THE NATIONAL UNIVERSITY
of SINGAPORE

School of Computing
Lower Kent Ridge Road, Singapore 119260

TRB6/07

Correlation-based Attribute Outlier Detection in XML

Judice L. Y. KOH , Mong Li LEE, Wynne HSU and Wee Tiong ANG

June 2007

Technical Report

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

JAFFAR, Joxan
Dean of School

Correlation-based Attribute Outlier Detection in XML

Judice L. Y. Koh[1] , Mong Li Lee[1], Wynne Hsu[1], and Wee Tiong Ang[2]

[1] School of Computing, National University of Singapore,
3 Science Drive 2, Singapore 119260
[2] Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119260
judice.koh@utoronto.ca, {leeml, whsu}@comp.nus.edu.sg, wtang@i2r.a-star.edu.sg

Abstract. Outlier detection - the problem of identifying deviating patterns in data sets, has important applications in data cleaning, fraud detection, and stock market analysis. Compared to relational data models, the hierarchical structure of semi-structured data such as XML provides semantically meaningful neighborhoods advancing existing outlier detection methods. In this paper, we propose a systematic framework - XODDS (XML Outlier Detection from Data Subspace) for outlier detection in XML data. XODDS utilizes the correlation between attributes to adaptively identify outliers.

XODDS consists of four key steps: (1) attribute aggregation defines summarizing elements in the hierarchical XML structures, (2) subspace identification determines contextually informative neighborhoods for outlier detection, (3) outlier scoring computes the extent of outlier-ness using correlation-based metrics, and (4) outlier identification adaptively determine the optimal thresholds distinguishing the outliers from non-outliers. Experimental results on both synthetic and real-world data sets indicate that XODDS is effective in detecting outliers in XML data.

Keywords: Outlier Detection, XML, Data Mining.

1 Introduction

Outlier detection is the problem of identifying deviations from the general patterns characterizing a data set. Detecting outliers is important in data cleaning, where outliers are often data noise or errors diminishing the accuracy of data mining. Outlier detection is also the core of applications such as fraud detection, stock market analysis, intrusion detection, marketing, network sensors, and email spam detection, where irregular patterns entail special attention.

Recent years have seen a rapid proliferation of semi-structured data models. For example, XML (eXtensible Markup Language) data models are increasingly popular as new standards for data representation and exchange on the WWW. Although outlier detection methods have been established for relational data, adapting them directly to XML data is limited because XML and relational data models differ in several aspects.

First, XML data contain multiple levels of nested elements (or attributes) organized in a tree-based structure, whereas relational data models have a flat tabular structure.

In fact, the hierarchical structure of XML data gives rise to a well-defined directionality lacking in relational data. Also, the modeling objectives for XML and relational data differ, and therefore the relationships represented. In relational data models, the primary-foreign key relationships between entities form the basis for data normalization and referential integrity. On the contrary, relationships between the XML elements are encoded in hierarchies, often with direct semantic correspondence to the real-world relations such as containment and composition.

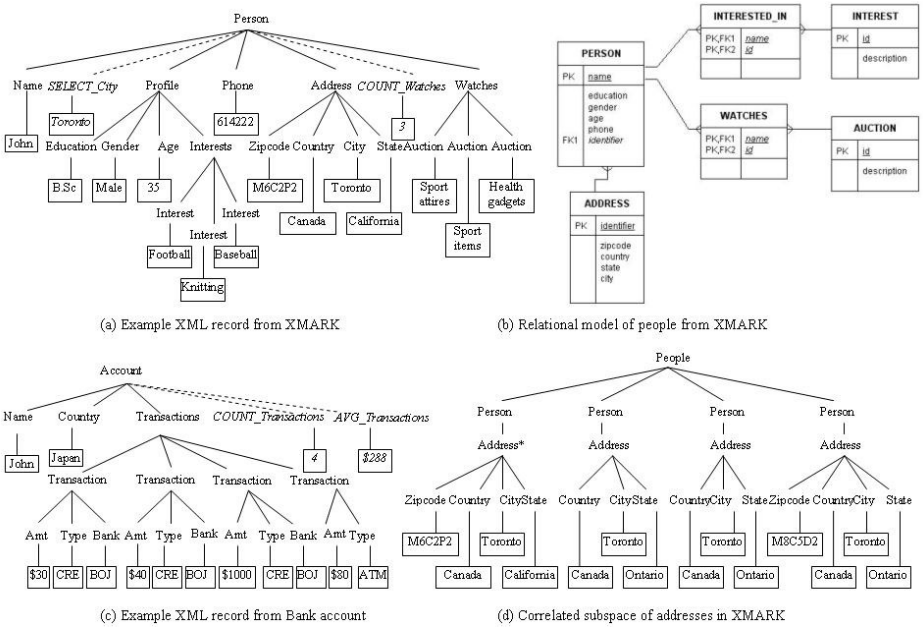

Figure 1. XML and correlated subspace.

To illustrate, we use an example from the XMARK benchmark for XML databases. From the XML structure of a <person> entity defined in XMARK schema, as shown in Figure 1(a), the following deductions can be computationally derived:

An address is composed of zipcode, city, country, and state.

A person watches a set of auctions.

A person has a number of interests.

On the other hand, deducing the same from the corresponding relational data model in Figure 1(b) requires domain knowledge or human interpretation, in order to determine whether Tim watches the auctions or the auction watches him. While the use of associative entities (watches and interested_in relations) perfectly normalized the M:N relationships of person, interests and auctions, the directionality is lost. Also missing in Figure 1(b) are the logical groupings of semantically correlated entities – A person is represented by his name, phone number, address, watched auctions and profile which, in turn, comprises of education, gender, age and interests. Basically, these groupings are “flattened” into tables when they are mapped to relational models.

Clearly, the hierarchical structure of XML data enables it to encode additional context information that is lacking in its relational counterpart. This paper leverages on such information to advance the outlier detection process. But before we delve further into the detection method, let us first consider what constitutes an outlier in XML data.

There are two known types of outliers – the class and the attribute outliers. Class outliers refer to complete records that do not fit into any class, by definition of distance, nearest neighbors, density or distribution [1, 7, 13, 16]. Since XML is semi-structured, well-defined classes of records do not exists, so detecting class outliers in XML makes little sense. Rather, we focus on identifying attribute outliers - *univariate points that exhibit deviating correlation behavior with respect to other attributes* [8]. Attribute outliers are typically associated with errors caused by mis-annotations, typographical, or entry mistakes. Nevertheless, not all of them are results of errors. The element <State:California> in Figure 1(a) may be an entry error, but <Amt:$1000> in Figure 1(c) cannot be labeled as an error technically although it is definitely unusual because the other transactional amounts are in magnitudes of less than $100. Such pattern requires further investigation because it may be fraudulent.

According to the definition given in [8], attribute outlier-ness is a bivariate property of a *target attribute* and a *neighborhood of its correlated attributes*. For example, intuitively, we know that <Country>, <State> and <City> attributes are highly correlated (semantically), and therefore it is unusual to associate the target attribute <State:California> with <City:Toronto> and <Country:Canada>, both of which are often found with <State:Ontario> instead. On the other hand, it is meaningless to associate <State> element to the <Phone> or <Interest> elements. We introduce the notion of *correlated subspaces* that leverage on the nested, hierarchical structure of XML to derive groups of attributes that are logically correlated in XML. As shown in Figure 1(d), the elements <Country>, <State>, <City> and <Zipcode> form a correlated subspace of the <Address> of each <Person> across all <People> in the XMARK dataset. Similarly, the <Amt>, <Bank> and <Type> elements of <Transaction> elements in Figure 1(b) form a subspace of all transactions of John's account.

Few known metrics for attribute outliers exist, even in relational data (details in Section 2). In order to quantitatively determine the extent of outlier-ness of a target attribute in a correlated subspace, we developed two new correlation-based outlier metrics – *xO-Measure* and *xQ-measure* which rely on the correlation behavior of individual to determine the attribute outlier-ness. Consider Figure 1(d) as an example, notice that <Country:Canada> and <City:Toronto> each co-occur more frequently with <State:Ontario> than with <State:California>. This simple idea forms the basis of *xO-Measure* and *xQ-measure*.

While it is not complicated to determine attribute outlier-ness of attributes with leaf nodes, computing the same for non-leaf nodes such as the <Watches> element in Figure 1(a) and the <Transaction> element in Figure 1(c) is not as straightforward. To resolve this, we utilize *aggregate attributes* to summarize sub-structures in XML through computing a scalar value from a set of multiple elements using aggregate functions (e.g. AVG, COUNT, MIN, MAX, and SUM). The use of aggregate attributes in XML is conceptually similar to that for relational databases [4, 5]. Aggregate attributes can be used to facilitate the identification of outliers at higher

level of abstractions. For example, through the correlation patterns of the <COUNT_Transactions>, <AVG_Transactions> and <Country> elements in Figure 1(c), we can identify accounts with unusually low (or high) transactional averages or counts, compared to other accounts from the same country. In fact, a transactional average of $228, the equivalence of less than USD$3, would have been very uncommon in Japan.

Summarizing, attribute outlier detection is an important yet new problem for XML. In this work, we utilize the context information inherent in the XML data models to determine correlated subspaces and derive aggregate attributes to advance the outlier detection method. Specifically, this paper makes the following contributions:

1. The notion of correlated subspaces - a meaningful grouping of correlated objects based on the XML data structures.
2. Aggregate attributes as summarizing elements in the hierarchical XML structures. The use of aggregate attributes enables the detection of attribute outliers at higher level of abstraction.
3. Two correlation-based attribute outlier metrics for XML, namely the *xO-Measure* and *xQ-Measure*.
4. A complete, systematic framework for detecting attribute outliers in XML called **XODDS** (for **X**ML attribute **O**utlier **D**etection from **D**ata **S**ubspaces) and demonstrate its effectiveness on real world datasets.

The rest of this chapter is organized as follows. Section 2 discusses related work. Section 3 presents formal definitions used in XODDS and Section 4 describes the XODDS framework. Section 5 presents other interestingness metrics, which we explore usages on the attribute outlier detection problem. An experimental evaluation of XODDS is given in Section 6, and we conclude in Section 7.

2 Related Works

Existing outlier detection methods has focused on class outliers. Numerous methods for identifying class outliers broadly classifies into distribution-based, clustering-based, distance-based and density-based approaches. Conversely, attribute outlier detection has received less attention. Among the few research related to attribute outlier detection are data polishing approaches to attribute outlier detection problem construct for each dimension a classifier based on the remaining dimensions and the class dimension [13, 16]. Incorrect predictions are labeled as attribute outliers. The accuracy of data polishing method varies depending on the classifier used and they are limited to finding attribute outliers resulting in change of class membership.

Research on cleaning XML data is also at its infancy. The focus is on XML duplicate detection by matching values of the corresponding fields of two objects and the structural similarity between the XML objects [9, 11, 14].

3 Preliminaries

Before detailing XODDS, we first present the terminologies used in this paper. An XML document is modeled as a tree $T(V, E, r, L)$ where V is a set of n nodes, E is a set of edges $E \subseteq V \times V$ and r is the root node. Each node represents an element or an attribute.

Let L be a countable set of labels. The labeling function *label*: $V \rightarrow L$ maps each $v \in V$ to some $l \in L$; different nodes may have the same label. Each $e \in E$ is an ordered pair of nodes, $e = (v_i, v_j)$ where $v_i \in V$ is the *parent* of $v_j \in V$, and v_j is called the *child* of v_i.

We use the function *value*(v) to denote the value of a leaf node. It follows that if v is a non-leaf node, value(v) = Ø. For every node $v \in V$, there is a unique path from root node r to v, denoted by $path(r, v) = (v_0 = r, v_1 ..., v)$. The number of edges from r to v is *dist*(r, v). Without loss of generality, we say that r is an *dth-ancestor* of v, *denoted* $\hat{A}^d(v)$ where $d = dist(r, v)$.

Consider the example in Figure 1(a). Let T be the tree rooted on Person node. It follows that {"Profile", "Interests", "Address", "Name", "State"} $\subset L$, <City:Toronto>, <Country:Canada> are the child nodes of <Address>, and $\hat{A}^2$(<City:Toronto>)=<Person>.

Given a tree $T(V, E, r, L)$, an *object* $Obj(v_i)$ rooted at node $v_i \in V$ is a set of nodes $v \in V$ such that $dist(v_i, v)=1$ and value(v)≠Ø. Simply, an object is denoted by its children leaf nodes.

So, Figure 1(a) shows 5 objects *Obj*(<Person>), *Obj*(<Profile>), *Obj*(<Interests>), *Obj*(<Watches>), and *Obj*(<Address>)={<Zipcode:M6C2P2>, <Country:Canada>, <State:California>, <City:Toronto>}.

4 Outlier Detection Framework

Given the complexity, it is not possible to resolve the attribute outlier detection problem in XML in a single computational step. Therefore, we streamline the various algorithmic components of our outlier detection method into a systematic, generalized framework, which we called XODDS. Figure 2 details the user specification requirements (for aggregate attributes and choice of outlier metric) and the four key processes in XODDS, namely attribute aggregation, subspace identification, outlier scoring and outlier identification.

4.1 Attribute Aggregation

As discussed in Section 1, aggregate attributes are summarizing elements that can be used to encapsulate nested XML elements. They are essential for identifying attribute outliers at higher levels of the abstraction. Let us formally define an aggregate attribute in XML data model.

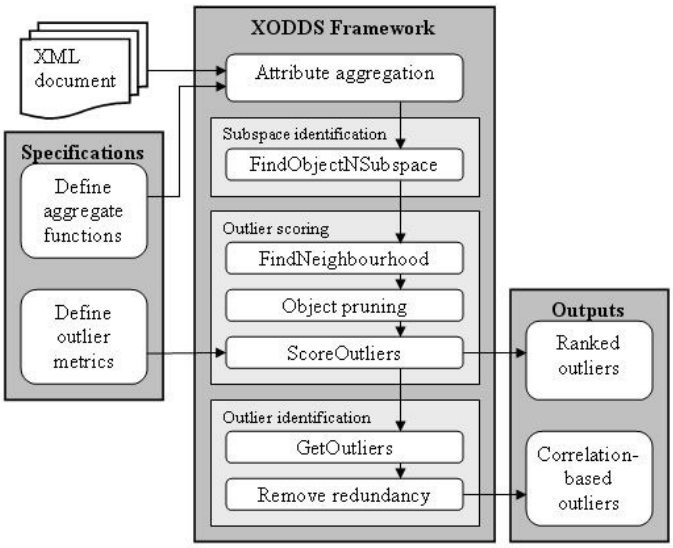

Figure 2. XODDS XML Outlier Detection Framework.

Definition 1 (Aggregate Attribute). Given T, an aggregate attribute $v_a(F, v_i, V_f\}$ is a leaf node derived from applying the *aggregate function F* on a set of nodes $V_f = \{v \in V \mid label(v) = l_a$ and $\hat{A}^d(v) = v_i$, $0<d<dist(v_i, v)$ and $l_a \in L\}$. v_a can be inserted to T as a sibling node of v_i.

For example, in Figure 1(b), let F = AVG be applied to all <Amt> nodes. This generates the leaf node <AVG_Transactions:$288>. Since the objective is to summarize all the transactional amounts in <Transactions>, it makes sense to insert the new leaf node as a sibling node to <Transactions> in the XML document.

Formulation of aggregate attributes requires contextual information from the users and cannot be determined automatically by the system. Attribute aggregation is a specification step in XODDS, which allows users to define and add aggregate attributes to the XML document. For each aggregate attribute, the user must specify, through the user interface of XODDS, (1) aggregate functions F, (2) v_i node to aggregate, and (3) the set of V_f nodes to apply aggregate function F.

4.2 Subspace Identification

Correlated subspaces are logical groupings of attributes in XML and attribute outliers are determined by comparing the objects in each logical group. Two objects $Obj(v_i)$ and $Obj(v_j)$, defined over nodes v_i and v_j respectively are said to be *comparable* if they exist in the same correlated subspace. Intuitively, a correlated subspace is a container of comparable objects that ideally correspond to real-world entities, and each object is described by leaf nodes.

Since XML is semi-structured, two elements with the same label may not represent comparable real-world entities. For instance, a person's home address as well as his work address can be tagged with <Address> label. Nevertheless, the path

of the two <Address> nodes from the root node provides additional information to differentiate them. We leverage on this structural information to group objects belonging to the same correlated subspace. However, instead of using the root as the pivoting node, the correlated subspace is defined over the *nearest common ancestor* (NCA) to facilitate navigation within a correlated subspace in a tree-based structure. Both NCA and correlated subspace are formally defined as follows:

Definition 2 (Nearest Common Ancestor). Given two nodes, $v_i, v_j \in V$, $v_c \in V$ said to be the *common ancestor* of v_i and v_j if $v_c=\hat{A}^k(v_i)=\hat{A}^k(v_j)$ and for each $\hat{A}^d(v_i)$ and $\hat{A}^d(v_j)$, $1 \leq d \leq k-1$, $\text{label}(\hat{A}^d(v_i)) = \text{label}(\hat{A}^d(v_j))$. v_c is called the *nearest common ancestor* of v_i and v_j, denoted as $NCA(v_i, v_j) \in V$ if the distance between v_i and v_j through v_c is shorter than any $v_{c'} \in V$.

For example, in Figure 1(d), *NCA*(<State:Ontario>,<State:California>)=<People>.

Definition 3 (Correlated Subspace). Given a node $v_p \in V$, which we call the *pivoting node* of the subspace, a correlated subspace $V_c(v_p) \subseteq V$ is a set of nodes such that

- $\forall\ v \in V_c$, $Obj(v) \neq \emptyset$,
- for any $v_i, v_j \in V_c$ $\text{label}(v_i) = \text{label}(v_j)$, and
- $NCA(v_i, v_j) = v_p$

Following the last example, the <Address> objects form a subspace with pivoting node v_p=<People> as shown in Figure 1(d).

The second step of XODDS - subspace identification implements the identification of correlated subspaces in a given XML document. If the elements or attributes are numerical, we discretize them into categorical values through binning into equi-width intervals. There exists a spectrum of discretization approaches such as K-interval discretization, iterative discretization, χ^2 binning, among many others. Rather than evaluating them, this paper utilizes a common discretization technique and focuses on the core method of attribute outlier detection. Besides binning, another pre-processing step in XODDS is to filter leaf nodes that contain unique values, usually the identifiers. For example, <Name:John> and <Phone:614222> in Figure 1(a) do not contain information that is relevant to attribute outlier detection.

In XODDS, objects and correlated subspaces are identified in the Procedure *FindObjectNSubspace*. After parsing the XML document into a DOM tree, the procedure performs a depth-first search on the tree [Line 1-3]. Each object node is collected into an array *Object*. If the node has the same label as its sibling nodes, it is immediately identified as a subspace with its parent as the pivoting node [Line 6-7]. Otherwise, the object is stored into a list, which is screened at Line 13-25. Following Definition 2, object nodes are checked if they belong to the subspace V_c based on their nearest common neighbors and XPATHs [Line 15, 22]. The function *isObjectNode* simply checks if the given node contains leaf nodes. The output of Procedure *FindObjectNSubspace* is a list of correlated subspaces in the XML document.

Input: XML document. Output: A list of correlated subspaces V_c

```
procedure FindObjectNSubspace
1.  Parse XML into DOM tree, T
2.  V_c ← Ø, Obj ← Ø, Objects ← Ø
3.  For each v_i of T do
4.    If isObjectNode(v_i) then
5.      Obj(v_i) ← children(v_i)
6.      If label(sibling(v_i)) EQ label(v_i) then
7.        V_c(parent(v_i), label(v_i)) ← Obj(v_i)
8.      Else
9.        Objects ← Obj(v_i), label(v_i), XPATH(v_i)
10.     Endif
11.   Endif
12. Endfor
13. For each Obj(v_i), label(v_i), XPATH(v_i) in Objects do
14.   For each V_c(v_p, label(v_j))
15.     If label(v_i) EQ label(v_j) AND NCA(v_i,v_j) EQ v_p then
16.       V_c(v_p, label(v_j)) ← Obj(v_i)
17.       GOTO Line 25
18.     Endif
19.   EndFor
20.   For each remaining Obj(v_j), label(v_j), XPATH(v_j) in
    Objects do
21.     If label(v_i) EQ label(v_j) AND XPATH(v_i) EQ
        XPATH(v_j) then
22.       V_c (NCA(v_i, v_j), label(v_i)) ← Obj(v_i)
23.     Endif
24.   EndFor
25. EndFor
```

4.3 Outlier Scoring

Outlier-ness is not a binary property; classifying an attribute as an outlier or non-outlier serves little value. An attribute outlier metric should ideally reflect the outlier-ness of each attribute in quantitative terms that denote the strength of the outlier-ness. In XML, this outlier-ness represents the extent of deviation of a target attribute in its correlated neighborhood. Before we detail the algorithmic steps of scoring an outlier in XODDS, we first formally define the notions of a correlated neighborhood, and the metrics that are used in the outlier scoring step.

Definition 4 (Correlated Neighborhood). Given a correlated subspace V_c in *T and a node* $v_i \in V_c$, a *correlated neighborhood* is any k-subset of the children nodes of v_i, $Obj^k(v_i) \subseteq Obj(v_i)$ and k is called the degree. For a *target attribute* $v_t \in Obj^k(v_i)$, the *neighbours* of v_t is defined $N(v_t, Obj^k(v_i)) = \{v \in Obj^k(v_i) \mid v \neq v_t\}$.

In Figure 1(d), Obj^3(<Address>*) = {<State:California>, <Country:Canada>, <City:Toronto>} is a 3-degree correlated neighborhood defined over node <Address>* and N(<State:California>, Obj^3(<Address>*)) = {<Country:Canada>, <City:Toronto>} .

Definition 5 (Support). Given an correlated neighborhood $Obj^k(v_i)$ in V_c, the support of $Obj^k(v_i)$, denoted $sup(Obj^k(v_i))$is the count of the number of nodes $v_c \in V_c$ such that $\forall\ v_j \in Obj^k(v_i)$, there exists *a node* $v_k \in Obj(v_c)$ such that *label*(v_j)=*label*(v_k) *and value*(v_j)=*value*(v_k).

In more intuitive sense, the support of a set of nodes in $Obj^k(v_i)$ is the count of objects in the same subspace, with leaf nodes that have the same labels and values as those in $Obj^k(v_i)$. Following the previous example in Figure 1(d), $sup(N$(<State:California>), Obj^3(<Address>*)) = 4 are the number of <Address> objects containing leaf nodes <Country:Canada>, and <City:Toronto>}.

Definition 6 (*x*O-measure). Given a correlated neighborhood $Obj^k(v_i)$ defined over $v_i \in V_c$. The *x*O-measure of a node $v_s \in Obj^k(v_i)$, denoted *x*O-measure(v_s, $Obj^k(v_i)$) is defined as

$$xO-measure(v_s, \mathrm{Obj}^k(v_i)) = \frac{\sum_{v_j \neq v_s, v_j \in \mathrm{Obj}^k(v_i)} \sup(N(v_j, \mathrm{Obj}^k(v_i)))}{\sup(N(v_s, V_s^k))} \quad (1)$$

*x*O-measure is the quotient of the co-occurrences of a target attribute v_s with its neighbors and the co-occurrence of its neighbours in its absence. For example, since $sup(N$(<Country:Canada>), Obj^3(<Address>*)) = 1 and $sup(N$(<City:Toronto>), Obj^3(<Address>*)) = 1, the *x*O-measure of <State:California> in Obj^3(<Address>*) = 1+1/4=0.5, which is comparatively lower than *x*O-measure of <State:Ontario> in the similar correlated neighbourhood = (3+3)/4=1.5. The former clearly has greater extent of outlier-ness. Note that for *x*O-measure (and *x*Q-measure) **the lower the metric score, the higher the degree of outlier-ness.**

Definition 7 (*x*Q-measure). The *x*Q-measure of $v_s \in Obj^k(v_i)$, denoted *x*Q-measure(v_s, $Obj^k(v_i)$) is defined as

$$xQ-measure(v_s, Obj^k(v_i)) = \frac{\sup(Obj^k(v_i))}{\sup(N(v_s, Obj^k(v_i))} \quad (2)$$

*x*Q-measure is a conditional probability - Given the neighbors of a target attribute v_s, *x*Q-measure is the probability that v_s is introduced in the data set.

Following the last example, *x*Q-measure(<State:California>, Obj^3(<Address>*) = 1/4 = 0.25 is also significantly lower than *x*Q-measure(<State:Ontario>, Obj^3(<Address>) =3/4=0.75.

Procedure *FindNeighbourhood* in the XODDS framework takes as input the list of subspaces identified and enumerates all posssible correlated neighborhoods in each subspace. The function *enumerateKSubsets* generates subsets from an object [Line 3]. Following Definition 5, the support of each correlated neighborhood is accumulated at Line 5-9. The output of the procedure is a list of correlated neighborhoods at varying degree k and their supports, organized according to each subspace.

Input: List of correlated subspaces V_c in *T*. Output: Hashtables SUP(V_c, *Obj*(k, v_i)) for each correlated subspace and correlated neighborhood

```
procedure FindNeighbourhood
1.  For each subspace Vc(vp, ls) do
2.    For each Obj(vi) in Vc do
3.      Objects ← enumerateKSubsets(Obj(vi))
4.      For each Obj(k, vi) in Objects do
5.        If SUP(Vc, Obj(k, vi)) exists then
6.           Increment SUP(Vc, Obj(k, vi)) by 1
7.        Else
8.           SUP(Vc, Obj(k, vi)) = 1
9.        EndIf
10.     EndFor
11.   EndFor
12. EndFor
```

Once the supports are computed, Procedure *ScoreOutliers* computes the outlier scores based on the *x*O-measure or *x*Q-measure metric, depending on the metric specification of the user. These metric scores are calculated in function *calculateScore*, according to Definition 7 and 8 respectively [Line 4].

Input: List of correlated neighbourhoods and supports. User option of outlier metrics. Output: List of outlier scores for each attribute *v* in each correlated neighborhood O(*v*, *Obj*(k, v_i))

```
procedure ScoreOutliers
1.  For each subspace Vc(vp, ls) do
2.    For each Obj(k, vi)  in SUP(Vc, Obj(k, vi))  do
3.       For each vs in Obj(k, vi) do
4.          O(vs, Obj(k,vi))←calculateScore(vs, Obj(k, vi))
5.       Endfor
```

```
6.    Endfor
7.  Endfor
```

4.4 Outlier Identification

Outlier-ness scoring measures the attributes according to their extent of outlier-ness, they do not differentiate between outliers and non-outliers. To isolate the attribute outliers from non-outliers, users typically need to define a threshold. This is not viable, given that the number of attribute outliers varies across different XML documents, and across attributes.

The final step of XODDS – outlier identification distinguishes the outliers from the non-outliers using an adaptive threshold derived from the input XML. For each target attribute in a correlated subspace, the optimal thresholds are determined from the maximal *Rate-of-change* which is the data-dictated outlier score separating the outliers and non-outliers. This removes the dependency of the outlier detection on any user-specified parameter. To evaluate the effectiveness of *Rate-of-change, w*e compare by experiments with the *Top-k* approach, which selects a certain top percentage of the data set as outliers (details in Section 6).

The input to Procedure *GetOutliers* is a list of k-degree neighborhoods for a target attribute and the corresponding outlier scores. The correlated neighborhoods are first sorted in ascending order of the outlier scores [Line 1]. Then Rate-of-Change is computed for each neighborhood and the maximal threshold is determined [Line 2-5]. Neighborhoods with the lowest metrics scores are output with the target attribute as correlation-based outliers.

Since the same attribute outlier may be detected in correlation neighborhoods of varying degrees, the outlier identification step also removes such redundancy.

Input: List of outlier scores for attribute v_s in k-degree neighborhoods. Output: Attribute outliers identified in XODDS.

```
procedure GetOutliers
1. B ← (v_s, Obj(k, v_i)) sorted in ascending order of O(v_s,
   Obj(k, v_i))
2. For each point b_i in B do
3.    Rate-of-change(b_i) ← (b_i - b_i-1)/ b_i
4. Endfor
5. ß = i with max Rate-of-change(b_i)
6. For each b_i, 1• j • ß do
7.    OUTPUT← b_i
8. Endfor
```

5 Interesting-ness measure

Interesting-ness measures are often used to rank the usefulness and significance of the discovered co-relations or co-occurrences. They may, therefore be appropriate for quantifying correlation-based attribute outliers. We compare the suitability of *x*O-

measure and *x*Q-measure, as well as 5 interesting-ness metrics to the attribute outlier detection problem in the next section.

Almost all correlation-based measures for categorical data can be defined in terms of the frequency counts (or supports) in a 2 ×2 contingency table depicting the co-presence, co-absence and cross-presence of two variables. For the attribute outliers detection problem, we are interested in the correlation behavior of a target attribute, denoted v_t, with other attributes in a localized neighborhood, denoted $N(v_t)$ (Figure 3).

	$N(v_t)$	$\neg N(v_t)$	
v_t	F_{11}	F_{10}	F_{1+}
$\neg v_t$	F_{01}	F_{00}	F_{0+}
	F_{+1}	F_{+0}	n

Figure 3. The 2×2 contingency table of a target attribute and its correlated neighborhood

Interesting-ness measures inspected in this comparative study are mainly described in [6, 12] and they include:

- Piatetsky-Shapiro (PS) $= \frac{F_{11}}{n} - \frac{F_{1+}F_{+1}}{n^2}$ (range -0.25 to 0.25)
- Interest $I = \frac{nF_{11}}{F_{1+}F_{+1}}$ (range 0 to ∞)
- Jaccard $\zeta = \frac{F_{11}}{F_{1+} + F_{+1} - F_{11}}$ (range 0 to 1)
- $H - measure = \frac{F_{10}F_{01}}{F_{1+}F_{+1}}$ (range 0 to 1)
- Probability $\Pr = \frac{F_{11}}{n}$ (range 0 to 1)

6 Experiment Evaluation

We evaluate the performance of the XODDS method on both synthetic Bank Account and real-world protein data sets. Experiments are performed on an Intel Core 2 dual 2 GHz computer with 1GB of main memory, and running Mac OS. Programs are written using Java. The following aspects are investigated in the experiments:

(1) Compare the accuracy of *x*O-measure, *x*Q-measure and other interesting-ness metrics using Top-k and Rate-of-Change selection methods.

(2) Evaluate the accuracy of various attribute outlier metrics in data sets with varying noise levels.

(3) Compare the accuracy of XODDS with the relational approach that uses χ^2 to derive the correlated subspaces in relational tables.

(4) Evaluate the effectiveness of aggregate attributes at identifying attribute outliers at higher level of abstractions. In particular, we are looking for "hidden" outliers which could be missed if not for the use of aggregate attributes.

(5) Evaluate the performance of using XODDS to detect discrepancies in annotations in a real-world protein database.

6.1 Bank Account

We extracted from the financial data set available at the web site http://lisp.vse.cz/pkdd99/Challenge/berka.htm, a Bank Account data set (denoted Bank) containing 4,500 accounts, 207,989 transactional records, 670 loan records, and 711 payment orders. The data set, originally in relational tables, is converted to the XML schema shown in Figure 4. 504 correlated subspaces are determined. The transactional records of separate account each form a correlated subspace because individual spending power and therefore transactional profile differs.

To prevent the implicit noise from interfering with the evaluation process, 4,884 transactional records which potentially contain outliers are removed, leaving 203,105 transactions.

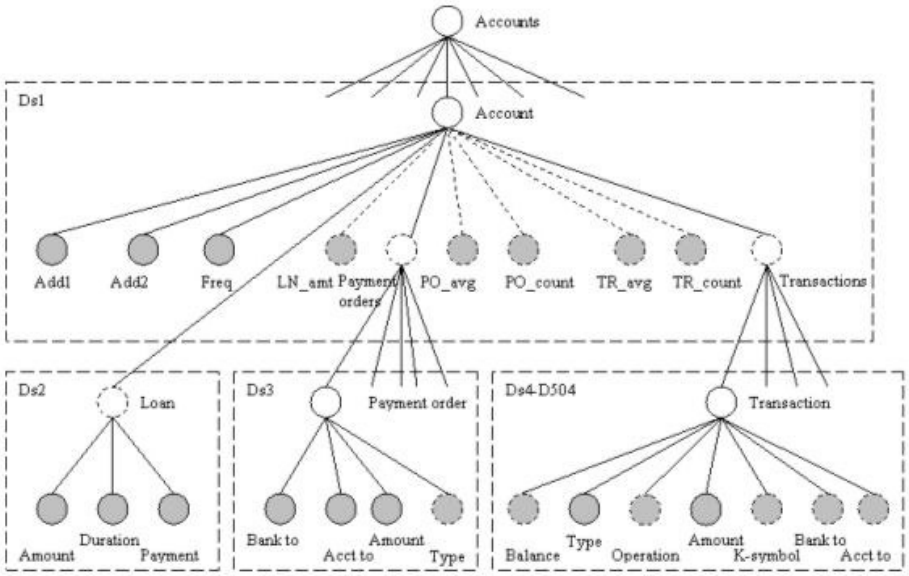

Figure 4. XML structure and correlated subspace of Bank Account

6.1.1 Top-k *vs* Rate-of-Change

On a Bank account data set which has 2% attribute outliers inserted, we first compare the performance of using Rate-of-Change as the selection criteria, compared to using Top-k. In Figure 5, Rate-of-Change (ROC) is applied to select more stringent thresholds from the top-k percent of the potential outliers; in Figure 6, no ROC

selection is made. With ROC, the F-scores of both *x*O-measure and *x*Q-measure converge to 80%. Figure 6 shows that the performance of the same metrics without ROC, have comparatively lower F-scores of at most 66%. Also noticed that at k=1%, the top-k approach achieves much higher F-scores than ROC across all metrics. This is expected because the input XML document contains 2% noise; selecting a "tighter bound" within the top 1% outliers merely serves to increase the number of FNs (false negatives).

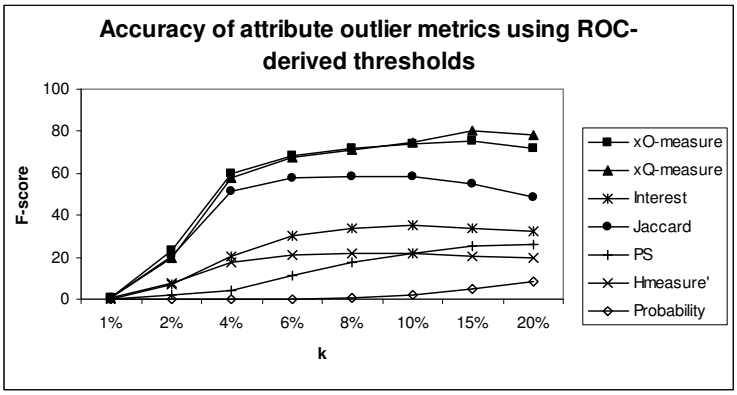

Figure 5. Performance of XODDS with various metrics using ROC derived thresholds

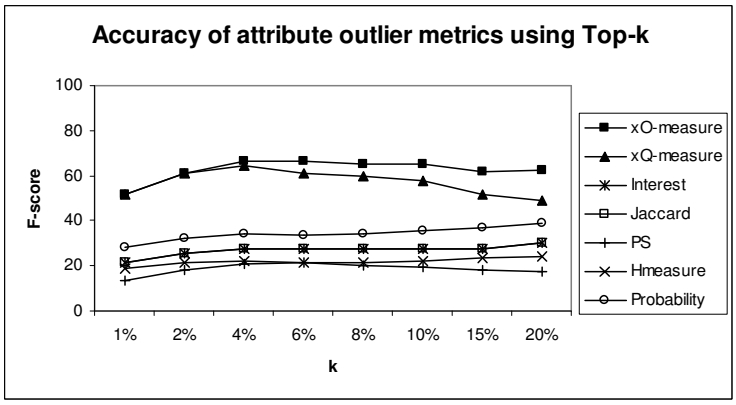

Figure 6. Performance of XODDS with various metrics using Top-k thresholds

Figures 5 and 6 also show that *x*O-measure and *x*Q-measure generally outperform the interesting-ness metrics. From a realistic perspective, we inserted attribute outliers as well as rare attributes into the input XML document because both types of patterns entail attention. Rare attributes such as <Amt:$1000> in Figure 1(c) deserve as much attention as abnormally low (or high) transactional averages compared to other accounts of the same country. Hence, metrics that do not differentiate between them (*x*O-measure, *x*Q-measure and Jaccard) generally achieves performs better with Bank. One other reason for the poor performance of Interest and PS metrics is they are

highly dependent on the vast co-absence of XML data (high F_{00}), which is naturally sparse.

6.1.2 Varying Noise Levels

Figure 7 compares the F-scores of various metrics at varying noise levels. The outlier thresholds are adaptively selected by XODDS using ROC. *x*O-measure and *x*Q-measure perform consistently better than the other metrics, even as the noise level increases. F-scores range between 72-81% for *x*O-measure and between 75%-80% for *x*Q-measure.

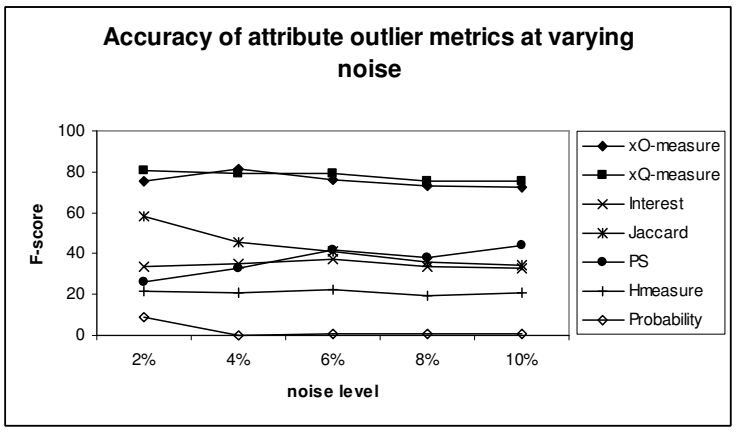

Figure 7. Performance of XODDS with various metrics using Top-k thresholds

6.1.3 Varying Noise Levels

XODDS utilizes the inherent hierarchical structure of XML to derive meaningful subspaces for outlier detection. Similarly, given a relational table, dimensions or columns can be divided into subspaces through *attribute clustering*. In order to compare the accuracy of detecting outliers using these two approaches, we "relationalize" the Bank data set (denoted RBank), and cluster the attributes based on Chi-square χ^2.

The χ^2 test is typically used to determine correlations between dimensions containing categorical data. For non-parametric data or continuous data, rank-order correlation calculations such *Spearman-Rho* and *Kendall-tau* or the *Pearson* test can be used instead. χ^2 test of independence is based on the difference of the observed frequencies with the corresponding expected frequencies. If $\chi^2 = 0$, the two attribute vectors are statistically independent. The χ^2 is computed by:

$$\chi^2 = \sum \frac{\left(f_{obs} - f_{\exp}\right)^2}{f_{\exp}}$$

where f_{exp} refers to the expected frequency of an attribute pair in the contingency table and f_{obs} is the observed frequency.

Table 1. Attribute subspaces derived in RBank using χ^2

Subspace	Attributes
1	Freq, Add1, Add2, Transaction/Acct to, Transaction/Bank to, Payment order/Acct to
2	Loan/Amount, Loan/Duration, Loan/Payment
3	Payment order/Bank to, Payment order/Amount, Payment order/Type
4	Transaction/Type, Transaction/Operation, Transaction/Amount, Transaction/Balance, Transaction/K-symbol

Table 1 shows the groups of attributes derived from the relational table using χ^2. It makes sense that the region(Add1) or district(Add2) where the customer open his bank account is likely dependent on the banks he often transacts to, so it is not surprising to find that Transaction/Bank to, Transaction/Acct to, and Payment order/Acct to are correlated to the Bank branch addresses (Add1 and Add2) in Subspace 1. We apply the same outlier scoring and selection algorithms to the relational subspaces. Figure 8 shows the performance of XODDS compared with the relational approach at varying noise levels. The poor performance of the latter approach may be due to the increased redundancy when an object is replicated across multiple tuples when "flattened", and thus affecting the distribution of each dimension in the converted relational table.

As the noise level increases, the correlation between the attributes decreases, thus affecting the accuracy of the outlier detection. Also notice that in Figure 8, the accuracy using *x*Q-measure (QM) in XODDS is generally higher than with *x*O-measure (OM), particularly when the level of noise increases. In fact, even on the Bank Account data set with 10% noise, the F-score that can be achieved using XODDS with *x*Q-measure is 75%.

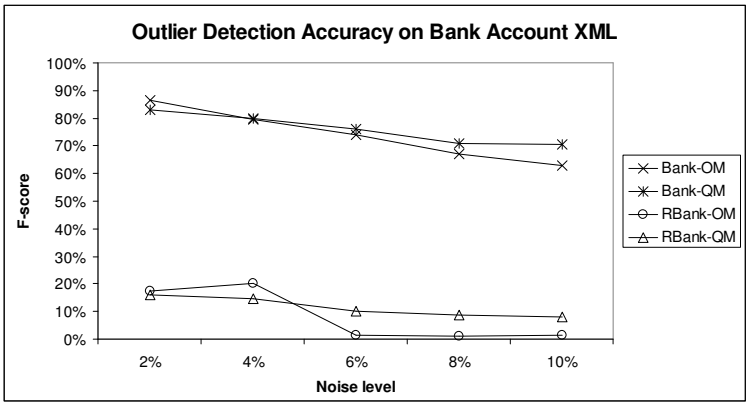

Figure 8. Performance of XODDS with various metrics using Top-k thresholds

6.1.4 Aggregate Attributes

The purpose of aggregate attributes is to summarize nested XML structures so that detection of attribute outliers at higher level of abstraction is possible. As shown in Figure 4, we introduced 5 aggregate attributes to the Bank Account data set: *TR_count* and *TR_avg* are the number of transactions and the average amount of transactions of an account. *PO_count* and *PO_avg* are the number of payment orders and the average amount of the payment orders. *LN_amt* is the loan amount. Since each account is restricted to one loan, the loan amount is merely "promoted" to the account level.

The aggregate attributes are compared across all 4,500 accounts in Bank; the outliers identified are shown in Figure 9. We are interested to determine any inherent outliers in the raw data set. We anticipated that removing 4,884 transactional records with possible outliers in the transaction subspace in our pre-filtering step may not necessarily "clean up" the account subspace. This is justified by applying XODDS on the clean data set which does not contain any inserted outliers. Some of the interesting inherent outliers which are uncovered are:

1. An account which has less than 10 transactional records.
2. Loan amount of less than $1,000 issued to 2 accounts.
3. Loan amount of more than $100,000 issued to 1 account.
4. 18 accounts with transactional averages of less than $100 in Hl.m. Praha city of Czech Republic

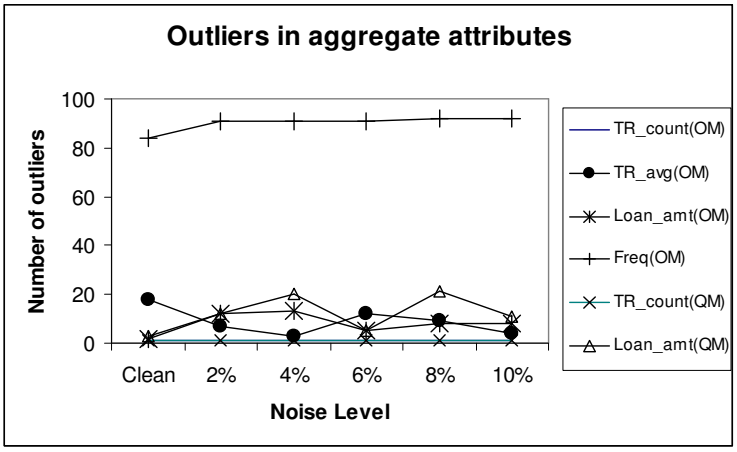

Figure 9. Number of outliers in the Account subspace across varying noise

These accounts are highlighted because the typical transactional averages of accounts opened in Hl.m. Praha is usually in magnitude of $100 or more. In general, the number of aggregate attribute outliers does not increase significantly as noise level increases, even for the averaging attributes. Inserting fraudulently high transactional amount to the transaction object may increase the outliers at the account level (e.g. transactional average increases) but it may also average out the outlier behavior.

6.2 UniProt Data Set

The second part of the experiment study evaluates the performance of XODDS in detecting annotation errors in the UniProt/TrEMBL database. The UniProt/TrEMBL database (release 7.1) consists of 2,826,395 protein sequence records collected from multiple sources of large-scale sequencing projects. It is frequently accessed by the world-wide biological researchers for analysis and data mining [2].

UniProt/TrEMBL records are computationally annotated. Therefore, the protein functions are predicted rather than verified experimentally and they contain a significant portion of mis-annotations or erroneous information [3, 15]. The purpose of applying XODDS on the UniProt/TrEMBL database is to identify discrepant protein annotations.

Given the high complexity of biological data, the original XML schema of UniProt/TrEMBL is extensive. For purpose of this study, we only focused on selected subspaces of the gene ontologies and the keywords. Each protein object in UniProt/TrEMBL is annotated with a set of gene ontology (GO) nodes corresponding to the GO controlled vocabulary of proteins' properties. Similarly, a protein contains a set of keyword subject references. Table 2 shows the detected outliers. The k-neighborhoods indicate the size of the neighborhoods of the outliers.

Table 2. Outliers detected from the UniProt/TrEMBL Gene Ontologies and Keywords annotations.

k-Neighborhoods	GO (OM)	GO(QM)	KW(OM)	KW(QM)
3	46	184	5	40
4	91	491	6	53
5	54	251	49	30
6	60	85	53	25
7	10	13	75	11
8	904	473	65	2
9	532	99	215	228
10	9	9	43	44
11	3712	9595	4	4
12	1621	4252	1	1
13	524	1391	2	2
14	117	317	-	-
15	16	45	-	-
16	1	3	-	-
17	3	5731	-	-
18	1208	765	-	-
19	75	50	-	-
Total	**8983**	**23754**	**418**	**440**

A biologist through manual verification verifies the detected attribute outliers. Table 3 shows the accuracy of the GO outliers detected using XODDS (with *x*O-measure). True positive TP indicates an uncommon association of the target attribute with the other attributes in the projected relation. False positive FP indicates that no peculiarity is found in the correlation behavior of the target attribute. Indeterminable means that further investigation is required.

Table 3. Annotation results of outliers detected from the UniProt/TrEMBL Gene Ontologies.

k-Neighborhood	TP	FP	Indeterminant	k-Neighborhood
3	9	12	25	3
4	28	7	56	4
5	23	3	28	5
6	35	0	25	6
7	6	0	4	7
8	283	48	573	8
9	189	39	304	9
10	4	1	4	10
11	2347	5	1360	11
12	1167	0	454	12
13	419	0	105	13
14	102	0	15	14
15	15	0	1	15
16	1	0	0	16
17	1	0	2	17
18	354	33	821	18
19	21	1	53	19
Total	5004	149	3830	Total

The manual verification step largely depends on the knowledge level of the biologist and his decisive-ness. Table 3 shows that a large percentage 43% of the outliers require further investigation because the biologist lacks the domain knowledge to justify if the annotation is erroneous or it is only exceptional. Only 3% out of those which can be annotated are false positives. The remaining 97% of the gene ontology outliers are confirmed erroneous annotations.

To mention a few interesting examples, a Mitochondrion protein Q35127 which occurs outside the nucleus is annotated to be involved in "DNA repair" - a process which occurs only inside the nucleus, a Chloroplast protein Q5UVG7 which is a plant cell organelle, is involved in "defense response to pathogen" and another Mitochondrion protein Q9B8V5 is annotated to contain a "nucleus". Some of these erroneous annotations are corrected in the latest UniProtKB/TrEMBL release.

7 Conclusion

In this paper, we have presented a novel framework called XODDS that utilizes the correlations between attributes to identify, attribute outliers. We introduced in XODDS, two new concepts of correlated subspace and aggregate attributes which are derived from the hierarchical structural of XML data models and two attribute outier metrics - *x*O-measure and *x*Q-measure. Through evaluation with synthetic and real-world data, we have shown that XODDS can achieve F-score of up to 80% and it can determine up to 97% erroneous annotations on a real-world data set.

References

1. Aggrawal, C.C., Yu, P.S. An Effective and Efficient Algorithm for High-dimensional Outlier Detection. *VLDB 14*, 2005.
2. Apweiler, R., Bairoch, A., Wu, C.H., et al. UniProt: the Universal Protein Knowledgebase. *Nucl. Acids Res. 32*, 2004.
3. Gilks, W.R., Audit, B., De Angelis, D., et al. Modeling the percolation of annotation errors in a database of protein sequences. *Bioinformatics 18*, 12, 2002.
4. Gray, J., Chaudhuri1, S., Bosworth1, A., et al. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery*, 1(1), 1997.
5. Gupta, A., Harinarayan, V. and Quass, D. Aggregate-Query Processing in Data Warehousing Environments. *VLDB*, 1995.
6. He, B., Chang, K.C., Han J. Discovering complex matchings across Web query interfaces: A correlation-mining approach. *ACM SIGKDD*, 2004.
7. Knorr, E.M., Ng, R.T. Finding Intensional Knowledge of Distance-based Outliers. *VLDB*, 1999.
8. Koh, J.L.Y., Lee, M.L., Hsu, W., Lam, K.T. Correlation-based Detection of Attribute Outliers. *DASFAA*, 2007.

9. Low, W. L., Tok, W. H., Lee, M. L. and Ling, T. W. Data Cleaning and XML : The DBLP Experience. Poster in IEEE ICDE, 2002.

10. Piatetsky-Shapiro, G. Discovery, analysis and presentation of strong rules. *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. Frawley eds, MIT Press, Cambridge, MA: 2299-2480, 1991.

11. Puhlmann, S., Weis, M., Naumann, F. XML Duplicate Detection Using Sorted Neighborhoods. *EDBT*, 2006.

12. Tan, P.N., Kumar, V. and Srivastava, J. Selecting the right interestingness measure for association patterns. *ACM SIGKDD*, 2002, 32-41.

13. Teng, C.M. Polishing Blemishes: Issues in Data Correction. *IEEE Intelligent Systems*, 19, 2, 2004.

14. Weis, M. and Naumann, F. DogmatiX tracks down duplicates in XML. *ACM SIGMOD*, 2005.

15. Wieser,D., Kretschmann,E., Apweiler,R.: Filtering erroneous protein annotation. Bioinformatics, 20, 2004.

16. Zhu, X. and Wu, X. Class Noise vs. Attribute Noise: A Quantitative Study of their Impacts. *Artificial Intelligence Review*, 22, 3, 2004.