Anomaly detection through quasi-functional dependency analysis

Giulia Bruno[1], Paolo Garza[1], Elisa Quintarelli[2], Rosalba Rossato[2]
[1] Dipartimento di Automatica e Informatica – Politecnico di Torino
Italy
{giulia.bruno,paolo.garza}@polito.it
[2] Dipartimento di Elettronica e Informazione – Politecnico di Milano
Italy
{quintare,rossato}@elet.polimi.it

Journal of Digital Information Management

ABSTRACT: *Anomaly detection problems have been investigated in several research areas such as database, machine learning, knowledge discovery, and logic programming, with the main goal of identifying objects of a given population whose behavior is anomalous with respect to a set of commonly accepted rules that are part of the knowledge base. In this paper we focus our attention on the analysis of anomaly detection in databases. We propose a method, based on data mining algorithms, which allows one to infer the "normal behavior" of objects, by extracting frequent "rules" from a given dataset. These rules are described in the form of quasi-functional dependencies and mined from the dataset by using association rules. Our approach allows us to consequently analyze anomalies with respect to the previously inferred dependencies: given a quasi-functional dependency, it is possible to discover the related anomalies by querying either the original database or the association rules previously stored. By further investigating the nature of such anomalies, we can either derive the presence of erroneous data or highlight novel information which represents significant exceptions of frequent rules. Our method is independent of the considered database and directly infers rules from the data. The applicability of the proposed approach is validated through a set of experiments on XML databases, whose results are here reported.*

Categories and Subject Descriptors
H.2.8[Database Applications];Data Mining **H.2.3 [Languages**]: Query languages

General Terms
Data Mining, XML databases, Database research

Keywords: Data mining, Quasi-functional dependencies, Association rules, Anomaly detection.

Received 21 December 2006; Revised 08 Mar. 2007 and 26 April 2007; Accepted 03 May 2007

1. Introduction

Integrity constraints are used in structured and unstructured databases to capture real-world semantics observed in the modeled application domain. Thus, they are introduced at design time to describe a-priori knowledge, and consequently, the valid instances of the database are those satisfying simultaneously all constraints. Nevertheless, collected data can hide interesting and not known in advance information on unstated constraints. For example, it happens when data is the result of an integration process of several sources or when represents dynamic aspects.

The analysis of collected and heterogeneous data with the aim of detecting implicit information is clearly a fascinating task, which can be complex due to the size of datasets. In our opinion, there are two kinds of interesting knowledge to discover from data sources: (i) *frequent trends* and (ii) *anomalies with respect to such frequent trends*. In fact, many techniques have been exploited to discover frequent trends in data (e.g. mined association rules). On the other hand, infrequent behaviors can augment the knowledge about a data source. In particular, anomaly detection problems have been investigated in several research areas.

In all these situations the main goal is to identify objects of a given population whose behavior is anomalous with respect to a set of rules part of the knowledge base, usually expressed by means of some statistical kind of computation on the given population. These exceptional situations are usually referred as *outliers* in the literature. Indeed, "*an outlier is an observation that lies an abnormal distance from other values in a random sample from a population*" [11]. In this work, we define such infrequent situations as *anomalies*.

The problem of analyzing anomalies is interesting, since they represent errors or semantically correct, albeit infrequent, situations. In both cases detecting anomalies is a challenging task either to allow one to correct erroneous data or to investigate the meaning of exceptions.

In the database research field and in real applications, the problem of identifying and correcting anomalies has received a lot of attention in recent years [3][4][9]: outlier detection is used to examine the database in order to derive more complex forms of constraints. These tasks are of great importance also in a variety of application fields, especially for biological and clinical data, where it is important to detect anomalies in order to clean errors or discover exceptions needing a further investigation by specialists [7].

In this paper we consider the anomaly detection problem as being a part of the data mining approach and we introduce a proposal for discovering the nature of anomalies on datasets. This method is independent of the type of the target database. Our technique allows the extraction of frequent "rules", called *quasi-functional dependencies*, that represent the normal behavior of the considered domain application. A quasi-functional dependency is an approximate functional dependency derived from data [6], and represents an implication among attributes (in the context of relational databases) or among elements (with respect to XML documents), which frequently holds in the analyzed dataset. The detection of quasi-functional dependencies is performed by means of association rules: for each mined quasi-functional dependency, the set of association rules used to infer the dependency is queried in order to detect anomalies. In particular, the association rules with confidence that is lower than a fixed threshold are selected, and further investigated in order to characterize them by distinguishing interesting anomalies from erroneous data.

As an example, let us to consider a dataset describing research publications. Given a publication title, we expect to be able to identify always the corresponding year of publication. That is we expect the title to determine the publication year. A quasi-functional dependency constraint is violated only by few cases (or tuples, when working on relational datasets). The few cases that invalidate the exact functional dependency constraint highlight either errors or interesting situations. An anomaly to the functional dependency cited above, we derive from the data, would be the existence of two publications with the same title and different publication years. Is it an error or an exception?

The main contributions of our work can be summarized as follows.

- Functional dependencies and quasi-functional dependencies (which are not known a-priori) are directly inferred from the current instance of the considered data sources, by abstracting sets of similar previously mined association rules. This first step allows discovering new relationships among attributes (or elements), which occur in the analyzed database instances. If the size of the considered databases is statistically significant, the discovered relationships can be considered "independent" of the used instances and represent frequent rules holding on datasets having the same schema (i.e., they can suggest dependencies on the considered application domain).
- Quasi-functional dependencies are analyzed to single out anomalies in the datasets.
- Distinguishing erroneous data from interesting exceptions is performed by querying the set of mined association rules or the considered datasets and by analyzing the frequency of retrieved anomalies. We show the SQL and XQuery expressions that can be used to extract anomalous situations. The applicability of the approach has been validated through a set of experiments on XML databases.

The paper is organized as follows. Section 2 reports the definitions of functional dependencies and association rules and then describes quasi-functional dependencies and the methodology to detect them by means of association rules. The usage of quasi-functional dependencies to find anomalies is reported in Section 3; Section 4 presents the experiments performed on real XML databases. Section 5 discusses related works, while Section 6 draws conclusions and presents future developments of the proposed approach.

2. Quasi-functional dependencies

In the context of relational databases, functional dependencies are constraints among sets of attributes. Given a relation *R*, a functional dependency between two sets of attributes *X* and *Y* of a relation *R* imposes the following constraint on an instance *r* of *R*. Any two tuples t_1 and t_2 of r that agree on the value of *X* (i.e. $t_1[X] = t_2[X]$) must agree on the value of *Y* (i.e. they must also have $t_1[Y] = t_2[Y]$). The functional dependency between the two sets of attributes *X* and *Y* of *R* is denoted by $X \rightarrow Y$ [10].

Functional dependencies for XML have been defined in [5] by using *tree tuples,* which describe the paths of an XML document whose schema is expressed by a DTD (Document Type Definition).

Let us assume the following disjoint sets *El* of element names, *Att* of attribute names, *Str* of possible values of string-valued attributes, and *Vert* of node identifiers. All attribute names start with the symbol @, whereas symbols ε and S represent element type declarations EMPTY and #PCDATA.

In [5] a DTD is defined to be D = (E, A, P, R, r), where:

- $E \subseteq El$ is a finite set of element types,
- $A \subseteq Att$ is a finite set of attributes,
- P is a mapping from E to element type definitions: given $\tau \in E$, P(τ) = S or P(τ) is a regular expression α defined as $\alpha ::= \varepsilon \mid \tau' \mid \alpha|\alpha \mid \alpha, \alpha \mid \alpha^*$, where ε is the empty sequence, $\tau' \in E$, and "|", ",", and "*" denote union, concatenation, and the Kleene closure, respectively.
- R is a mapping from E to the powerset of A. If $@m \in R(\tau)$, @m is defined for τ.
- $r \in E$ and is called the element type of the root; without loss of generality, it is assumed that r does not occur in P(τ) for any $\tau \in E$.

Given a DTD D = (E, A, P, R, r), a string $w = w_1 \ldots w_n$ is a path in D if $w_1 = r$, w_i is in the alphabet of P(w_{i-1}), for each $i \in [2, n-1]$, and w_n is in the alphabet of P(w_{n-1}) or $w_n = @m$ for some $@m \in R(w_{n-1})$. The notation *paths(D)* stand for the set of all paths in D and *EPaths(D)* for the set of all paths that end with an element type (rather than an attribute or S).

A *tree tuple t* in a DTD *D* is a function that assigns to each path in *D* a value that represents a node identifier, or a string (for the content of leaf elements), in such a way that *t* represents a finite tree with paths from *D* containing at most one occurrence of each path.

Formally, given a DTD D = (E, A, P, R, r), a *tree tuple t* in D is a function from *paths(D)* to $Vert \cup Str \cup \{\perp\}$ such that:

- for $p \in EPaths(D)$, $t(p) \in Vert \cup \{\perp\}$, and $t(r) \neq \perp$
- for $p \in paths(D) - EPaths(D)$, $t(p) \in Str \cup \{\perp\}$
- if $t(p_1) = t(p_2)$ and $t(p_1) \in Vert$, then $p_1 = p_2$
- if $t(p_1) = \perp$ and p_1 is a prefix of p_2, then $t(p_2) = \perp$
- $\{p \in paths(D) \mid t(p) \neq \perp\}$ is finite.

Given a DTD *D*, a functional dependency over *D* is an expression of the form $S_1 \rightarrow S_2$, where S_1, S_2 are finite nonempty subsets of *paths(D)*. Given an XML tree *T* valid w.r.t. *D*, *T* satisfies $S_1 \rightarrow S_2$, if for every tree tuple t_1 and t_2 in *T*, $t_1[S_1] = t_2[S_1]$ and $t_1[S_1] \neq \perp$ imply $t_1[S_2] = t_2[S_2]$.

Association rules describe the co-occurrence of data items (i.e., couples of the form (attribute, value)) in a large amount of collected data [1]. Rules are usually represented as implications in the form $A \rightarrow B$, where *A* and *B* are two arbitrary sets of data items, such that $A \cap B = \varnothing$. The quality of an association rule is usually measured by means of *support (s)* and *confidence (c)*. Support corresponds to the frequency of the set $A \cup B$ in the dataset, while confidence corresponds to the conditional probability of finding *B*, having found *A* and is given by

$$c = \frac{s(A \cup B)}{s(A)}$$

As demonstrated in [6], a functional dependency can be detected from data by analyzing all the previously mined association rules (with a significant support) determining the correlation between the values of the attributes *X* and *Y*, and computing the dependency degree between them.

The *dependency degree* of a mined functional dependency between the sets *X* and *Y* is computed according to the following formula:

$$p = \sum_{i \in AR} s_i \cdot c_i$$

where *AR* is the set of all association rules relating attributes *X* and *Y*. s_i and c_i are respectively the support and confidence values of each rule. If the dependency degree is equal to one, we can state that a functional dependency has been mined. If the degree is close to one (does not decrease below a specific threshold), we define the involved attributes as *quasi-functional dependent*.

We introduce the notion of quasi-functional dependency by using a simple example.

2.1. Example

Let us now consider a relational database containing information about people of a country. In particular, we consider only a table *People* having as (partial) schema (*CitizenID*, *City*, *Province*, ..). Each person is described by some attributes, among which there are the city (*City*) and the province (*Province*) where he/she lives. Table 1 reports a subset of the relational table *People*.

CitizenID	*City*	*Province*	….
1	Piobesi	Torino	…
2	Piobesi	Torino	…
3	Piobesi	Torino	…
4	Piobesi	Cuneo	…
5	Piobesi	Cuneo	…
…	…	…	…
101	Recetto	Novara	…
102	Recetto	Milano	…
103	Recetto	Novara	…
…	…	…	…

Table 1. A portion of *People* table

During the design phase, some functional dependencies can be specified in order to describe the semantics of the considered application. In this example, the *People* table has as primary key the *CitizenId* attribute. Thus, we assume that the functional dependency *CitizenId* $\rightarrow$ *City* holds at design time.

Moreover, according to the nature and the content of the considered relation *People*, from a first analysis of the dataset, we expect to find the functional dependency *City* $\rightarrow$ *Province*. Hence, we expect exclusively to find association rules with confidence 100% between the attributes *City* and *Province*; on the contrary, we detect also association rules with confidence lower than 100%. Table 2 reports examples of those rules that allow us to infer a quasi-function dependency between *City* and *Province* (support and confidence are the values extracted by the mining algorithm applied on the entire dataset only sketched in Table 1).

Body	**Head**	**Sup**	**Conf**
City=Piobesi	Province=Torino	45.1%	75.2%
City=Piobesi	Province=Cuneo	14.9%	24.8%
City=Recetto	Province=Novara	28.7%	99.7%
City=Recetto	Province=Milano	0.1%	0.3%

Table 2. Examples of association rules found in the database

By computing the dependency degree between *City* and *Province* (by considering all the extracted rules and not only those that are reported in Table 2), we obtain a value of 0.95. Supposing that this value is high enough, we can state that there is a quasi-functional dependency between the two attributes, as reported in Table 3.

Quasi-functional dependency	***P***
City $\rightarrow$ Province	0.95

Table 3. Dependency degree between *City* and *Province*.

3. Anomaly detection

In this section we describe our proposal to retrieve anomalies and investigate their nature. After detecting quasi-functional dependencies that involve two (or more) attributes, we have two possibilities:

1. we query the original datasets to extract the instances that violate the dependencies;
2. for each quasi-functional dependency relating the sets *X* and *Y*, we query all the stored association rules that involve *X* and *Y*, with a low confidence (i.e. with a confidence lower that a fixed threshold). Such rules allow us to infer that the dependency is a quasi-functional dependency and not an exact one (see the formula in Section 2 to compute the dependency degree).

For investigating the nature of detected anomalies, we analyze the confidence of each rule that involves the attributes of the quasi-functional dependency. If this value is very low (compared to the confidence value of the other rules), we can conclude that this is an error, otherwise it is a correct exception. We will further explain the previous example to clarify these concepts.

Another possibility to distinguish semantically correct anomalies from errors, with a semi-automatic procedure, is to apply the query to detect anomalies to other databases in the same application domain. By comparing the results of the distributed query, we can infer that an anomaly is an error if it does not occur in more than one data source.

Otherwise, if we discover the same anomaly more times, we can assume it is an admissible situation, which needs a further investigation. To this aim, a rewriting procedure can be required to apply the query to several data sources represented by different models and with different schemas.

3.1. Example

With respect to the relation *People* introduced in Example 2.1, we now investigate the rules that involve *City* and *Province* with a low confidence (i.e., lower than 50%). We find the following rules:

1. City=Piobesi $\rightarrow$ Province=Cuneo [s=14.9%, c=24.8%]
2. City=Recetto $\rightarrow$ Province=Milano [s=0.1%, c=0.3%]

Both of them represent interesting cases. The first one is a correct, albeit infrequent, relationship, since there are two cities with the same name (Piobesi) in two different provinces (Cuneo and Torino). The second one is an error, since there is only a city named Recetto in the Novara province. We are able to distinguish between the two cases by analysing the confidence value. The value of the second rule (0.3%) is at least one order of magnitude smaller than the average confidence values of the other rules. Hence, we can conclude that this is an error.

3.2. Errors vs. interesting outliers

In this section, we describe in more detail our proposal to distinguish erroneous situations from interesting anomalies. Once the association rules and the related quasi-functional dependencies have been mined from a dataset, we store them either in a relational database or as XML documents [16] on the basis of the data sources we are analyzing. The following tables form the relational databases of rules (**RULE** and **ITEMR**) and quasi-functional dependencies (**CONSTR** and **ITEMD**).

```
RULE (ID, Sup, Conf, NumItemHead,NumItemBody)
ITEMR (IDRule, DataElement, Value, Head_Body)
CONSTR (ID, P_Degree, N_ItemHead, N_ItemBody)
ITEMD (IDDep, DataElement, Head_Body)
```

In particular, each tuple of the table **RULE** keeps trace of the identifier (ID) of an extracted association rule, and stores its support (Sup), confidence (Conf), the number of item appearing in the head (NumItemHead) and in the body (NumItemBody) of the rule itself. The table **ITEMR** allows one to store the structure of each extracted association rule. In particular, IDRule is the identifier of a rule, DataElement and Value describe the name and the value of an attribute/element that appears in the considered rule, and Head_Body is a flag attribute that denotes if the considered attribute/element appears in the head or in the body of the current rule.

Analogously, the table **CONSTR** stores the identifier of a quasi-functional dependency (ID), its dependency degree (P_Degree), and the number of attributes that appear in the head (N_ItemHead) and in the body (N_ItemBody) of the dependency itself. The tuples of **ITEMD** table store the structures of quasi-functional dependencies.

Given a quasi-functional dependency $X \rightarrow Y$ with a dependency degree p, we can discover the related anomalies by applying one of the following two queries.

1. By querying the original database (where the rules have been mined) it is possible to extract the tuples violating the exact functional dependency and thus, satisfying the quasi-functional dependency. For simplicity reasons, we suppose to have a super table R representing the database. Thus, the SQL query we have to apply is the following.

```
SELECT X, Y FROM R
WHERE X IN
        (SELECT X FROM R
         GROUP BY X
         HAVING COUNT(DISTINCT(Y)>1))
```

   For distinguishing erroneous data from interesting exceptions, we analyze the frequency of retrieved anomalies or we apply the same queries to other datasets in the same application filed.

2. By using the tables that store the association rules, we apply the following query to retrieve the list of attributes and the respective values of association rules with a low confidence.

```
SELECT IDRule, DataElement, Value, Head_Body, Conf FROM ITEMR
WHERE IDRule IN
   (SELECT ID FROM RULE JOIN ITEMR ON ID=IDRule
    WHERE Conf<Threshold AND NumItemHead =1 AND
          NumItemBody =1
   AND DataElement='X' AND HEAD_BODY='body'
    INTERSECT
   SELECT ID FROM RULE JOIN ITEMR ON ID=IDRule
    WHERE Conf<Threshold AND NumItemHead =1 AND
          NumItemBody =1
   AND DataElement='Y' AND HEAD_BODY='head')
 ORDER BY IDRule
```

   Once again, by analyzing the confidence value of the retrieved anomalies we can distinguish between errors and interesting outliers.

When dealing with XML-based data sources, in order to have a homogeneous representation of data and queries, the set of association rules can be represented as an XML document with the following DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT RuleSet (AssRule+)>
<!ELEMENT AssRule (RuleBody, RuleHead)>
<!ATTLIST    AssRule NumberItemHead CDATA #REQUIRED
             NumberItemBody CDATA #REQUIRED
             support CDATA #REQUIRED
             confidence CDATA #REQUIRED>
<!ELEMENT RuleBody (item+)>
<!ELEMENT RuleHead (item+)>
<!ELEMENT Item (#PCDATA)>
<!ATTLIST   Item DataElement CDATA #REQUIRED>
```

In this work, we consider a subset of the functional dependencies defined for XML in [5]; in particular, we infer only dependencies between sets of paths that reach leaf nodes of the considered XML document. The quasi-functional dependencies are represented by an XML documents with the following DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT DepSet (Constraint+)>
<!ELEMENT Constraint (Body, Head)>
<!ATTLIST Constraint NumItemHead CDATA  #REQUIRED
           NumItemBody CDATA #REQUIRED
          Prec_degree CDATA #REQUIRED>
<!ELEMENT Body (item+)>
<!ELEMENT Head (item+)>
<!ELEMENT Item (#PCDATA)>
```

The XQuery [17] expression to retrieve the anomalies from the document of association rules (i.e., the rules with a low confidence) is the following.

```
FOR $r IN doc("AssociationRule.xml")//AssRule
WHERE $r[@confidence<Threshold] AND
    $r[@NumberItemHead=1] AND $r[@NumberItemBody=1]
AND $r/RuleBody/item/@DataElement='X' AND
   $r/RuleHead/item/@DataElement='Y'
return $r
```

In order to discover an anomaly from the XML documents used to mine the association rules relating the elements X and Y, we should write an XQuery expression to find out X elements which are related to more than one Y element. We do not explicitly indicate here this possibility because the syntax of the XQuery expression is strictly related to the DTD of the data source.

The described SQL and XML queries can be generalized when dependencies with more that two items; indeed, they allow domain experts to retrieve anomalies with respect to quasi-functional dependencies and consequently to concentrate on a small portion of data (the anomalies) to find out interesting outlier situations, by manually or semi-automatically discarding errors.

4. Experimental results

The experiments have been performed on a 3.2GHz Pentium IV system with 2GB RAM, running Kubuntu 6.10. For the itemset and association rule extraction we use a publicly available version of Apriori [1] downloaded from [18].

We validated our approach by performing a set of experiments on the TPC-H database [15], the DBLP database [14], and 8 UCI databases [25] saved in XML format . TPC-H is a suite of synthetic datasets generated from the TPC-H relational tables and saved in XML format[1]. We consider two tables (ORDER and LINEITEM), both of them stored as an XML file. Each transaction in the ORDER XML file includes elements that characterize an order (Orderkey, Orderstatus, Custkey, etc.), while the LINEITEM XML file includes a transaction for each LineItem value (characterized by the elements Orderkey, Partekey, Linestatus, Receipt date, Commit date, etc.).

DBLP is a real-life dataset where each transaction includes elements that characterize an article (Authors, Title, Year, Conference name, etc.). The 8 selected UCI databases cover a wide range of different real life domains (census data, solar flare data, etc.). The main characteristics of each database are reported in Table 4.

Since anomaly detection in XML is a more challenging problem, we concentrate our analysis on XML datasets. However, the proposed approach can also be directly applied to any relational database.

4.1. Quasi-functional dependency and anomaly detection

We applied our method to the TPC-H, DBLP, and UCI databases and we detected some quasi-functional dependencies; the number of quasi-functional dependencies with a high value of *p* (i.e., $0.95<p<1$), extracted for each considered database, is reported in Table 5.

The number of extracted quasi-functional dependencies depends on the considered datasets. Some databases are characterized by several quasi-functional dependencies (e.g., chess), while some others are characterized by few or zero quasi-functional dependencies (e.g., zoo, voting) with a high value of *p*.

The number of functional dependencies is related, *in a not predictable way*, to the number of attributes, to the domain of each attribute, and to the domain of the considered database.

Dataset	Number of quasi-functional dependencies with $1>p>0.95$
TPC-H order.xml	1
TPC-H lineitem.xml	4
DBLP	1
Census	5
Chess	181
Flare	12
Mushroom	21
Nursery	0
Tic-tac-toe	0
Voting	0
Zoo	2

Table 5 – Quasi-functional dependencies with 0.95<p<1 in the TPC-H, DBLP, and UCI databases

Since the number of extracted quasi-functional dependencies is relatively high, we report detailed results only for the quasi-dependencies extracted from the DBLP and the TPC-H datasets. We decided to analyze in detail the TPC-H and the DBLP datasets since they cover well-known domains; however, similar results have been obtained for the UCI datasets.

Dataset	File Size (MB)	Description	Number of elements
TPC-H order.xml	26	Order table in XML	9
TPC-H lineitem.xml	154	Lineitem table in XML	16
DBLP	259	DBLP XML records	14
Census	3.8	Census data	15
Chess	0.25	Board configurations at the end of chess games	37
Flare	0.028	Solar flare data	13
Mushroom	0.365	Mushroom records drawn from the Audubon Society Field Guide to North American Mushrooms	23
Nursery	1.1	Nursery-school application ranking	9
Tic-tac-toe	0.026	Complete set of possible board configurations at the end of tic-tac-toe games	10
Voting	0.018	1984 United Stated Congressional Voting Records	17
Zoo	0.004	Classification of animals	18

Table 4 – Main database characteristics

[1] The scale factor parameter has been set to 0.05. The number of generated tuples is related to the scale factor.

Table 6, Table 7, and Table 8 report the quasi-functional dependencies with a high value of *p* (i.e., 0.95<*p*<1) for the DBLP and the TPC-H datasets.

Quasi-functional dependency	*P*
Title $\rightarrow$ Year	0.981

Table 6 – DBLP: dependency degree between elements in the DBLP database.

Quasi-functional dependency	*P*
Lineitem-receipt-date $\rightarrow$ Linestatus	0.996
Lineitem-returnflag $\rightarrow$ Linestatus	0.987
Lineitem-commit-date $\rightarrow$ Linestatus	0.981
Orderkey $\rightarrow$ Linestatus	0.987

Table 7 – TPC-H: dependency degree between elements in the LINEITEM XML document.

Quasi-functional dependency	*P*
Order-date $\rightarrow$ Order-status	0.978

Table 8 - TPC-H: dependency degree between elements in the ORDERS XML document.

After retrieving the quasi-functional dependencies, we have applied the anomaly detection approach to their XML representation. As an example, the XML representation of the dependencies mined from the LINEITEM XML document of the TPC-H dataset is reported in Table 9.

Analogously to quasi-functional dependencies analysis, we report detailed results only for the anomalies of the DBLP and the TPC-H datasets. However, similar analyses can be performed for the UCI datasets.

In our experiments we have first investigated the DBLP dataset, composed by a number N of articles (N= 618145). By considering the quasi-functional dependencies reported in Table 6, we examined the association rules with a low confidence that involve the elements *Title* and *Year*. In particular, we applied the query proposed in Section 3.2 and extracted all the rules with a confidence lower than 100%; some of the most representative rules are reported in Table 10. In order to preserve authors' privacy we do not report real values of the elements *Title*, *Author*, and *Booktitle*.

Title	*Year*	Support	Confidence
Title 1	1992	1/N	50%
Title 1	1994	1/N	50%
Title 2	1995	1/N	50%
Title 2	1996	1/N	50%

Table 10 - Examples of association rules found in the DBLP database by considering the Title and Year attributes

The rules with a confidence lower than 100% highlight articles with the same title but different years. It is a rare

```
<?xml version="1.0" encoding="UTF-8"?>
<DepSet>
  <Constraint  NumItemHead = “1”  NumItemBody = “1”   Prec_degree = “0.996” >
                    <Body>
                        <item> Lineitem-receipt-date </item>
                    </Body>
                    <Head>
                        <item> Linestatus </item>
                    </Head>
  </Constraint>
  <Constraint  NumItemHead = “1”  NumItemBody = “1”   Prec_degree = “0.987” >
                    <Body>
                        <item> Lineitem-returnflag </item>
                    </Body>
                    <Head>
                        <item> Linestatus </item>
                    </Head>
  </Constraint>
  <Constraint  NumItemHead = “1”  NumItemBody = “1”   Prec_degree = “0.981” >
                    <Body>
                        <item> Lineitem-commit-date </item>
                    </Body>
                    <Head>
                        <item> Linestatus </item>
                    </Head>
  </Constraint>
  <Constraint  NumItemHead = “1”  NumItemBody = “1”   Prec_degree = “0.987” >
                    <Body>
                        <item> Orderkey </item>
                    </Body>
                    <Head>
                        <item> Linestatus </item>
                    </Head>
  </Constraint>
</DepSet>
```

Table 9. XML representation of quasi-functional dependencies about LINEITEM XML document

situation, since articles usually have different titles. We find two causes of such interesting anomaly.

The first one is when the same article is published with the same title on a conference and on a journal in different years, as reported in Table 11. The second one is when the same article is published with the same title on two different conferences in different years, as reported in Table 12.

In our opinion, both cases represent interesting exceptions.

All the association rules with a confidence lower than 100% that involve the elements *Title* and *Year* highlight anomalies that can be classified in one of the two classes discussed above.

```
<Article>
        <Title>Title 1</Title>
        <Author>author X</Author>
        <Author>author Y</Author>
        <Year>1992</Year>
        <Booktitle>ConferenceX</Booktitle>
</Article>
..
..
<Article>
        <Title>Title 1</Title>
        <Author>author X</Author>
        <Author>author Y</Author>
         <Year>1994</Year>
        <Booktitle>JournalY</Booktitle>
```

Table 11. Detected anomaly: the same article is published with the same title on a conference and on a journal.

```
<Article>
        <Title>Title 2</Title>
        <Author>author X</Author>
        <Author>author Y</Author>
        <Year>1995</Year>
        <Booktitle>ConferenceX</Booktitle>
</Article>
..
..
<Article>
        <Title>Title 2</Title>
        <Author>author X</Author>
        <Author>author Y</Author>
        <Year>1996</Year>
        <Booktitle>ConferenceY</Booktitle>
</Article>
```

Table 12. Detected anomaly: the same article is published with the same title on two different conferences.

With regard to the TPC-H dataset, five quasi-functional dependencies have been extracted. A quasi-functional dependency is related to the pair of elements (Order-date,Order-status), while the other four quasi-functional dependencies are related to the value of the *Linestatus* element (LINEITEM XML document), which depends on the order date and other dates (Commit date, Receipt date, etc.). For example, all the order lines characterized by the same commit date are almost all characterized by the same value of the *Linestatus* element.

Those lines which are characterized by a status value that is different from the most common value are considered as anomalies (they need the same time interval of the other Lineitem elements to be committed, delivered, etc.).

Our approach highlights these anomalies. For example, we extracted the following rules:

Lineitem-commit-date='4/23/1995" $\rightarrow$ Linestatus='F' [s=0.037%, c=89.5%]

Lineitem-commit-date='4/23/1995" $\rightarrow$ Linestatus='O' [s=0.004%, c=10.5%]

The order status (*Linestatus*) can assume the value "finished" (F) or "open" (O). The first rule represents normal cases (almost all the lines that have been committed in '4/23/1995' have been closed), while the second rule highlights anomalies cases (old lines that have not been already closed).

Similar considerations are valid also for the other quasi-functional dependencies reported in Table 7 and Table 8.

4.2. Extraction time for quasi-functional dependencies

The extraction task is composed by two main steps: association rule mining and dependency degree computation.

The execution time of both steps depends on the number of elements (attributes), and in particular on the number of possible pairs of elements (i.e., the number of combinations). Hence, it increases approximately linearly with the number of possible pairs of elements. More precisely, the execution time increases almost linearly with the number of extracted rules. The execution time of the two steps for each dataset is reported in Table 13.

The proposed approach supposes that no a-priori knowledge is available. Since it analyzes all the possible pairs of elements, these results represent the execution time required to extract all possible association rules composed by two items and the related quasi-functional dependencies.

Database	Number of rules	Rule mining (s)	Dependency degree computation (s)
TPC-H order.xml	2,520,432	46	224
TPC-H lineitem.xml	27,921,090	499	4241
DBLP	32,940,372	525	3194
Census	776,436	13.772	27.645
Chess	5,164	0.312	0.035
Flare	1,522	0.047	0.014
Mushroom	7,054	0.382	0.073
Nursery	846	0.141	0.011
Tic-tac-toe	756	0.029	0.007
Voting	2328	0.052	0.019
Zoo	4804	0.094	0.073

Table 13. Rule mining and dependency degree computation execution time

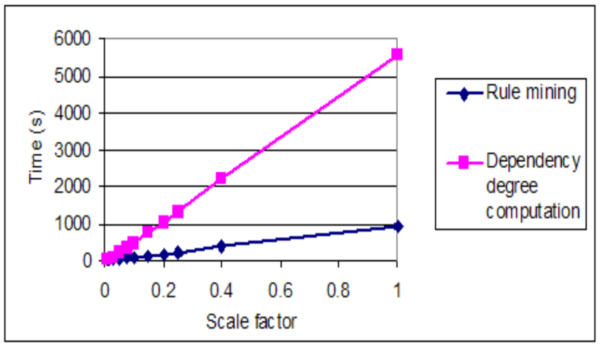


Figure 1. TPC-H orders.xml: execution time when varying the scale factor.

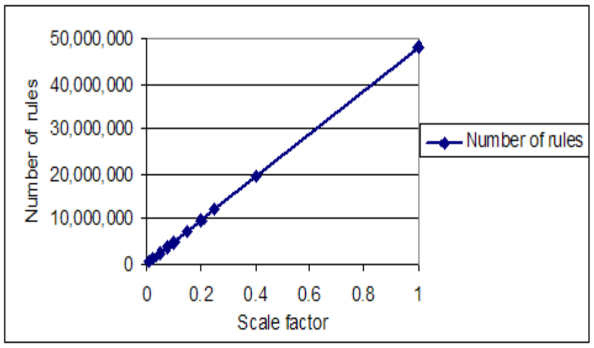


Figure 2. TPC-H orders.xml: number of extracted rules when varying the scale factor.

However, if some a-priori knowledge is available, we can exploit it and reduce the execution time, e.g., if we know that some elements are not related with any other element, we can remove them before the rule mining step.

We performed a set of experiments to analyze the scalability of the proposed approach. In particular, we analyzed the extraction time when varying the scale factor of the TPC-H dataset. Figure 1 shows the execution time of the two main steps, i.e., rule mining and dependency degree computation, for the TPC-H order.xml database.

Both the execution time of the rule mining and the execution time of the dependency degree computation task increase almost linearly with the scale factor (i.e., with the number of records).

Figure 2 shows the number of extracted rules when varying the scale factor; also the number of extracted rules increases almost linearly with the scale factor.

Hence, we can conclude that the extraction time for quasi-functional dependencies and the number of extracted rules are characterized by the same trend when varying the file size.

5. Related work

In this section we discuss the main relevant work dealing with the anomaly or outlier detection problems.

Statistical approaches were the first proposals for the outlier detection techniques; the basic component is a probabilistic model that can be either a-priori established or automatically derived by analyzed data. This model can also include unknown parameters that are determined by means of data mining techniques. Objects that do not suit the probabilistic model are considered as outliers [8] [12][19][20].

Once a probabilistic model is given or constructed, statistical methods are very efficient; however, they have several disadvantages which make their use in data mining systems inconvenient. First, they strictly require the usage of a data model; in the case the model is parameterized, complex procedures for finding values they assume are necessary. Moreover, it is not guaranteed that the data of the observed population match the assumed distribution law if there is no estimate of the distribution density based on the empirical data.

Most popular approaches for the outlier detection problems are the so-called *distance-based methods* [2] [13]; outliers are quantitatively characterized by means of a distance function between objects of the database. The main advantage of distance-based approaches is that a probabilistic model is not constructed, but the majority of these algorithms have a quadratic complexity. Moreover, when they are applied to real information systems, containing heterogeneous data with a complex structure, the definition of a distance function between such objects is a nontrivial problem; methods based on kernel functions [21] and fuzzy approaches [22] can be used to achieve this goal.

Several other interesting approaches have already been developed to solve outlier detection by means of database [9] and data mining techniques [4][7], including machine learning [23], knowledge discovery [24], and logic programming [3].

In [3], outliers have been formalized in the context of logic programming-based knowledge systems. The authors propose a basic framework where observations (outliers) are described by means of a set of facts encoding some aspects of the current status of the world, while the background knowledge of the system is described by means of a logic program. Outliers are identified on the basis of some disagreement with the background knowledge and supported by some evidence in the observed data, called witness sets. Moreover, the basic framework is extended to the case the observations of the current status of the world has to be captured by means of more complex form, i.e. no ground facts but logical rules.

The main difference of our approach is that we discover “rules” directly from data and do not consider them as part of the knowledge base. The rules we infer have the form of quasi-functional dependencies and can be easily written as a logic program. For example, if we consider the relational table *People* in Table 1, we can encode each attribute value as ground fact of the unary predicates *CitizenID*, *City*, and *Province*, respectively. Moreover, we can translate each pair of values of the *City* and *Province* attributes, appearing in tuples of the *People* table, as facts of the binary predicate IN(X,Y). We infer from association rules the quasi functional dependency *City* ⇀*Province*, which corresponds to the following two rules:

CityIN1Province(X) ← City(X), ¬ not CityIN2Province

CityIN2Province(X) ← City(X), Province(Y),Province(Z), Y ≠ Z, IN(X,Y), IN(X,Z)

Our approach infers the sets {IN(Piobesi,Torino),IN(Piobesi,Milano)}and {IN(Recetto,Novara),IN(Recetto,Milano)} as anomalies of the two logic rules.

In our opinion, it could be interesting to extend the applicability of our approach to more general “rules”, not only functional dependency, which can be abstracted from mined association rules.

The inference of functional dependencies from data in order to detect anomalies has already been applied in [7]. The authors present results on protein structure databases, without formalizing the concepts of quasi-functional dependency and without defining a general method to retrieve the anomalies. In this paper we formalize the notion of quasi-functional dependency and propose a method to retrieve anomalies and to discover their possible nature. We mainly focalize on the application of the proposed method to XML databases.

In [9] the authors introduce the notion of pseudo-constraints, which are predicates having significantly few violations. The authors use this pattern to identify rare events in databases. The aim of the work is similar to our main purpose: they define this data mining pattern to detect interesting anomalies. However, our approach differs from [9] for two reasons. Firstly, they define the notion of pseudo-constraint on the Entity-Relationship (ER) model, whereas we use association rules to define a quasi-functional constraint. Then, they focus on cyclic pseudo-constraints and propose an algorithm for extracting this kind of cyclic pattern. On the contrary, our notion of dependency is an implication between sets of elements and is not related to the structure of the data source used to mine the pattern. Moreover, we do not apply our notion only to relational databases: we have generalized the method to XML datasets and experimental results confirm the generality of the proposal.

6. Conclusions and future work

Anomaly detection problems have been investigated in several research areas; most approaches single out abnormalities from a given population by considering statistical characteristics.

According to this perspective, in this work we have described an approach to discover anomalies on data by considering quasi-functional dependencies mined by using association rules. The proposal defined in the paper is general and can be applied in any context.

As an ongoing work we are extending our approach to more complex dependencies by considering graph-based association rules in deeply nested XML documents (and not only simple patterns). Moreover, we are applying this technique to biological datasets where quasi-functional dependencies can be mined and anomalies must be detected. We are developing a tool that first mines functional dependencies from a database and then, in the case of quasi-functional dependencies, automatically rewrites the query to single out anomalies, in order to apply it to other relational and XML datasets.

Acknowledgment

We like to thank Elena Baralis and Letizia Tanca for the precious collaboration in the setting foundations of our research. A special thank to Cristiana Bolchini and Daniele Apiletti for their help.

References

[1] Agrawal, R., Srikant, R (1994).Fast algorithms for mining association rules in large data-bases. *In:* International Conference on Very Large Data Bases, 478-499. Morgan Kaufmann.

[2] Aggarwal , C. C., Yu, P. S (2001). Outlier Detection for High Dimensional Data. Proceedings of SIGMOD Conference, 37-46.

[3] Angiulli, F., Greco, G., Palopoli, L (2006). Outlier Detection by Logic Programming. *ACM Transaction on Computational Logic*. To appear.

[4] Angiulli, F., Pizzuti, C (2005). Outlier mining in large high dimensional datasets. IEEE Transactions on Knowledge and Data Engineering, 17 (2) 203-215.

[5] Arenas, M., Libkin, L (2004). A Normal Form for XML Documents. *ACM Transactions on Database Systems,* 29 (1) 195-232.

[6] Baralis, E., Garza, P., Quintarelli, E., Tanca, L. (2004). Answering Queries on XML Data by means of Association Rules. Current Trends in Database Technology, Vol. 3268. Springer-Verlag.

[7] Apiletti, D., Bruno, G., Ficarra, E., Baralis, E. (2006). Data Cleaning and Semantic Improvement in Biological Databases. *Journal of Integrative Bioinformatics*., 3(2).

[8] Breunig, M., Kriegel, H., Hg, R., Sander, J. (2000). LOF: Identifying density-based local outliers, *In*: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, 93-104.

[9] Ceri, S., Di Guinta, F., Lanzi, P. (to appear). Mining Constraint Violations. ACM-Transactions on Database Systems.

[10] Elmasri, R., Navathe, S. B. (2005). Foundamentals of DATABASE SYSTEMS. Pearson, Addison Wesley.

[11] NIST/SEMATECH e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/, 2007.

[12] Knorr, E., Hg, R., Tucakov, R (2000). Distance-Based Outlier : Algorithms and Applications. VLDB Journal, 8(3-4), 237-253.

[13] Papadimitriou, S., Kitagawa, H., Gibbons, P., Faloutsos, C (2003). LOCI: Fast outlier detection using the local correlation integral. Proceedings of 19th International Conference on Data Engineering (ICDE), 315-326.

[14] Lay, M. The DBLP bibliography server, http://dblp.uni-trier.de/xml.

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

[16] World Wide Web Consortium (1998). Extensible Markup Language (XML) 1.0, http://www.w3C.org/TR/REC-xml/.

[17] World Wide Web Consortium (2002). XQuery: An XML Query Language, http://www.w3C.org/TR/REC-xml/.

[18] Goethals, B., Zaki, M (2003). Advances in frequent itemset mining implementation: Report on FIMI'03. SIGKDD Explorations Newsletter 6 (1) 109-117.

[19] Han, J., Kamber, M (2000). Data Mining: Concepts and Techniques. Morgan Kaufmann, 2000.

[20] Yamanishi, K., Takeichi, J., Williams, G. (2000). On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms. Proceedings of 6Th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 320-324.

[21] Scholkopf, B., Smola, A.J (2002). Learning with Kernels. Cambridge, London, MIT.

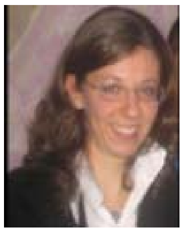
[22] Girolami, M (2001). Mercer Kernel Based Clustering in Feature Space. *IEEE Transactions on Neural Networks*, 13 (4) 780-784.

[23] Breitenbach, M., Grudic, G. Z. (2005). Clustering through ranking on manifolds, In: Proceedings of International Conference on Machine Learning (ICML 2005), 73-80.

[24] Abe, N., Zadrozny, B., Langford, J. (2006). Outlier detection by active learning. Proceedings of International

Conference on Knowledge Discovery and Data Mining (KDD 2006), 504-509.

[25] Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.



Giulia Bruno is a Ph.D. student in the Database and Data Mining Group at the Dipartimento di Automatica e Informatica of the Politecnico di Torino since January 2006. She holds a Master degree in Computer Engineering from Politecnico di Torino in Septembre 2005. She is currently working in the field of data mining and bioinformatics. Her activity is focused on the analysis of gene expression data, gene network modelling, data cleaning and semantic information discovery. Her research activities are also devoted to classification of physiological signals to monitor patient conditions for clinical analysis.

Paolo Garza is a research assistant in the Database and Data Mining Group at the Dipartimento di Automatica e Informatica of the Politecnico di Torino since January 2005. He holds a Master degree and a Ph.D in Computer Engineering, both from Politecnico di Torino. His current research interests include Data Mining and Database Systems. In particular, he has worked to supervised classification of structured and unstructured data, clustering, and itemsets mining algorithms.

Elisa Quintarelli received her master's degree in Computer Science from the University of Verona, Italy. On January 2002 she completed the Ph.D. program in Computer and Automation Engineering at Politecnico di Milano and is now an assistant professor at the Dipartimento di Elettronica e Informazione, Politecnico di Milano. Her main research interests concern the study of efficient and flexible techniques for specifying and querying semistructured and temporal data, the application of data-mining techniques to provide intensional query answering. More recently, her research has been concentrated on context aware data management.

Rosalba Rossato received the master degree in Computer Science from the University of Verona; in 2006 she received the Ph.D. degree in Computer Science from the same university. She was from January to May 2006 a Post-Doc fellow at the Department of Computer Science of University of Verona. Since June 2006, she is a Post-Doc fellow at the Dipartimento di Elettronica e Informazione, Politecnico di Milano. Her main research interests are related to the database field and include the management of temporal information, the application of data-mining techniques to the management of structured and semistructured information. Recently, her research has been devoted to the context aware data management.