XQuake as a Constraint-Based Mining Language

Valerio Grossi and **Andrea Romei** [1]

Abstract. XQuake is a language for data mining inspired by the inductive databases theory. This work extends XQuake with the definition of domain-specific constraints. An ontology is used to describe the domain knowledge. We give the main idea of the work-in-progress discussing its possibilities and advantages.

1 INTRODUCTION AND MOTIVATION

The use of constraints in a mining application has rapidly become a challenging topic for research community. The aim is to find a unified approach for the definition of a *constraint-based mining language*. The use of constraints helps the analyst to model mining problems by specifying desirable properties of the mined patterns. This feature involves several aspects. In fact, even when the set of *domain-specific constraints* is known, it is challenging to provide both a query language for formalizing them and a system that can process their properties in an efficient way. For example, the extraction of association rules implies the discovery of a large quantity of useless rules. An approach that permits to extract rules by specifying properties based on the analyst needs is required.

Our goal is the definition of a constraint-based language that enables the specification of domain-specific constraints with the aid of an ontology. Specifically, this paper proposes an extension of XQuake [5, 6], a language and system for supporting complex mining tasks out of XML data. Surprisingly, no previous work has addressed the use of both ontology-based mining constraints and, at the same time, a coherent and uniform framework for representing data, mining models, and the KDD process. Since XQuake already supports the second aspect, we concentrate on its extension to express domain-dependent constraints.

2 XQUAKE AT A GLANCE

XQuake is a language and system for programming data mining processes over native XML databases, in the spirit of inductive databases [4]. While we refer the reader to [5] for a detailed description, here we summarize its main features as follows: *(i)* it satisfies the closure principle, since both data and mining models are represented in XML; *(ii)* it represents the KDD process in a declarative way, by means of an XQuery [9] program extended with mining primitives; *(iii)* it permits to evaluate constraints via XQuery predicates, with the aid of a built-in library; *(iv)* the system architecture is conceived to automatically capture the proprieties of such constraints (e.g. monotonicity and anti-monotonicity), to be used directly during the extraction of the mining model.

[1] Department of Computer Science, University of Pisa, Italy. Email: {vgrossi,romei}@di.unipi.it

The question now is, why the choice of a mining language out of XML data? We provide three answers. *(i)* The amount of information XML-coded is steadily growing. *(ii)* XQuake takes advantage of the XML philosophy also for representing the results of the mining process, according to the PMML standard [7]. *(iii)* Ontologies represented as OWL [8] are de facto XML documents, and they can be used to describe a domain knowledge. This latter point is exactly what we introduce in the next section.

3 A SIMPLE APPLICATION SCENARIO

We propose a scenario involving a simple Market Basket Analysis application inspired by our previous work [2]. The study of constrained association rule mining and constraint programming is well-known in literature (e.g. see [3]). Here, we introduce a mining language for defining, in an uniform approach, both mining tasks and domain-specific constraints on the extracted knowledge.

Application scenario. Let us suppose that a domain expert investigates for a future promotional campaign during the holidays. The goal is to study the relation between the most frequent drinks promoted in Easter, and the most frequent cakes promoted in Christmas in the past.

Input data. Data of a national supermarket has been translated from a relational format (see [2]) to an XML format, stored in the native XML database of XQuake. An XML fragment of the `MBA.xml` document is reported below.

```
<MBA>
  <store @id="id00193">
    <purchase @date="05/01/2012">
      <item @qty="2" @price="3.5">vodka lemon</item>...
    </purchase>...
  </store>...
</MBA>
```

Here, each `<store>` tag represents a sequence of `<purchase>` elements, each encoding our transaction. The attribute `@date` denotes the date of that purchase. Each transaction is made up of a not empty set of purchased items, encoded in the `<item>` XML element. Two XML attributes denote the quantity and the price of that item. The resulting XML database contains 775,000 transactions and about 30,000 distinct items.

Domain knowledge. We enrich our data with the domain knowledge represented as an OWL document containing a description of each item and their hierarchical organization. A fragment of the `items.owl` ontology is depicted in Fig. 1.

Data constraints. Input data is filtered by considering *(i)* only purchases made between the 1^{st} December and the 1^{st} May and, *(ii)* for each transaction, only the items having a total price (i.e. `@qty` * `@price`) greater than 5 Euros.

Simple knowledge constraints. We aim at extracting association rules having exactly one item in the consequent, a minimum support of 0.5 and a minimum confidence of 0.7.

Advanced domain-specific constraints. Our purpose is to extract

attribute	value
hasColour	*neutral*
hasAlcoholic	*high*
hasFlavour	*lemon*
hasBrand	*Best*
hasSize	*70 cl*
hasPrice	*EUR 7.56*
hasPromotion	*Christmas*

Figure 1. A fragment of the hierarchical structure of `items.owl` (left). Data properties for 'Vodka Lemon' at the lower level (right).

```
(: Transactions, items and data constraints specif. :)
1.  for data $trans in doc("MBA")/store/purchase
        [@date > "01/01/2012" and @date < "01/05/2012"]
2.  let group $item := $trans/item[@price * @qty > 5.0]

(: Domain knowledge specification :)
3.  let supplementary $hasRec :=
      (<drink>{local:hasRec("Drink","Easter")}</drink>,
       <cake>{local:hasRec("Cake","Christmas")}</cake>)

(: Output constraints specification :)
4.  having (some $v in mfn:supplementary-body()
              satisfies $v/drink) and
5.    count(mfn:supplementary-head()) = 1 and
6.    mfn:supplementary-head[1]/cake and
7.    mfn:support() > 0.50 and mfn:confidence() > 0.70

8. return pmml
```

Figure 2. A possible implementation of the MBA scenario with XQuake.

association rules of the following form:

$$(i_1 \in \text{EasterDrink}) \text{ and } (i_2 \in \text{AnyItem}) \text{ and ... and } (i_n \in \text{AnyItem}) \Rightarrow (i_{n+1} \in \text{ChristmasCake}) \text{ [supp] [conf]}$$

where `EasterDrink` (resp. `ChristmasCake`) is the class of items that are drinks (resp. cakes) having an Easter (resp. Christmas) promotion, and `AnyItem` is the entire set of distinct items.

Our proposed solution. Among the plethora of languages for querying RDF and OWL documents, we integrate in XQuake an adaptation of [1], where, XQuery is employed for querying and reasoning with OWL and RDF ontologies. This solution appears to be adequate to our purposes for two main reasons. First, since XQuake extends XQuery with mining primitives, in turn, the implementation of an extension of XQuery for querying OWL documents is quite intuitive to be used by the user. Second, this permits to maintain the principle of closure that is at the basis of the inductive databases theory.

Fig. 2 provides a snapshot of the XQuake implementation, for further details see [5, 6] and [1]. The query is composed by four parts. First, the set of transactions (i.e. the `<purchase>` elements) is specified by the `for data` clause (row 1). Items of each transaction (i.e. the elements `<item>`) are defined in the `let group` clause (row 2).

Second, the `let supplementary` clause is evaluated for each frequent item (row 3). Here, a pair of XML elements are constructed by using the user-defined function reported below[2]:

```
declare function local:hasRec($class, $prop)
as xs:boolean {
  let $owl := owldoc("items.owl")
  return sw:hasSuperclass($owl, mfn:item(), $class)
         and sw:hasProperty($owl, mfn:item(),
                            $prop, "hasRecurrency")]
};
```

Third, the `having` clause operates on the output result (rows 4-7). It contains an XQuery predicate that is evaluated for each mined association rule. Basically, it evaluates the required constraints on the domain knowledge defined at row 3[3].

Finally, the `return` clause (row 8) is evaluated once, to return a PMML document containing the association rules satisfying the constraints.

Flexibility. It is important to note that our language is flexible both as far as a *modification of the domain knowledge* (since we use special constructs to traverse the ontology) and as far as the *introduction of different kinds of constraints* (since we use XQuery predicates for expressing them).

[2] If the current frequent item i (obtained through the built-in mining function `mfn:item()`) belongs to the superclass c in the ontology d, the built-in function `sw:hasSuperclass(d,i,c)` returns true. The build-in function `sw:hasProperty(d,i,p,v)` returns true if, for the item i, the data property p has value v in d

[3] The `mfn:supplementary-body()` (resp. `mfn:supplementary-head()`) built-in function returns the domain knowledge associated to the items of the body (resp. head) of the rule.

4 CONCLUSION

This work focuses on a general solution for XML data mining, and more generally, for a data mining query language. The solution proposed, even if at a preliminary stage, gives an additional improvement, in terms of the definition of complex domain-specific constraints. Our main advantages consist in a framework in which one can define, in an **expressive**, **declarative** and **uniform** way, both the KDD process and the mining constraints on the extracted knowledge. While the former aspect has been discussed in [6], this paper investigated the latter aspect. Basically, constraints are expressed through XQuery expressions supported by a built-in library. The proposed study can be refined in several directions. Here, we mention only the exploitation of domain-specific constraints for other kinds of models, such as sequential patterns and clusters.

ACKNOWLEDGMENTS

This work was supported by the EU FET-Open project "ICON - Inductive Constraint Programming", contract number FP7-284715.

REFERENCES

[1] J. M. Almendros-Jiménez, 'Querying and reasoning with RDF(S)/OWL in XQuery', in *Proceedings of the* 13^{th} *Web Technologies and Applications Conference (APWeb)*, pp. 450–459, (2011).

[2] A. Bellandi, B. Furletti, V. Grossi, and A. Romei, 'Ontology-driven association rules extraction: a case study', in *Proceedings of the International Workshop on Contexts and Ontologies: Representation and Reasoning (C&O:RR)*, (2007).

[3] T. Guns, S. Nijssen, and L. De Raedt, 'Itemset mining: A constraint programming perspective', *Artif. Intell.*, **175**(12-13), 1951–1983, (2011).

[4] T. Imielinski and H. Mannila, 'A database perspective on knowledge discovery', *Comm. Of The Acm*, **39**(11), 58–64, (1996).

[5] A. Romei and F. Turini, 'XML data mining', *Softw., Pract. Exper.*, **40**(2), 101–130, (2010).

[6] A. Romei and F. Turini, 'Programming the KDD process using XQuery', in *Proceedings of the* 4^{th} *International Conference on Knowledge Discovery and Information Retrieval (KDIR)*, pp. 131–139, (2011).

[7] The Data Mining Group. The Predictive Model Markup Language (PMML). Version 4.1. `www.dmg.org`, 2012.

[8] W3C. OWL Web Ontology Language. W3C Recommendation 10 February 2004. `www.w3.org/TR/owl-features`, 2004.

[9] W3C. XQuery 3.0: An XML Query Language. W3C Working Draft 14 December 2010. `www.w3.org/TR/xquery-30/`, 2010.