Discovering temporal change patterns in the presence of taxonomies

Discovering temporal change patterns in the presence of taxonomies

Luca Cagliero

L. Cagliero is with the Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy. E-mail:luca.cagliero@polito.it. The author would like to thank Telecom Italia Lab. for having provided the source data and Prof. Elena Baralis, Dr. Tania Cerquitelli, and Dr. Paolo Garza for their insightful suggestions and comments.

Abstract

Frequent itemset mining is a widely exploratory technique that focuses on discovering recurrent correlations among data. The steadfast evolution of markets and business environments prompts the need of data mining algorithms to discover significant correlation changes in order to reactively suit product and service provision to customer needs. Change mining, in the context of frequent itemsets, focuses on detecting and reporting significant changes in the set of mined itemsets from one time period to another. The discovery of frequent generalized itemsets, i.e., itemsets that (i) frequently occur in the source data, and (ii) provide a high level abstraction of the mined knowledge, issues new challenges in the analysis of itemsets that become rare, and thus are no longer extracted, from a certain point.

This paper proposes a novel kind of dynamic pattern, namely the HIGEN (HISTORY GENERALIZED PATTERN), that represents the evolution of an itemset in consecutive time periods, by reporting the information about its frequent generalizations characterized by minimal redundancy (i.e., minimum level of abstraction) in case it becomes infrequent in a certain time period. To address HIGEN mining, it proposes HIGEN MINER, an algorithm that focuses on avoiding itemset mining followed by postprocessing by exploiting a support-driven itemset generalization approach. To focus the attention on the minimally redundant frequent generalizations and thus reduce the amount of the generated patterns, the discovery of a smart subset of HIGENS, namely the NON-REDUNDANT HIGENS, is addressed as well.

Experiments performed on both real and synthetic datasets show the efficiency and the effectiveness of the proposed approach as well as its usefulness in a real application context.

Index Terms

Data Mining, Mining Methods and Algorithms

I. INTRODUCTION

The problem of discovering relevant data recurrences and their most significant temporal trends is becoming an increasingly appealing research topic. The application of frequent itemset mining and association rule extraction algorithms [1] to discover valuable correlations among data has been thoroughly investigated in a number of different application contexts (e.g., market basket analysis [1], medical image processing [4]). In the last years, the steady growth of business-oriented applications tailored to the extracted knowledge prompted the need of analyzing the evolution of the discovered patterns. Since, in many business environments, companies are expected to reactively suit product and service provision to customer needs, the investigation of the most notable changes between the set of frequent itemsets or association rules mined from different time periods has become an appealing research topic [2], [9], [12], [19], [23], [27].

Frequent itemset mining activity is constrained by a minimum support threshold to discover patterns whose observed frequency in the source data (i.e., the support) is equal to or exceeds a given threshold [1]. However, the enforcement of low support thresholds may entail generating a very large amount of patterns which may become hard to look into. On the other hand, higher support threshold enforcement may also prune relevant but not enough frequent recurrences. To overcome these issues, generalized itemset extraction could be exploited. Generalized itemsets, which have been first introduced in [3] in the context of market basket analysis, are itemsets that provide a high level abstraction of the mined knowledge. By exploiting a taxonomy (i.e., a is-a hierarchy)

Date	Time	Location	Product description
2009-01-02	11:00 p.m.	Turin	T-shirt
2009-01-03	11:10 p.m.	Rome	T-shirt
2009-01-05	8:40 a.m.	Paris	Jacket
2009-01-30	5:05 p.m.	Paris	Jacket
2009-01-31	8:40 a.m.	Cannes	Jacket

(a) Dataset D_1. Product sales in January 2009

Date	Time	Location	Product description
2009-02-02	11:10 p.m.	Turin	T-shirt
2009-02-02	10:27 p.m.	Turin	T-shirt
2009-02-05	8:50 a.m.	Paris	Jacket
2009-02-05	5:00 p.m.	Cannes	Jacket
2009-02-28	5:00 p.m.	Rome	Jacket

(b) Dataset D_2. Product sales in February 2009

TABLE I

RUNNING EXAMPLE DATASETS

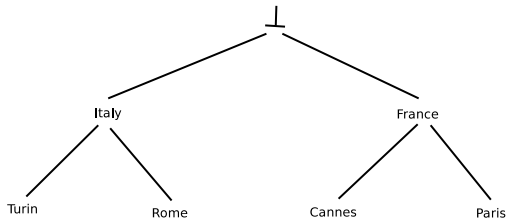


Fig. 1. Generalization hierarchy for the location attribute in example datasets D_1 and D_2

over data items, items are aggregated into higher level (generalized) ones. Tables I(a) and I(b) report two example datasets, associated, respectively, with two consecutive months of year 2009 (i.e., January and February). Each record corresponds to a product sale. In particular, for each sale the product description, the date, the time, and the location in which it took place are reported. For the sake of simplicity, in this preliminary analysis I am interested in mining generalized itemsets involving just the location or the product description. Figure 1 shows a simple hierarchy defined on the location attribute. Table II(a) reports the set of all possible frequent generalized and not generalized itemsets, and their corresponding absolute support values (i.e., observed frequencies), mined, by means of a traditional approach [3], from the example datasets D_1 and D_2, by exploiting the taxonomy reported in Figure 1 and by enforcing an absolute minimum support threshold equal to 2. Readers can notice that some of the itemsets satisfy the support threshold in both D_1 and D_2, while the others are frequent in only one out of two months.

Not generalized itemset	**Sup** D_1	**Sup** D_2
{Turin}	1 (Inf.)	2
{Paris}	2	1 (Inf.)
{T-shirt}	2	2
{Jacket}	3	3
{T-shirt, Turin}	1 (Inf.)	2
{Jacket, Paris}	2	1 (Inf.)
Generalized itemset	**Sup** D_1	**Sup** D_2
{Italy}	2	3
{France}	3	2
{T-shirt, Italy}	2	2
{Jacket, France}	3	2

(a) Generalized and not generalized itemsets mined from D_1 and D_2.

HIGENs **from** D_1 **to** D_2
{**T-shirt**} [sup = 2] $\rightsquigarrow$ {T-shirt} [sup = 2]
{**Jacket**} [sup = 3] $\rightsquigarrow$ {Jacket} [sup = 3]
{**Paris**} [sup = 2] $\nearrow$ {France} [sup = 2]
{T-shirt, Italy} [sup = 2] $\searrow$ {**T-shirt, Turin**} [sup = 2]
{**Jacket, Paris**} [sup = 2] $\nearrow$ {Jacket, France} [sup = 2]
{Italy} [sup = 3] $\searrow$ {**Turin**} [sup = 2]

(b) Extracted HIGENs.

TABLE II

EXTRACTED PATTERNS. min_sup = 2.

This paper focuses on change mining in the context of frequent itemsets by exploiting generalized itemsets to represent patterns that become rare with respect to the support threshold, and thus are no longer extracted, at a certain point. Previous approaches allow both keeping track of the evolution of the most significant pattern quality indexes (e.g., [2], [7], [8]) and discovering their most fundamental changes (e.g., [5], [12], [19]). Consider, for instance, the itemsets {*T-shirt, Turin*} and {*Jacket, Paris*}. The former itemset is infrequent in D_1 (i.e., its support value in D_1 is lower than the support threshold) and becomes frequent in D_2 due to a support value increase from January to February, while the latter shows an opposite trend. Albeit these pieces of information might be of interest for business decision making, the discovery of higher level correlations, in the form of generalized itemsets, issues new challenges in the investigation of pattern temporal trends. Itemset generalization may allow preventing infrequent knowledge discarding. At the same time, experts may look into the information provided by the sequence of generalizations or specializations of the same pattern in consecutive time periods. For instance, the fact that the generalized itemset {*T-shirt, Italy*} (frequent in both January and February 2009) has a specialization (i.e., {*T-shirt, Turin*}) that is infrequent in January while becomes frequent in the next month may be deemed relevant for decision making and, thus, might be reported as it highlights a temporal correlation among spatial recurrences regarding product *T-shirt*. Similarly, the information that, although {*Jacket, Paris*} becomes infrequent with respect to the minimum support threshold moving from January to February, its generalization {*Jacket, France*} remains frequent

in both months could be relevant for analyst decision making as well. However, to the best of my knowledge, this kind of information cannot be neither directly mined nor concisely represented by means of any previously proposed approach.

This paper proposes a novel kind of dynamic pattern, namely the HIGENs (HIstory GENeralized Pattern). A HIGEN compactly represents the minimum sequence of generalizations needed to keep knowledge provided by a not generalized itemset frequent, with respect to the minimum support threshold, in each time period. If an itemset is frequent in each time period, the corresponding HIGEN just reports its support variations. Otherwise, when the itemset becomes infrequent at a certain point, the HIGEN reports the minimum number of generalizations (and the corresponding generalized itemsets) needed to make its covered knowledge frequent at a higher level of abstraction. I argue that a frequent generalization at minimum abstraction level represents the rare knowledge with minimal redundancy. In case an infrequent not generalized itemset has multiple generalizations belonging to the same minimal aggregation level, many HIGENs associated with the same itemset are generated. To focus the attention of the analysts on the frequent generalizations of a rare itemset covering the same knowledge with a minimal amount of redundancy and, thus, reduce the number of the generated patterns, a more selective type of HIGEN, i.e., the NON-REDUNDANT HIGEN, is proposed as well. NON-REDUNDANT HIGENs are HIGENs that include, for each time period, the frequent generalizations of the reference itemset of minimal generalization level characterized by minimal support.

In Table II(b) the set of HIGENs, mined from the example dataset by enforcing an absolute minimum support threshold equal to 2, is reported. The reference itemset is written in boldface. The relationship $\nearrow$ means that the right-hand side itemset is a generalization of the left-hand side one, $\searrow$ implies a specialization relationship, while $\rightsquigarrow$ implies that no abstraction level change occurs. For instance, $\{$*Jacket, Paris*$\} \nearrow \{$*Jacket, France*$\}$ is a HIGEN stating that, from January (D_1) to February (D_2) 2009, the not generalized itemset $\{$*Jacket, Paris*$\}$ becomes infrequent with respect to the minimum support threshold while its upper level generalization (see Figure 1) remains frequent. All the HIGENs reported in Table II(b) refer to a different not generalized reference itemset. In case the extraction would generate more HIGENs per itemset, the generation of the NON-REDUNDANT HIGENs generates just the HIGENs covering the minimal amount of redundancy, i.e., the ones including generalizations with minimal support.

To address HIGEN mining, the HIGEN MINER (HIstory GENeralized Pattern MINER) has been proposed. Instead of generating the HIGENs by means of a postprocessing step that follows the generalized itemset mining process performed on data collected at each time interval, the HIGEN MINER algorithm directly addresses the HIGEN mining by extending a support-driven generalization approach, first proposed in [6], to a dynamic context. The proposed algorithm avoids both redundant knowledge extraction followed by postpruning and multiple taxonomy evaluations over the same pattern mined from different time intervals. A slightly modified version of the HIGEN MINER algorithm is also proposed to address NON-REDUNDANT HIGEN mining.

The proposed approach may be profitably exploited in a number of different application domains (e.g., context-aware domain, network traffic analysis, social network analysis). Experiments, reported in Section VI, show (i) the

effectiveness of the proposed approach in supporting expert decision making through the analysis of context data coming from a context-aware mobile environment and (ii) the efficiency and the scalability of the proposed HIGEN MINER algorithm on synthetic datasets.

This paper is organized as follows. Section II discusses and compares state-of-the-art approaches concerning both change mining and generalized itemset mining with the proposed approach. Section III introduces preliminary notions, while Section IV formally states the HIGEN mining problem. Section V describes the HIGEN MINER algorithm. Section VI evaluates the efficiency and the effectiveness of the proposed approach. Finally, Section VII draws conclusions and presents related future work.

II. RELATED WORK

Frequent itemset mining has been first proposed in [1], in the context of market basket analysis, as the first step of the association rule extraction process. The problem of discovering relevant changes in the history of itemsets and association rules has been already addressed by a number of research papers (e.g., [2], [5], [7], [8], [12], [19]). Active data mining [2] was the first attempt to represent and query the history pattern of the discovered association rule quality indexes. It first mines rules, from datasets collected in different time periods, by adding rules and their related quality indexes (e.g., support and confidence [1]) to a common rule base. Next, the analyst could specify a history pattern in a trigger which is fired when such a pattern trend is exhibiting. Albeit history patterns allow tracking most notable pattern index changes, they neither keep the information covered by rare patterns nor show links among patterns at different abstraction levels. More recently, other time-related data mining frameworks tailored to monitor and detect changes in rule support and confidence have been proposed [7], [8], [25]. In [8] patterns are evaluated and pruned based on both subjective and objective interestingness measures. The aim of [7] is to monitor the mined patterns and reduce the effort spent in data mining. To achieve this goal, new patterns are observed as soon as they emerge, and old patterns are removed from the rule base as soon as they become extinct. To further reduce the computational cost, at one time period a subset of rules is selected and monitored, while data changes that occur in subsequent periods are measured by their impact on the rules being monitored. Similarly, [25] addresses itemset change mining from time-varying data streams. However, the approaches presented in [7], [8], [25] do not address rare knowledge representation by means of generalized itemsets. Differently, [20] deals with rule change mining by discovering two main types of rules: (i) the stable rules, i.e., rules that do not change a great deal over time and, thus, are more reliable and could be trusted, and (ii) the trend rules, i.e., rules that indicate some underlying systematic trends of potential interest. Similarly, this paper categorizes the HIGENs based on their temporal trend. However, their evolution has been classified in terms of itemset generalizations or specializations that occur over time. The problem of discovering the subset of most relevant changes in association rules has been addressed by (i) evaluating rule changes that occur between two time periods by means of chi-square test [19], (ii) searching for border descriptions of emerging patterns extracted from a pair of datasets [12], and (iii) applying a fuzzy approach to rule change evaluation [5]. Unlike [5], [12], [19], the proposed approach allows both comparing more than two time periods at a time and representing patterns that become infrequent in a certain time period by

means of one of their generalizations characterized by minimal redundancy.

A parallel issue concerns the detection of most notable changes in multidimensional data measures. [18] first introduces the concept of cubegrade and compares the multidimensional data cube cells with their gradient cells, namely their ancestors, descendants, and siblings. [11] extends the work proposed in [18] by pushing anti-monotone constraints into the mining of highly similar data cube cell pairs with hugely different measure values. The HIGEN mining problem is somehow related to but different from that addressed in [11], [18] as it concerns the discovery, from relational data, of temporal change patterns composed of frequent generalized itemsets at minimal abstraction level.

A parallel effort has been devoted to proposing more efficient and effective algorithms to address generalized frequent itemset mining. The problem of frequent generalized itemset mining has been first introduced in [3] in the context of market basket analysis as an extension of the traditional frequent itemset extraction task [1]. It generates itemsets by considering, for each item, all its parents in a taxonomy (i.e., a is-a hierarchy defined on data items). Hence, all combinations of candidate frequent itemsets are generated by exhaustively evaluating the taxonomy. To prevent redundant knowledge extraction, several optimization strategies have been proposed (e.g., [6], [15]–[17]). Among them, [6] proposed a support-driven itemset generalization approach, i.e., a frequent generalized itemset is extracted only if it has at least an infrequent descendant. This paper extends the generalization procedure proposed in [6] to a dynamic context to address HIGEN mining from a sequence of time-related datasets without the need of a postprocessing step. Unlike [6], it does not extract all frequent generalizations of an infrequent pattern, but rather generates only the ones characterized by minimal redundancy.

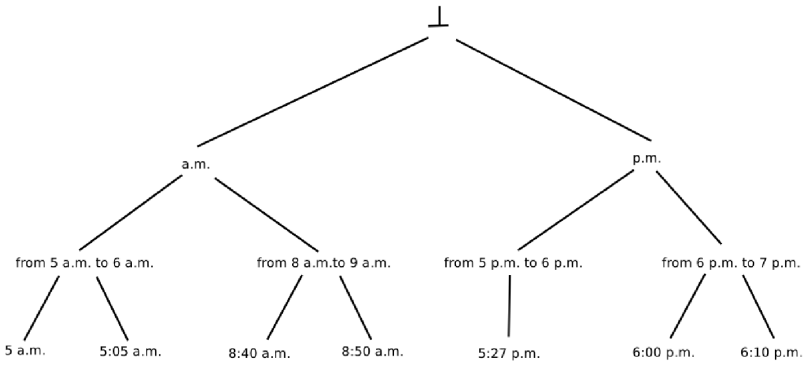
III. PRELIMINARY DEFINITIONS

This section introduces main notions concerning dynamic generalized itemset mining from structured data.

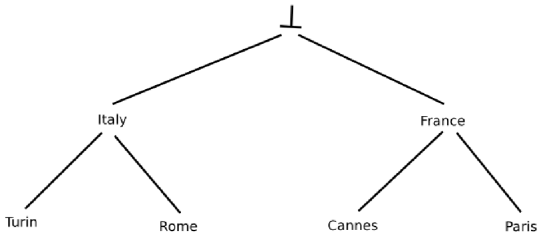
A structured dataset is a collection of records, where each record is a set of pairs $(attribute, value)$ (e.g., *(date, 5:05 p.m.)*), called items, which identify specific data features (e.g., the time) and their values in the corresponding domains (e.g., *5:05 p.m.*). Dynamic data mining considers datasets associated with different time periods. Thus, I introduce the notion of timestamped structured dataset. Its formal definition follows.

Definition 3.1: **Timestamped structured dataset.** Let $\mathcal{T}$=$\{t_1, t_2, \ldots, t_n\}$ be a set of data features and Ω=$\{\Omega_1, \Omega_2, \ldots, \Omega_n\}$ its corresponding domains. t_i may be either a categorical or a numeric discrete data feature. Let $W = [w_{start}, w_{end}]$ be a time interval. A timestamped structured dataset D is a collection of records, where (i) each record r is a set of pairs $(t_i, value_i)$ where $t_i \in \mathcal{T}$ and $value_i \in \Omega_i$, (ii) each $t_i \in \mathcal{T}$, also called attribute, may occur at most once in any record, and (iii) each record has a time stamp $r_{ts} \in W$.

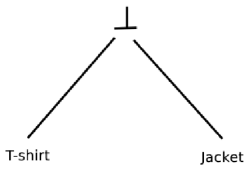
In the case of datasets with continuous attributes, the value range should be discretized into intervals, and the intervals mapped into consecutive positive integers. Consider again the example datasets D_1 and D_2 reported in Tables I(a) and I(b). They are examples of timestamped structured datasets composed of 4 attributes (i.e., date, time, location, and product description), among which the date attribute is selected as time stamp. Dataset D_1 includes



(a) Time - Generalization Hierarchy GH_{time}



(b) Location - Generalization Hierarchy $GH_{location}$



(c) Product description - Generalization Hierarchy $GH_{product}$

Fig. 2. Taxonomy over data items in D_1 and D_2

product sales (i.e., records) whose time stamp ranges from 2009-01-01 to 2009-01-31 (see Table I(a)), while D_2 includes product sales whose time stamp ranges from 2009-02-01 to 2009-02-28 (see Table I(b)).

Generalized itemset mining exploits a set of generalization hierarchies, i.e., a taxonomy, built over data items to generalize at higher levels of abstraction items represented in a timestamped structured dataset. A taxonomy is a is-a hierarchy built over data item values composed of aggregation hierarchies defined on each data attribute.

Definition 3.2: **Taxonomy.** Let $\mathcal{T}$=$\{t_1, t_2, \ldots, t_n\}$ be a set of attributes and Ω=$\{\Omega_1, \Omega_2, \ldots, \Omega_n\}$ its corresponding domains. Let ρ = $\{GH_1, \ldots, GH_m\}$ be a set of generalization hierarchies defined on $\mathcal{T}$. Each GH_i represents a predefined hierarchy of aggregations over values in Ω_i. GH_i leaves are all the values in Ω_i. Each non-leaf node in GH_i is an aggregation of all its children. The root node (denoted as $\perp$) aggregates all values for attribute t_i. A taxonomy $\Gamma \subseteq \rho$, is a forest of generalization hierarchies. Γ contains one generalization hierarchy $GH_i \in \rho$ for each attribute t_i in $\mathcal{T}$.

Taxonomies may be either analyst-provided or inferred by means of ad-hoc algorithms (e.g., [10], [14]). Figure 2 reports three examples of generalization hierarchies constructed over the attributes of the example timestamped datasets reported in Tables I(a) and I(b), exception for the time stamp attribute (i.e., the date) which, by construction, is not considered. According to Definition 3.2, the set of all the generalization hierarchies in Figure 2 is a taxonomy.

A pair *(t_i, $aggregation\_value_i$)*, where $aggregation\_value_i$ is a not-leaf node belonging to the generalization hierarchy associated with the i-th attribute t_i, is a generalized item. In the following I denote as $leaves(aggregation\_value_i) \subseteq \Omega_i$ the set of leaf nodes descendants of value $aggregation\_value_i$ in an arbitrary generalization hierarchy GH_i associated with the i-th data attribute. For instance, in the generalization hierarchy $GH_{location}$ shown in Figure 2(b), *(location, Italy)* is an example of generalized item whose set of leaves includes items *(location, Rome)* and *(location, Turin)*. Most relevant quality indexes that characterize a generalized item in terms of, respectively, a timestamped dataset (Cf. Definition 3.1) and a taxonomy (Cf. Definition 3.2) are its support and generalization level. The support of $(t_i, aggregation\_value_i)$ is defined as the sum of the observed frequencies of the corresponding leaves $leaves(aggregation\_value_i)$ in the source dataset. For instance, the support of the generalized item *(location, Italy)* in D_1 is $\frac{2}{5}$ as it covers 2 out of 5 records in D_1 (see Table I(a)). The generalization level classifies items based on their generalization grade in the corresponding taxonomy. The item level $L[(t_i, aggregation\_value_i)]$ is defined as the height of the subtree nested in GH_i and rooted in $aggregation\_value_i$ plus 1, i.e., it is the length of the longest downward path in GH_i to a leaf from that node plus 1. Consider again the generalization hierarchy $GH_{location}$ on the product sale location shown in Figure 2(b). The level of the generalized item *(location, Italy)* is 2 provided that the longest path on $GH_{location}$ from $Italy$ to a leaf node has length 1.

In the context of relational data, a (not generalized) k-itemset, i.e., an itemset of length k, is a set of k items, each one belonging to a distinct data attribute. Generalized itemsets represent high level recurrences hidden in the analyzed data. A generalized itemset of length k (i.e., a generalized k-itemset) is a set of generalized or not generalized items including at least one generalized item. A more formal definition follows.

Definition 3.3: **Generalized itemset.** Let $\mathcal{T}$=$\{t_1, t_2, \ldots, t_n\}$ be a set of attributes belonging to a timestamped structured dataset and $\mathcal{I}$ the enumeration of all the items in the corresponding dataset. Let Γ be a taxonomy over $\mathcal{T}$, and $\mathcal{E}$ the set of generalized items derived by all the generalization hierarchies in Γ. A generalized itemset Y is a subset of $\mathcal{I} \bigcup \mathcal{E}$ including at least one generalized item in $\mathcal{E}$. Each attribute $t_i \in \mathcal{T}$ may occur at most once in Y. For instance, $\{$*(product, T-shirt), (location, Italy)*$\}$ is a generalized 2-itemset, while $\{$*(product, T-shirt), (product, Jacket)*$\}$ does not.

To characterize generalized and not generalized itemsets, I extend previously discussed notions of item support and level to itemsets. To define the (generalized) itemset support, I first introduce the concept of (generalized) itemset matching and coverage set. A (generalized) itemset matches a given record if all its items either (i) belong to the record, or (ii) are ancestors of items that belong to the record.

Definition 3.4: **(Generalized) itemset matching.** Let D be a timestamped structured dataset and Γ a taxonomy built on D. A (generalized) itemset X matches an arbitrary record $r \in D$ if and only if for all (possibly generalized) items $x \in X$

1) $x \in \mathcal{I}$ (i.e., x is an item) and $x \in r$, or
2) $x \in \mathcal{E}$ (i.e., x is a generalized item) and $\exists\ i \in leaves(x)$ such that $i \in r$.

The coverage set $cov(X, D)$ of a (generalized) itemset X is given by the set of records in D matched by X [22].

Definition 3.5: **(Generalized) itemset support.** Let D be a timestamped structured dataset and Γ a taxonomy on D. The support of a generalized or not generalized itemset X is given by $\frac{|cov(X,D)|}{|D|}$.

Definition 3.6: **(Generalized) itemset level.** Let X=$\{(t_1, value_1), \ldots, (t_k, value_k)\}$ be a (generalized) itemset of length k. Its level $L[X]$ is the maximum item generalization level by considering items in X, i.e., $L[X] = max_{1 \leq j \leq k}\{L[(t_j, value_j)]\}$.

The generalization level of an itemset is affected by the highest generalized item level. Consider again the generalized itemset {*(product, T-shirt), (location, Italy)*}. Its support is $\frac{2}{5}$ in both datasets D_1 and D_2 (See Tables I(a) and I(b)). Since, according to the taxonomy reported in Figure 2, L[*(product, T-Shirt)*] = 1 and L[*(location, Italy)*] = 2 its generalization level is 2.

A generalized itemset may be an ancestor or a descendant of another one, according to a given taxonomy, as stated by the following definition.

Definition 3.7: **(Generalized) itemset ancestor and descendant.** Let $X, Y \subseteq \mathcal{I}$ be two different (generalized) k-itemsets and Γ be a taxonomy. X is an ancestor of Y on Γ, denoted as $X \in Anc[Y, \Gamma]$, if and only if for any (generalized) item $y_i \in Y$ exists a (generalized) item $x_i \in X$ such that x_i is an ancestor of y_i in GH_i. If X is an ancestor of Y, then Y is a descendant of X, denoted as $Y \in Desc[X, \Gamma]$.

Consider again the generalized itemset I = {*(product, T-shirt), (location, Italy)*}. Since, according to the taxonomy reported in Figure 2, $Italy$ is an ancestor of $Turin$, {*(product, T-shirt), (location, Turin)*} is an example of descendant of I.

The redundancy of an ancestor with respect to one of its descendants is defined in terms of their respective coverage sets.

Definition 3.8: **Ancestor redundancy.** Let D be a structured dataset and Γ a taxonomy. Let $X, Y \subseteq \mathcal{I}$ be two (generalized) k-itemsets such that $X \in Anc[Y, \Gamma]$. The redundancy $red(X, Y)$ of X with respect to Y is given by $\frac{|cov(X,D)|}{|cov(Y,D)|}$.

Theorem 3.1: The ancestors of a reference (generalized) itemset with minimal redundancy are the ones with minimal support.

Proof: Suppose that X_1,$X_2 \in Anc[Y, \Gamma]$ and $red(X_1, Y) > red(X_2, Y)$. Since $|cov(X_1, D)| > |cov(X_2, D)|$ it follows that $sup(X_1, D) > sup(X_2, D)$. ■

IV. Problem statement

This section formalizes the concepts of HIGEN (HIstory GENeralized Pattern) and NON-REDUNDANT HIGEN (Non-redundant HIstory GENeralized Pattern) as well as formally states the mining problem addressed by the paper.

The frequent (generalized) itemset mining problem [3] focuses on discovering, from the analyzed data, generalized and not generalized itemsets (Cf. Definition 3.3) whose support value is equal to or exceeds a minimum support threshold. Given an ordered sequence of timestamped structured datasets (Cf. Definition 3.1) relative to different time periods, a taxonomy (Cf. Definition 3.2), and a minimum support threshold, dynamic change mining [2], in the context of frequent itemsets, investigates the changes and the evolution of the extracted itemsets, in terms of

their main quality indexes (e.g., the support), from one time period to another. This paper extends the dynamic change mining problem, in the context of frequent itemsets, by exploiting frequent generalized itemsets to represent knowledge associated with infrequent patterns. In the following, I introduce the concepts of HIGEN and NON-REDUNDANT HIGEN.

a) The HIGEN*:* A HIGEN, associated with both a not generalized itemset it and an ordered sequence of timestamped datasets $\mathcal{D}$ = $\{D_1, D_2, \ldots, D_n\}$, is a ordered sequence of generalized itemsets $g_1, g_2, \ldots, g_n$ that represents the evolution of the knowledge covered by it in $\mathcal{D}$. Each g_i is a frequent (generalized) itemset extracted from D_i. g_i may be either (i) it, in case it is frequent in D_i with respect to the minimum support threshold, or (ii) one of the frequent generalizations of it in D_i characterized by minimum generalization level otherwise. A more formal definition of HIGEN follows.

Definition 4.1: HIGEN**.** Let $\mathcal{D}$=$\{D_1, D_2, \ldots, D_n\}$ be an ordered sequence of timestamped structured datasets and Γ be a taxonomy built over data items in $D_i \in \mathcal{D}\ \forall\ i$. Let it be a not generalized itemset and min_sup be a minimum support threshold. A HIGEN HG_{it}, associated with it, is an ordered sequence of itemsets or generalized itemsets $g_1, g_2, \ldots, g_n$ such that:

- if $sup(it, D_i) \geq min\_sup$ then g_i= it
- else g_i= git, where git is an ancestor of it, with respect to Γ, frequent in D_i and characterized by a minimum generalization level among the set of frequent ancestors of it, i.e., $git \in Anc[it, \Gamma] \wedge sup(git, D_i) \geq min\_sup$ and $\nexists\ git_2 \in Anc[it, \Gamma]$ such that $L[git, \Gamma] > L[git_2, \Gamma]$ and $sup(git_2, D_i) \geq min\_sup$

A HIGEN HG_{it} may be represented as a sequence

$$g_1\ \mathrm{R}_1\ g_2\ \mathrm{R}_2\ \ldots\ g_{n-1}\ \mathrm{R}_{n-1}\ g_n$$

where R_i is a relationship holding between (generalized) itemsets g_i and g_{i+1} and may be represented as $\nearrow$ if g_i is a descendant of g_{i+1}, $\searrow$ if g_i is an ancestor of g_{i+1} or $\rightsquigarrow$ if $L[g_i, \Gamma]$ = $L[g_{i+1}, \Gamma]$.

Table II(b) reports the set of HIGENs mined from the timestamped datasets D_1 and D_2 (See Tables I(a) and I(b)) by enforcing an absolute minimum support threshold equal to 2 and by exploiting the taxonomy reported in Figure 2. For instance, $\{$*(location, Paris)*$\}$ $\nearrow$ $\{$*(location, France)*$\}$ is a HIGEN associated with itemset $\{$*(location, Paris)*$\}$ given that $\{$*(location, Paris)*$\}$ is a descendant of $\{$*(location, France)*$\}$ and it is frequent in D_1 but infrequent in D_2. Consider now the time attribute in D_1 and D_2 (See Tables I(a) and I(b)) and the corresponding generalization hierarchy (see Figure 2(a)). $\{$*(time, from 5 p.m. to 6 p.m.)*$\}$ $\searrow$ $\{$*(time, 5 p.m.)*$\}$ is a HIGEN associated with itemset $\{$*(time, 5 p.m.)*$\}$ while $\{$*(time, p.m.)*$\}$ $\searrow$ $\{$*(time, 5 p.m.)*$\}$ does not as it exists an ancestor of $\{$*(time, 5 p.m.)*$\}$, i.e., $\{$*(time, from 5 p.m. to 6 p.m.)*$\}$, that is frequent in D_1 and such that L[$\{$*(time, from 5 p.m. to 6 p.m.)*$\}$, Γ] $<$ L[$\{$*(time, p.m.)*$\}$, Γ].

b) The NON-REDUNDANT HIGEN*:* Since a not generalized itemset it may have several ancestors characterized by the same generalization level, it trivially follows that an itemset may have many associated HIGENs. However,

analysts may prefer to look into a more concise set of temporal change patterns instead of the whole HIGEN set. Among the possible generalizations of an infrequent itemset belonging to the minimal abstraction level, the ones with minimal support are the generalizations that cover its knowledge with the minimal amount of redundancy (Cf. Theorem 3.1).

To reduce the number of considered patterns, the notion of NON-REDUNDANT HIGEN is introduced. Among the set of HIGENs relative to the same generalized itemset it, the NON-REDUNDANT HIGENs are the HIGENs that include generalized itemsets with minimal redundancy, i.e., the ones with minimal support, in case the reference itemset becomes infrequent in a certain time period.

Definition 4.2: NON-REDUNDANT HIGEN**.** Let $\mathcal{D}$ be an ordered sequence of timestamped structured datasets and Γ a taxonomy built over data items in $D_i \in \mathcal{D} \ \forall \ i$. A NON-REDUNDANT HIGEN SHG_{it}, associated with it, is a HIGEN composed of an ordered sequence of itemsets or generalized itemsets $g_1, g_2, \ldots, g_n$ such that for each generalized itemset g_i in SHG_{it} its redundancy with respect to it is minimal, i.e., it does not exist any frequent ancestor g_i^* of it such that $sup(g_i^*, D_i) < sup(g_i, D_i)$.

From the above definition, it trivially follows that the number of NON-REDUNDANT HIGENs associated with each not generalized itemset it is lower than the number of the corresponding HIGENs. Consider, for instance, an item *(time, 5:35 p.m.)* and two of its possible generalizations *(time, from 5 p.m. to 6 p.m.)* and *(time, from 5:30 p.m. to 6:30 p.m.)* belonging to the same generalization level. Suppose that *(time, 5:35 p.m.)* $\nearrow$ *(time, from 5 p.m. to 6 p.m.)* and *(time, 5:35 p.m.)* $\nearrow$ *(time, from 5:30 p.m. to 6:30 p.m.)* are both HIGENs, according to Definition 4.1. Among them, the analyst may prefer the one that covers the same knowledge supported by the item *(time, 5:35 p.m.)* with a minimal amount of redundancy. If, for instance, the support of *(time, from 5 p.m. to 6 p.m.)* is strictly lower than the support of *(time, from 5:30 p.m. to 6:30 p.m.)* in the second time period the former HIGEN is selected as NON-REDUNDANT HIGEN. An empirical evaluation, performed on synthetic datasets, of the average number of frequent generalizations, with minimum generalization level and support value, of each infrequent itemset is reported in Section VI-C.

c) Problem statement: Given an ordered set of timestamped structured datasets, a taxonomy Γ, and a minimum support threshold min_sup, this paper addresses the problem of mining all HIGENs, according to Definitions 4.1.

To efficiently accomplish the HIGEN extraction task, in the next section I present the HIGEN MINER (HIstory GENeralized Pattern MINER). The main HIGEN MINER algorithm modifications needed to address the NON-REDUNDANT HIGEN extraction (Cf. Definition 4.2) are discussed as well.

V. THE HIGEN MINER ALGORITHM

The HIGEN MINER (HIstory GENeralized Pattern MINER) algorithm addresses the extraction of the HIGENs, according to Definition 4.1.

HIGEN mining may be addressed by means of a postprocessing step after performing the traditional generalized itemset mining step [3], constrained by the minimum support threshold and driven by the input taxonomy, from

each timestamped dataset. However, this approach may become computationally expensive, especially at lower support thresholds, as it requires (i) generating all the possible item combinations by exhaustively evaluating the taxonomy, (ii) performing multiple taxonomy evaluations over the same pattern mined several times from different time periods, and (iii) selecting HIGENs by means of a, possibly time-consuming, postprocessing step.

To address the above issues, I propose a more efficient algorithm, called HIGEN MINER. It introduces the following expedients: (i) To avoid generating all the possible combinations, it adopts, similarly to [6], an Apriori-based support-driven generalized itemset mining approach, in which the generalization procedure is triggered on infrequent itemsets only. Unlike [6], the generalization process does not generate all possible ancestors of an infrequent itemset at any abstraction level, but it stops at the generalization level in which at least a frequent ancestor occurs, (ii) to prevent multiple taxonomy evaluations over the same pattern, the generalization process of each itemset is postponed after its support evaluation in all timestamped datasets and iteratively applied on infrequent generalized itemsets of increasing generalization level, and (iii) to reduce the extraction time, HIGEN generation is performed on-the-fly, without the need of an ad-hoc postprocessing step. Furthermore, a slightly modified version of the HIGEN MINER algorithm is proposed to address NON-REDUNDANT HIGEN extraction. A description of the main algorithm modifications is given in Section V-B.

As a drawback, the HIGEN MINER algorithm automatically selects the subset of generalized itemsets of interest at the cost of a higher number of dataset scans with respect to a traditional Apriori-based miner. Consider a timestamped datasets of n attributes and a taxonomy of height H_{max}, the HIGEN MINER algorithm requires up to $H_{max} \cdot n$ dataset scans, while a traditional Apriori-like miner requires up to n dataset scans. However, traditional mining approaches still require a postprocessing step to select the HIGENs of interest (Cf. Definition 4.1). The experimental evaluation, reported in Section VI, shows that the HIGEN MINER algorithm yields good performance, in terms of both pattern pruning selectivity, with respect to previous generalized itemset mining approaches (i.e., [3], [6]), and execution time. In Section V-A, a pseudo-code of the HIGEN MINER is reported and thoroughly described while, in Section V-B, the selection and categorization of the discovered HIGENs is addressed.

A. *The* HIGEN MINER *algorithm pseudo-code*

Algorithm 1 reports the pseudo-code of the HIGEN MINER. The HIGEN MINER algorithm iteratively extracts frequent generalized itemsets of increasing length from each timestamped dataset by following an Apriori-based level-wise approach and directly includes them into the HIGEN set. At an arbitrary iteration k, HIGEN MINER performs the following three steps: (i) k-itemset generation from each timestamped dataset in $\mathcal{D}$ (line 3), (ii) support counting and generalization of infrequent (generalized) k-itemsets of increasing generalization level (lines 6-37), (iii) generation of candidate itemsets of length $k+1$ by joining k-itemsets and infrequent candidate pruning (line 39). Since HIGEN MINER discovers HIGENs from relational datasets it adopts the well-known strategy to consider the structure of the input datasets to avoid generating combinations that do not comply with the relational data format. After being generated, frequent itemsets of length k are added to the corresponding HIGENs in the HG set (line 9), while infrequent ones are generalized by means of the taxonomy evaluation procedure (line 17). Given an infrequent

itemset c of level l and a taxonomy Γ, the taxonomy evaluation procedure generates a set of generalized itemsets of level $l+1$ by applying, on each item $(t_j, value_j)$ of c, the corresponding generalization hierarchy $GH_j \in \Gamma$ (see Definition 3.2). All the itemsets obtained by replacing one or more items in c with their generalized versions of level $l+1$ are generated and included into the Gen set (line 21). Finally, generalized itemset supports are computed by performing a dataset scan (line 26). Frequent generalizations of an infrequent candidate c, characterized by level $l+1$, are first added to the corresponding HIGEN set and then removed from the Gen set when their lower level infrequent descendants in each time period have been fully covered (lines 27- 32). In such a way, their further generalizations at higher abstraction levels are prevented. Notice that the taxonomy evaluation over an arbitrary candidate of length k is postponed when the support of all candidates of length k and generalization level l in each timestamped dataset is available. This approach allows triggering the generalization on distinct candidates of level l that are infrequent in at least one timestamped dataset in $\mathcal{D}$. The sequence of support values of an itemset that is infrequent in a given time period is store and reported provided that (i) it has at least a frequent generalization in the same time period, and (ii) it is frequent in at least one of the remaining time periods. The generalization procedure stops, at a certain level, when the Gen set is empty, i.e., when either the taxonomy evaluation procedure does not generate any new generalization or all the considered generalizations are frequent in each time period and, thus, have been pruned (line 30) to prevent further knowledge aggregations.

The HIGEN MINER algorithm ends the mining loop when the set of candidate itemsets is empty (line 40).

B. HIGEN *categorization and selection*

Domain experts are commonly in charge of looking into the discovered temporal change patterns to highlight most notable trends. To ease the domain expert validation task, a preliminary analysis of the set of extracted HIGENs may (i) categorize them based on their time-related trends, or (ii) prune them to select a subset of interest, i.e., the NON-REDUNDANT HIGENs.

HIGEN *categorization:* To better highlight HIGEN significance, HIGENs are categorized, based on their time-related trend, in: (i) *Stable* HIGEN*s*, i.e., HIGENs that include generalized itemsets belonging to the same generalization level, (ii) *monotonous* HIGEN*s*, i.e., HIGENs that include a sequence of generalized itemsets whose generalization level shows a monotonous trend, and (iii) *oscillatory* HIGEN*s*, i.e., HIGENs that include a sequence of generalized itemsets whose generalization level shows a variable and non-monotonous trend. Since, according to Definition 3.6, a generalized itemset of level l may have several generalizations of level $l+1$ and taxonomies may have unbalanced data item distributions, *stable* HIGENs may be further partitioned in: (i) *Strongly stable* HIGENs, i.e., stable HIGENs, in which items, appearing in its generalized itemsets and belonging to same data attribute, are characterized by the same generalization level, and (ii) *Weakly stable* HIGENs, i.e., stable HIGENs in which items, appearing in its generalized itemsets and belonging to the same attribute, may be characterized by different generalization levels. Examples of HIGENs, extracted from a real-life dataset, belonging to each category are reported in Section VI.

NON-REDUNDANT HIGEN *selection:* To reduce the amount of generated patterns, analysts may focus on the

set of NON-REDUNDANT HIGENs (Cf. Definition 4.2). Since they represent the subset of HIGENs whose generalizations cover the same knowledge covered by the reference infrequent itemset with minimal redundancy, they may be deemed relevant by domain experts. Notice that the extraction of the NON-REDUNDANT HIGENs may be accomplished by slightly modifying the HIGEN MINER algorithm (see Algorithm 1). A sketch of the main algorithm modifications follows. At each algorithm iteration, once the the list of Gen set is populated, the set of infrequent descendants of patterns in Gen is visited. A nested loop iterates on Gen in order of increasing support and selects its generalizations with minimal support. Finally, the set HG of discovered temporal change patterns is updated accordingly.

VI. EXPERIMENTAL RESULTS

I evaluated the HIGEN MINER (HIstory GENeralized Pattern MINER) algorithm by means of a large set of experiments addressing the following issues: (i) The usefulness and the characteristics of the HIGENs mined from a real-life dataset (Section VI-A), (ii) the pruning selectivity, in terms of the number of generalized itemsets extracted by HIGEN MINER with respect to previous generalized itemset mining algorithms, i.e., [3], [6] (Section VI-B), (iii) the performance, in terms of the number of extracted temporal patterns, of the HIGEN MINER algorithm (Section VI-C), and (iv) the scalability, in terms of execution time, of the proposed approach (Section VI-D). All the experiments were performed on a 3.2 GHz Pentium IV system with 2 GB RAM, running Ubuntu 8.04. The HIGEN MINER algorithm was implemented in the Python programming language [21].

A. Domain expert validation

To validate the significance of the proposed approach, a campaign of experiments has been conducted on a real-life dataset coming from a context-aware mobile system. The extracted HIGENs have been first processed, based on the procedure described in Section V-B. Next, they have been analyzed by a domain expert to assess their usefulness in the context of mobile user and service profiling.

This Section is organized as follows. Section VI-A.1 provides a brief description of the adopted dataset. Section VI-A.2 reports a selection of the discovered HIGENs while Section VI-A.3 describes the characteristics of the discovered HIGENs based on the proposed categorization (see Section V-B).

1) Real dataset: A real dataset, called *mDesktop*, is provided by a research hub, namely the Telecom Italia Lab. It collects contextual information about user application requests submitted, through mobile devices, over the time period of three months (i.e., from August to October). From the whole context data collection, I generated 3 different timestamped datasets, each one corresponding to a 1-month time period. Regarding privacy concerns related to real context data usage, please notice that (i) experimental data were collected from voluntary users that provide their whole informed consent on personal data treatment for this research project, and (ii) real user names were hidden throughout the paper to preserve identities. A more detailed description of the adopted data collection follows.

mDesktop dataset. The Telecom Italia mobile desktop application provides a wide range of services to users through mobile devices. Some of them provide recommendations to users on restaurants, museums, movies, and other entertainment activities. For instance, they allow end users to request for a recommendation (GET_REC service), enter a score ($VOTE$ service), or update a score for an entertainment center ($UPDATE$ service). Other services allow users to upload files, photos, or videos and share them with the other users. The dataset includes 1,197 user requests concerning file sharing and uploading, 5814 records concerning user recommendations, and 4487 records regarding other kinds of services (e.g., weather forecasts, SMS, and call requests). To perform HIGEN mining, a taxonomy including the following generalization hierarchies has been defined.

- **time stamp** $\rightarrow$ hour $\rightarrow$ time slot (two-hour time slots) $\rightarrow$ time slot (six-hour time slots) $\rightarrow$ day period (AM/PM)
- **service** $\rightarrow$ service category
- **latitude:longitude** $\rightarrow$ city $\rightarrow$ country
- **phone number** $\rightarrow$ call type (PERSONAL/BUSINESS)

In particular, service characterization is performed by exploiting the generalization hierarchy over the *Service* attribute domain reported in Figure 3.

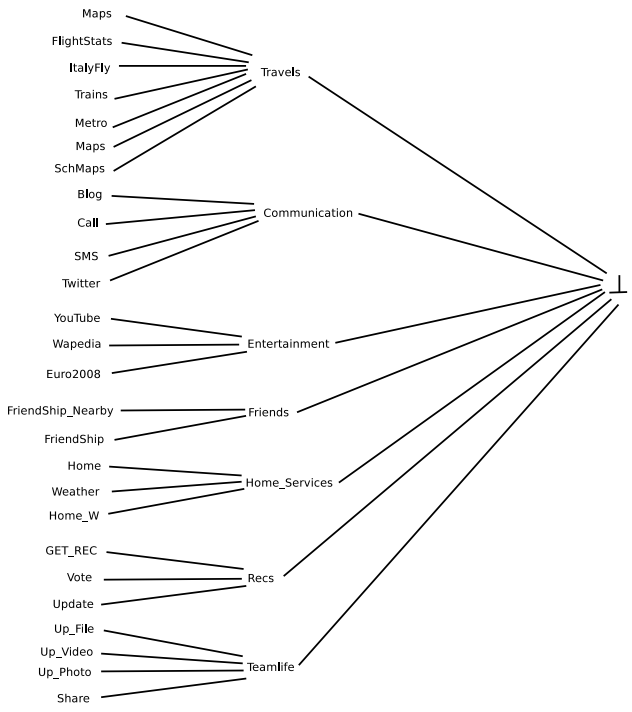


Fig. 3. Generalization hierarchy over the *Service* attribute

I also considered (i) different time periods, and (ii) different generalization hierarchies for the *time stamp* attribute. In particular, I also considered 1-week time periods and 4-hour and 8-hour time slots. Some remarkable examples of HIGENs mined by using these configurations are reported in Section VI-A.2.

2) Discovered pattern analysis: In the following, a selection of the discovered HIGENs is reported. Furthermore, possible scenarios of usage for the selected patterns suggested by the expert are discussed. The reported HIGENs are classified based on the categorization presented in Section V-B. Note that all the reported HIGENs, selected by the expert as most notable patterns, are also NON-REDUNDANT HIGENs. Indeed, the pruning of the not NON-REDUNDANT HIGENs did not relevantly affect the effectiveness of the knowledge discovery process.

a) **Examples of stable** HIGEN**s***: Stable* HIGENs are HIGENs whose generalized itemsets have the generalization level in common. Table III reports two examples of extracted strongly stable HIGENs, namely the HIGENs 1 and 2, and one example of weakly stable HIGEN, namely the HIGEN 3, generated by enforcing a minimum support threshold $min\_sup = 0.001\%$.

Both the reported strongly stable HIGENs concern the usage time of service *Twitter*, which provides mobile access to a famous online community. Since in HIGENs 1 and 2 all the items belonging to attribute *Service* are characterized by generalization level 1, while the ones belonging to attribute *Time* belong, respectively, to levels 2 and 3, the considered HIGENs are categorized as strongly stable HIGENs. The first strongly stable HIGEN states that requests for *Twitter* are frequently submitted between 9 p.m. and 10 p.m. within each considered month. The second one states a similar recurrence at a higher temporal abstraction level, i.e., between 7 p.m. and 13 p.m.. Their joint extraction implies that service *Twitter* is frequently requested from 9 p.m. to 10 p.m. within each month, but exists at least one hourly time slot between 7 p.m. and 13 p.m., different from [9 p.m., 10 p.m.], such that is infrequent within each considered month. I also analyzed the recurrences in service *Twitter* usage mined from shorter time periods. Even by considering weekly time periods, analogous trends are shown. Differently, the third stable HIGEN concerns the usage time of the whole application service by a user called John. Although, according to Definition 3.6, all its generalized itemsets have the same generalization level (i.e., 3), it is categorized as weakly stable HIGEN provided that the level of the items belonging to attribute *User* changes from month to month. This means that the information stored in the generalized itemsets changes its abstraction level, with respect to the input taxonomy, for a certain extent. However, since the overall itemset generalization level is affected by the mostly generalized attribute values (i.e., values associated with attribute *Time*), whose level does not vary over time, it still retains a certain degree of stability.

The information provided by stable HIGENs may be deemed relevant for service provisioning. For instance, the analyst may profitably exploit this knowledge for suiting bandwidth shaping to the actual user needs. In particular, he may either allocate dedicated bandwidth to frequently requested services at specific time slots or use free bandwidth at less congested time slots for network monitoring purposes.

b) **Examples of monotonous** HIGEN**s***: Monotonous* HIGEN s are HIGENs whose sequence of generalized itemsets is characterized by generalization levels with monotonous trend due to a steadfast and significant decrease/increase of the itemset support values, with respect to the support threshold, in consecutive time periods. Consider the monotonous HIGENs reported in Table III, which have been generated by enforcing a minimum support threshold $min\_sup = 0.001\%$.

HIGENs mined between August and October				
Time period	**Generalized itemset**	**Support (%)**	**Gen. level**	**Relationship**
Strongly Stable HIGEN 1				
August	{(Service, Twitter), (time, from 9 p.m. to 10 p.m.)}	0.051	2	$\rightsquigarrow$
September	{(Service, Twitter), (time, from 9 p.m. to 10 p.m.)}	0.020	2	$\rightsquigarrow$
October	{(Service, Twitter), (time, from 9 p.m. to 10 p.m.)}	0.022	2	-
Strongly Stable HIGEN 2				
August	{(Service, Twitter), (time, from 7 p.m. to 13 p.m.)}	0.071	3	$\rightsquigarrow$
September	{(Service, Twitter), (time, from 7 p.m. to 13 p.m.)}	0.031	3	$\rightsquigarrow$
October	{(Service, Twitter), (time, from 7 p.m. to 13 p.m.)}	0.044	3	-
Weakly Stable HIGEN 3				
August	{(User, Employee), (time, from 7 p.m. to 13 p.m.)}	0.051	3	$\rightsquigarrow$
September	{(User, Employee), (time, from 7 p.m. to 13 p.m.)}	0.062	3	$\rightsquigarrow$
October	{(User, John), (time, from 7 p.m. to 13 p.m.)}	0.024	3	-
Monotonous HIGEN 4				
August	{(Service, Home_W), (User, John)}	0.005	1	$\nearrow$
September	{(Service, Home_Services), (User, John)}	0.017	2	$\rightsquigarrow$
October	{(Service, Home_Services), (User, John)}	0.021	2	-
Monotonous HIGEN 5				
August	{(Service, Travels), (User, John)}	0.009	2	$\searrow$
September	{(Service, Trains), (User, John)}	0.008	1	$\rightsquigarrow$
October	{(Service, Trains), (User, John)}	0.007	1	-
Oscillatory HIGEN 6				
August	{(Service, Travels), (User, Terry)}	0.015	2	$\searrow$
September	{(Service, Maps), (User, Terry)}	0.053	1	$\nearrow$
October	{(Service, Travels), (User, Terry)}	0.017	2	-
Oscillatory HIGEN 7				
August	{(Service, MMS), (Time, from 13 p.m. to 14 p.m.)}	0.004	2	$\nearrow$
September	{(Service, MMS), (Time, from 13 p.m. to 15 p.m.)}	0.005	3	$\searrow$
October	{(Service, MMS), (Time, from 13 p.m. to 14 p.m.)}	0.002	2	-

TABLE III

mDesktop DATASET. EXAMPLES OF HIGENS. $min\_sup = 0.001\%$.

Consider service *Home_W* belonging to category *Home_Services*, which is targeted to provide facilities to users who are at home. The first monotonous HIGEN (i.e., HIGEN 4) shows a recurrence in user John *mDesktop* application service usage. Requests for service *Home_W* are frequently submitted in August, while become infrequent in both September ($sup = 0.0001\%$) and October ($sup = 0.0008\%$). Nevertheless, the corresponding category *Home_Services* remains frequent in all the considered months. Readers can notice that weak monotonicity holds as the corresponding itemset generalization level increases from August to September while remains constant from September to October. It is worth mentioning that, by considering weekly time periods separately within each month, the reference itemset

$\{$*(Service, Home_W), (User, John)*$\}$ generates a strongly stable HIGEN over August, while it generates oscillatory HIGENs in the next months. Differently, the monotonous HIGEN 5 highlights an opposite trend. It regards service *Trains*, which belongs to service category *Travels* and provides information on rail transports. User John appears less interested in service *Trains* in August (sup = 0.0002%), possibly due to holiday vacations, while the service increases its attractiveness in the following months.

The above information is deemed relevant by domain expert for both user and service profiling. The analyst could (i) shape service bandwidth depending on the time period, (ii) personalize user $John$ service promotions, and (iii) plan long-term promotions to counteract service usage decrease at certain times of the year. The extracted information may be also used for cross-selling purposes, i.e., by suggesting others services belonging to category *Home_Services* (e.g., service *Weather*) to either increase user interest in the service category or counteract user migration from occasional use services (i.e., classes of services, such as $Travels$, on which users focus their interest for short time periods) to commonly used services (e.g., service $Home\_W$).

c) **Examples of oscillatory** HIGENs*: Oscillatory* HIGENs are HIGENs whose sequence of generalized itemsets is characterized by generalization levels with variable and non-monotonous trend. Consider, for instance, the oscillatory HIGENs, reported in Table III, mined by enforcing a minimum support threshold equal to 0.01%.

The first oscillatory HIGEN (i.e., HIGEN 6) concerns service category $Travels$, which provides to users information about travels, and its service $Maps$, which allows browsing of geographical maps. User Terry seems interested in services belonging to category $Travels$ in each considered month. Nevertheless, in September, he focused his interest in service $Maps$, possibly due to work travels. Differently, the second oscillatory HIGEN (i.e., HIGEN 7) provides information about the usage time of service MMS. In August and October, the service MMS is frequently requested between 13 p.m. and 14 p.m., while in September user interest in service MMS in the same time slot decreases but still holds between 13 p.m. and 15 p.m. Notice that, by using a different 2-hour time stamp (between 12 p.m. and 14 p.m.) the temporal correlation does not hold anymore. By looking into the discovered oscillatory HIGENs, analysts may, for instance, investigate the provision of occasional use services (e.g., service $Maps$) and perform the following actions: (i) suggest complementary services (e.g., service $Flightstats$), (ii) plan promotions targeted to specific user profiles, and (iii) discover mostly used service parameters to automatically suggest (e.g., frequently requested GPS coordinates for service $Maps$).

3) Characteristics of the discovered HIGENs*:* The expert also analyzes the frequency of stable, monotonous, and oscillatory HIGENs inherent in a certain service category. To address this issue, he adopts the following three-step approach: (i) HIGEN mining from *mDesktop* by means of the HIGEN MINER algorithm, by enforcing a minimum support threshold $min\_sup = 0.001\%$, (ii) selection of the HIGENs whose generalized itemsets include items belonging to the *Service* attribute (i.e., service description), and (iii) HIGEN categorization based on the service categories reported in Figure 3. For instance, the monotonous HIGEN $\{(Service, SMS)\} \nearrow \{(Service, Communication)\}$ would be associated with category $Communication$. Table IV reports the distribution of the HIGENs within the service categories.

Service category	**Number of stable** HIGENs **(%)**			**Number of monotonous** HIGENs **(%)**	**Number of oscillatory** HIGENs **(%)**
	Weak	**Strong**	**Total**		
Travels	18	9	27	32	41
Communication	19	33	52	22	26
Entertainment	20	7	27	34	39
Friends	14	12	26	44	30
Home_Services	17	28	45	32	23
Recs	15	14	29	36	35
Teamlife	14	22	36	27	37

TABLE IV

mDesktop DATASET. HIGEN DISTRIBUTION. $min\_sup = 0.001\%$

Categories that include commonly used services (e.g., category $Communication$) are mainly represented by stable HIGENs (e.g., 52% of the HIGENs belonging to $Communication$ are stable against 22% of monotonous HIGENs and 26% of oscillatory HIGENs), while categories that group occasional use services are mostly covered by monotonous or oscillatory HIGENs. A significant percentage of strongly stable HIGENs characterizes service categories with a great regularity in their context of use (e.g., category $Home\_Services$). For these services, analysts may design long-term resource and promotion plans. Service categories with a significant percentage of weakly stable or monotonous HIGENs (e.g., *Entertainment* and *Friends*) are usually suitable for medium-term plans, as they show periodical (e.g., seasonal) variations of their context of usage. Finally, oscillatory HIGENs are often characterized by non-deterministic behaviors. For instance, notice that service category $Travels$ includes a significant percentage of oscillatory HIGENs as its usage changes significantly and unexpectedly from month to month.

B. HIGEN MINER *pruning selectivity*

I evaluated the pruning selectivity of the HIGEN MINER generalization procedure on synthetic datasets. To this aim, I compared the number of frequent itemset and generalized itemsets mined from each generated timestamped dataset with that extracted by the following generalized frequent itemset mining algorithms: (i) a traditional algorithm, i.e., Cumulate [3], which performs an exhaustive taxonomy evaluation by generating all possible frequent combinations of generalized and not generalized itemsets, and (ii) a recently proposed support-driven approach to itemset generalization, i.e., GENIO [6], which generates a generalized itemset only if it has at least an infrequent descendant (Cf. Definition 3.7). The set of experiments was performed on synthetic datasets generated by means of the TPC-H generator [26], whose main configuration settings follow.

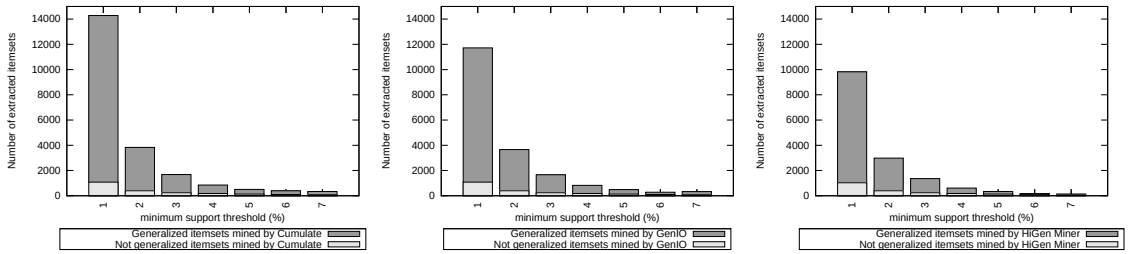


Fig. 4. Generalized and not generalized itemsets extracted from data-1

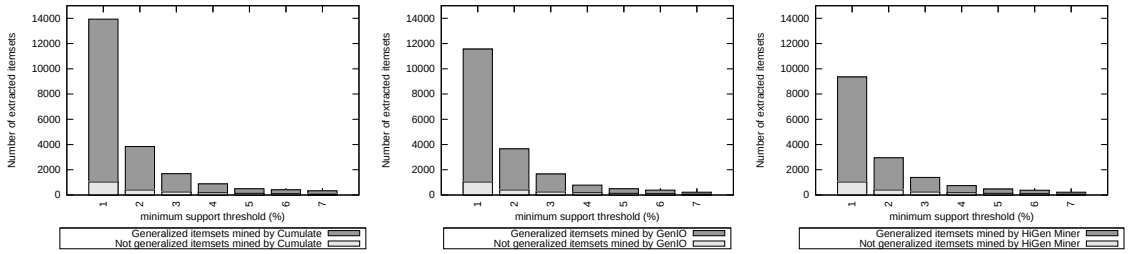


Fig. 5. Generalized and not generalized itemsets extracted from data-2

1) Synthetic datasets: The TPC-H data generator [26] consists of a suite of business-oriented ad-hoc database queries. By varying the scale factor parameter, files with different size could be generated. Generalized itemsets have been mined from a dataset generated from the lineitem table by setting a scale factor equal to 0.075 (i.e., around 450,000 records). Hierarchies on line item categorical attributes have been generated by using the part, nation, and region tables. To generate generalization hierarchies over the continuous attributes, I applied several data equi-depth discretization steps of finer granularities. The finest discretized values are considered as data item values and, thus, become the taxonomy leaves, while the coarser discretization procedures are exploited to aggregate the corresponding lower level values. More specifically, pairs of consecutive discretized values are aggregated in higher level items. To partition the whole dataset in three distinct time-related data collections I queried the source data by enforcing different constraints on the shipping date value (attribute $ShipDate$). More specifically, I partitioned line items shipped in the three following time periods: [1992-01-01, 1994-02-31], [1994-03-01, 1996-05-31] [1996-06-01, 1998-12-01]. For the sake of brevity, I will denote the corresponding datasets as data-1, data-2, and data-3 in the rest of this section.

2) Performance comparison: Since the enforcement of the minimum support threshold during the itemset mining step significantly affects the number of extracted itemsets, I performed different mining sessions, for all combinations of algorithms and datasets, by varying the minimum support threshold value. In Figures 4, 5, and 6 I plotted the number of itemsets mined, respectively, from data-1, data-2, and data-3. To test the HIGEN MINER algorithm, I considered the generated datasets in order of increasing shipment date interval. To better highlight the pruning selectivity on the cardinality of the mined generalized itemsets, I distinguished between generalized itemsets and

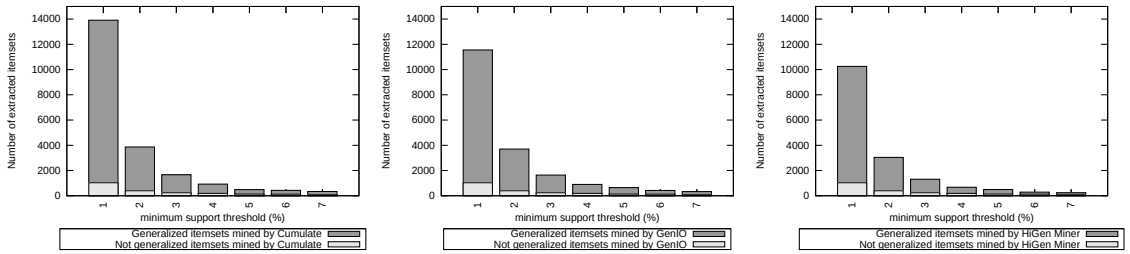


Fig. 6. Generalized and not generalized itemsets extracted from data-3

not.

For all the tested algorithms and datasets and for most of the settings of minimum support value, the percentage of extracted frequent itemsets including at least one generalized item is significant (i.e., higher than 60%). For both Cumulate and GENIO, the number of mined (generalized) itemsets significantly increases when lower minimum support values are enforced (e.g., 1%). Thus, the analysis of the temporal evolution of the extracted itemsets may require a computationally expensive postpruning step. In GENIO, infrequent items are aggregated during the extraction process and all frequent ancestors of an infrequent itemset are extracted. However, most of the extracted generalization seem redundant as they represent the same infrequent knowledge at increasing abstraction levels. The less redundant generalizations could be selected as the less redundant frequent representatives. By following this approach, the HIGEN MINER algorithm selects just the frequent ancestors with minimal generalization level (i.e., with minimal redundancy). The pruning selectivity, in terms of the number of extracted generalized itemsets, achieved by HIGEN MINER within each time period appears more evident when lower support thresholds (e.g., 2%) are enforced (see Figures 4, 5, and 6). In fact, when high support thresholds (e.g., 7%) are enforced, most of the frequent generalizations have already a high generalization level and, thus, the pruning effectiveness is less evident. Oppositely, when lower support thresholds are enforced (e.g., 2%), the extraction of a significant amount of redundant generalized itemsets, generated by both Cumulate and GENIO, is prevented. The pruning rate on the number of mined higher level (generalized) itemsets is between 5% and 15% with respect to GENIO for any support threshold value.

C. HIGEN MINER *performance analysis*

I analyzed the performance of the HIGEN MINER algorithm, in terms of the number extracted HIGENS and NON-REDUNDANT HIGENS, by analyzing the impact of the following factors: (i) minimum support threshold, (ii) number of time periods, and (iii) taxonomy characteristics. A set of experiments was performed on synthetic datasets generated by means of the TPC-H generator [26]. The baseline configuration used for data generation is similar to that described in Section VI-B.1.

a) Impact of the support threshold: I analyzed, on synthetic datasets, the impact of the minimum support threshold on the number of discovered HIGENS and NON-REDUNDANT HIGENS. In Figure 7 I report the number

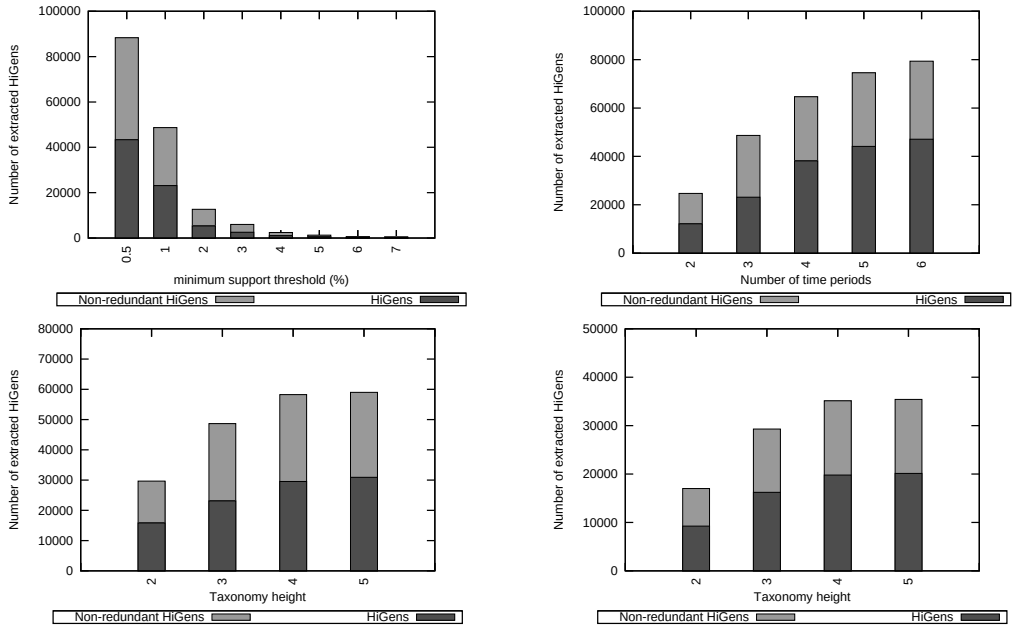


Fig. 7. HIGEN MINER: number of mined HIGENs and NON-REDUNDANT HIGENs. Default parameters: min_sup=1%, Taxonomy height = 3, Time periods = 3.

of HIGENs mined from data-1, data-2, and data-3 by varying the minimum support threshold. To test the HIGEN MINER algorithm, datasets are considered in order of increasing shipment date interval. To better highlight the pruning selectivity yielded by mining NON-REDUNDANT HIGENs, I distinguished between NON-REDUNDANT HIGENs and not.

As expected, the number of generated HIGENs increases more than linearly when the minimum support threshold decreases. The selection of the NON-REDUNDANT HIGENs significantly reduces the number of generated temporal change patterns. In fact, the number of possible generalizations with minimal generalization level of each infrequent itemset significantly affects the cardinality of the set of mined HIGENs. Let min_sup be a minimum support threshold. Let n be the number of considered time periods (i.e., the number of timestamped datasets) and let $|\{p_i\}|$ be the average number of not generalized patterns p_i that are frequent, with respect to min_sup, in an arbitrary time period at any abstraction level. Let $avg\_level\_sharing$ be the average number of frequent generalizations, with minimum generalization level, of an infrequent pattern. An estimation of the number of HIGENs ($|HiGen|$), mined by enforcing a support threshold min_sup, is:

$$|HiGen| \approx avg\_level\_sharing \cdot (n-1) \cdot |\{p_i\}| \tag{1}$$

By setting n = 3 and by approximating the $\{p_i\}$ set cardinality in Formula 1 as the greatest common divisor of the number of generalized itemsets extracted from data-1, data-2, and data-3, for each support threshold (see Figures 4, 5, and 6), I can compute the approximated value achieved by $avg\_level\_sharing$, for any support threshold, starting

from the results reported in Figure 7. The average number $avg\_level\_sharing$ of frequent generalizations, with minimal generalization level, of each infrequent itemset turns out to range from 2 to 4 for any support threshold. The achieved results are close to the expectations. By selecting the NON-REDUNDANT HIGENs, the reduction, in terms of the number of HIGENs, is greater than 35% for any support threshold. The obtained results depend on the considered data item distribution.

b) Impact of the number of time periods: I analyzed the impact of the number of considered time periods on the cardinality of the extracted patterns. To this aim, I uniformly partitioned the $ShipDate$ attribute domain of the lineitem table in an increasing number of intervals. Figure 7 reports the number of mined HIGENs and NON-REDUNDANT HIGENs. The cardinality of the discovered HIGENs grows when the number of time periods increases due to both the presence of new reference itemsets that are frequent, at any abstraction level, in at least one time period and the generation of multiple combinations of least generalized itemsets. As expected, the increase is less relevant when considering just NON-REDUNDANT HIGENs.

c) Impact of the taxonomy characteristics: I also analyzed, on synthetic datasets, the impact of the main taxonomy characteristics on the number of extracted HIGENs and NON-REDUNDANT HIGENs. To evaluate the impact of the taxonomy height, I generated ad-hoc taxonomies of increasing height by varying the number of discretization steps adopted for aggregating continuous data attribute values. To also evaluate the impact of different multiple-level data item distributions, Figures 7 reports the number of extracted temporal patterns, when varying the taxonomy height, by applying two representative discretization procedures, i.e., an equi-depth procedure [24] and an entropy-based one [13], on numeric data attributes. A similar trend is shown by both the tested item distributions. By using a taxonomy composed of only 2 levels many generalizations are still infrequent in some time periods and, thus, the corresponding HIGENs are not extracted. When the taxonomy height increases from 2 to 4 a significant number of reference infrequent itemsets becomes frequent at a higher generalization level and, thus, the number of extracted combinations relevantly grows. Further taxonomy height increments do not produce any significant increase of the HIGEN cardinality. Since the data distribution generated by the entropy-based discretization is sparser than that generated by the equi-depth procedure, it produces a smaller number of temporal correlations.

D. HIGEN MINER *scalability*

I also analyzed, on synthetic datasets, the scalability of the HIGEN MINER algorithm, in terms of the execution time. To analyze the scalability with the number of dataset records, I exploited the TPC-H data generator [26] and I varied the scale factor to generate datasets of increasing size, ranging from $150,000$ to $900,000$ records. To test the HIGEN MINER, I exploited the same configurations for the data generator already described in Section VI-B.1. Figure 8 plots the time spent in HIGEN mining for different support values. The computational complexity significantly increases for lower minimum support threshold values (e.g., 1%) due to the higher number of extracted itemsets. However, the execution time scales roughly linearly with the number of dataset records for any support threshold.

I also analyzed on synthetic datasets the scalability of the HIGEN MINER with both the taxonomy height and the

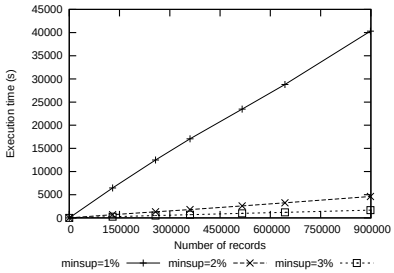


Fig. 8. Scalability of the HIGEN MINER algorithm. Time periods = 3. Taxonomy height = 3

number of time periods. Results, not reported here due to the lack of space, show that HIGEN MINER averagely scales more than linearly with both taxonomy height and number of time periods due to the non-linear growth of the number of considered (potentially relevant) combinations. However, the HIGEN MINER execution time is still acceptable even when coping with quite complex taxonomies evaluated over a number of time periods (e.g., around $20,000$ seconds with min_sup=1%, time periods = 6, taxonomy height = 5, number of records = $300,000$).

VII. CONCLUSIONS AND FUTURE WORK

This paper addresses the problem of change mining in the context of frequent itemsets. To represent the evolution of itemsets in different time periods without discarding relevant but rare knowledge due to minimum support threshold enforcement, it proposes to extract generalized itemsets characterized by minimal redundancy (i.e., minimum abstraction level) in case one itemset becomes infrequent in a certain time period. To this aim, two novel kinds of dynamic patterns, namely the HIGENS and the NON-REDUNDANT HIGENS, have been introduced. The usefulness of the proposed approach to support user and service profiling in a mobile context-aware environment has been validated by a domain expert.

Future works will address: (i) the automatic inference and use of multiple taxonomies from data coming from real application contexts (e.g., social network data), and (ii) the pushing of more complex HIGEN quality constraints into the mining process.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, pages 207–216, 1993.

[2] R. Agrawal and G. Psaila. Active data mining. In *Proceedings of the 1st international conference on knowledge discovery and data mining*, pages 3–8, New York, NY, USA, 1995. AAAI.

[3] R. Agrawal and R. Srikant. Mining generalized association rules. *VLDB'95*, pages 407–419, 1995.

[4] M. L. Antonie, O. R. Zaiane, and A. Coman. Application of data mining techniques for medical image classification. *MDM/KDD'01*, 2001.

[5] W.-H. Au and K. C. C. Chan. Mining changes in association rules: a fuzzy approach. *Fuzzy Sets Syst.*, 149:87–104, January 2005.

[6] E. Baralis, L. Cagliero, T. Cerquitelli, V. D'Elia, and P. Garza. Support driven opportunistic aggregation for generalized itemset extraction. In *IS 2010*, 2010.

[7] S. Baron, M. Spiliopoulou, and O. G?nther. Efficient monitoring of patterns in data mining environments. In L. Kalinichenko, R. Manthey, B. Thalheim, and U. Wloka, editors, *Advances in Databases and Information Systems*, volume 2798 of *Lecture Notes in Computer Science*, pages 253–265. Springer Berlin Heidelberg, 2003.

[8] M. Böttcher, D. Nauck, D. Ruta, and M. Spott. Towards a framework for change detection in data sets. In M. Bramer, F. Coenen, and A. Tuson, editors, *Research and Development in Intelligent Systems XXIII*, pages 115–128. Springer London, 2007.

[9] D. W.-L. Cheung, J. Han, V. Ng, and C. Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proceedings of the Twelfth International Conference on Data Engineering*, ICDE '96, pages 106–114, Washington, DC, USA, 1996. IEEE Computer Society.

[10] C. Clifton, R. Cooley, and J. Rennie. Topcat: Data mining for topic identification in a text corpus. In *In Proceedings of the 3rd European Conference of Principles and Practice of Knowledge Discovery in Databases*, 2002.

[11] G. Dong, J. Han, J. M. W. Lam, J. Pei, K. Wang, and W. Zou. Mining constrained gradients in large databases. *IEEE Trans. on Knowl. and Data Eng.*, 16:922–938, August 2004.

[12] G. Dong and J. Li. Mining border descriptions of emerging patterns from dataset pairs. *Knowl. Inf. Syst.*, 8:178–202, August 2005.

[13] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1029, 1993.

[14] S. C. Gates, W. Teiken, and K.-S. F. Cheng. Taxonomies by the numbers: building high-performance taxonomies. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 568–577, New York, NY, USA, 2005. ACM.

[15] P. Giannikopoulos, I. Varlamis, and M. Eirinaki. Mining frequent generalized patterns for web personalization in the presence of taxonomies. *IJDWM*, 6(1):58–76, 2010.

[16] J. Han and Y. Fu. Mining multiple-level association rules in large databases. *IEEE Transactions on knowledge and data engireering*, 11(7):798–805, 1999.

[17] J. Hipp, A. Myka, R. Wirth, and U. Guntzer. A new algorithm for faster mining of generalized association rules. *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD98)*, pages 74–82, 1998.

[18] T. Imieliski, L. Khachiyan, and A. Abdulghani. Cubegrades: Generalizing association rules. *Data Mining and Knowledge Discovery*, 6:219–257, 2002. 10.1023/A:1015417610840.

[19] B. Liu, W. Hsu, and Y. Ma. Discovering the set of fundamental rule changes. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 335–340, New York, NY, USA, 2001. ACM.

[20] B. Liu, Y. Ma, and R. Lee. Analyzing the interestingness of association rules from the temporal dimension. In *ICDM*, pages 377–384, 2001.

[21] Python. Python website, 2009.

[22] L. D. Raedt. Constraint-based pattern set mining. In *In SIAM International Conference on Data Mining*, pages 237–248, 2007.

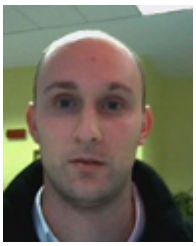
[23] B. Shen, M. Yao, Z. Wu, and Y. Gao. Mining dynamic association rules with comments. *Knowl. Inf. Syst.*, 23:73–98, April 2010.

[24] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 32–41, July 2002.

[25] Y. Tao and M. T. Özsu. Mining frequent itemsets in time-varying data streams. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1521–1524, New York, NY, USA, 2009. ACM.

[26] TPC-H. The TPC benchmark H. Transaction Processing Performance Council, 2009. http://www.tpc.org/tpch/default.asp.

[27] K. Verma and O. P. Vyas. Efficient calendar based temporal association rule. *SIGMOD Rec.*, 34:63–70, September 2005.



Luca Cagliero is a Ph.D. student at the Dipartimento di Automatica e Informatica of the Politecnico di Torino since January 2008. He holds a Master degree in Computer and Communication Networks from Politecnico di Torino. His current research interests include Data Mining and Database Systems. He has worked on user and service profiling in context-aware systems, ontology construction, and social network analysis by using itemset and association rule mining algorithms.

Algorithm 1 HIGEN MINER: HIstory GENeralized Pattern MINER

Input: ordered set of timestamped structured datasets $\mathcal{D}$=$\{D_1, D_2, \ldots, D_n\}$, minimum support threshold *min_sup*, taxonomy Γ

Output: set of HIGENs HG

k = 1 // Candidate length
$HG = \emptyset$
C_k = set of distinct k-itemsets in D
repeat
  for all $c \in C_k$ **do**
    scan D_i and count the support sup(c,D_i) $\forall$ $D_i \in \mathcal{D}$
  end for
  $L_k^i = \{$itemsets $c \in C_k |$ sup(c,D_i) $\geq$ *min_sup* for some $D_i \in D\}$
  HG = update_HIGEN_set(L_k^i, HG)
  l = 1 // Candidate generalization level
  $Gen = \emptyset$ // generalized itemset container
  repeat
    for all c in C_k of level l **do**
      $D_c^{\inf}$ = $\{D_i \in D \mid$ sup(c,D_i) $<$ *min_sup*$\}$ // datasets for which c is infrequent
      if $D_c^{\inf} \neq \emptyset$ **then**
        $gen(c)$ = set of new generalizations of itemset c of level $l+1$
        $gen(c)$ = taxonomy_evaluation(Θ, c)
        for all $gen \in gen(c)$ **do**
          $gen.Desc$ = c // Generalized itemset descendant
        end for
        Gen = $Gen \cup$ $gen(c)$
      end if
    end for
    if $Gen \neq \emptyset$ **then**
      for all $gen \in Gen$ **do**
        scan D_i and count the support of gen $\forall$ $D_i \in D_{gen.Desc}^{\inf}$
        for all $gen|$ sup(gen,D_i) $\geq$ *min_sup* for some $D_i \in D_{gen.Desc}^{\inf}$ **do**
          HG = update_HIGEN_set(gen, HG)
          if sup(gen,D_i) $\geq$ *min_sup* for all $D_i \in D_{gen.Desc}^{\inf}$ **then**
            remove gen from Gen
          end if
        end for
      end for
      C_k = $C_k \cup Gen$
    end if
    l = l + 1
  until Gen = $\emptyset$
  $k = k + 1$
  C_{k+1} = candidate_generation($\bigcup_i C_k^i$)
until C_k = $\emptyset$
return HG