: at each iteration C is the set of clusters in the minimum summary $min(HS)$ (clearly, $min(HS)$ changes

at each iteration as clusters are progressively merged and HS is expanded), while M stores an arc for each couple of candidate mergeable clusters, i.e., clusters with direct containment (Line 2). We recall that safely restricting to consider direct containment is possible thanks to Theorem 3; indeed, at each step, Algorithm 1 merges the two most similar clusters, which are necessarily two clusters related by direct containment. We use the term *candidate* since, while with the Top and Bottom strategies mergeability is ensured by the building rules of UC, the Medoid strategy requires a further check since the changes occurred in the cluster composition may affect mergeability.

The algorithm starts by initializing HS with singleton clusters, one for each FI (Line 3); this can be seen as a degenerate h-summary since no merge took place so far. At Line 4 we initialize $UC = (C, M)$: C is the set of singleton clusters, while M stores the couples of singletons whose FIs are directly contained into one another with reference to $\mathcal{F}$.

Then, while the cardinality of $min(HS)$ is above the target cardinality k and some pairs of mergeable clusters exist (Line 5), we pick the pair of mergeable clusters $\overline{c}'$ and $\overline{c}''$ whose representatives have maximum similarity (Line 6); since the arcs in M are sorted by descending similarity, this simply means getting the first arc. These two clusters are then merged into cluster $\overline{c}$ (Line 8) and the h-summary HS is updated by adding the $\overline{c}$ as the father of $\overline{c}'$ and $\overline{c}''$ (Line 9). Finally, at Line 10, UC is updated as described by Algorithm 2.

Algorithm 2 updates UC according to the representative strategy adopted. Figure 8 shows which arcs are kept and which ones are dropped when $\overline{c}'$ and $\overline{c}''$ are merged. Note that arc $(\hat{c}, \overline{c}')$ cannot belong to M since otherwise it would be $rep(\hat{c}) \sqsubseteq rep(\overline{c}') \sqsubseteq rep(\overline{c}'')$, but then $(\hat{c}, \overline{c}'')$ would not be a direct containment (with reference to the set of all cluster representatives), which is impossible by construction (see Line 12). Symmetrically, the same holds for $(\check{c}, \overline{c}'')$. Arc $(\overline{c}', \overline{c}'')$ is dropped in all the strategies; moreover: in Top, arc $(\hat{c}, \overline{c}'')$ is dropped since with this strategy the representative of $\overline{c}$ is the representative of $\overline{c}'$ (i.e., the most general one), thus $\hat{c}$ and $\overline{c}$ are not mergeable since $rep(\overline{c}) = rep(\overline{c}') \not\sqsubseteq rep(\hat{c})$; Bottom is the dual case of *Top*; in Medoid, containment of representatives is

Algorithm 1 Create H-Summary

Input $\mathcal{F}$: POS of FIs, k: number of clusters, *strat* representative strategy
Output HS: h-summary

$C = \{\{I\}$ s.t. $I \in \mathcal{F}\}$ ▷ Set of singleton clusters, one for each FI
$M \leftarrow \{(c', c'')$ s.t. $c' \in C \wedge c'' \in C \wedge rep(c') \dot{\sqsubseteq}_{\mathcal{F}} rep(c'')\}$ ▷ Couples of clusters with direct containment
$HS \leftarrow C$ ▷ HS is initialized
$UC \leftarrow (C, M)$ ▷ UC is initialized
while $(M \neq \varnothing) \wedge (|min(HS)| > k)$ **do** ▷ While the h-summary is expandable...
  $(\overline{c}', \overline{c}'') \leftarrow argmax_{\{(c', c'') \in M\}} sim(rep(c'), rep(c''))$ ▷ ...find the pair of candidate mergeable clusters with most similar representatives;
  if $\overline{c}' \leftrightarrow \overline{c}''$ **then** ▷ if they are actually mergeable according to *strat*...
    $\overline{c} \leftarrow \overline{c}' \cup \overline{c}''$ ▷ ...merge them,
    $AddFather(HS, \overline{c}, \overline{c}', \overline{c}'')$ ▷ update HS,
    $Update(UC, \overline{c}, \overline{c}', \overline{c}'', strat)$ ▷ and update UC
return HS

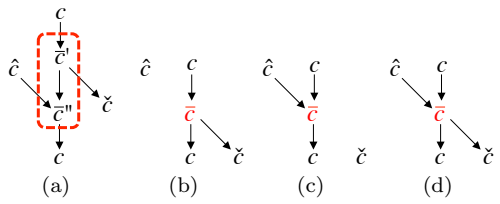


Figure 8: Changes in UC when clusters $\overline{c}'$ and $\overline{c}''$ are merged: before merging (a), after Top merging (b), after Bottom merging (c), and after Medoid merging (d)

not necessary, so all clusters remain potentially mergeable and no more arcs are removed.

Example 7. *Here we show some steps of Algorithm 1 using the Bottom strategy, with reference to the POS of FIs shown in Figure 5. The h-summary HS is initialized with 9 singleton clusters (gray boxes in Figure 9(a)). Among the 11 pairs of mergeable clusters whose FIs are directly contained one into the other with reference to* $\mathcal{F}$ *(the pairs connected by arrows in Figure 9(a)), the pair* $(\{\text{Store}\}, \{\text{Macy's}\})$ *is the one with highest similarity (0.96). So these two clusters are merged; the representative of the new cluster is FI* $\{\text{Macy's}\}$ *as shown in Figure 9(b). At the second iteration, among the 8 pairs of mergeable clusters, the one with highest similarity is* $(\{\text{Amenity}, \text{Low}\}, \{\text{Store}, \text{Low}\})$ *(Figure 9(b)); the representative of the merged cluster is* $\{\text{Store}, \text{Low}\}$*. After 7 iterations, two clusters are obtained (shown in Figure 10(b)); since these two clusters are not mergeable, the algorithm stops and returns an h-summary including these two*

Algorithm 2 Update

Input $UC = (C, M)$: graph of useful comparisons, $\overline{c}', \overline{c}''$: clusters to be merged (with $(\overline{c}', \overline{c}'') \in M$), $\overline{c}$: cluster resulting from the merge, *strat*: representative strategy
Output UC: updated graph of useful comparisons
1: $C \leftarrow C \cup \overline{c}$
2: **switch** *strat* **do**
3: **case** *Top*
4: $M \leftarrow M \cup \{(\overline{c}, c)$ s.t. $(\overline{c}'', c) \in M \vee (\overline{c}', c) \in M\}$ ▷ $\overline{c}$ inherits outgoing arcs of $\overline{c}''$ and $\overline{c}'$
5: $M \leftarrow M \cup \{(c, \overline{c})$ s.t. $(c, \overline{c}') \in M\}$ ▷ $\overline{c}$ inherits the incoming arcs of $\overline{c}'$
6: **case** *Bottom*
7: $M \leftarrow M \cup \{(c, \overline{c})$ s.t. $(c, \overline{c}') \in M \vee (c, \overline{c}'') \in M\}$ ▷ $\overline{c}$ inherits incoming arcs of $\overline{c}'$ and $\overline{c}''$
8: $M \leftarrow M \cup \{(\overline{c}, c)$ s.t. $(\overline{c}'', c) \in M\}$ ▷ $\overline{c}$ inherits the outgoing arcs of $\overline{c}''$
9: **case** *Medoid*
10: $M \leftarrow M \cup \{(c, \overline{c})$ s.t. $(c, \overline{c}') \in M \vee (c, \overline{c}'') \in M\}$ ▷ $\overline{c}$ inherits incoming arcs of $\overline{c}'$ and $\overline{c}''$
11: $M \leftarrow M \cup \{(\overline{c}, c)$ s.t. $(\overline{c}', c) \in M \vee (\overline{c}'', c) \in M\}$ ▷ $\overline{c}$ inherits outgoing arcs of $\overline{c}'$ and $\overline{c}''$
12: $Clear(UC, \overline{c}', \overline{c}'')$ ▷ Remove $\overline{c}'$ and $\overline{c}''$ from C and all arcs in M that (i) involve $\overline{c}'$ or $\overline{c}''$, (ii) do not represent direct containment
13: **return** UC

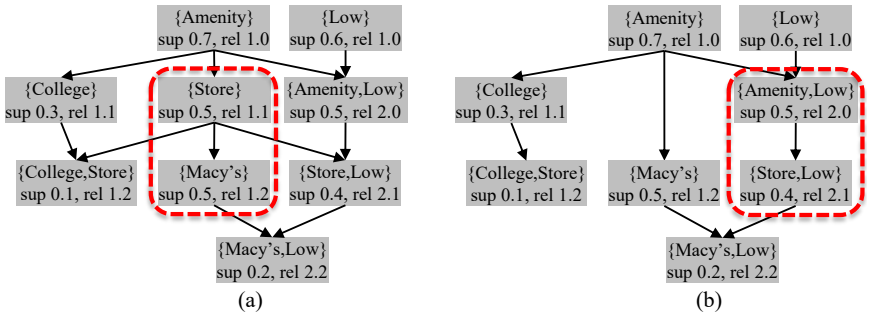


Figure 9: Clustering steps shown on graph UC using the bottom strategy; first (a) and second (b) iterations (each cluster is shown by its representative, arrows represent cluster mergeability, dashed boxes show the next couple of clusters to be merged)

roots. Figure 10 also shows the minimum summaries obtained using the Top and Medoid strategies, emphasizing the differences between the composition and the representative of the resulting clusters.

5.4. Complexity

The computational complexity of hierarchical clustering has a two-faceted nature: initialization and iteration. While the initialization cost (Algorithm 1, Line 2) has been discussed in Section 5.2, we now consider the iteration cost. Note that the complexity of our algorithms does not depend on the number of transactions in the dataset, but it depends on the number of FIs (i.e., on $|\mathcal{F}|$).

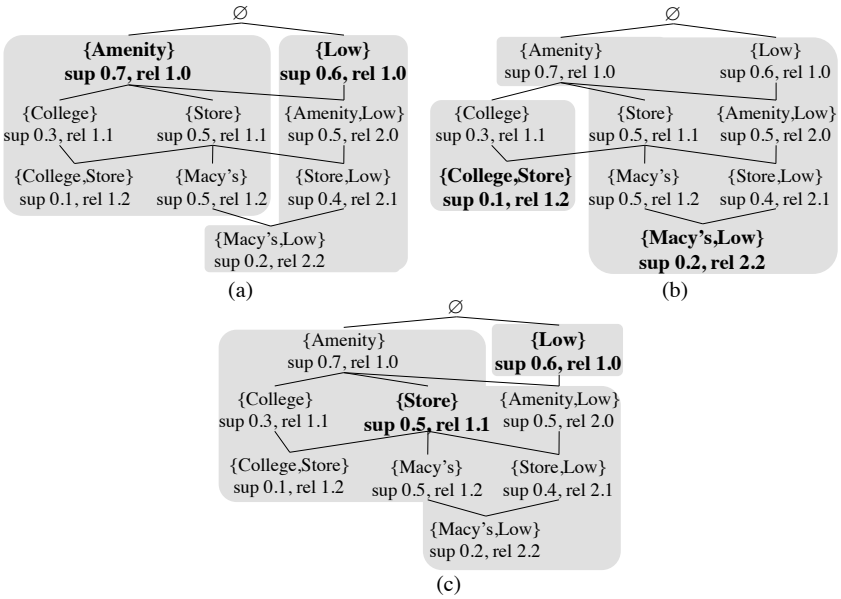


Figure 10: Minimum summaries obtained using the Top (a), Bottom (b), and Medoid (c) strategies shown on $\mathcal{F}$ (gray areas represent clusters, their representatives are in bold)

For instance, in the case of colossal trajectory mining [40, 47], even only a few dozens of transactions can produce a huge number of FIs.

By relying on a priority queue (Algorithm 1, Line 6), the worst-case complexity of hierarchical clustering is $O(|\mathcal{F}|^2 log|\mathcal{F}|)$ [12]: clustering requires $|\mathcal{F}|-1$ merging steps, each of which requires $|\mathcal{F}|$ similarity computations to be stored in the priority queue. Editing a priority queue has a $log|\mathcal{F}|$ complexity. However, merging two clusters entails different complexities according to the adopted strategy. On the one hand, Bottom and Top require no validation for merged cluster, so the actual complexity of clustering is $O(|\mathcal{F}|^2 log|\mathcal{F}|)$. On the other hand, the computation of a feasible medoid for two clusters c' and c'' has quadratic complexity, resulting in a complexity $O(|\mathcal{F}|^3))$. Although medoid approximations exist (e.g., [32, 4]), they require statistical assumptions that are not know while merging the clusters[1]. Remarkably, the medoid approxima-

[1]The estimation of statistical parameters proved to produce "bad" medoids when the cluster population was not sufficient to provide a precise estimation or when global parameters

tion med^* proposed in Section 5.1 has linear complexity, resulting in an overall complexity $O(|\mathcal{F}|^2 log|\mathcal{F}|)$.

Note that in our setting the worst-case scenario is actually quite unlikely, since it requires that: (i) the goal is to aggregate all FIs into a single cluster; (ii) only in the last iteration of the algorithm (when there is a single cluster) the optimal and approximated medoid complexities are $O(|\mathcal{F}|^3)$ and $O(|\mathcal{F}|^2 log|\mathcal{F}|)$; and (iii) no direct containment relationships between FIs are present, so that the the antimonoticity of the similarity function cannot be used to prune the search space.

As to space complexity, hierarchical clustering requires to store both the graph of useful comparisons (UC in Algorithm 1) and the priority queue that sorts the mergeability arcs. Storing UC requires to store both its set of nodes, C, and its set of arcs, M. As to C, all the FIs in $\mathcal{F}$ (i.e., the initial singleton clusters) are stored in memory; if f is the average space taken by an FI (f depends on the size of each item and on the number of items per FI), storing C requires $|\mathcal{F}| \cdot f$ bytes. As to M, each mergeability arc includes a reference to the two clusters (8 bytes in total) and the similarity between their representatives (4 bytes), yielding $|M| \cdot 12$ bytes overall. Finally, the priority queue contains the references to all mergeability arcs, overall $|M| \cdot 4$ bytes. Thus, the total space taken in memory is $|\mathcal{F}| \cdot f + |M| \cdot 16$ bytes. Differently from the computational time, this space does not depend on the summarization strategy as the clustering process does not require to copy new objects in memory, but only to partition graph UC (see Section 7).

6. Visualizing and exploring summaries

Though several summarization approaches have been devised as discussed in Section 2, most of them do not provide user-friendly ways to visually explore the summaries obtained. To make data analysis more effective, SUSHI builds

where extended even to small clusters.

on an interactive visual interface (available at `http://semantic.csr.unibo.it/sushi`) that enables analysts to unveil information hidden in summaries.

Initially, when visualizing an h-summary HS, an overview of the minimum summary $min(HS)$ is provided by showing the top k clusters sorted by cardinality. If $|min(HS)| > k$, to avoid scaling issues the remaining clusters are grouped into a fictitious "other" cluster. SUSHI then provides two different visual approaches that fulfill orthogonal requirements to navigate HS:

- The *graph-based* approach overviews the *entire* h-summary by always showing a complete summary, as in the classical approaches to FI visualization discussed in Section 2, and enables analysts to expand/collapse clusters. This approach highlights inter-cluster relationships (e.g., in our profiling case study, how common behaviors relate to each other).

- In the *tree-based* approach, the context of visualization is a single cluster, shown with its children and grandchildren within the h-summary; thus, the relationships between its representative and the other FIs in the cluster are emphasized (e.g., in our profiling case study, how common a behavior is). Analysts can zoom in and out one cluster.

These two approaches are pursued using well-known visualization layouts, respectively, directed acyclic graphs (DAGs) [18] and treemaps [6]. In both approaches, colors code both the feature with the highest relevance of the cluster representative (hue) and its support (saturation).

Both visualization approaches share two interaction possibilities (see Figure 11(c)):

- *options*: analysts are allowed to select the data source (e.g., a file containing the FIs), the representative strategy (Bottom, Top, or Medoid), the desired number of clusters k, and the similarity coefficient λ;

- *filter*: the data source can be filtered by applying a support threshold and by specifying the specific items the analyst is interested in (e.g., visualizing

only the FIs that contain items (worksIn,Harlem) or (frequents,Museum) with a support between 0.25 and 0.5).

However, as discussed in the following subsections, they differ in the primitives enabling analysts to navigate h-summaries.

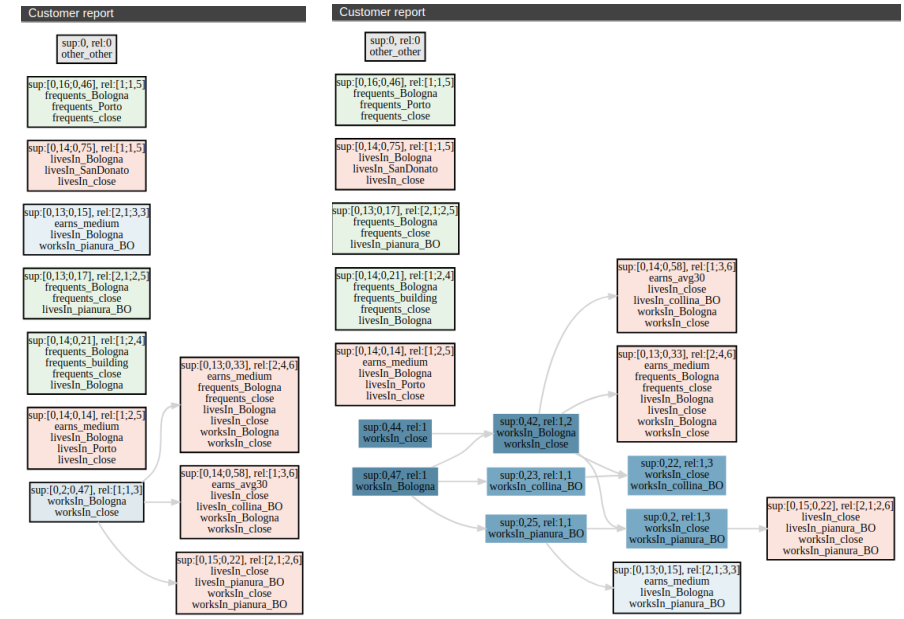
6.1. Graph-based visualization

DAGs gracefully represent inter-cluster relationships within a summary in terms of direct containment between cluster representatives. At each iteration, analysts can expand a thick-bordered cluster or collapse a cluster. Thin-bordered clusters represents singleton clusters that cannot be further expanded. At the maximum level of detail, i.e., when all clusters have been completely expanded, the summary including all singleton clusters is displayed. More precisely, given the summary S currently displayed through the DAG and a cluster $c \in S$: (i) the expansion of c replaces c with its k most similar descendants in HS which completely and disjointly cover the FIs in c, so as to obtain a new summary S'; (ii) the collapse of c undoes its expansion (similarly to the zoom-in, extend, and zoom-out primitives in [37]).

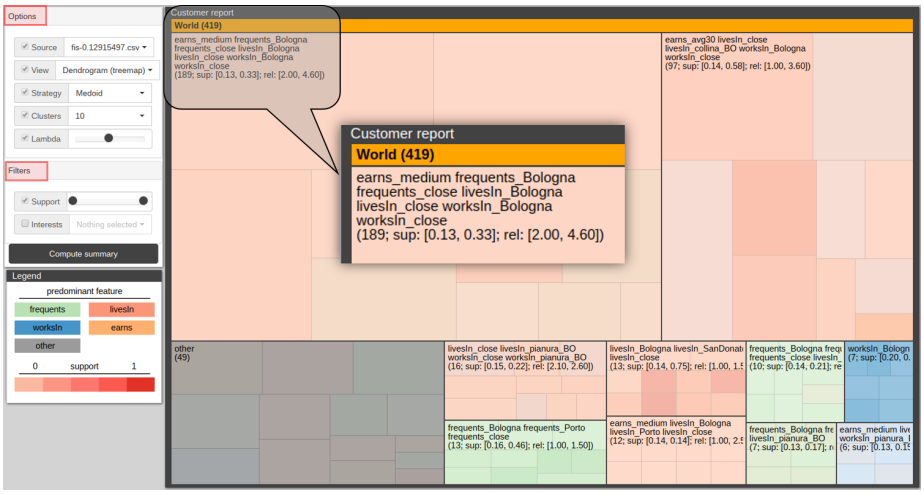
Figure 11(a) shows an example of the DAG representing $min(HS)$ ($k = 10$), while Figure 11(b) shows how the graph changes when a cluster is expanded. DAG visualization is based on the Graphviz open source library [15]. The DAG should be examined from left to right; leftmost clusters have representatives with low item cardinality (high support, low relevance), while rightmost clusters have representatives with high item cardinality (low support, high relevance). This helps analysts in finding containment paths between cluster representatives, and is the reason for keeping FIs sorted in space due to the antimonotonic and monotonic properties of $sup()$ and $rel()$, respectively.

6.2. Tree-based visualization

Ordered treemaps are popular visualization tools for large hierarchical datasets, and we use them to map h-summaries into 2D areas [36]. Figure 11(c) shows



(a) (b)



(c)

Figure 11: Graph-based visualization, before (a) and after (b) the expansion of cluster $\{(\textsf{worksIn}, \text{Bologna}), (\textsf{worksIn}, \text{close})\}$; tree-based visualization (c)

a treemap (drawn using the D3 library) in which the visualization area is partitioned into $k = 10$ (plus one "others") rectangles, each corresponding to a cluster in $min(HS)$. The area of rectangles is proportional to the cluster cardinality, hence, this visualization emphasizes the volume of FIs summarized by each representative. Each rectangle is divided into sub-rectangles corresponding to its k descendants with highest similarity $sim(c)$ in HS. By hovering over these sub-rectangles, analysts can get further details. Physical nesting is used instead of arcs to represent inter-cluster relationships, so the visualization is clearer and easier to interpret, and scalability is improved.

At each iteration, analysts can zoom-in or zoom-out a cluster c. More precisely, (i) a zoom-in of c displays its k most similar descendants in HS which completely and disjointly cover FIs in c; (ii) a zoom-out of c undoes the last zoom-in (similarly to the zoom-in, filter, and zoom-out primitives in [37]).

7. Experimental tests

Summarization can be evaluated in terms of (i) effectiveness (summary compaction, cohesion, information loss, and interestingness), (ii) efficiency (computational performance), and (iii) understandability of the summary. We evaluate SUSHI using two datasets: a real one, called ProfilingDS, related to the Profiling domain schema, and a synthetic one, called SyntheticDS.

- ProfilingDS describes the behavior of 20000 mall customers in the city of Bologna (Italy). The profile of each customer is obtained from her daily GPS trajectories, and modeled through a transaction set $\mathcal{T}$ using multiple features (where she lives and works, the places she frequents, and how much she earns). Transactions are then enriched with multi-level and multi-dimensional knowledge from external open data sources, namely, Open Street Map (`https://www.openstreetmap.org/`) and the Italian Statistic Institute ISTAT (`https://www.istat.it/`); the classical Apriori algorithm [2] is applied to extract the set of FIs $\mathcal{F}$ out of $\mathcal{T}$.

Table 2: Notation summary

Notation	Meaning
$\|\mathcal{H}\| \in [3, 6]$	Number of hierarchies
$\|L\| \in [2, 5]$	Number of levels per hierarchy
$\nu = 20$	Number of values per level
$\lambda \in [0, 0.6]$	Similarity constant
$k = 10$	Number of desired clusters
$\mathcal{F}$	Set of FIs
HS	H-summary
$min(HS)$	Minimum summary
$\|M\|$	Number of mergeability comparisons

- SyntheticDS includes multiple POSs of FIs. While all these POSs share the same number of hierarchies ($|\mathcal{H}| = 4$) and values per level ($\nu = 20$), they have different cardinalities $|\mathcal{F}|$ and different hierarchy depths $|L|$. Additional details will be given in Section 7.1.

Finally, for both dataset, $\mathcal{F}$ is summarized into an h-summary HS. The notation we adopt is summarized in Table 2.

7.1. Effectiveness of the summarization strategies

Effectiveness is evaluated from different perspectives: compaction gain, information loss, interestingness, and cluster cohesion. The first two have been defined in [9] to evaluate a summary S of the set of FIs $\mathcal{F}$, the third one in [24][2]:

- *Compaction gain*: the reduction with respect to $\mathcal{F}$,

$$Gain(S) = |\mathcal{F}|/|S|$$

- *Information loss*: the total amount of information missing from S,

$$Loss(S) = \sum_{c \in S} \sum_{I \in c} \sum_{f \in Feat(I)} loss_f(I, c)$$

[2] In [24], the authors also provide a metric for intelligibility which is not applicable to our approach since it works on itemsets with a fixed schema, while we use schemaless itemsets.

where $loss_f(I, c)$ counts the number of values for feature f that are present in I but not in $rep(c)$.

- *Interestingness*: how different the summary elements are (intuitively, the more unbalanced the summary, the higher its interestingness):

$$Int(S) = \sum_{c \in S} \frac{|c| \cdot (|c| - 1)}{|\mathcal{F}| \cdot (|\mathcal{F}| - 1)}$$

Compaction gain, information loss, and interestingness do not measure how similar the elements of each cluster are to the cluster representative; for this reason we complement them with the *cohesion* of the clusters in S, defined as:

$$Coh(S) = \frac{1}{|S|} \sum_{c \in S} \sum_{I \in c, I \neq rep(c)} \frac{sim(rep(c), I)}{|c| - 1}$$

Figure 12 compares —in terms of cohesion, compaction gain, information loss, and interestingness — the minimum summaries $min(HS)$ produced by the Bottom, Medoid, and Top strategies of SUSHI for ProfilingDS. We comment below the main outcomes:

- Bottom and Medoid outperform Top in terms of cohesion and information loss as a direct consequence of how they are conceived. As to interestingness, Medoid outperforms both other strategies, while Bottom achieves a lower interestingness than Top as it produces more fragmented clusters.

- Higher cohesion values could be obtained by increasing the number of clusters in the h-summary, $|min(HS)|$, which however might make the summary very complex. Figure 13 compares the three strategies for increasing numbers of clusters, showing that Medoid outperforms Top since it produces higher cohesion and lower loss of information; additionally, Medoid can be applied even for more compact summaries (i.e., for low values of $|min(HS)|$). Conversely, Bottom shows its limitations: since the number of maximal FIs is a lower bound to $|min(HS)|$, it always produces a larger number of clusters.

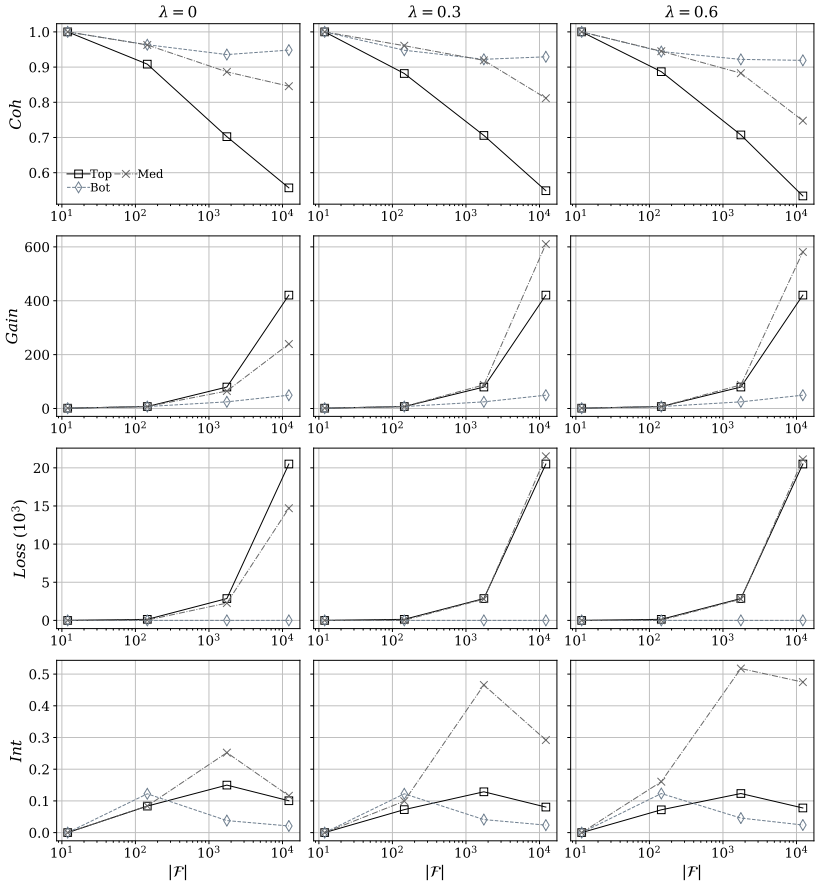


Figure 12: Comparison of the Bottom, Medoid, and Top strategies in terms of effectiveness (with $k = 20$) for ProfilingDS

- SUSHI allows users to tune the relative weight λ of support-based and feature-based similarity when computing itemset relevance. Indeed, as claimed in Section 4, clusters with similar features but different support might underlie different correlations (in ProfilingDS, different customer behaviors). The impact of λ is more apparent for Medoid, while Top and Bottom are less sensitive to it.

- As to information loss, for a fixed summary cardinality, Top and Bottom are at the highest and lowest ends of the range. This is not surprising since, intuitively, Top and Bottom tend to generalize and to preserve the

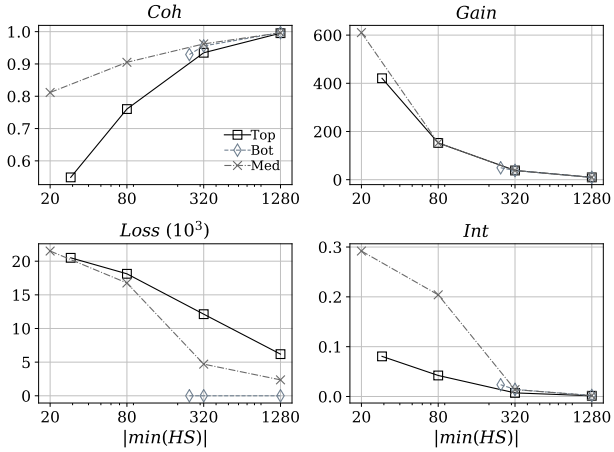


Figure 13: Comparison of the Bottom, Medoid, and Top strategies in terms of effectiveness (with $|\mathcal{F}| = 12000$, $\lambda = 0.3$) for ProfilingDS

summarized information, respectively.

SUSHI explicitly keeps hierarchy into account, thus it is interesting to analyze how the hierarchy structure impacts on effectiveness. To this end we rely on our synthetic dataset, SyntheticDS. First of all, in Figure 14 we compare the three strategies on four POSs including 5000 FIs when the depth $|L|$ of the four hierarchies changes from 2 to 5. Medoid outperforms Bottom and Top in terms of compaction gain and interestingness. For shallow hierarchies, Top and Medoid provide the most compact summaries but, as expected, Medoid produces more cohesive summaries. Conversely, Bottom produces clusters with high cohesion and no information loss (by definition) at the cost of a lower compaction gain (i.e., several small clusters). Noticeably, Medoid —which is always better than Top in terms of cohesion and information loss— also overcomes Top in terms of gain for deep hierarchies since, in this case, each medoid is similar to several FIs (see Definition 7).

The h-summaries returned by SUSHI are inherently hierarchical. While the tests described above only analyze the properties of the minimum summary, it is also interesting to analyze how the properties of summaries change when moving from large and very detailed ones to compact and less informative ones. To

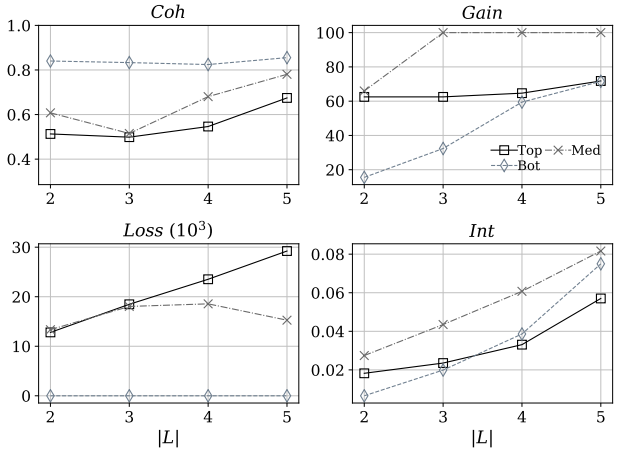


Figure 14: Comparison of the Bottom, Medoid, and Top strategies in terms of effectiveness (with $k = 20$, $\lambda = 0.3$) for SynteticDS

do this, we analyze how the properties of the minimum summary change as Algorithm 1 reduces its cardinality. In particular, Figure 15 shows how cohesion, number $|M|$ of mergeability comparisons, compaction gain, information loss, and interestingness evolve for increasing iteration steps. Not surprisingly, Bottom and Top prune more mergeability arcs than Medoid, with Bottom being the strategy that requires less steps to conclude the algorithm. The three strategies merge progressively less cohesive clusters as iterations proceed. While at early iterations the three strategies merge small clusters, towards the end (approximately after the first 5000 iterations) the pruning of mergeability arcs favors the creations of larger clusters.

Overall, we can conclude that the Top strategy produces compact but not cohesive summaries (because it tends to generalize), the Bottom strategy produces cohesive but not compact summaries (because it preserves details), while the Medoid strategy achieves the best balance between compactness and cohesion. In the process of FI exploration, the three strategies provide different insights: Top enables users to uncover general behaviors supported by a large FI population; Bottom uncovers idiosyncratic behaviors supported by a population of cohesive FIs; Medoid mitigates these effects, uncovering behaviors supported

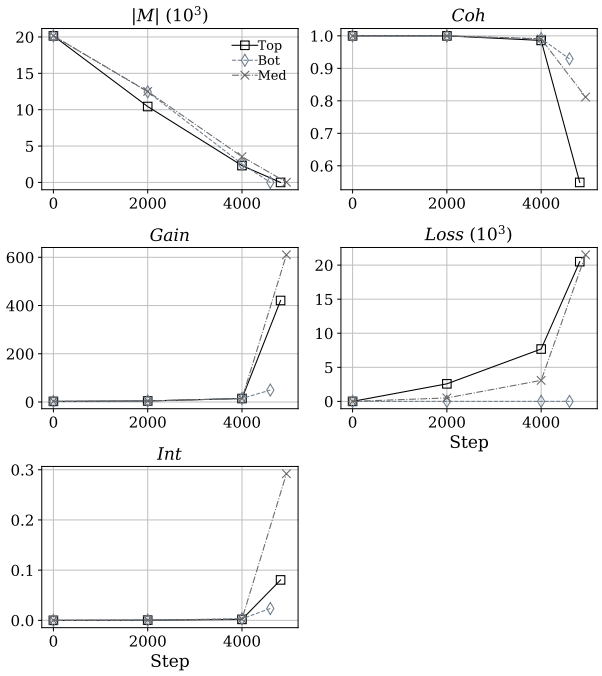


Figure 15: Comparison of the Bottom, Top, and Medoid strategies while building the h-summary ($k = 20$, $|\mathcal{F}| = 12000$, $\lambda = 0.3$) for ProfilingDS

by the most cohesive FIs.

7.1.1. Optimal vs. approximated medoid

As described in Section 5.1, for the sake of scalability, we also implemented an approximated version of the optimal medoid. At every merging step, the percentage variation in cluster cohesion is less than 1% (i.e., even when the approximated medoid is different from the optimal one, they are still highly similar). However, following an iterative process, changing the representative of even a few clusters might produce different summary results. Figure 16 shows how the summaries produced by the approximated and optimal medoids affect the Medoid strategy for increasing summary cardinalities. Noticeably, the provided summaries have the same compaction gain. By obtaining summaries with the same number of clusters and similar cluster medoids, the variation in information loss is also minimal. The summary produced by the approximated

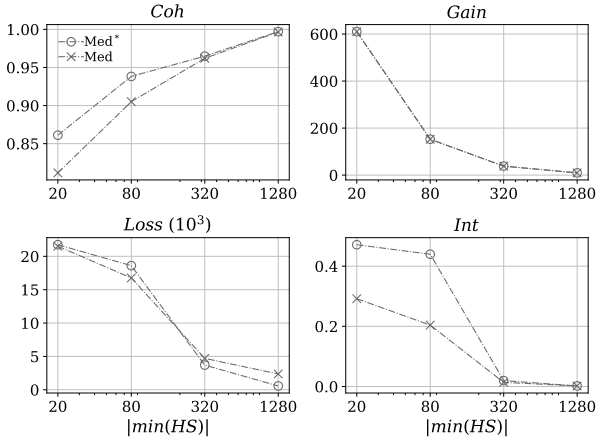


Figure 16: Comparison of the optimal (Med) and approximated (Med*) Medoid strategies in terms of effectiveness ($|\mathcal{F}| = 12000$, $\lambda = 0.3$) for ProfilingDS

strategy tends to be slightly more unbalanced as the medoid is not always centered in the cluster, resulting in higher interestingness (due to the quadratic effect of cluster cardinality, small cardinality variations are amplified in the interestingness metric). Due to unbalancing, the approximated medoid produces a larger number of smaller clusters than the optimal medoid, resulting in a slightly higher average cohesion (cohesion only differs by 0.05 in the worst case). We can conclude that the approximated medoid does represent a valid alternative to the optimal one.

7.2. Efficiency of the summarization strategies

We ran the tests on a machine equipped with Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz CPU and 4GB RAM; all measures are in seconds. We emphasize that we implemented SUSHI in a centralized and sequential architecture; an implementation in a big data distributed solution is out of the paper scope.

As depicted in Figure 17-left and Figure 18-left, SUSHI runs in near-real time even when thousands of FIs are considered. It is apparent that Medoid is computationally heavier than Bottom and Top, due to the quadratic complexity required to compute the medoids. Remarkably, when the approximated medoid

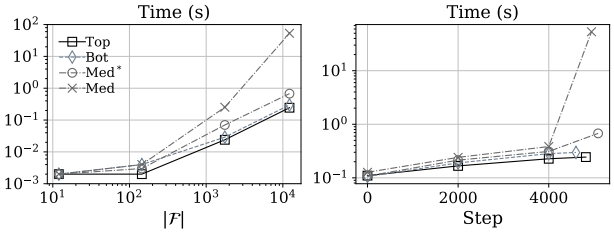


Figure 17: Efficiency in function of (left) the number of FIs $|\mathcal{F}|$ ($k = 20$, $\lambda = 0.3$) and (right) the number of algorithm steps ($k = 20$, $\lambda = 0.3$, $|\mathcal{F}| = 12000$) for ProfilingDS

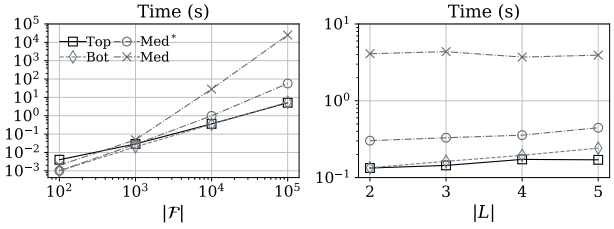


Figure 18: Efficiency in function of (left) the number of FIs $|\mathcal{F}|$ and (right) the hierarchy depth $|L|$ ($k = 20$, $\lambda = 0.3$) for SyntheticDS

is used, the time for summarization is comparable to that of Top and Bottom. The computational gap becomes larger as the average cluster size grows at the different steps of Algorithm 1 (see Figure 17-right). Overall, the performances of the summarization strategies are not sensible to the hierarchy structure (Figure 18-right).

As shown in Figure 19-left, the memory usage clearly increases with the number of FIs, $|\mathcal{F}|$. However, summarizing 10^5 FIs only requires 400MB. This happens also in Figure 19-right by increasing the number of hierarchy levels $|L|$ and by keeping a fixed amount of itemsets, $|\mathcal{F}| = 5000$: in fact, deeper hierarchies produce more mergeability arcs in Algorithm 1. As anticipated in Section 5.4, the memory usage does not depend on the summarization strategies as all the produced summaries are hierarchical partitions of the existing $\mathcal{F}$.

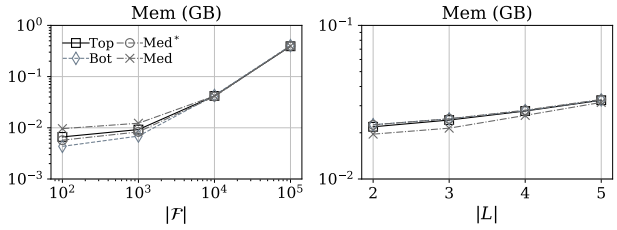


Figure 19: Memory usage in function of (left) the number of FIs $|\mathcal{F}|$ and (right) the hierarchy depth $|L|$ ($k = 20$, $\lambda = 0.3$) for SyntheticDS

7.3. Comparison against BUS and MBUS

To the best of our knowledge, no previous approaches address the summarization of multi-level and multi-dimensional FIs. The closest contribution to SUSHI are [9] and its extension [24], in which the authors introduce the BUS and MBUS algorithms to summarize transactions with a fixed schema. Before performing a quantitative comparison, we report the key differences between SUSHI and (M)BUS:

1. SUSHI relies on a multi-dimensional and multi-level similarity (Definition 7), while in (M)BUS similarity is expressed in terms of set containment (i.e., an FI I summarizes I' if I is a subset of I').

2. SUSHI relies on hierarchical summaries expressed as dendrograms, which natively code containment and similarity relationships between clusters; this allows to interactively navigate h-summaries (i.e., expansion/collapse in Section 6.1 and zoom-in/zoom-out in Section 6.2) by following containment paths. Conversely, (M)BUS generates "flat" summaries (i.e., plain sets of FIs); in (M)BUS, summary navigation requires multiple runs of the algorithm with different values of $|min(HS)|$, which does not preserve the relationships among clusters.

3. SUSHI implements three summarization strategies, while (M)BUS implements only the Top strategy.

Due to these differences, BUS and MBUS cannot be directly compared to

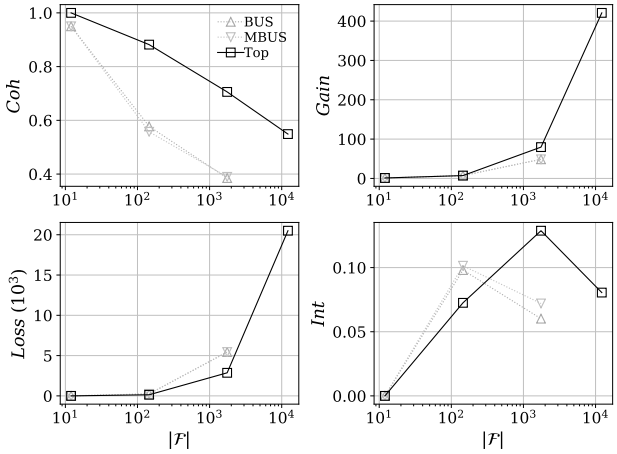


Figure 20: Comparison of the Top strategy, BUS ,and MBUS in terms of effectiveness (with $k = 20$, $\lambda = 0.3$) for ProfilingDS

SUSHI. Thus, to compare the three approaches in terms of effectiveness we limit SUSHI to the Top strategy and only consider the minimum summaries it produces. Figure 20 compares —in terms of cohesion, compaction gain, information loss, and interestingness— the minimum summaries $min(HS)$ produced by the Top strategy of SUSHI and the summaries produced by BUS and MBUS. Clearly, while the three approaches are comparable as to compaction gain, information loss, and interestingness, SUSHI significantly outperforms the others in terms of cohesion. This happens because the SUSHI similarity function captures hierarchical similarity between FIs, while in (M)BUS similarity is limited to set containment. So, for instance, in (M)BUS the two FIs $I = \{(\mathsf{frequents}, \text{Store})\}$ and $I' = \{(\mathsf{frequents}, \text{Macy's})\}$ have null similarity, hence, they cannot be clustered together.

As to efficiency, Figure 21 shows that all three SUSHI strategies outperform BUS and MBUS by orders of magnitude. The execution of the algorithms was stopped when $|\mathcal{F}| = 1100$, since obtaining larger numbers of FIs would require hours. The dramatic improvement of SUSHI is strictly related to its capability of reducing the number of comparisons between FIs by exploiting hierarchies, as formally described in Section 5.2.

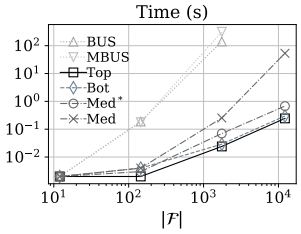


Figure 21: Comparison of SUSHI, BUS, and MBUS in terms of efficiency in function of the number of FIs ($k = 20$, $\lambda = 0.3$) for ProfilingDS

To sum up, from the discussion above it clearly emerges that without a dedicated approach, such as SUSHI, the summarization of multi-level and multi-dimensional FIs yields poor performances from both points of view of effectiveness and efficiency.

7.4. Summary understandability

To assess the quality of the summary and the visual experience with SUSHI, we conducted a set of tests with 15 users, mainly master students in data science with basic or advanced knowledge of FI mining. After a 15-minutes introduction to SUSHI and to our profiling case study, the users were asked to answer 6 analytical questions in 6 minutes each, and finally to fill out a qualitative questionnaire. Of the 6 questions, two were answered using a spreadsheet software with a plain CSV file, two using graph-based visualization, and two using tree-based visualization; the dataset included 419 FIs (analyzing a larger dataset in a plain file would have been too complex). Here is an example of question:

Which common behavior do the mall customers show among the following ones?

1. $\{(\mathsf{livesIn}, \text{Bologna}), (\mathsf{livesIn}, \text{close}), (\mathsf{earns}, \text{Medium})\}$
2. $\{(\mathsf{livesIn}, \text{Bologna}), (\mathsf{worksIn}, \text{far}), (\mathsf{earns}, \text{High})\}$
3. $\{(\mathsf{livesIn}, \text{Bologna}), (\mathsf{earns}, \text{10to35}), (\mathsf{frequents}, \text{HillsInBologna})\}$

Table 3 summarizes the quantitative results. From the comments made in the questionnaire, it emerges that the spreadsheet is too dispersive and hardly

Table 3: Outcome of user evaluation

	Spreadsheet	Graph-based	Tree-based
Preferences	2	3	10
Complexity	7	6.7	5.8
Time (s)	262	308	231
Score	0.64	0.63	0.7

manageable for large datasets; graph-based visualization provides an intuitive overview of the summary and well highlights FI relationships, though navigation is difficult when several clusters are displayed; tree-based visualization gives an intuitive overview of the summary, a focused navigation, and good readability.

Overall, users mostly appreciated tree-based visualization, which yields the lowest complexity in understanding summaries, friendly in-depth navigation, and the lowest time per answer. Both SUSHI visualizations provide intuitive overviews of the summary, with tree-based visualization allowing the most focused navigation. The representativeness of the visualized FI with respect to the FIs within the same group is also well perceived. Though the overall scores are comparable, spreadsheet analysis is deemed to be too dispersive and not effective for FI summarization. The longer answering time with graph-based visualization with respect to spreadsheet is explained by considering that the brief training made for SUSHI could not balance the previous experience of users with spreadsheets.

8. Conclusion

The new applications emerging in the era of analytics and big data ask for the study of new machine learning techniques as well as for revamping established ones. In particular, our work has been inspired by a real profiling study based on the tracking of GPS positions of people. Although multi-level and multi-dimensional FIs are a perfect way to represent the behavior of clusters of customers, the huge number of FIs mined hinders their effective analysis. For this reason we proposed SUSHI, an original approach to FI summarization and visualization based on an innovative similarity function. SUSHI turned out

to outperform previous approaches both from the efficiency and effectiveness points of view. Moreover, the proposed similarity and visualization techniques were successful in the tests with real users, proving to be a valuable tool to expedite the analysis of FIs.

References

[1] Afrati, F.N., Gionis, A., Mannila, H., 2004. Approximating a collection of frequent sets, in: Proc. SIGKDD, pp. 12–19.

[2] Agrawal, R., Srikant, R., 1994. Fast algorithms for mining association rules in large databases, in: Proc. VLDB, pp. 487–499.

[3] Ahmed, M., 2019. Data summarization: a survey. Knowl. Inf. Syst. 58, 249–273.

[4] Bagaria, V.K., Kamath, G.M., Ntranos, V., Zhang, M.J., Tse, D., 2018. Medoids in almost-linear time via multi-armed bandits, in: AISTATS, PMLR. pp. 500–509.

[5] Baralis, E., Cagliero, L., Cerquitelli, T., D'Elia, V., Garza, P., 2010. Support driven opportunistic aggregation for generalized itemset extraction, in: Proc. IS, pp. 102–107.

[6] Bederson, B.B., Shneiderman, B., Wattenberg, M., 2002. Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. ACM Trans. Graph. 21, 833–854.

[7] Bothorel, G., Serrurier, M., Hurter, C., 2013. Visualization of frequent itemsets with nested circular layout and bundling algorithm, in: Proc. ISVC, pp. 396–405.

[8] Carbonell, J.G., Goldstein, J., 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries, in: Proc. SIGIR, pp. 335–336.

[9] Chandola, V., Kumar, V., 2007. Summarization — compressing data into an informative representation. Knowl. Inf. Syst. 12, 355–378.

[10] Chon, K., Hwang, S., Kim, M., 2018. Gminer: A fast gpu-based frequent itemset mining method for large-scale data. Inf. Sci. 439-440, 19–38.

[11] Davidson, I., Ravi, S.S., 2009. Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. Data Min. Knowl. Discov. 18, 257–282.

[12] Day, W.H., Edelsbrunner, H., 1984. Efficient algorithms for agglomerative hierarchical clustering methods. Journal of classification 1, 7–24.

[13] Djenouri, Y., Drias, H., Bendjoudi, A., 2014. Pruning irrelevant association rules using knowledge mining. IJBIDM 9, 112–144.

[14] Djenouri, Y., Lin, J.C., Nørvåg, K., Ramampiaro, H., 2019. Highly efficient pattern mining based on transaction decomposition, in: ICDE, IEEE. pp. 1646–1649.

[15] Ellson, J., Gansner, E.R., Koutsofios, E., North, S.C., Woodhull, G., 2001. Graphviz - open source graph drawing tools, in: Graph Drawing, pp. 483–484.

[16] Ertek, G., Demiriz, A., 2006. A framework for visualizing association mining results, in: Proc. ISCIS, pp. 593–602.

[17] Francia, M., Golfarelli, M., Rizzi, S., 2018. A similarity function for multi-level and multi-dimensional itemsets, in: Proc. SEBD.

[18] Gansner, E.R., Koutsofios, E., North, S.C., Vo, K., 1993. A technique for drawing directed graphs. IEEE Trans. Software Eng. 19, 214–230.

[19] Golfarelli, M., Rizzi, S., 2009. Data warehouse design: Modern principles and methodologies. McGraw-Hill.

[20] Gunopulos, D., Khardon, R., Mannila, H., Toivonen, H., 1997. Data mining, hypergraph transversals, and machine learning, in: Proc. SIGACT-SIGMOD-SIGART, pp. 209–216.

[21] Han, J., Cheng, H., Xin, D., Yan, X., 2007. Frequent pattern mining: current status and future directions. Data Min. Knowl. Discov. 15, 55–86.

[22] Han, J., Fu, Y., 1999. Mining multiple-level association rules in large databases. IEEE Trans. Knowl. Data Eng. 11, 798–804.

[23] Han, J., Pei, J., Yin, Y., Mao, R., 2004. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Min. Knowl. Discov. 8, 53–87.

[24] Hoplaros, D., Tari, Z., Khalil, I., 2014. Data summarization for network traffic monitoring. J. Network and Computer Applications 37, 194–205.

[25] Jin, R., Abu-Ata, M., Xiang, Y., Ruan, N., 2008. Effective and efficient itemset pattern summarization: regression-based approaches, in: Proc. SIGKDD, pp. 399–407.

[26] Keim, D.A., 2002. Information visualization and visual data mining. IEEE Trans. Vis. Comput. Graph. 8, 1–8.

[27] Leung, C.K., Carmichael, C.L., 2009. FpVAT: a visual analytic tool for supporting frequent pattern mining. SIGKDD Explorations 11, 39–48.

[28] Lim, S.J., 2009. On a visual frequent itemset mining, in: Proc. ICDIM, pp. 46–51.

[29] Liu, G., Zhang, H., Wong, L., 2014. A flexible approach to finding representative pattern sets. IEEE Trans. Knowl. Data Eng. 26, 1562–1574.

[30] Luna, J.M., Fournier-Viger, P., Ventura, S., 2019. Frequent itemset mining: A 25 years review. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 9.

[31] Mampaey, M., Vreeken, J., 2013. Summarizing categorical data by clustering attributes. Data Min. Knowl. Discov. 26, 130–173.

[32] Newling, J., Fleuret, F., 2017. A sub-quadratic exact medoid algorithm, in: AISTATS, PMLR. pp. 185–193.

[33] Nguyen, L.T.T., Vu, V.V., Lam, M.T.H., Duong, T.T.M., Manh, L.T., Nguyen, T.T.T., Vo, B., Fujita, H., 2019. An efficient method for mining high utility closed itemsets. Inf. Sci. 495, 78–99.

[34] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L., 1999. Discovering frequent closed itemsets for association rules, in: Proc. ICDT, pp. 398–416.

[35] Poernomo, A.K., Gopalkrishnan, V., 2009. CP-summary: a concise representation for browsing frequent itemsets, in: Proc. SIGKDD, pp. 687–696.

[36] Shneiderman, B., 1992. Tree visualization with tree-maps: 2-d space-filling approach. ACM Trans. Graph. 11, 92–99.

[37] Shneiderman, B., 1996. The eyes have it: A task by data type taxonomy for information visualizations, in: Proc. Symposium on Visual Languages, pp. 336–343.

[38] Song, W., Liu, M., 2014. A visualizer for high utility itemset mining, in: Proc. CSE, pp. 244–248.

[39] Takigawa, I., Mamitsuka, H., 2011. Efficiently mining δ-tolerance closed frequent subgraphs. Machine Learning 82, 95–121.

[40] Vanahalli, M.K., Patil, N., 2019. An efficient parallel row enumerated algorithm for mining frequent colossal closed itemsets from high dimensional datasets. Inf. Sci. 496, 343–362.

[41] Wang, J., Karypis, G., 2006. On efficiently summarizing categorical databases. Knowl. Inf. Syst. 9, 19–37.

[42] Xiang, Y., Jin, R., Fuhry, D., Dragan, F.F., 2011. Summarizing transactional databases with overlapped hyperrectangles. Data Min. Knowl. Discov. 23, 215–251.

[43] Xin, D., Han, J., Yan, X., Cheng, H., 2005. Mining compressed frequent-pattern sets, in: Proc. VLDB, pp. 709–720.

[44] Yan, X., Cheng, H., Han, J., Xin, D., 2005. Summarizing itemset patterns: a profile-based approach, in: Proc. SIGKDD, pp. 314–323.

[45] Yang, L., 2005. Pruning and visualizing generalized association rules in parallel coordinates. IEEE Trans. Knowl. Data Eng. 17, 60–70.

[46] Yu, W., 2019. Discovering frequent movement paths from taxi trajectory data using spatially embedded networks and association rules. IEEE Trans. Intelligent Transportation Systems 20, 855–866.

[47] Zaki, F.A.M., Zulkurnain, N.F., 2018. RARE: mining colossal closed itemset in high dimensional data. Knowl.-Based Syst. 161, 1–11.

[48] Zhang, S., Jin, Z., Lu, J., 2010. Summary queries for frequent itemsets mining. Journal of Systems and Software 83, 405–411.

[49] Zhang, X., Deng, Z., 2015. Mining summarization of high utility itemsets. Knowl.-Based Syst. 84, 67–77.

[50] Zihayat, M., An, A., 2014. Mining top-k high utility patterns over data streams. Inf. Sci. 285, 138–161.