POLITECNICO DI TORINO
Repository ISTITUZIONALE

Generalized association rule mining with constraints

Original
Generalized association rule mining with constraints / Baralis, ELENA MARIA; Cagliero, Luca; Cerquitelli, Tania; Garza, Paolo. - In: INFORMATION SCIENCES. - ISSN 0020-0255. - STAMPA. - 194:(2012), pp. 68-84. [10.1016/j.ins.2011.05.016]

Availability:
This version is available at: 11583/2460918 since:

Publisher:
ELSEVIER

Published
DOI:10.1016/j.ins.2011.05.016

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

12 October 2024

Generalized association rule mining with constraints

Elena Baralis, Luca Cagliero, Tania Cerquitelli

Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy

Paolo Garza*

Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133, Milano, Italy

Abstract

Generalized association rule extraction is a powerful tool to discover a high level view of the interesting patterns hidden in the analyzed data. However, since the patterns are extracted at any level of abstraction, the mined rule set may be too large to be effectively exploited in the decision making process. Thus, to discover valuable and interesting knowledge a post-processing step is usually required.

This paper presents the CoGAR framework to efficiently support constrained generalized association rule mining. The generalization process of CoGAR exploits a (user-provided) multiple-taxonomy to drive an opportunistic itemset generalization process, which prevents discarding relevant but infrequent knowledge by aggregating features at different granularity lev-

*Corresponding author. Tel.: +39 02 2399 3507.

Email addresses: `elena.baralis@polito.it` (Elena Baralis), `luca.cagliero@polito.it` (Luca Cagliero), `tania.cerquitelli@polito.it` (Tania Cerquitelli), `garza@elet.polimi.it` (Paolo Garza)

Preprint submitted to

els. Besides the traditional support and confidence constraints, two further constraints are enforced: (i) schema constraints and (ii) the opportunistic confidence constraint. Schema constraints allow the analyst to specify the structure of the patterns of interest and drive the itemset mining phase. The opportunistic confidence constraint, a new constraint proposed in this paper, allows us to discriminate between significant and redundant rules by analyzing similar rules belonging to different abstraction levels. This constraint is enforced during the rule generation step.

Experiments performed on real datasets collected in two different application domains show the effectiveness and the efficiency of the proposed framework in mining constrained generalized association rules.
Keywords: Generalized association rules, data mining algorithms, knowledge discovery, context-aware mining, network traffic analysis, mining with constraints.

---

1. Introduction

Generalized association rule extraction [2] is a widely used exploratory technique that allows discovering hidden correlations among data. By evaluating a taxonomy (is-a hierarchy) over data items, items can be aggregated according to different granularity levels. The aggregated concepts are called generalized items. Consider, for example, the jacket, coat, mittens, and hat items. Outerwear might be their corresponding generalized item. Thus, generalized items and itemsets provide a high level view of the patterns hidden in the analyzed data. They have been profitably exploited in different application domains (e.g., market-basket analysis [2, 18], network traffic domain [3])

Table 1: An example service invocation log dataset

user	user location	service	request time	date
Paolo	Milano	PhoneCall	9 a.m.	20/01/2010
Paolo	Milano	SMS	10 a.m.	13/01/2010
Paolo	Torino	WeatherForecast	18 p.m.	18/06/2010
Paolo	Carmagnola	WeatherForecast	21 p.m.	18/06/2010
Luca	Cuneo	SMS	20 p.m.	18/09/2010
Luca	Fossano	PhoneCall	14 p.m.	19/09/2010

to provide a high level abstraction of the mined knowledge.

The exhaustive evaluation of taxonomies may cause the extraction of a huge amount of patterns. Thus, a post-processing step is usually performed to select valuable patterns. Consider, for example, a customer care analyst interested in profiling service requests to provide personalized services and in analyzing system performance to guarantee high-quality services. To this aim, generalized association rules can be profitably exploited to highlight service usage (e.g., when services are mostly requested by a given set of users). However, a large number of uninteresting or redundant patterns may be discovered besides the interesting ones.

In this paper, we present the CoGAR framework to efficiently support constrained generalized association rule mining. It enforces (application-dependent) schema constraints and the new opportunistic confidence constraint during the generalized rule extraction process to improve both efficiency and effectiveness of the mining activity.

Consider again the previous customer care analyst and the running example dataset reported in Table 1. An association rule mining algorithm mines all the frequent rules by considering all the possible items combinations. However, the analyst is interested in analyzing exclusively the correlations (i) between user and service and (ii) between service and service request

time, while he is not interested in other kinds of correlations (e.g., between user and service request time, or other attribute combinations). To this aim, the schema constraints (1) $\{user, service\}$ and (2) $\{service, request\ time\}$ may drive the mining process to extract only the patterns of interest. For example, the rule $\{(user, Paolo)\} \Rightarrow \{(service, SMS)\}$ is mined, while the rule $\{(user, Paolo)\} \Rightarrow \{(user\ location, Torino)\}$ is not extracted. Instead of specifying a complex set of item constraints by means of boolean expressions (e.g., [18]), we compactly represent these kinds of item constraints at the attribute level to make them more easy to use.

To further prune the set of generated patterns, we also investigate the relationships holding among similar patterns at different abstraction levels. Consider again the generalized rule $\{(user, Paolo)\} \Rightarrow \{(service, SMS)\}$ and a taxonomy (i.e., is-a hierarchy) built over data items that aggregates services such as SMS and $PhoneCalls$ into the higher level category $Communication$. Suppose that it is extracted as it satisfies schema, minimum support, and confidence constraints. However, the higher level rule $\{(user, Paolo)\} \Rightarrow \{(service, Communication)\}$ is extracted as well. It has the same rule body of the former (i.e., $(user, Paolo)$) and its head $(service, Communication)$ is just a generalization of $(service, SMS)$. Thus, its support and confidence are definitely higher than the minimum support and confidence thresholds. However, the latter rule may be deemed redundant for decision making because its support and confidence increase is only due to the generalization of the rule head. The new *opportunistic confidence constraint* proposed in this paper allows us to prune these kinds of redundant rules to reduce the set of generated rules and facilitate the decision making process.

This paper presents the CoGAR (Constrained Generalized Association Rules) framework to efficiently mine generalized association rules by enforcing interesting constraints. Given a minimum support and confidence threshold, a set of schema constraints on the structure of interesting patterns, and the opportunistic confidence constraint, the mining task follows the traditional two-step approach [1]: (i) Extraction of the frequent generalized itemsets driven by support and schema constraints, and (ii) generation of the corresponding generalized rules, driven by both confidence and opportunistic confidence constraints. To prevent discarding relevant but infrequent knowledge, the itemset generalization process is driven by a (user-provided) taxonomy composed of aggregation hierarchies. An opportunistic aggregation approach [5] is exploited to evaluate the taxonomy, i.e., an itemset is generalized only if it is infrequent with respect to the minimum support threshold. Unlike previous works [5, 6, 8], the itemset mining algorithm used by CoGAR is also able to evaluate multiple hierarchies on the same attribute to analyze at the same time different facets (i.e., aggregation hierarchies) of the same feature. To effectively support end-users in exploring the extracted knowledge, mined patterns and multiple-taxonomies are stored in XML files that can be queried by means of XQuery [22].

Experimental results on real life datasets show the effectiveness of CoGAR in mining interesting patterns satisfying both user-specified schema constraints and the opportunistic confidence constraint, while experiments on synthetic datasets show the scalability of the mining algorithm.

The paper is organized as follows. Section 2 introduces definitions and notations exploited in the paper, while Section 3 presents an overview of the

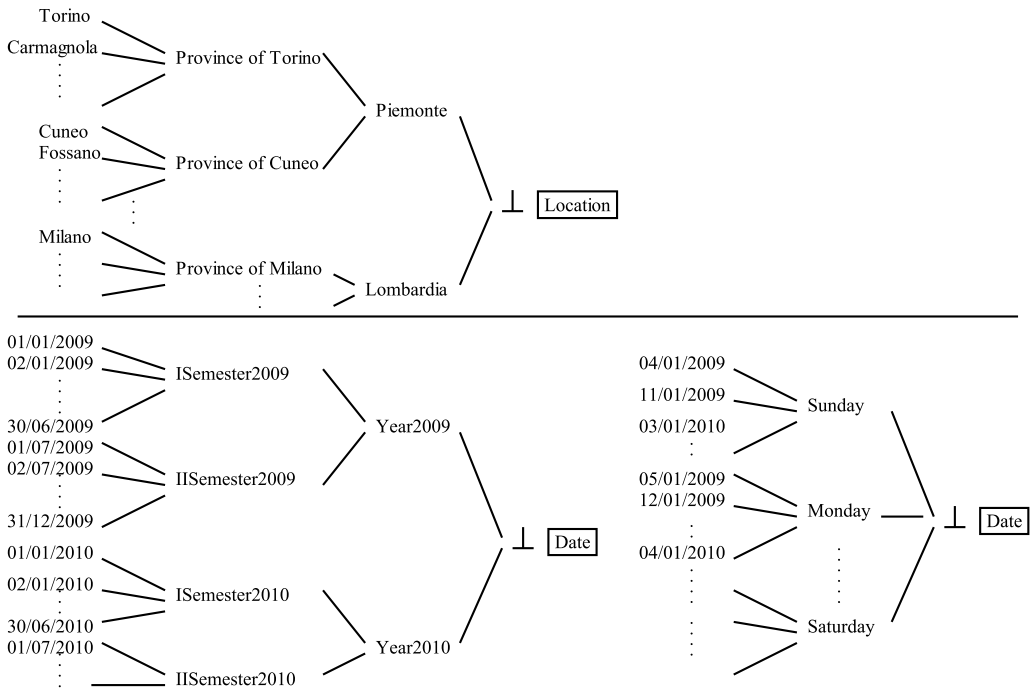


Figure 1: An example of multiple-taxonomy

CoGAR framework and describes the main features of its building blocks. Section 4 introduces the exploited mining algorithms. Section 5 discusses the experiments performed to validate the proposed approach. Section 6 discusses related works, while Section 7 draws conclusions and discusses future works.

2. Definitions and notations

In the following we introduce a set of notions and definitions preparatory to the constrained rule mining problem statement.

Let $\mathcal{T}=\{t_1, \ldots, t_n\}$ be a set of labels, called attributes, which describe

data features, and Ω=$\{\Omega_1, \ldots, \Omega_n\}$ be the corresponding attribute domains. An item is a pair $(t_i, value_i)$ which assigns value $value_i \in \Omega_i$ to attribute t_i. An itemset is a set of items.

A structured dataset D is a collection of records, where each record is a set of items and contains at most one item for each attribute in $\mathcal{T}$. When itemsets are mined from structured datasets, each attribute t_i may occur at most once in each itemset. In the rest of this section, we will exploit the running example dataset reported in Table 1 to facilitate the understanding of the introduced definitions.

Since we are interested in generalized rules, we suppose that a taxonomy (i.e., a is-a hierarchy) is defined on the items of the dataset. The taxonomy is used to aggregate items at different levels. In the running example, the taxonomy reported in Figure 1 will be considered. A set of aggregations over items belonging to each attribute may be represented as an aggregation tree.

Definition 1. ***Aggregation tree.*** *Let t_i be an attribute and Ω_i the corresponding domain. The aggregation tree AT_i is a tree whose leaves are values in Ω_i, while each non-leaf node in the tree is an aggregation of its children, and may be further generalized by its father. Nodes are mutually exclusive and collectively exhaustive. The root node of AT_i aggregates all values for attribute t_i.*

Each tree in Figure 1 is an aggregation tree defined over an attribute. Note that many aggregation trees can be defined for each attribute. In Figure 1 two aggregation trees, related to two different facets, are defined for the date attribute.

A forest of aggregation trees is called taxonomy.

Definition 2. ***Taxonomy.*** *Let* $\mathcal{T}$=$\{t_1, \ldots, t_n\}$ *be a set of attributes and* ρ=$\{AT_1, \ldots, AT_m\}$ *a set of aggregation trees defined on* $\mathcal{T}$*. A taxonomy* $\Gamma \subseteq \rho$*, is a forest of aggregation trees.* Γ *contains at most one aggregation tree* AT_i *for each attribute* t_i *in* $\mathcal{T}$*.*

We introduce the concept of multiple-taxonomy to allow different aggregation trees over the same attribute.

Definition 3. ***Multiple-taxonomy.*** *Let* $\mathcal{T}$=$\{t_1, \ldots, t_n\}$ *be a set of attributes. A multiple-taxonomy* $\Theta = \{\bigcup_k AT_{1k}, \ldots, \bigcup_j AT_{nj}\}$ *is a forest of aggregation trees, where* $\bigcup_j AT_{ij}$ *is the set of aggregation trees defined on attribute* t_i*.*

The taxonomy reported in Figure 1 is a multiple-taxonomy because the date attribute is associated with two aggregation trees.

The multiple-taxonomy is exploited to aggregate items.

Definition 4. ***Generalized item.*** *Let* t_i *be an attribute,* Ω_i *the corresponding domain, and* AT_i *an aggregation tree defined on* t_i*. A generalized item is a pair* $(t_i, expression_i)$*, where* $expression_i$ *is a non-leaf node in* AT_i *defining an aggregation value over values in* Ω_i*.*

The item $(date, ISemester2010)$ is a generalized item aggregating all the values of *date* in the range [01/01/2010-30/06/2010].

Definition 5. ***Generalized itemset.*** *Let* $\mathcal{I}$=$\{(t_1, value_1), (t_2, value_2), \ldots, (t_n, value_n)\}$ *be the enumeration of all the items in the structured dataset. Let* Θ *be a multiple-taxonomy on* $\mathcal{T}$*, and* $\mathcal{E}$=$\{(t_1, expression_1), (t_2, expression_2),$

..., $(t_m, expression_m)\}$ be the set of generalized items derived by all aggregation trees in Θ. A generalized itemset Y is a subset of $\mathcal{I} \bigcup \mathcal{E}$. Each attribute $t_i \in \mathcal{T}$ may occur at most once in Y.

$\{(location, Province\ of\ Torino), (date, ISemester2010)\}$ is an example generalized itemset of length 2.

To define generalized itemset support, we first introduce the concept of generalized itemset matching.

Definition 6. ***Generalized itemset matching.*** *Let D be a structured dataset and $\Theta = \{\bigcup_k AT_{1k}, \ldots, \bigcup_j AT_{nj}\}$ a multiple-taxonomy on D. Let $leaves(expression_i) \subseteq \Omega_i$ be the set of leaf nodes descendants of $expression_i$ in $AT_{ik} \in \Theta$. 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 $i \in r$*

Definition 7. ***Generalized itemset support.*** *Let D be a structured dataset and Θ a multiple-taxonomy on D. The support of a generalized itemset X is given by the number of records $r \in D$ matching X divided by the cardinality of D.*

The support of the itemset $\{(location, Province\ of\ Torino), (date, ISemester2010)\}$, mined from the running example dataset, is equal to $\frac{2}{6}$ because the number of records containing simultaneously a descendant of $(location, Province\ of\ Torino)$ and a descendant of $(date, ISemester2010)$ is equal to two.

The concept of generalized itemset descendant is introduced as it is exploited by one of the proposed constraints.

Definition 8. Generalized Itemset Descendant. *A (generalized) itemset X is a descendant of a generalized itemset Y with respect to a multiple-taxonomy Θ if (i) X and Y have the same length and (ii) for each item $y \in Y$ there exists an item $x \in X$ that is a descendant of y in Θ .*

For instance, the itemset $\{(location, Torino), (date, ISemester2010)\}$ is a descendant of the itemset $\{(location, Piemonte), (date, Year2010)\}$. In the following, we will denote as $Desc[Y]$ the set of descendants of Y.

By combining frequent generalized itemsets, generalized rules can be mined.

Definition 9. ***Generalized association rule.*** *A generalized association rule $X \Rightarrow Y$ is an association rule in which $X \cup Y$ is a generalized itemset.*

$\{(location, Piemonte) \Rightarrow (date, Year2010)\}$ is an example of generalized association rule.

Generalized rules are usually characterized by their support and confidence values. Given a (generalized) rule $X \Rightarrow Y$, its support is equal to the support of the itemset $X \cup Y$, while its confidence is defined as $conf(X \Rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)}$.

Two different types of constraints are enforced by our framework on generalized rules: (i) schema constraints and (ii) the opportunistic confidence constraint. In the following, their formal definitions are given.

Schema constraint. A schema constraint restricts the set of attributes that may appear in an itemset.

Definition 10. ***Schema constraint.*** *Let $\mathcal{T}$=$\{t_1, \ldots, t_n\}$ be a set of attributes. A schema constraint $Sc \subseteq \mathcal{T}$ of length k is a set of k distinct attributes. A generalized itemset X satisfies constraint Sc iff $attr(X) \subseteq Sc$, where $attr(X)$ is the set of attributes in X.*

An example of a schema constraint is $\{user, service\}$. The example schema constraint specifies that the only itemsets of interest are those of length 2 of type $\{user = value_{user}, service = value_{service}\}$ or those of length 1 related to one of the two attributes of interest ($\{user = value_{user}\}$ and $\{service = value_{service}\}$).

Definition 11. ***Schema constraint satisfaction.*** *Let X be a generalized itemset, $attr(X)$ the set of attributes in X, and $\mathcal{S}$=$\{Sc_1, \ldots, Sc_n\} \neq \emptyset$ a set of schema constraints. X satisfies constraints in $\mathcal{S}$ iff $attr(X) \subseteq Sc_i$ for at least an $i \in [1, n]$.*

When $\mathcal{S} = \emptyset$ no schema constraint is enforced. From the above definition, it trivially follows that if itemset X satisfies a constraint Sc, then any itemset $Y \subseteq X$ also satisfies Sc. This property can be profitably exploited to apply schema constraints during the itemset mining step.

For example the schema constraint $\{user, service\}$ is satisfied by the itemset $\{user = Paolo, service = SMS\}$ and also by the itemsets $\{user = Paolo\}$ and $\{service = SMS\}$.

The opportunistic confidence constraint. We propose a new constraint, called opportunistic confidence constraint, to prune a set of generalized rules that are considered useless. The opportunistic confidence constraint is based

on the confidence measure and compares each generalized rule r with a subset $R(r)_{desc}$ of its descendants. $R(r)_{desc}$ includes all the descendant rules of r such that the body is the same body of r while the head is a descendant of the head of r. Given a minimum confidence threshold, we consider a generalized rule r useful if it satisfies the enforced minimum confidence threshold while no rule in $R(r)_{desc}$ satisfies it.

Definition 12. ***Opportunistic confidence constraint.*** *Given a minimum confidence threshold* $minconf$*, an arbitrary generalized rule* $r : X \Rightarrow Y$ *satisfies the opportunistic confidence constraint iff (i)* $conf(r) \geq minconf$ *and (ii)* $\nexists\ r_d : X \Rightarrow Z$*, with* $Z \in Desc[Y]$*, such that* $conf(r_d) \geq minconf$*.*

Given a structured dataset D, a multiple-taxonomy Θ, a set of schema constraints $\mathcal{S}$, a minimum support threshold *minsup*, a minimum confidence threshold $minconf$, and the opportunistic confidence constraint, the CoGAR framework discovers generalized association rules satisfying all constraints.

3. The CoGAR Framework

The CoGAR (Constrained Generalized Association Rules) framework mines generalized rules satisfying both user-provided schema constraints and the opportunistic confidence constraint. It has been exploited to extract valuable knowledge from different application domains (e.g., network traffic analysis, mobile service analysis).

Figure 2 shows the building blocks of the CoGAR framework, which mainly performs three activities. (i) *Data integration and pre-processing.*

Data to be analyzed is usually provided by different and heterogeneous sources. Before performing the knowledge discovery process, data is cleaned by removing irrelevant and redundant information and integrated into a common data structure. (ii) *Generalized association rule mining with constraints.* The mining block is the core of CoGAR. It exploits a (user-provided) multiple-taxonomy to drive the mining of generalized association rules. To tailor the mining process to specific user targets, schema constraints are enforced during the itemset mining step. Moreover, the opportunistic confidence constraint is enforced to prune generalized rules that do not provide new interesting knowledge with respect to similar rules at lower abstraction levels. (iii) *Rule querying and selection.* To allow end-users to easily exploit the mined knowledge, both the set of generalized rules and the multiple-taxonomy are stored in an XML data repository, which can be queried by means of the XQuery language [22].

A more detailed description of the functionalities of each CoGAR architectural block follows.

3.1. Data integration and preprocessing

This block collects data coming from different sources, integrates them, and applies preprocessing tasks (e.g., data discretization) to transform data into a common data structure. During the preprocessing phase, data cleaning and redundant data pruning may be performed. Finally, preprocessed data are stored in a common data repository. Independently of the domain, the data integration step is based on a global as view (GAV) approach. A relation global view is defined on the available sources by means of semantic mappings between the schemata of the sources and the schema of the global view.

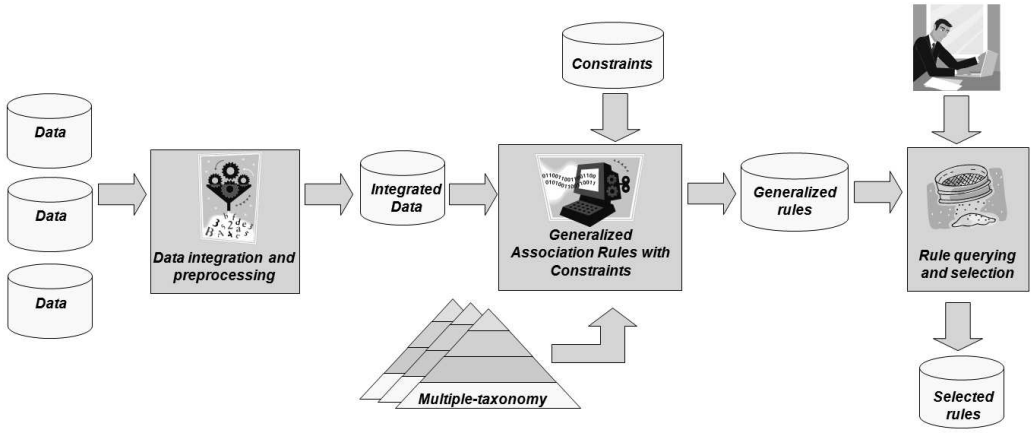


Figure 2: The CoGAR Framework Architecture

The global view provides an integrated view that is exploited by the mining block. The semantic mappings, defined by a domain expert, depend on the application domain.

Consider a context-aware mobile application with two data sources. The first source stores the positions of users, while the second source stores user requests. In the first data source the user position can be either a GPS coordinate or a mobile phone cell, depending on the used device (e.g., laptops, mobile phones). A proper data transformation is needed to obtain the same value when the position of the user is semantically the same, independently of the original format. To obtain a uniform representation, each position can be, for example, mapped to the corresponding city. This semantic mapping allows obtaining a uniform representation of user locations in the global view defined on the original data sources. To obtain a global view integrating user positions and service requests, the two data sources must be joined by exploiting the time information available in both sources. However, also in this

case a proper mapping must be defined to obtain a uniform representation of the time information as data sources exploit different formats.

3.2. Generalized association rule mining with constraints

Given the common data repository generated by the first block of CoGAR, a multiple-taxonomy, schema constraints, and the opportunistic confidence constraint this block performs the extraction of frequent generalized association rules satisfying constraints. The extracted rules are stored in an XML repository. The mining task follows the usual two-step approach [1]: (i) Extraction of frequent generalized itemsets and (ii) generation of rules.

To improve the efficiency of the mining task, schema constraints should be enforced as soon as possible. We propose an efficient itemset mining algorithm (i.e., CI-Miner) that directly mines generalized itemsets by pushing schema constraints into the itemset mining step. Generalized association rules satisfying the same schema constraints may be mined by applying any traditional rule mining algorithm on the extracted itemsets. We used our implementation of the traditional rule mining procedure proposed in [1]. Finally, a post-processing step is applied to enforce the opportunistic confidence constraint.

Section 4 reports more detailed information about the used mining algorithms.

3.3. Rule querying and selection

The mining block exclusively extracts rules satisfying the enforced constraints. Further exploration may allow end-users to focus their attention on specific subsets of rules depending on their goals. The rule querying and

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT ruleSet (rule*)>
<!ELEMENT rule (measure+, body, head)>
<!ELEMENT measure (#PCDATA)>
<!ATTLIST measure name CDATA #REQUIRED>
<!ELEMENT body (item+)>
<!ATTLIST body length CDATA #REQUIRED>
<!ELEMENT head (item+)>
<!ATTLIST head length CDATA #REQUIRED>
<!ELEMENT item EMPTY>
<!ATTLIST item attributeName CDATA #REQUIRED attributeValue CDATA #REQUIRED>
```

Figure 3: DTD of the XML document for association rule representation

selection block of CoGAR allows end-users to retrieve subsets of rules, or ranking them, according to their actual analysis target. We use XML to store both the extracted rules and the exploited multiple-taxonomy, while XQuery [22] is adopted to query and retrieve the rules of interest.

The DTD reported in Figure 3 describes the schema of the XML documents used to represent the mined rule sets. The `rule` element is used to represent rules, while the `body` and the `head` elements respectively represent the antecedent and the consequent of the rules. To store the value of the measures of interest the `measure` element is used. An XML document representing a rule set including two simple rules is reported in Figure 4.

The proposed XML rule representation is easily and efficiently queryable by means of the XQuery language [22]. Figure 5 reports an example query, which retrieves all the rules that include the item (*user*, *Paolo*) in their body and sorts them by descending confidence. According to user interests, similar queries may be easily written by the end-users of CoGAR.

Also the exploited taxonomy is represented by means of XML files. Figure 6 represents the DTD associated with the XML files used to store taxonomies, while Figure 7 represents a part of an example taxonomy. For each

```
<ruleSet>
   <rule>
      <measure name="support">5.8</measure> <measure name="confidence">50.5</measure>
      <body length="1">
         <item attributeName="user">Paolo</item>
      </body>
      <head length="2">
         <item attributeName="date">2008-12-24</item>
         <item attributeName="hour">13:24</item>
      </head>
   </rule>
   <rule>
      <measure name="support">7.5</measure> <measure name="confidence">30.0</measure>
      <body length="2">
         <item attributeName="user">Tania</item>
         <item attributeName="date">2008-11-02</item>
      </body>
      <head length="1">
         <item attributeName="service">Chat</attribute>
      </head>
   </rule>
</ruleSet>
```

Figure 4: An example of an XML document representing a rule set including two rules

```
<resultSet> { for $rule in doc("MinedRuleSet.xml")/ruleSet/rule
              where exists($rule/body/item[@attributeName="user" and @attributeValue="Paolo"])
              order by $rule/measure[@name="confidence"] descending
              return $rule) } </resultSet>
```

Figure 5: Selection of the rules including the item (*user*, *Paolo*) in the rule body, sorted by descending confidence

generalized item the `listOfChildren` element is used to represent its children. Each child item can be either a generalized or not generalized item. Figure 7 shows part of a taxonomy composed of two aggregation trees. The first part of Figure 7 reports an aggregation tree defined on the date attribute (dates are aggregate in months, semesters, and years) while the second part defines an aggregation over the service attribute.

The XML representation of the used taxonomy allows performing more complex queries by considering contemporaneously the XML file representing the mined rules and the one representing the taxonomy. For example, by

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT taxonomy (item+)>
<!ELEMENT item (listOfChildren*)>
<!ATTLIST item attributeName CDATA #REQUIRED attributeValue CDATA #REQUIRED>
<!ELEMENT listOfChildren (item+)>
```

Figure 6: DTD of the XML document for taxonomy representation

```
<taxonomy>
   <item attributeName="date" attributeValue="Year2010">
      <listOfChildren>
         <item attributeName="date" attributeValue="FirstSemester2010">
            <listOfChildren>
               <item attributeName="date" attributeValue="January2010"/>
               <item attributeName="date" attributeValue="February2010"/>
               .....
            </listOfChildren>
         </item>
         <item attributeName="date" attributeValue="SecondSemester2010">
            <listOfChildren>
               <item attributeName="date" attributeValue="July2010"/>
               .....
            </listOfChildren>
         </item>
      <listOfChildren>
   </item> .....
   <item attributeName="service" attributeValue="Communication">
      <listOfChildren>
         <item attributeName="service" attributeValue="PhoneCall"/>
         <item attributeName="service" attributeValue="SMS"/>
      <listOfChildren>
   </item> .....
</taxonomy>
```

Figure 7: An example of an XML document representing a taxonomy

using the two XML files, it is possible to select all the rules containing a descendant of the generalized item (*service*, *Communication*) in the rule consequent. The corresponding XQuery is reported in Figure 8.

4. Mining algorithms

The mining block of CoGAR is based on three components: (i) a schema constrained itemset mining algorithm (CI-Miner), (ii) a rule mining algo-

```
<resultSet> {
  for $rule in doc("MinedRuleSet.xml")/ruleSet/rule
  where $rule/head/@length="1"
  and exists(for $item in doc("Taxonomy.xml")//item[@attributeName="service" and
             @attributeValue="Communication"]//item
             where $item/@attributeName=$rule/head/item/@attributeName
             and $item/@attributeValue=$rule/head/item/@attributeValue
             return $item)
  return $rule } </resultSet>
```

Figure 8: Selection of the rules including a descendant of the item (*service*, *Communication*) in the rule head

rithm (RuleGen), and (iii) a post-processing filtering algorithm (CR-Filter). These three algorithms are sequentially invoked.

The first applied algorithm is CI-Miner. It extends the GenIO algorithm [5] by pushing the schema constraint into the itemset mining process. Given a structured dataset D, a multiple-taxonomy Θ, a minimum support threshold *minsup*, and a set of schema constraints $\mathcal{S}$, the CI-Miner algorithm extracts frequent generalized itemsets satisfying constraints in $\mathcal{S}$ by means of an opportunistic approach [5]. The opportunistic approach generalizes an itemset only if it is infrequent with respect to the minimum support threshold. It exploits a lazy taxonomy evaluation, which is triggered on infrequent itemsets only. Hence, an arbitrary frequent generalized itemset Y is extracted by CI-Miner iff (i) there exists at least a constraint $Sc \in \mathcal{S}$ such that $attr(Y) \subseteq Sc$ and (ii) Y has at least an infrequent descendant.

CI-Miner iteratively generates generalized itemsets by means of a level-wise approach (i.e., it is an Apriori-like approach). In an arbitrary iteration k, CI-Miner performs two steps: (i) Support counting and selection of frequent itemsets with length equal to k and (ii) generation of candidate itemsets of length $k+1$ by joining k-itemsets. However, differently from traditional algorithms (e.g., [1]), at each iteration CI-Miner enforces a set of schema

constraints $\mathcal{S}$ to prune the set of candidate itemsets. Only the candidate itemsets satisfying at least one of the enforced schema constraints are considered in the following iterations. This allows pruning useless candidates. CI-Miner also exploits the maximum schema constraint length to break the Apriori-like mining process earlier. In particular, itemsets longer than the maximum enforced constraint length are not extracted as they do not satisfy any constraint in $\mathcal{S}$.

The set of generalized itemsets mined by CI-Miner is exploited by our implementation of the traditional rule mining procedure proposed in [1] (denoted as RuleGen in the following). Given the set of frequent generalized itemsets and a minimum confidence threshold ($minconf$), the rule mining procedure generates the set of generalized association rules with a confidence value at least equal to $minconf$. The mined set will be denoted as R throughout the section.

Finally, the opportunistic confidence constraint is enforced on R by a post-processing algorithm, called CR-Filter. Algorithm 1 reports the pseudocode of CR-Filter. To check if an arbitrary rule r satisfies the opportunistic confidence constraint, r must be exclusively compared with the set of rules having its own antecedent. This property is exploited by CR-Filter to enforce the opportunistic confidence constraint. Hence, rules are initially partitioned by considering their antecedent (lines 1-3). Then, the algorithm considers one rule set R_X at a time (lines 4-9) and checks which rules in R_X satisfy the opportunistic confidence constraint (lines 5-7). The initial partitioning of the rule set significantly reduces the number of comparisons.

Algorithm 1 CR-Filter: Constrained Rule Filtering

Input: set of rules R,multiple-taxonomy Θ
Output: $R_{selected}$, set of generalized rules satisfying the opportunistic confidence constraint

/**Rule partitioning by considering their antecedent**/
for all itemset X in $\{X|r : X \Rightarrow Y \in R\}$ **do**
  $R_X = \{r|r \in R$ and $r.antecedent = X\}$ /*R_X *is the set of rules with the itemset* X *as antecedent**/
end for

/**Enforcement of the opportunistic confidence constraint**/
for all rule set R_X **do**
  for all rule r in R_X **do**
    delete r from R_X if $\exists r_l \in R_X$ such that $r_l.consequent$ is a descendant of $r.consequent$
  end for
  $R_{selected}$=$R_{selected} \cup R_X$
end for
return $R_{selected}$

5. Experimental results

We evaluated the CoGAR framework by means of a large set of experiments addressing the following issues: (i) The effectiveness of the proposed approach in mining valuable knowledge from different application domains (Section 5.2), (ii) the effect of the enforced constraints on the set of mined generalization rules (Section 5.3), and (iii) the scalability of the proposed generalized rule miner, in terms of execution time on synthetic datasets, with respect to (a) the number of records, and (b) the taxonomy height (Section 5.4).

5.1. Experimental settings

Experiments were performed on both real and synthetic datasets. Two real datasets collected from the context-aware domain and the network traffic domain were used. In Sections 5.1.1 and 5.1.2 the main characteristics of these datasets and the set of enforced schema constraints are reported. The characteristics of the synthetic datasets, used to analyze the scalability of the proposed approach, are described in Section 5.4.

All the experiments were performed on a 2.66 GHz Pentium IV system with 8 GB RAM, running Ubuntu Release 9.10. The CoGAR framework was implemented in the Python programming language [17].

5.1.1. Context-aware dataset

Telecom Italia Lab[1] provided us the *Recs* dataset containing a set of requests submitted to a real context-aware service provider system (the *Recs* system). The *Recs* system provides recommendations to mobile device users on restaurants, museums, movies, and other entertainment activities. Each user can request a recommendation (GET_REC service), enter a score (VOTE service), or update a score (UPDATE_VOTE service) for an entertainment activity. The analyzed dataset was obtained by logging the requests of 20 users and their locations over a time period of three months. The dataset contains 5814 records (i.e., requests). Each record is characterized by the request type, the parameters of the request, the user and its context information (user location, date, and time). To perform generalized rules mining, a single taxonomy including the following aggregation trees has been initially defined.

- date → month → trimester → year
- timestamp → hour → timeslot (two hours timeslots)→ day period (AM/PM)
- latitude:longitude → city → country

The usage of a multiple-taxonomy is discussed in Section 5.2.3.

[1]TILab is the Telecom Italia Group research hub

Table 2: Recs dataset: enforced schema constraints

Constraint	Rule Example
$\{user, date, time\}$	$\{(user, John)\} \Rightarrow \{(date, June), (time, morning)\}$
$\{user, time, place\}$	$\{(user, John)\} \Rightarrow \{(time, morning), (place, office)\}$
$\{user, time, param\}$	$\{(user, John), (time, morning)\} \Rightarrow \{(param, OUT)\}$
$\{user, date, param\}$	$\{(user, John), (date, winter)\} \Rightarrow \{(param, OUT)\}$
$\{user, date, place\}$	$\{(user, John)\} \Rightarrow \{(date, winter), (place, office)\}$
$\{user, place, param\}$	$\{(user, John), (place, office)\} \Rightarrow \{(param, OUT)\}$
$\{user, service, time\}$	$\{(user, John), (service, CALL)\} \Rightarrow \{(time, 2-6p.m.)\}$
$\{user, service, date\}$	$\{(user, John), (service, CALL)\} \Rightarrow \{(date, December)\}$
$\{user, service, place\}$	$\{(user, John), (service, CALL)\} \Rightarrow \{(place, office)\}$
$\{user, service, param\}$	$\{(user, John), (service, CALL)\} \Rightarrow \{(param, OUT)\}$
$\{service, date, time\}$	$\{(service, CALL), (date, winter), (time, afternoon)\}$
$\{service, place, time\}$	$\{(service, WEATHER)\} \Rightarrow \{(place, home), (time, evening)\}$
$\{service, place, date\}$	$\{(service, WEATHER), (place, home)\} \Rightarrow \{(date, summer)\}$
$\{service, param, date\}$	$\{(service, CALL)\} \Rightarrow \{(param, OUT), (date, week-end)\}$
$\{service, param, time\}$	$\{(service, CALL)\} \Rightarrow \{(param, OUT), (time, afternoon)\}$
$\{service, place, param\}$	$\{(service, WEATHER), (place, home)\} \Rightarrow \{(param, TODAY)\}$

Enforced schema constraints. A domain expert in charge of service provisioning provided us the schema constraints reported in Table 2, together with examples of compliant rules. He was interested in profiling users and services. Thus, constraints include the user and/or the service attribute. The user attribute is exploited to characterize the user behavior, while the service attribute is exploited to profile service usage. Additional information deemed relevant was the user/service context information (e.g., service invocation date and time, user location).

5.1.2. Network traffic dataset

NetCapture is a network traffic dataset obtained by performing different capture sessions with the open-source Network Analyzer tool [13] on a backbone link of our campus network. Captured traffic has been aggregated in traffic flows (i.e., records summarizing a group of similar and temporally contiguous packets). Each flow is characterized by six attributes: Source IP address, destination IP address, source port, destination port, flow size (i.e.,

the size of the flow expressed in byte), and number of IP packets aggregated in that flow. The *NetCapture* dataset is characterized by $16,783$ records.

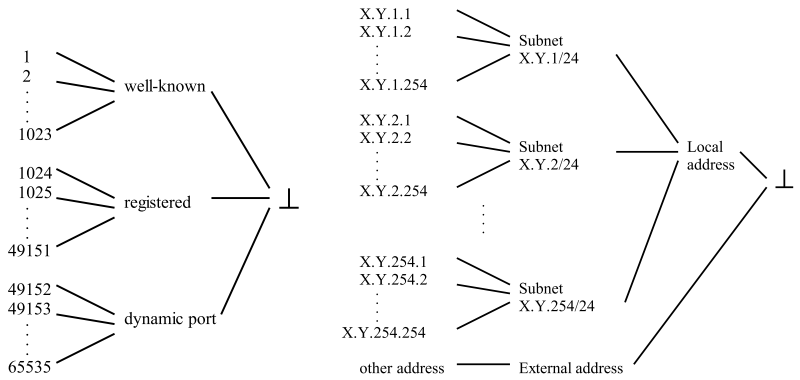
The taxonomy used in the experiments aggregates infrequent items according to the following aggregation trees. (1) Source and destination ports are aggregated by exploiting the aggregation tree shown in Figure 9(a), which introduces three aggregation values (i.e., *well known, registered, dynamic*). (2) Source and destination IP addresses are aggregated by exploiting the aggregation tree shown in Figure 9(b)[2]. IP addresses are aggregated in *subnet* if they are local to our campus network. IP addresses not belonging to the campus network are aggregated in a more general *external address* node. Furthermore, both the flow size (bytes) and the number of IP packets attributes are uniformly discretized in 4 bins, whose intervals are [1,1000), [1000, 2000), [2000, 3000), and equal or greater than 3000.

Enforced schema constraints. A network analyst was interested in identifying pairs of network devices (each device is identified by its IP) that frequently communicated together and the characteristics of the generated traffic (e.g., used ports, number of transmitted packets). He was also interested in identifying single network devices (IPs) with a huge amount of incoming/outgoing data on specific ports. He provided us the schema constraints reported in Table 3 to drive the network traffic analysis.

5.2. Effectiveness of the CoGAR *framework*

The effectiveness of the proposed approach in mining valuable generalized association rules in different application domains is discussed in Sections 5.2.1

[2]For privacy reasons, the first 16 bits of the IP addresses are hidden.



(a) Aggregation tree AT_{port} for the source and destination port attributes

(b) Aggregation tree $AT_{IP-address}$ for the source and destination IP address attributes

Figure 9: Aggregation trees for the port and IP attributes

(context-aware data) and 5.2.2 (network traffic data). Section 5.2.3 introduces the enforcement of a multiple-taxonomy to enrich the knowledge discovery process.

5.2.1. Knowledge discovery from context-aware data

The habits of specific users (or user categories) may be characterized by some kind of recurrence. By selecting from Table 2 only the schema con-

Table 3: *NetCapture* dataset: enforced schema constraints

Constraint	**Rule example**
$\{IPsource, IPdest, Portsource\}$	$\{(IPsource, X.Y/16), (Portsource, 184)\} \Rightarrow \{(IPdest, Extern)\}$
$\{IPsource, IPdest, Portdest\}$	$\{(IPsource, X.Y/16)\} \Rightarrow \{(IPdest, Extern), (Portdest, 184)\}$
$\{IPsource, Portsource, Flowsize\}$	$\{(IPsource, X.Y/16), (Portsource, 50)\} \Rightarrow \{(Flowsize, 10)\}$
$\{IPdest, Portdest, Flowsize\}$	$\{(IPdest, X.Y/16), (Portdest, 50)\} \Rightarrow \{(Flowsize, 10)\}$
$\{IPsource, Portsource, PacketsNr\}$	$\{IPsource, X.Y/16), (Portsource, 50)\} \Rightarrow \{(PacketsNr, 10)\}$
$\{IPdest, Portdest, PacketsNr\}$	$\{(IPdest, X.Y/16), (Portdest, 50)\} \Rightarrow \{(PacketsNr, 10)\}$
$\{IPsource, IPdest, PacketsNr\}$	$\{(IPsource, X.Y/16), (IPdest, Extern)\} \Rightarrow \{(PacketsNr, 10)\}$
$\{IPsource, IPdest, FlowSize\}$	$\{(IPsource, X.Y/16), (IPdest, Extern)\} \Rightarrow \{(Flowsize, 10)\}$
$\{Portsource, Portdest, FlowSize\}$	$\{(Portsource, 1024), (Portdest, 184)\} \Rightarrow \{(Flowsize, 10)\}$
$\{Portsource, Portdest, PacketsNr\}$	$\{(Portsource, 1024), (Portdest, 184)\} \Rightarrow \{(PacketsNr, 10)\}$

straints involving the *user* attribute (i.e., the first ten schema constraints), generalized rule mining is tailored to user profiling. For example, the following generalized rule allows the discovery of valuable knowledge about a generic user of the *Recs* application, named *Rossi*[3].

{ `(user, Rossi)`} $\Rightarrow$ { `(date, II Trimester 2009) (service, GET_REC)` } $(sup = 9.8\%, conf = 95\%)$

This rule has been mined by enforcing a support threshold equal to 1% (i.e., absolute threshold=58) and by exploiting the user-provided taxonomy reported in Section 5.1.1.
The rule highlights that user *Rossi* is interested in getting recommendations in the second trimester of year 2009, with rule confidence 95%. Thus, it provides relevant knowledge on this user attitudes. In particular, for specific users (user categories), rules satisfying the above constraints may highlight the service type users are mainly interested in, the context in which requests are commonly submitted, and the parameters which are frequently used.
Consider now the following rule:

{ `(user, Rossi)`} $\Rightarrow$ { `(date, Year 2009) (service, GET_REC)` } $(sup = 10.4\%, conf = 98\%)$

It highlights a higher level recurrence that does not provide additional knowledge with respect to the former one, as the confidence increase from the former to the latter one is exclusively due to the rule head generalization on the date attribute (i.e., from trimester to year). The opportunistic confidence

[3]Actual individual names are not provided for privacy reasons.

constraint allows sharply pruning this kind of rules which are considered rendudant and, thus, not useful for analyst decision making.

By enforcing all the constraints reported in Table 2, the following generalized rule is also extracted.

{ `(place, Italy)` `(date, II Trimester 2009)`} $\Rightarrow$ {`(service, GET_REC)`} $(sup = 17\%, conf = 97\%)$

This rule shows a different application-oriented recurrence. In particular, it emphasizes that the GET_REC service is frequently requested when the location is Italy and the date is in the second semester of year 2009.

5.2.2. Knowledge discovery from network traffic dataset

To discover interesting correlations hidden in the network traffic trace, the itemset mining process might be initially driven by the whole schema constraint set reported in Table 3 and by a minimum support threshold equal to $minsup$=1.5%, while the rule generation should be driven by a minimum confidence threshold $minconf$=10% and the opportunistic confidence constraint. Among others, the following generalized rule is extracted.

(i) { `(IP source, Extern)`, `(Flow Size,` > 3000)} $\Rightarrow$ {`(IP destination, X.Y.85/24)` } $(sup = 1.7\%, conf = 15\%)$

The above rule highlights significant incoming external traffic flows to subnet $X.Y.85/24$. Indeed, the network domain expert should consider to monitor the most significant incoming flows involving subnet $X.Y.85/24$ to understand problems coming from network traffic overloading.

Beyond the former rule, a traditional generalized rule miner, driven by support and confidence constraints only, would extract the following higher

level rule as well:

(ii) { `(IP source, Extern), (Flow Size,` > 3000`)`} $\Rightarrow$ {`(IP destination, X.Y/16)` } $(sup = 2.8\%, conf = 32\%)$

Its extraction may mislead the expert in decision making, as it could lead him to carry out a monitoring campaign on a larger set of IP addresses ($X.Y/16$) even if the contemporaneous extraction of (i) suggests to restrict the monitoring space to the 24-bit subnet $X.Y.85/24$. The enforcement of the opportunistic confidence allows avoiding redundant and possibly misleading knowledge extraction, thus, easing the knowledge discovery process.

A more insightful analysis tailored to traffic volume monitoring may focus on how hosts belonging to subnet $X.Y.85/24$ are contacted on specific well-known ports (e.g., port 1000). By focusing on recurrences involving couples source/destination IP addresses and ports only (i.e., enforcing just the first two mining constraints in Table 3) and by lowering the support thresholds ($minsup$=0.1%), the following rule is mined.

(iii) { `(IP destination, X.Y.85.189)` } $\Rightarrow$ {`(Destination Port, 1000)`} $(sup = 1.2\%, conf = 88.8\%)$

It highlights relevant incoming connections through well-known port 1000 of internal an IP address belonging to subnet $X.Y.85/24$. The analyst may deem this knowledge relevant as he monitors service usage on specific ports to prevent and manage network overloading situations.

5.2.3. Multiple-taxonomy evaluation to enhance the knowledge discovery

For several application domains the analysis on different facets of the same feature is desiderable. Multiple-taxonomy evaluation provides the capability

to explore at the same time different aggregation trees on the same attribute. For example, suppose to enrich the taxonomy proposed in Section 5.1.1 by extending the *date* attribute taxonomy with the week day, week, and bimester information as follows.

- date → month → trimester → year
- date → bimester
- date → week day → week
- timestamp → hour → timeslot (two hours timeslots)→ day period (AM/PM)
- latitude:longitude → city → country

The usage of different aggregation trees for the *date* attribute could be semantically interpreted as a faceted knowledge classification, in which facets are different axes along which items are aggregated.

By enforcing a minimum support threshold equal to 2% and the same constraints proposed in Section 5.1.1, the following generalized association rules are mined (among others).

{ `(user, Rossi)` } ⇒ { `(date, Tuesday), (service, GET_REC)` } ($sup = 3.4\%, conf = 33\%$)

{ `(user, Rossi)` } ⇒ { `(date, May-June 2009), (service, GET_REC)` } ($sup = 7.5\%, conf = 73\%$)

These rules provide a more detailed knowledge on user *Rossi* habits. They emphasize a different facet of the *date* attribute (the day of the week) that

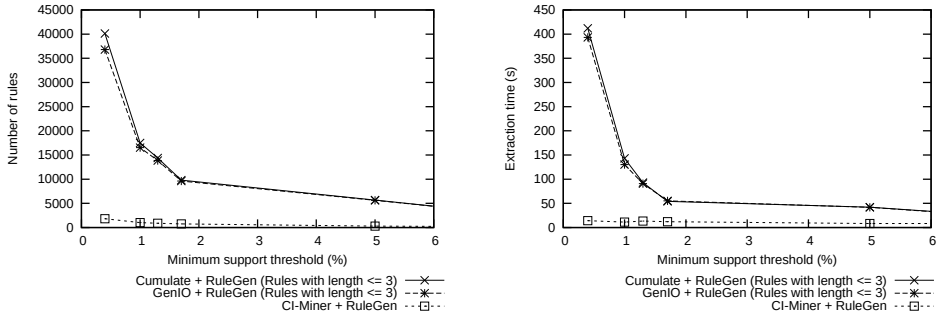
may better support domain experts in user profiling (e.g., to personalize daily promotions depending upon the yearly time period). They also allow the specialization of previously mined knowledge by highlighting a smaller time slice (i.e., May-June vs. May-July). When a single aggregation tree per attribute is allowed, multiple extraction sessions are needed to obtain the same information.

5.3. Effect of the enforced constraints

The value of the minimum support and confidence thresholds significantly affect the number of mined rules. To avoid the extraction of uninteresting correlations, we propose to enforce analyst-provided schema constraints and the new opportunistic confidence constraint as well. In this section, we analyze the effect of the enforced constraints in terms of both the number of extracted generalized rules and execution time. To this aim, we separately analyze the effect of schema and opportunistic confidence constraints.

5.3.1. Effect of the schema constraints

The CoGAR framework exploits the CI-Miner algorithm to perform generalized itemset mining. Then, RuleGen is exploited to perform the rule mining procedure. Itemset mining is commonly constrained by a minimum support threshold. The CI-Miner algorithm also enforces schema constraints to further reduce the amount of uninteresting extracted itemsets and, consequently, the number of rules. Figures 10(a) and 11(a) report for the *Recs* and *NetCapture* datasets (i) the number of mined rules and (ii) the corresponding extraction time when varying the minimum support threshold and enforcing no confidence threshold (i.e., $minconf$=0). We compared the



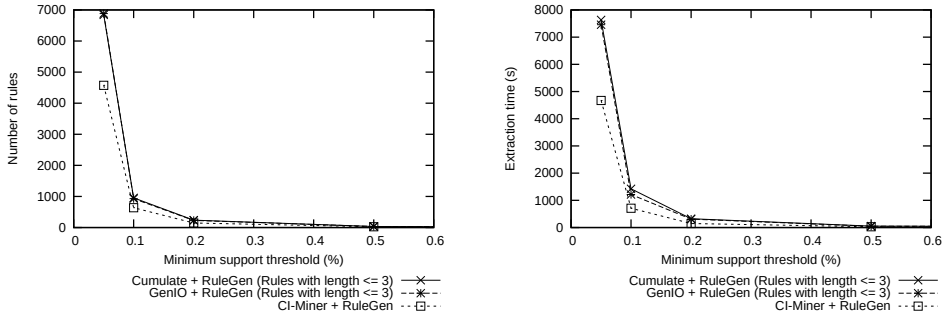
(a) Number of extracted rules

(b) Extraction time

Figure 10: *Recs* dataset: Effect of the minimum support threshold on the number of mined rules and corresponding execution time.

number of rules mined by the mining block based on the CI-Miner itemset mining algorithm followed by RuleGen (i.e., the rules satisfying the enforced constraints and the minimum thresholds) with the number of rules mined by exploiting both *Cumulate* [2] and GenIO [5] in the initial itemset set mining phase. Unlike CI-Miner, *Cumulate* and GenIO do not enforce any schema constraints, indeed a post-processing step is required to extract the same knowledge of interest. To perform a fair comparison between the three extraction processes, we limited the maximum length of the mined itemsets (and rules) to the maximum constraint length (i.e., max_len=3 for schema constraints in Tables 2 and 3) also when the *Cumulate* and GenIO algorithms are executed. The enforcement of schema constraints into the mining process significantly reduces the amount of extracted irrelevant knowledge, thus improving the efficiency of the knowledge discovery process.

Rule mining based on GenIO slightly outperforms the one based on *Cumulate*, at small and medium support thresholds (e.g., $minsup$=1%), in



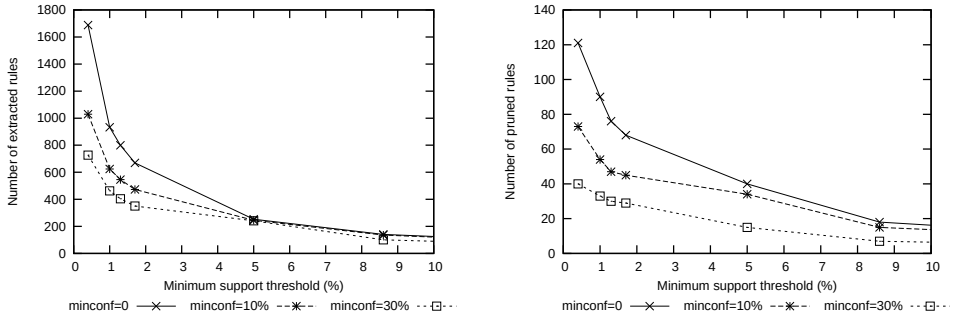
(a) Number of extracted rules

(b) Extraction time

Figure 11: *NetCapture* dataset: Effect of the minimum support threshold on the number of mined rules and corresponding execution time.

terms of rule pruning selectivity due to the support-driven approach to pattern generalization [5]. For the *Recs* dataset (see Figures 10(a)) and 10(b)), schema constraint enforcement yields a reduction larger than 90% of the rule cardinality for low support threshold (i.e., *minsup*=0.3%). The time reduction is significant also for medium support thresholds (e.g., *minsup*=3%), while it becomes more and more relevant (i.e., even larger than 95%) when further decreasing the minimum support threshold.

Similar considerations hold for the *NetCapture* dataset (see Figure 11(a)) and 11(b)). Mined rule set cardinality reduction and time reduction are less significant for the *NetCapture* dataset, with respect to the ones obtained on *Recs*, because *NetCapture* is characterized by fewer attributes than *Recs*. Thus, the number of extracted rules is lower and the corresponding time reduction is less significant.



(a) Number of extracted rules

(b) Pruning selectivity

Figure 12: *Recs* dataset: Effect of the opportunistic confidence constraint.

5.3.2. Effect of the opportunistic confidence constraint

To perform rule generation from the set of extracted frequent itemsets, the CoGAR framework exploits RuleGen followed by the CR-Filter postpruning algorithm. The rule mining step is constrained by a minimum confidence threshold. Besides, we enforced the opportunistic confidence constraint to further prune the set of extracted generalized rules.

Figure 12(a) reports the impact of the support and confidence thresholds on the number of rules mined from the *Recs* dataset when the opportunistic confidence constraint is enforced.

As expected, the pruning selectivity is more relevant when higher support and confidence thresholds are enforced. The opportunistic confidence constraint prunes a subset of the mined rule set that (i) includes at least a generalized item in the rule consequent, and (ii) may be considered as redundant (Cf. Definition 12). To evaluate the pruning selectivity of the new constraint, we consider the Recs dataset, as a representative example,

since the extracted pattern sets include around 65%-80% of rules containing at least a generalized item in the rule head. Figure 12(b) reports the number of rules pruned by enforcing the opportunistic confidence constraint and by varying the support and confidence thresholds. The percentage of rules pruned by the new constraint assumes values in the range [6%-12%] for every combination of support and confidence values.

The balancing between the confidence threshold and the new opportunistic confidence constraint could be highlighted by comparing the curves reported in Figure 12(b). When no minimum confidence is enforced (i.e., $minconf$=0), the generation of a (lower level) rule, satisfying the minimum support threshold, prevents the generation of all the frequent rules characterized by (i) the same body, and (ii) an ancestor (i.e., higher level itemset) of its rule head. Indeed, the opportunistic confidence constraint pruning effectiveness is maximum. When, instead, the minimum confidence threshold increases, some of the lower level rules are discarded due to the minimum confidence constraint and, thus, the pruning effectiveness of the opportunistic confidence constraint decreases.

We also separately analyzed the execution time of the steps of the mining activity (i.e., itemset mining constrained by the minimum support threshold and schema constraints and rule mining constrained by both confidence threshold and the new opportunistic confidence constraint). On average, the execution time of the itemset mining step typically accounts for more than 90% of the total execution time, while the remaining time is devoted to the rule mining and post-processing steps.

5.4. Scalability of the mining process

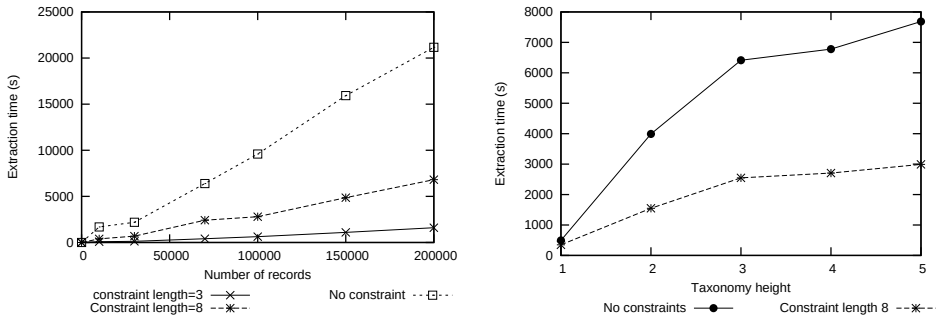
We analyzed the scalability of the rule mining process with respect to (i) the number of dataset records and (ii) the taxonomy height. To perform the scalability analysis we exploited a synthetic data generator based on the IBM data generator [11].

To allow generating taxonomies of different heights, we properly extended the original code of the synthetic generator. The taxonomy is generated by means of the following procedure. For each attribute, all values are considered as leaves (i.e., level 1) of the corresponding aggregation tree. Next, attribute values are sorted in a lexicographical order and grouped together based on a constant aggregation factor f. Finally, each group of items is collapsed in a newly generated upper level item. The above procedure is iterated until a unique root is available. For all attributes, we set the factor f to $\lceil \sqrt[(h-1)]{n} \rceil$, where n is the attribute domain cardinality. This leads to the creation of an aggregation tree, composed of h aggregation levels, such that the ratio between the number of items at level l and the number of items at level $l-1$ keeps constant.

Since some attributes are continuous, we performed a discretization step based on an equi-width technique by setting the number of bins to 10.

5.4.1. Scalability with respect to the dataset cardinality

To analyze the scalability of our approach with respect to the cardinality of the dataset, we generated datasets of size ranging from 5,000 to 200,000 records with 12 categorical attributes and corresponding taxonomies having height equal to 5. A minimum support threshold equal to 1% was enforced during the itemset mining step, while no minimum confidence threshold (i.e.,



(a) Scalability on the number of records. Taxonomy height=5.

(b) Scalability on the taxonomy height. Number of records=70,000.

Figure 13: Scalability of the IBM datasets ($minsup$=1%, $minconf$=0).

$minconf$=0) was enforced during the rule generation step. Three different schema constraint configurations are evaluated: (i) All the possible combinations of the first three attributes, (ii) all the possible combinations of the first eight attributes, and (iii) no schema constraints.

Figure 13(a) plots the overall extraction time (i.e., comprehensive of itemset and rule mining time and rule post-processing time) for the different constraint settings by varying the number of records. It shows that the proposed algorithm scales almost linearly with respect to the number of records. The overall CPU time is still acceptable also when dealing with larger datasets, and even when no constraint is enforced. Obviously, the number and type of schema constraints significantly impact on the execution time.

5.4.2. Scalability with respect to the taxonomy height

To evaluate the impact of the taxonomy height, we generated different taxonomies with height ranging from 1 to 5. For each attribute, a 5-level ag-

gregation tree is synthetically generated by means of the procedure described in Section 5.4. Next, the top level is pruned and a 4-level aggregation tree is generated. By iteratively applying this procedure on each aggregation tree, the taxonomy height is reduced by one at each iteration. At the end of the generation process, five different taxonomies of decreasing height are available for testing.

We set the number of records to $70,000$. A minimum support threshold equal to 1% and a minimum confidence equal to 0% are enforced during the mining process. Two different schema constraint configurations have been evaluated: (i) All the combinations of the first eight attributes, and (ii) no schema constraint. Figure 13(b) plots the extraction time by varying the taxonomy height. Since each increase of the taxonomy height corresponds to an increase of the total number of items, when increasing the taxonomy height also the number of extracted rules and the execution time increase. However, the increase does not scale linearly with respect to the taxonomy height. The difference between the number of items of the l-th taxonomy level and the number of items of the $(l\text{-}1)$-th taxonomy level depends on the value of l. According to the taxonomy generation process, the higher is the value of l, the lower is the difference between the number of items of the two considered taxonomies. As expected, the execution time increase is higher when moving from height 1 to 2, while it is lower for higher height values.

The taxonomy height also affects the computational cost of the post-processing steps. The time spent in rule post-processing scales roughly linearly with the taxonomy height. However, its impact on the overall execution time is negligible.

6. Related work

The generalized association rule mining problem was firstly introduced in [2]. The algorithm proposed in [2] is based on the Apriori principle and generates generalized itemsets by considering, for each item, all its parents in the hierarchy. Hence, generalized itemsets are exhaustively generated. One step further towards a more efficient extraction process for generalized association rule mining was based on new optimization strategies [9, 10]. In [10] a faster support counting is provided by exploiting the TID intersection computation, which is common in rule mining algorithms designed for the vertical data format. Differently, in [9] an optimization based on a top-down hierarchy traversal and multiple-support thresholds is proposed. It aims at identifying in advance generalized itemsets that cannot be frequent by means of an Apriori-like principle and uses different support for each item depending on its level in the taxonomy to prune redundant generalized itemsets. To further increase the efficiency of generalized rule mining algorithms, in [16] a FP-tree based algorithm is proposed, while in [19] both subset-superset and parent-child relationships in the lattice of generalized itemsets are exploited to avoid generating meaningless patterns.

The generalization process of all the state-of-the-art algorithms is driven by a taxonomy composed of at most one aggregation hierarchy for each given attribute. Hence, to analyze data characterized by different levels of abstraction on the same feature, many mining sessions are needed. CoGAR overcomes the above issue by means of the CI-Miner algorithm that allows exploiting multiple-taxonomies enabling generalized knowledge extraction by considering simultaneously different facets of the same attribute.

Typically, the analyst is not interested in all the frequent (generalized) itemsets or rules. Hence, many previous works [4, 7, 18, 21] has been devoted to enforcing constraints to extract only a subset of the patterns of interest. Some of them are based on the items of interest according to the analyst preferences (e.g., [4, 18]) while others are based on statistical and objective measures (e.g., [7, 21]). Since the analyst commonly knows the items or the type of patterns he/she is mainly interested in, this knowledge can be used to prune the search space. The first algorithm that allows the user to specify a set of constraints on the patterns of interest is proposed in [18]. It allows specifying a set of item constraints, which are used to specify the items of interest and how they could be combined, by means of boolean expressions. Unlike [18], in our work we exploit schema constraints to focus on a subset of itemsets of interest. Schema constraints are similar to item constraints. However, they work at a higher level (at the attribute level). Each schema constraint could be expressed by means of a set of item constraints. However, the usage of item constraints requires to explicate all the items that can appear together and those that cannot. Hence, schema constraints are definitely more compact and easy to use. In [4] an ad-hoc language to enforce constraints on the characteristics of the rule body and head has been proposed. Other approaches (e.g., [21]) exploit objective measures and statistical tests to perform pattern selection, instead of analyst preferences. For example, in [21] a set of statistical tests have been used to select significant patterns. The proposed approach is orthogonal with respect to the aforementioned ones [4, 18] as it could be applied to select the most statistically significant patterns among those of analyst's interest.

All the approaches mentioned above analyze each rule as itself. Differently, the opportunistic confidence constraint compares each generalized rule with a set of similar rules. In particular, only generalized rules whose knowledge has not been already described by any other pattern at a lower abstraction level are included in the mined rule set. A previous approach that does not analyze each rule as itself is proposed in [7]. The authors of [7] propose to select a rule depending also on the characteristics of its simplifications. Given a rule $r : X \rightarrow Y$, another rule $r_s : X_s \rightarrow Y$ is a simplification of r if $X_s \subset X$. The selection criterion proposed in [7] selects a rule if and only if the difference between its confidence and the confidence of any of its simplifications is positive. Hence, a rule r is considered of interest if and only of it provides an improvement, in term of confidence, with respect to its simplifications. Also the opportunistic confidence constraint proposed in our work compares each rule with a set of similar rules and exploits the confidence measure. However, our constraint aims at pruning generalized rules instead of not generalized rules.

Finally, the idea of exploiting generalized association rules for context-aware user and service profiling and for network traffic analysis was firstly introduced in [6] and [3] respectively. The focus in [6] is on profiling users and services in a mobile context-aware application, while in [3] is on the characterization of stream network data. Both in [6] and [3], the mining activity is performed by exploiting the GenIO algorithm [5], while the CoGAR framework exploits a different mining algorithm and enforce different constraints (schema constraints and the opportunistic confidence constraint) to remove uninteresting or redundant patterns. Many other approaches have been pro-

posed to analyze network data (e.g., [14, 15]). However, they usually focus on identifying devices with anomaly behaviors or on clustering of devices. Differently, generalized rules aim at giving a high level representation of the characteristics of the analyzed network.

7. Conclusion and Future Works

In this paper we presented the CoGAR framework to discover interesting generalized association rules. The generalization process is driven by a multiple taxonomy that allows the opportunistic extraction of knowledge at different aggregation levels. Besides the minimum support threshold, a set of constraints on the structure of interesting patterns drives the itemset mining process to extract only those patterns that satisfy user-provided schema constraints. Furthermore, we propose the new opportunistic confidence constraint, which is enforced during rule generation step to further prune from the rule set a subset of patterns deemed redundant for decision making. Experimental results, on both real and synthetic datasets, show the effectiveness of CoGAR in mining interesting generalized association rules, as well as its good scalability.

Future extensions of the CoGAR framework will address (i) the automatic inference of taxonomies from datasets, and (ii) the exploitation of more efficient rule extraction algorithms (e.g., LCM [20]) or closed itemset mining algorithms (e.g., TD-Close [12]) to mine generalized rules with constraints.

References

[1] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: J.B. Bocca, M. Jarke, C. Zaniolo (Eds.), International Conference on Very Large Data Bases, Morgan Kaufmann, San Francisco, CA, 1994, pp. 487–499.

[2] R. Agrawal, R. Srikant, Mining generalized association rules, in: U. Dayal, P.M.D. Gray, S. Nishio (Eds.), International Conference on Very Large Data Bases, Morgan Kaufmann, San Francisco, CA, 1995, pp. 407–419.

[3] D. Apiletti, E. Baralis, T. Cerquitelli, V. D'Elia, Characterizing network traffic by means of the netmine framework, Computer Networks 53 (2009) 774–789.

[4] A. Appice, M. Berardi, M. Ceci, D. Malerba, Mining and filtering multilevel spatial association rules with ares, in: M.S. Hacid, N.V. Murray, Z.W. Ras, S. Tsumoto (Eds.), Foundations of Intelligent Systems, 15th International Symposium, Springer, Berlin/Heidelberg, Germany, 2005, pp. 342–353.

[5] E. Baralis, L. Cagliero, T. Cerquitelli, V. D'Elia, P. Garza, Support driven opportunistic aggregation for generalized itemset extraction, in: 5th IEEE International Conference on Intelligent Systems, IEEE, Piscataway, NJ, 2010, pp. 102–107.

[6] E. Baralis, L. Cagliero, T. Cerquitelli, P. Garza, M. Marchetti, Context-aware user and service profiling by means of generalized association

rules, in: J.D. Velasquez, S.A. Rios, R.J. Howlett, L.C. Jain (Eds.), International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Springer, Berlin/Heidelberg, Germany, 2009, pp. 50–57.

[7] R.J. Bayardo, R. Agrawal, D. Gunopulos, Constraint-based rule mining in large, dense databases, Data Min. Knowl. Discov. 4 (2000) 217–240.

[8] J. Han, Y. Fu, Discovery of multiple-level association rules from large databases, in: U. Dayal, P.M.D. Gray, S. Nishio (Eds.), Proceedings of 21th International Conference on Very Large Data Bases, Morgan Kaufmann, San Francisco, CA, 1995, pp. 420–431.

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

[10] J. Hipp, A. Myka, R. Wirth, U. Güntzer, A new algorithm for faster mining of generalized association rules, in: J.M. Zytkow, M. Quafafou (Eds.), Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery, Springer-Verlag, London, UK, 1998, pp. 74–82.

[11] IBM, IBM Quest Synthetic Data Generation Code, `http://www.almaden.ibm.com/cs/disciplines/iis/`, 2009.

[12] H. Liu, X. Wang, J. He, J. Han, D. Xin, Z. Shao, Top-down mining of frequent closed patterns from very high dimensional data, Information Sciences 179 (2009) 899–924.

[13] NetGroup, Analyzer 3.0, http://analyzer.polito.it, 2009.

[14] N.H. Park, S.H. Oh, W.S. Lee, Anomaly intrusion detection by clustering transactional audit streams in a host computer, Information Sciences 180 (2010) 2375 – 2389.

[15] S.T. Powers, J. He, A hybrid artificial immune system and self organising map for network intrusion detection, Information Sciences 178 (2008) 3024 – 3042.

[16] I. Pramudiono, M. Kitsuregawa, Fp-tax: tree structure based generalized association rule mining, in: G. Das, B. Liu, P.S. Yu (Eds.), Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, ACM, New York, NY, 2004, pp. 60–63.

[17] Python, Python website, http://www.python.org, 2009.

[18] R. Srikant, Q. Vu, R. Agrawal, Mining association rules with item constraints, in: D. Heckerman, H. Mannila, D. Pregibon (Eds.), Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, CA, 1997, pp. 67–73.

[19] K. Sriphaew, T. Theeramunkong, A new method for finding generalized frequent itemsets in generalized association rule mining, in: A. Corradi, M. Daneshmand (Eds.), Proceedings of the Seventh IEEE Symposium on Computers and Communications, IEEE Computer Society, Washington, DC, 2002, pp. 1040–1045.

[20] T. Uno, M. Kiyomi, H. Arimura, LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets, in: R.J.B. Jr., B. Goethals, M.J. Zaki (Eds.), Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, CEUR-WS.org, 2004.

[21] G.I. Webb, Discovering significant patterns, Machine Learning 71 (2008) 131.

[22] XQuery, Xquery w3c website, `www.w3.org/tr/xquery/`, 2009.