AN ONTOLOGY-BASED FRAMEWORK FOR MINING PATTERNS IN THE PRESENCE OF BACKGROUND KNOWLEDGE

C. Antunes*

*Instituto Superior Técnico / Technical University of Lisbon, Portugal
claudia.antunes@ist.utl.pt

Keywords: Pattern mining, Background knowledge, Ontology, Framework

Abstract

Since its formulation, pattern mining has been focused both in the development of mining algorithms and in its extension to address new challenges, like the discovery of structured patterns. In both cases, existing approaches are now effective and efficient. However, and despite pattern mining is a constrained problem by definition, there is no generally accepted solution to incorporate domain knowledge into the mining process. In this paper, we propose the *Onto4AR* framework and explain why it can be that solution. The framework uses a domain ontology to represent background knowledge, that can be used to impose constraints in the mining process. The user benefits from a set of pre-defined constraints, specified over general relations among concepts, like *is-a* and *has-a* relations. These constraints provide a mining environment independent of the problem domain, which can be extended with the definition of new constraints, either general (temporal or structural, for instances) or domain dependent. The framework's potential is illustrated with its application in a case study.

1 Introduction

One of the current key issues on data mining is the incorporation of background knowledge into the discovery process [16]. However, and despite this is a central issue in the knowledge discovery process, it has been in the shadow for a long time, more exactly, since the first days of knowledge discovery in databases (see for example the work by Feldman on 1996 [8]). The reasons to this scenario are extensive, but mainly related to the difficulties on representing and acquiring domain knowledge.

Along the time, several attempts have been made, mainly with the introduction of constraints in the mining process ([2], [5], [9], [13], [14]). By using constraints, the user assumes the responsibility of choosing which of the aspects of domain knowledge are most important for the current task. As pointed by Bayardo, constraints play a critical role in solving the trade-offs of the generality of data mining algorithms, by focusing "the algorithm on regions of the trade-off curves (or space) known (or believed) to be most promising" [6]. Regardless of their undoubtedly contributions, constraints have not get their own space in industrial applications, mainly due to the existence of numerous approaches to specify them, but none accepted as a standard. Indeed, those approaches differ mostly on the specification language and scope (from logical formulations to domain-specific languages).

Association rules discovery follows the general situation, being a paradigmatic case of the described scenario. The problem was first introduced in 1993 [1], and is defined as the discovery of "all association rules that have support and confidence greater than the user-specified minimum support and minimum confidence, respectively". With an *association rule* corresponding to an implication of the form $A \Rightarrow B$, where A and B are propositions (sets of pairs attribute/value, most of the times named *items*), that expresses that when A occurs, B also occurs with a certain probability (the rule *confidence*). The *support* of the rule is given by the relative frequency of transactions that include A and B, simultaneously.

Note that the problem was formulated as a constrained problem, with support and confidence thresholds restricting the number of discovered rules. Probably due to this nature, the study of constraints on this area has begun early with the specification of several different interestingness measures to filter the discovered patterns [4]. These measures rank the discovered rules, by quantifying the usefulness and utility of them, discarding those with an evaluation less than a user-specified threshold. With these measures, it is possible to both reduce the number of discovered patterns and improve the performance of the algorithms, by pruning uninteresting ones.

Another category of constraints (here referred as *content constraints*) can be seen as filters over the content of the discovered patterns, instead of its relevance. While interestingness measures are quantitative metrics, content constraints capture application semantics and introduce it into the mining process. An example of such constraints is the *item constraint*, as proposed by Srikant *et al* [14]. More recently, but in the context of sequential pattern mining, the

use of regular expressions [8] and context-free languages [2] were proposed for specifying constraints. Content constraints have also been applied as post-processing mechanisms, to discard rules that seem to not be useful (see for example the works by Boulicaut [7]).

Nevertheless, existing approaches do not convince business agents of their advantages, and have not been applied consistently in real situations. Without their collaboration, is impossible to acquire and represent domain knowledge to enrich the mining process. Fortunately, the recent developments in the area of knowledge management make possible the representation of background knowledge as Ontologies, which have been received with a considerable success by industry.

In this paper, we propose a framework for discovering association rules – the *Onto4AR framework*. This framework provides an environment for specifying constraints that enables the user to control the mining process. The great advantage of this framework is the use of domain ontologies to express existing background knowledge. By making use of these formal models, it is possible to create a universal environment for mining association rules in any domain, and in the presence of any domain knowledge. On the other side, the framework establishes the precise definition of a set of different constraints, which again can be applied to different problems

The rest of the paper is organized as follows: next (in section 2), we overview the use of domain ontologies, succinctly describing their main concepts and presenting a domain ontology, which will be used later. Then in section 3, we describe the entire framework, reformulating the pattern mining problem and explaining how existing algorithms can be used in the new context. Section 4 presents some pre-defined constraints that are independent of the problem domain but universally applicable. The paper ends with the discussion of the benefits from using the framework and some guidelines for future work.

2 Domain Ontologies

The development of the Semantic Web contributed considerably to advance the area of ontologies, and now they are commonly accepted as a mean to represent and share existing knowledge.

An *ontology* is an explicit specification of a conceptualization [10], which means that is a specification of an abstract, simplified view of a domain. Formally, an ontology is a 5-tuple $O:=\{C, \mathcal{R}, \mathcal{H}^C, rel, \mathcal{A}^O\}$, where C is a set of concepts, which represent the entities in the ontology domain and $\mathcal{R}$ is a set of relations defined among concepts. $\mathcal{H}^C$ is a taxonomy or concept-hierarchy, which defines *is-a* relations among concepts: $\mathcal{H}^C(c_1, c_2)$ means that c_1 is a sub-concept of c_2, or in other words c_2 is a parent of c_1. The *rel* element corresponds to a function, $rel: \mathcal{R} \rightarrow C \times C$ that specifies the relations on $\mathcal{R}$: if $r \in \mathcal{R}$, $rel(r)=(c_1, c_2)$, also written as $r(c_1, c_2)$, and means that c_1 is related to c_2, but the inverse is not necessarily true. Finally, $\mathcal{A}^O$ is a set of axioms that describe constraints on the ontology, expliciting implicit facts [12]. Among the concepts, we consider *leaf concepts* as concepts that can be instantiated (that are not abstract concepts).

In the counterpart of ontologies are *knowledge bases*, which specify existing instantiations for a particular ontology. Formally, a *knowledge base* is a 4-tuple $\mathcal{KB}:=\{O, I, inst, instr\}$, where O is a ontology as defined above and I a set of instances. *inst* is a function from C to 2^I called *concept instantiation*, and *instr* the relation instantiation function defined from $\mathcal{R}$ to $2^{I \times I}$. Note that while ontologies try "to capture the conceptual structures of a domain of interest" [12], by defining its elements and describing the relations among them, a knowledge base defines a set of

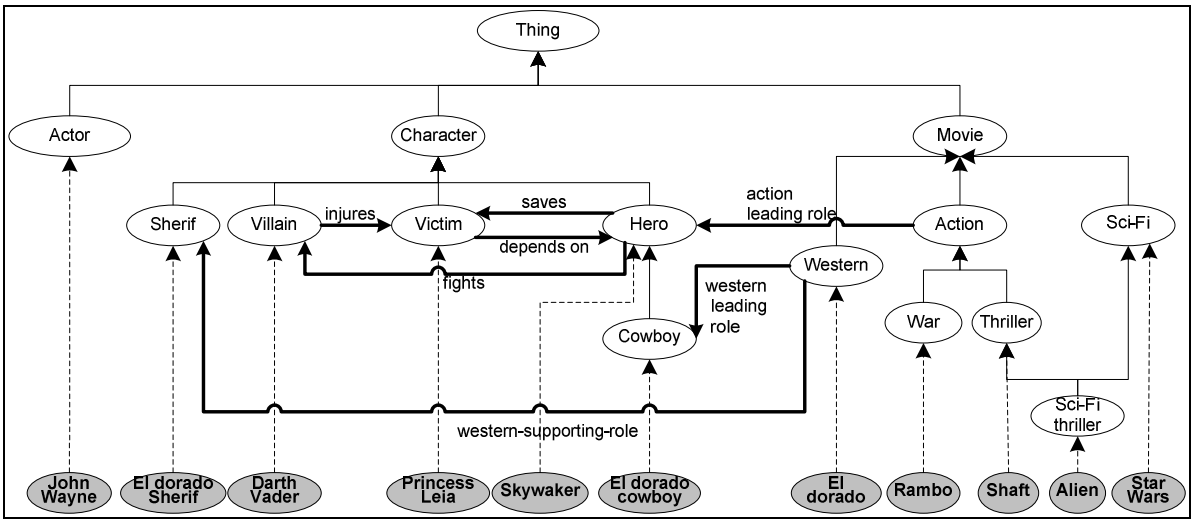

Figure 1 – Ontology and knowledge base for cinematographic domain

elements, which can be understood in that context.

2.1 An example

Figure 1 illustrates a simple knowledge base $\mathcal{KB}$ constituted by a simple ontology $\mathcal{O}$, on the cinematographic domain. The schema is represented in UML, where concepts are represented by ellipses (*Movie* and *Actor* are examples of concepts, and *Thing* represents the concept, from whom all other concepts descend), *is-a relations* as non-named arrows (for example *Action* is a particular case of *Movie*), and other relations by named strong arrows (*fights* for example). Instances are represented by shadowed ellipses and instantiations by dotted arrows (*Rambo* is an instance of *Action*). This knowledge base will be used later to illustrate the use of the *Onto4AR* framework described next.

3 The *Onto4AR* framework

The balance between discovering unknown information and managing large quantities of patterns is far from being reached, but achievable only by incorporating domain knowledge in the mining process, either by representing and using it in mining algorithms, or just by making use of users' expertise. As been shown, the use of constraints is one of the few effective approaches to implement that approach.

The *Onto4AR* framework defines a formal environment for incorporating background knowledge into the process of discovering association rules that is independent of the problem domain and the nature of the data. The framework is centred in a precisely definition of constraint, and assumes a new formulation of the problem.

Definition 1. *Let $\mathcal{KB}:=\{\mathcal{O}, \mathcal{I}, inst, instr\}$ be a knowledge base with $\mathcal{O}:=\{\mathcal{C}, \mathcal{R}, \mathcal{H}^{\mathcal{C}}, rel, \mathcal{A}^{\mathcal{O}}\}$, and $\mathcal{D}$ a set of transactions, where each transaction $\mathcal{T}$ is a set of instances, such that $\mathcal{T}\subseteq\mathcal{I}$. Let $\mathcal{L}$ be the set of items for this domain, where each item corresponds to a leaf concept of $\mathcal{O}$ ($\mathcal{L}\subset\mathcal{C}$), and X a set of items such that $X\subseteq\mathcal{L}$.*
Given a set of transactions, the goal is to find all rules of the form A⇨B, where A and B are disjoint sets of items that occur in $\mathcal{D}$, and their union satisfy a set of constraints $\mathcal{C}_{\mathcal{O}}$, defined over the ontology $\mathcal{O}$.

Note that this definition differs from the usual one in two aspects. First, there is no imposition on the number of times that *A* and *B* occur together. All depends on the constraints imposed by $\mathcal{C}_{\mathcal{O}}$. Second, a rule relates more than simple objects (instances), but specifies abstract relations, similar to generalized patterns proposed in [14]. Indeed, this aspect has been neglected, and the user has to represent the set of transactions at the right abstraction level. Even in the basket analysis problem, items do not correspond to real instances but to some abstraction (when a customer buys a particular beer and consumes it, other customers cannot buy it). In the *Onto4AR* framework, this issue is solved by specifying the meaning of the occurrence of a set of items in a transaction and the meaning of the equality between items, both defined outside the logic and scope of the mining algorithm.

Definition 2. *A set of items X* occurs *in a transaction T, or a transaction $\mathcal{T}$* contains *X ($\mathcal{T}\subseteq X$ for short), if all elements of X have an unique instantiation in $\mathcal{T}$, which means*

$$\forall x \in X, \exists t \in T : t = inst(x) \wedge T \setminus \{t\} \text{ contains } X \setminus \{x\}$$

For example, in the context of the knowledge base presented above, the set of items {*Western*, *Hero*} occurs in the transaction {*El dorado*, *El dorado cowboy*}, but {*Western*, *Hero*, *Cowboy*} does not: despite *El dorado cowboy* is an instance of *Hero* and *Cowboy*, he cannot be considered an instantiation of both items, simultaneously.

The equality among items, on the other hand, is defined by some axioms in the ontology. Specifically, each leaf concept has associated an *equal* predicate, defined by some axiom in the ontology. In this manner, it is possible to have multiple items (concepts) at different levels of abstraction, in the same context and rule.

In the context of the *Onto4AR* framework, a constraint is defined as in the related literature [6],

Definition 3. *A **constraint** is a predicate on the powerset of the set of items I, which means, that it is a function $c: 2^{\mathcal{I}} \rightarrow \{true, false\}$. An itemset S is said to* satisfy *c, if and only if, c(S) is true.*

In this manner, a constraint imposes some condition over the elements of the itemset, or the relations among them. These predicates can establish either qualitative or quantitative conditions. Indeed, at a first glance, two different categories of constraints can be distinguished: interestingness constraints (also known as interestingness measures) and content constraints. Note that other kind of constraints can be generally defined, like temporal or structural ones. They are not included at this point, since they are not essential to the traditional pattern mining problem.

Interestingness measures are constraints that impose quantitative conditions over the set of items in the rule, like the number of times that the set of items are transacted together, or the novelty introduced by the discovery of the rule.

Definition 4. *An **interestingness measure** is a composed function $f = f_\theta \circ g$ $[f(x)=g(f_\theta(x))]$, with $g: 2^{\mathcal{I}} \rightarrow \mathbb{R}$ and $f_\theta: \mathbb{R} \rightarrow \{true, false\}$, defined by the comparison of its argument with θ, a threshold value.*

Although interestingness constraints play a fundamental role on pattern mining, they only capture the knowledge about some quantity that is significant for the specific

business. As such, they are not able to represent any other knowledge about the business domain.

Content constraints introduce the ability of imposing that items in the rule have some specific characteristics. These characteristics can be selected among the ones represented in the domain ontology.

Definition 5. *A **content constraint** can be defined as a predicate* $c_O: 2^I \rightarrow \{true, false\}$ *that impose some qualitative condition on the items of its argument, expressed based in the domain knowledge represented in the ontology* O.

3.1 Algorithms

As in the rest of pattern mining area, efficient algorithms for identifying accepted patterns is mandatory in this framework. Algorithms for this purpose should be similar to general algorithms for pattern mining, which means that they would filter patterns that satisfy the imposed constraint in an internal pruning step. In this manner, the processing time would be significantly reduced due to the reduction on the explosion of discovered patterns.

It is important to note that this pruning step wouldn't require the navigation over the entire ontology. For each potential pattern (call it *candidate*) we need to verify if the candidate satisfies the constraint, this means, verifying if its items are linked among them, which only requires to follow the relations among their concepts in the ontology.

The last issue refers to candidate generation, since some of proposed constraints (namely item and weakly-connected constraints, as defined below) are non-anti monotonic. The simplest solution is to extend frequent k-patterns with frequent items, like it is done in constrained sequential pattern mining [9] and [2].

For algorithms that do not follow the candidates generation and test approach, like FP-growth [11], the changes needed are fewer, and similar to the ones addressed in candidate generation. Indeed, for FP-growth it is only necessary to change the process of creating the conditional trees: only accepted itemsets should be considered.

It is important to note that recent work on sequential pattern mining using more complex algorithms and constraints (deterministic finite and push-down automata) [2] demonstrates experimentally that the reduction on the number of discovered patterns has more impact in the performance, than the verification of the constraint satisfaction for each candidate.

Moreover, the *Onto4AR* just use the correspondence between instances and concepts, and the ontology structure – relations among concepts (instance relations are not used). In this manner, the ontology almost corresponds to an acyclic graph, which makes the verification process similar to the ones referred.

4 Pre-defined Constraints

Beside the establishment of a formal environment for mining patterns in the presence of background knowledge, our proposal benefits from the general acceptance of those formalisms by business agents. Since domain ontologies begin to be available at different domains, from finance to medicine, and from scientific research to industrial areas, it is expected that the *Onto4AR* framework could be used widely.

In order to facilitate its usage, we propose several constraints that are independent of the problem domain, and can be universally instantiated to each problem. Generally, interestingness measures satisfy this independence, but content constraints tend to be more dependent of the problem domain. Each introduced notion is illustrated by using the knowledge base introduced in section 2.1, and considering that each transaction represents a movie, with its title, genre, the list of its actors and corresponding characters.

4.1 Interestingness Measures

For allowing the discovery of association rules, as initially proposed in [1], we need to define the support and confidence constraints. Due to their nature, they are naturally defined as interestingness measures. Consider the predicate $>=_\theta$, which returns true if and only if its argument is greater than or equal to θ.

Definition 6. *The **support** is an interestingness measure resulting from composing the frequency function with the predicate* $\geq_\theta$.

In the same manner,

Definition 7. *The **confidence** is obtained by composing the conditional probability with the same predicate.*

4.2 Content Constraints

Among generic content constraints, we can distinguish between taxonomical and non-taxonomical constraints. Some additional notions help in their definition.

Definition 8. *A **child** is a predicate defined over the concepts in the ontology* O, *and child (x, y) is true if and only if* $\mathcal{H}^C(x, y)$ *is defined in the ontology. **Parent** is the inverse of child, and then parent (x, y) is true if and only if child (y, x) is true.*

The predicate *descendant* is defined in a recursive way:

Definition 9. *An item x is a **descendant** of y if and only if either x is child of y or exists a third concept z that is parent of x and is descendant of y, simultaneously.*

$$descendant(x,y) \Leftrightarrow child(x,y) \vee [\exists z \in C: child(x,z) \wedge descendant(z,y)]$$

Additionally, consider the notion of n^{th} *cousin*.

Definition 10. *An item is an* ***n^{th} cousin*** *of another item, if they are connected by a n degree collateral relation. An item x is* ***at least an n^{th}-cousin*** *of another item y if there is some $m \leq n$ such x is an m^{th}-cousin of y.,*

$$\forall x, y \in I; n \in N : n^{th}cousin(x, y, n) = true$$
$$\Leftrightarrow \begin{cases} descendant\ (x, parent(y)) \ , if\ n = 0 \\ n^{th}cousin(x,\ parent(y),\ n-1), otherwise \end{cases}$$

For example, x is a 0^{th} cousin of y if it descends from y's parent (it is either is "brother" or its "nephew"); x is a 1^{st} cousin of y, if it descends from y's grandparent (one degree of collaterality); x is a 2^{nd} cousin of y, if it descends from y's great-grandparent (two degrees of collaterality), and so on. Finally,

Definition 11. *Two items are* ***related*** *if there is a known non-taxonomical relation r (with $r \in \mathcal{R}$) among them.*

$$\forall x,y \in \mathcal{L}: related(x,y)=true \Leftrightarrow \exists r \in \mathcal{R}: r(x,y).$$

Taxonomical constraints. Taxonomical constraints establish restrictions based on the family ties among concepts, defined by the taxonomy. The first of this kind is the *item constraint* that imposes the presence of items in the rule [15].

Definition 12. *An* ***item constraint*** *is a content constraint c_S, implemented by the predicate $\in_S$, and then it only returns true if some items of its argument belong to the user-defined subset S of $\mathcal{L}$. Let X be an itemset*

$$X \text{ satisfies an item constraint } c_S \Leftrightarrow \exists x \in X: x \in S$$

For example, using the ontology described on section 2.1 the item constraint defined over the set $\{x \in \mathcal{L}: x = Western\}$ will only accept patterns that include some western movies.

Another class of taxonomical constraints is the *family constraints*, defined over the n^{th}*-cousin* and *at least* n^{th}*-cousin* predicates.

Definition 13. *An itemset satisfies the* ***same-family$_n$*** *constraint, if all their items share a common ancestor and all of them are at least n^{th} cousins to each other.*

$$X \text{ satisfies same-family}_n \Leftrightarrow$$
$$\forall x,y \in X\ \exists m \in \mathbb{R}: x \neq y \wedge m \leq n \wedge n^{th}cousin(x,y,m)$$

In the same context, the itemset {*War, Thriller*} satisfies the same-family constraint with n=0, since they share the same parent, but {*War*, *Sci-Fi Thriller*} satisfies the same-family constraint with n=1, since *War* descends from the grandparent of *Sci-Fi Thriller*. Note that every itemset satisfies this constraint for some n, since all concepts descend from the concept *Thing*.

Definition 14. *An itemset satisfies the* ***close-family$_n$*** *constraint, if every item in the set satisfies the at least an n^{th}-cousin predicate with another item in the set.*

$$X \text{ satisfies close-family}_n \Leftrightarrow$$
$$\forall x \in X\ \exists y \in X\ \exists m \in \mathbb{R}: x \neq y \wedge m \leq n \wedge n^{th}cousin(x,y,m)$$

In order to have an itemset that satisfies the close-family constraint and does not satisfy the same-family constraint, the itemset has to contain some concept that derives from more than one concept, this is, it has to have multiple parents. For example, the itemset {*Thriller, Sci-Fi Thriller, Sci-Fi*} satisfies the close-family constraint with n=1, because *Thriller* is a 1^{st}-cousin of *Sci-Fi Thriller*, and *Sci-Fi Thriller* is a 1^{st}-cousin of *Sci-Fi*.

It is important to note that item constraints are non-anti monotonic, but both family constraints introduced are.

Relational constraints. The incorporation of an ontology in the mining process, instead of a simple taxonomy, allows for the definition of other constraints that extend the notion of item constraint. Those new constraints are based on the relations among concepts, described in the ontology.

Definition 15. *An itemset is* ***weakly connected*** *if all its items are related with at least another item in the set.*

$$X \text{ is weakly connected} \Leftrightarrow \forall x \in X\ \exists y \in X: x \neq y \wedge related(x,y)$$

The itemset {*Western*, *Sherif*, *Cowboy*} is weakly connected, since there is the relation *western-leading-role* from *Western* to *Cowboy*, and another *western-supporting-role* from *Western* to *Sheriff*.

Definition 16. *An itemset is* ***softly connected*** *when there is a chain of relations among all its items.*

$$X \text{ is softly connected} \Leftrightarrow$$
$$\exists x,y \in X: x \neq y \wedge related(x,y) \wedge X \setminus \{x\} \text{ is softly connected}$$

In the context of the same ontology and knowledge base, the itemset {*Hero*, *Villain*, *Victim*} is softly connected, because there is a relation from *Hero* to *Villainr* (*fights*) and from *Villain* to *Victim* (*injures*), which means there is a chain of relations among the three items.

Definition 17. *An itemset is* ***strongly connected*** *if all its items are related to all the other items in the set:*

$$X \text{ is strongly connected} \Leftrightarrow \forall x,y \in X: x \neq y \wedge related(x,y)$$

In the same context, the itemset that are strongly connected are {*Hero*, *Victim*}, since they are mutually related.

These three predicates can be used to define new relational content constraints: *weak*, *soft* and *strong* constraints. From these, only weakly connected is non-anti monotonic.

4.3 Composition of constraints

Beside the existence of taxonomical and relational constraints, the *Onto4AR* framework provides another important artefact – the *composition of constraints*. As for

content constraints, the composition can be weak or strong.

Definition 18. *The **weak composition** of two constraints, say f and g, results in a new constraint f o g, which only accepts itemsets accepted by at least one of the original constraints.*

$$\forall X \in 2^{\mathcal{I}}: fog(X)=true \text{ iff } f(X)=true \text{ or } g(X)=true$$

On the other hand,

Definition 19. *The **strong composition** of two constraints, say f and g, results in a new constraint f o g, which only accepts itemsets accepted by both original constraints.*

$$\forall X \in 2^{\mathcal{I}}: fog(X)=true \text{ iff } f(X)=true \text{ and } g(X)=true.$$

With the composition of constraints, the set proposed in this paper is just a small picture of the potentialities introduced by the *Onto4AR* framework. Axiomatic, temporal and structural constraints are just some examples of other kind of constraints that could be incorporated, in order to be able to deal with more complex problems, like mining structured data.

5 Conclusions

The recent advances in the area of knowledge representation makes possible to represent background knowledge, in an effective way, using ontologies. Since one of the main drawbacks of data mining, in general, and of pattern mining, in particular, is to ignore domain knowledge, with those advances, it is time to surpass that feature.

This paper presents a framework that incorporates background knowledge in the core of the mining process, by using domain ontologies and by defining a set of constraints above them, which can guide the discovery process. The framework has three main advantages: the clear formulation of the pattern mining problem in the presence of domain knowledge, represented by an ontology, its wide applicability and its extensibility. The first advantage makes possible to perform the discovery at the right level of abstraction, without passing this responsibility to the final user. On the other hand, its applicability is due to its independence of the problem domain, which allows for the definition of constraints that are also domain independent. Finally, its extensibility derives from the ability to incorporate any constraints, general or domain-dependent.

With the ability of using background knowledge to guide the pattern mining problem, is clear that the complexity of the problems to address by this technique can be highly increased. For example, recent research [3] shows that this framework can be used to identify frequent molecular fragments: a task that has been addressed exclusively by graph-pattern mining algorithms.

References

[1] R. Agrawal, T. Imielinsky, A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proc. ACM SIGMOD Conf. on Management of Data*, 207-216 (1993)

[2] C. Antunes, A. Oliveira. Using Context-Free Grammars to Constrain Apriori-Based Algorithms for Mining Temporal Association Rules. In *Proc. Workshop on Temporal Data Mining* (2002)

[3] C. Antunes. Are graph-pattern mining algorithms really needed?, Technical Report, IST-UTL (2008)

[4] R.J. Bayardo. Mining the Most Interesting Rules. In *Proc. Int'l Conf. on Knowledge Discovery and Data Mining*, 145-154 (1999)

[5] R.J. Bayardo, R. Agrawal, D. Gunopulos. Constraint-Based Rule Mining in Large, Dense Databases. In *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, ACM Press, 145-154 (1999)

[6] R.J. Bayardo. The Many Roles of Constraints in Data Mining. In *SIGKDD Explorations*, vol. 4, nr. 1 (2002)

[7] J-F. Boulicaut, M. Klemettinen, H. Mannila, Querying inductive databases: a case study on the MINE RULE operator. *In Proc.Conf. Principles and Practice of Knowledge Discovery*, 194-202 (1998)

[8] R. Feldman, H. Hirsh. Mining Associations in Text in the Presence of Background Knowledge. In *Proc. Int'l Conf. on Knowledge Discovery in Databases and Data Mining*, 343-346 (1996)

[9] M. Garofalakis, R. Rastogi, K. Shim. SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. In *Proc. Very Large Databases Conf.*, 223-234 (1999)

[10] T.R. Gruber. A Translation Approach to Portable Ontology Specifications. In *Knowledge Acquisitions*, 5, 2. Academic Press, 21-66 (1998)

[11] J. Han, J. Pei, Y. Yin. *Mining Frequent Patterns Without Candidate Generation*, Technical Report TR-99-12, Simon Fraser University (1999)

[12] A. Maedche. *Ontology Learning for the Semantic Web*, Kluwer Academic Publishers (2002)

[13] J. Pei, J. Han. Constrained frequent pattern mining: a pattern-growth view. In *SIGKDD Explorations*, vol. 4 - 1. ACM Press, 31-39 (2002)

[14] R. Srikant and R. Agrawal. Mining Generalized Association Rules. In *Proc. Int'l Conference on Very Large Databases* (1995)

[15] R. Srikant, Q. Vu, R. Agrawal. Mining Association Rules with Item Constraints. In *Proc. Int'l Conf. on Knowledge Discovery and Data mining*, ACM Press, 67-73 (1997)

[16] Q. Yang, X. Wu. 10 Challenging Problems in Data Mining Research, in Int'l Journal of Information Technology & Decision Making, vol. 5, no. 4, pp. 594-604, World Scientific Publishing Company (2006)