INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

# Detecção de Duplicados em Bases de Dados XML

**Luís Miguel Gomes dos Santos Reis Leitão**

Dissertação para obtenção do Grau de Mestre em

**Engenharia Informática e de Computadores**

## Resumo Alargado

**Setembro 2007**

# Structure-Based Inference of XML Similarity for Fuzzy Duplicate Detection

Luis Leitao
IST/INESC-ID
Av. Professor Cavaco Silva
2744-016 Porto Salvo,
Portugal
lsrl@mega.ist.utl.pt

Pavel Calado
IST/INESC-ID
Av. Professor Cavaco Silva
2744-016 Porto Salvo,
Portugal
pavel.calado@tagus.ist.utl.pt

Melanie Weis
Hasso Plattner Institut
Prof.-Dr.-Helmert-Strasse 2-3
14482 Potsdam, Germany
melanie.weis@hpi.uni-potsdam.de

## ABSTRACT

Fuzzy duplicate detection aims at identifying multiple representations of real-world objects stored in a data source, and is a task of critical practical relevance in data cleaning, data mining, or data integration. It has a long history for relational data stored in a single table (or in multiple tables with equal schema). Algorithms for fuzzy duplicate detection in more complex structures, e.g., hierarchies of a data warehouse, XML data, or graph data have only recently emerged. These algorithms use similarity measures that consider the duplicate status of their direct neighbors, e.g., children in hierarchical data, to improve duplicate detection effectiveness. In this paper, we propose a novel method for fuzzy duplicate detection in hierarchical and semi-structured XML data. Unlike previous approaches, it not only considers the duplicate status of children, but rather the *probability of descendants* being duplicates. Probabilities are computed efficiently using a Bayesian network. Experiments show the proposed algorithm is able to maintain high precision and recall values, even when dealing with data containing a high amount of errors and missing information. Our proposal is also able to outperform a state-of-the-art duplicate detection system on three different XML databases.

## Categories and Subject Descriptors

H.2.5 [**Database Management**]: Heterogeneous Databases

## General Terms

Standardization, Algorithms, Experimentation

## Keywords

XML, Duplicate detection, Bayesian networks

## 1. INTRODUCTION

Fuzzy duplicate detection consists in the automatic determination of different representations of real-world objects stored in a data source. Fuzzy duplicate detection is of critical practical relevance in many applications, including data cleaning [20], data integration [7], and personal information management [8]. Ironically, the problem has been considered under various names, e.g., record linkage [28], merge/purge [12], reference reconciliation [8], or entity resolution [3]. In this paper, we refer to the problem as fuzzy duplicate detection, or *duplicate detection* for short.

Duplicate detection has been studied extensively for relational data stored in a single table. Algorithms performing duplicate detection in a single table generally compare tuples (each of which represents an object) based on attribute values. However, data usually comes in more complex structures, e.g., data stored in a relational table relates to data in other tables through foreign keys. Recently, duplicate detection algorithms for data stored in such complex structures have been proposed [2, 14, 26]. These approaches have in common that they do not only consider attribute values, but also relationships to related data.

We propose a novel method for duplicate detection in XML data. Detecting duplicates in XML is more challenging than detecting duplicates in relational data because there is no schematic distinction between *object types* among which duplicates are detected and *attribute types* describing objects. Furthermore, instances of a same object type may have different structure on instance level, whereas tuples within relations always have the same structure. We call these challenges candidate-description ambiguity and structural diversity [26]. On the other hand, XML duplicate detection allows to exploit the hierarchical structure for efficiency in addition to effectiveness, which is not the case when detecting duplicates in graph data.

The duplicate detection method we propose uses a Bayesian network model to compute the probability of any two XML objects, represented by XML elements, being duplicates. It considers the hierarchical structure of XML elements by considering probabilities for descendant XML elements as well. The model is built automatically based on the structure of the objects being compared. Since the structure contains no cycles, the duplicate probability can be determined efficiently. Our approach not only provides a strong formal basis for the proposed solution, but it is also flexible enough to easily adapt to different databases from different domains.

Experiments on artificial and real-world data show that the proposed method is able to achieve highly accurate results, yielding both high precision and recall in all cases. To further validate these results, a comparison was made with


*CIKM'07,* November 6–8, 2007, Lisboa, Portugal.

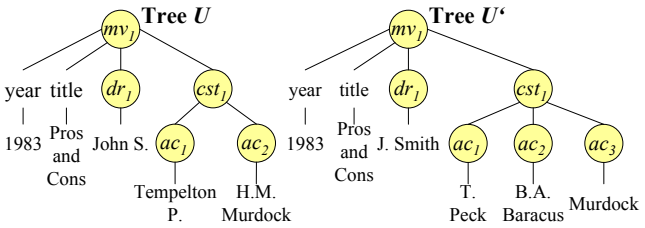


**Figure 1: Two XML trees, each representing a movie ($mv$) nesting directors ($dr$), a cast ($cst$) and actors ($ac$).**

a state-of-the-art XML duplicate detection system, DogmatiX [26], achieving consistently better results in all tested databases.

This paper makes the following main contributions: (i) it introduces a novel duplicate detection method, based on a *Bayesian network* model; (ii) the method not only considers children XML elements, but considers *descendant* XML elements, i.e., complete subtrees; (iii) the method not only considers the duplicate status (*duplicate* or *non-duplicate*) of children, but considers the *probability* that descendants are duplicates, thus allowing a more accurate computation of the final probability; and (iv) the method is shown to perform accurately in three different datasets, two of which derived form real-world XML databases.

The remainder of this paper is organized as follows. In Sec. 2, we describe the basic idea underlying our similarity measure. Secs. 3 and 4 then describe how XML duplicate detection is performed using the proposed Bayesian network model. We evaluate our algorithm in Sec. 5. Related work is covered in Sec. 6, before we conclude in Sec. 7.

# 2. MOTIVATING EXAMPLE

In this section, we first introduce XML duplicate detection, then present how a Bayesian network is constructed, and finally give an intuition of how XML similarity is computed.

## 2.1 XML Duplicate Detection

The goal of XML duplicate detection is to identify XML elements representing the same real-world object. In this work, we assume a schema mapping step has preceded duplicate detection, so that all XML documents comply to the same schema. As an example, consider the tree representation of two XML elements represented in Fig. 1 (nodes are labeled by their XML tag name and an index for future reference). Both trees represent XML elements named $mv$. These elements have two attributes, namely year and title. They nest further XML elements representing directors ($dr$) and casts ($cst$). A cast consists of several actors ($ac$), represented as children XML elements of $cst$. Year, title, $dr$, and $ac$ have a text node which stores the actual data. For instance, year has a text node containing 1983 as string value.

In this example, the goal of duplicate detection is to detect that both movies are duplicates, although director and actor names are represented differently. To do this, we make the following assumption: *The fact that two nodes are duplicates depends only on the fact that their values are duplicates and that their children are duplicates.*

This means that movies, represented by nodes tagged $mv$ are duplicates depending on whether or not their children nodes (tagged $dr$ and $cst$) and their values for attributes year and title are duplicates. Further, the nodes tagged $dr$ are duplicates depending on whether or not their values are duplicates and the nodes tagged $cst$ are duplicates depending on whether or not their children nodes (tagged $ac$) are duplicates. This process goes on recursively until the leaf nodes are reached. If we consider trees $U$ and $U'$ of Fig. 1, this process can be represented by a Bayesian network (BN), as explained in the following sections.

## 2.2 Bayesian Network for Duplicate Detection

Before we outline how the Bayesian network for XML duplicate detection is constructed, we provide some background knowledge and notation on Bayesian networks.

### 2.2.1 Bayesian Networks

Bayesian networks provide a graphical formalism to explicitly represent the dependencies among the variables of a domain, thus providing a concise specification of a joint probability distribution [18]. This representation is based on a directed acyclic graph where a set of random variables makes up the nodes of the network and a set of directed links connects pairs of nodes. In this graph, an edge from one node to another means that the first has a direct influence on the second. This influence is quantified through a *conditional probability distribution function* correlating the states of each node with the states of its parents.

To illustrate, let $X$ and $Y$ be two random variables and let $x$ and $y$ be two of their respective values. We use $X$ and $Y$ to refer to the random variables as well as to the nodes in the network associated with these variables. An edge directed from $Y$, the *parent* node, to $X$, the *child* node, represents the influence of the variable $Y$ on the variable $X$, which is quantified by the conditional probability $P(x|y)$. In general, let $\mathcal{P}$ be the set of all parent nodes of a node $X$. Further, let $p$ be a set of values for all the variables in $\mathcal{P}$ and let $x$ be a value of the variable $X$. The influence of $\mathcal{P}$ on $X$ can be modeled by any function $F$ such that $\sum_x F(x,p) = 1$ and $0 \leq F(x,p) \leq 1$. The function $F(x,p)$ provides a numerical quantification for $P(x|p)$.

A key advantage of Bayesian networks is their synthesized representation of probabilistic relationships. In fact, it is necessary to consider only the known independencies among the variables in a domain, rather than specifying a complete joint probability distribution. The dependencies declared at modeling time are used to infer *beliefs* for all variables in the network. The inference mechanism, though exponential in the worst case, is efficient in many practical situations, particularly in those shown in this work.

### 2.2.2 Bayesian Network Structure

For XML duplicate detection, we construct a Bayesian network as follows. Let us first consider the XML nodes tagged $mv$. As illustrated in Fig. 2, the BN will have a node labeled $mv_{11}$ representing the possibility of node $mv_1$ in the XML tree $U$ being a duplicate of node $mv_1$ in the XML tree $U'$. Node $mv_{11}$ is assigned a binary random variable. This variable takes the value 1 (active) to represent the fact that the XML $mv$ nodes in trees $U$ and $U'$ are duplicates. It takes the value 0 (inactive) to represent the fact that the nodes are not duplicates.

The probability of the two XML nodes being duplicates depends on (1) whether or not their values are duplicates,

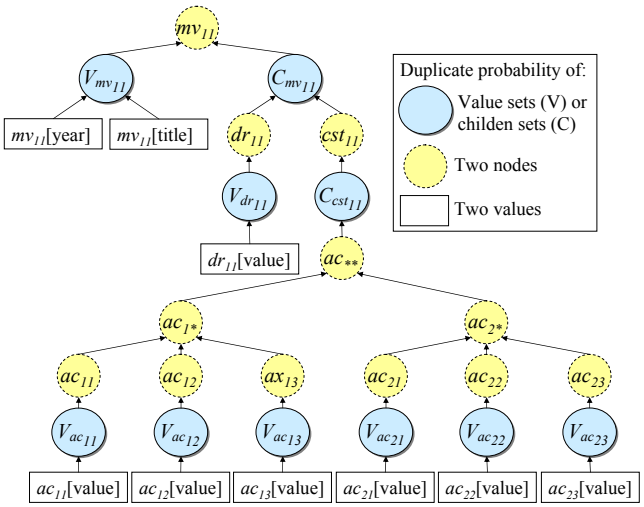


**Figure 2: Bayesian network to compute the similarity of the trees in Fig. 1.**

and (2) whether or not their children are duplicates. This is in accord with our assumption stated in Sec. 2.1. Thus, node $mv_{11}$ in the BN has two parent nodes, as shown in Fig. 2. Node $V_{mv_{11}}$ represents the possibility of the values in the $mv$ nodes being duplicates. Node $C_{mv_{11}}$ represents the possibility of the children of the $mv$ nodes being duplicates. As before, a binary random variable, that can be active or inactive, is assigned to these nodes, representing the fact that the values and children nodes are duplicates or non-duplicates, respectively.

We assume that the probability of the XML node values being duplicates depends on each attribute independently. This is represented in the network by adding new nodes for the attributes as parents of node $V_{mv_{11}}$, represented as rectangles in Fig. 2. In this case, these new nodes represent the possibility of the year values in the $mv$ nodes being duplicates and of the title values in the $mv$ nodes being duplicates.

Similarly, the probability of the children of the $mv$ nodes being duplicates depends on the probability of each pair of children nodes being duplicates. Thus, two more nodes are added as parents of node $C_{mv_{11}}$: node $dr_{11}$ represents the possibility of node $dr_1$ in tree $U$ being a duplicate of the node $dr_1$ in tree $U'$; node $cst_{11}$ represents the possibility of node $cst_1$ in tree $U$ being a duplicate of node $cst_1$ in tree $U'$.

We can now repeat the whole process for these two nodes. However, a slightly different procedure is taken when representing multiple nodes of the same type, as is the case for the XML nodes labeled $ac$. In this case, we wish to compare the full set of nodes, instead of each node independently. Thus, we say that the set of $ac$ nodes being duplicate depends on each $ac$ node in tree $U$ being a duplicate of any $ac$ node in tree $U'$. This is represented by nodes $ac_{**}$, $ac_{1*}$, and $ac_{2*}$ in the BN of Fig. 2.

Finally, each $ac_{ij}$ node represents the possibility that node $ac_i$ in tree $U$ is a duplicate of node $ac_j$ in tree $U'$. Since the $ac$ nodes have no children, their probability of being duplicates only depends on their values. Thus, each node $ac_{ij}$ in the network has only one parent node $V_{ac_{ij}}$. Since each $ac$ node has only one value, each node $V_{ac_{ij}}$ in the network has only one parent representing the possibility of both XML nodes, $ac_i$ and $ac_j$, having duplicate values.

A formalization of how the BN is constructed is provided in Sec. 3. Before that, we give readers an intuition of how this BN is used for XML duplicate detection.

| Conditional Probability |
|---|
| $P(V_{mv_{11}}\|mv_{11}[year], mv_{11}[title])$ |
| $P(V_{dr_{11}}\|dr_{11}[value])$ |
| $P(V_{ac_{ij}}\|ac_{ij}[value])$ |

(a) CP1

| Conditional Probability |
|---|
| $P(C_{mv_{11}}\|dr_{11}, cst_{11})$ |
| $P(C_{cst_{11}}\|ac_{**})$ |

(b) CP2

| Conditional Probability |
|---|
| $P(mv_{11}\|V_{mv_{11}}, C_{mv_{11}})$ |
| $P(dr_{11}\|V_{dr_{11}})$ |
| $P(cst_{11}\|C_{cst_{11}})$ |
| $P(ac_{ij}\|V_{ac_{ij}})$ |

(c) CP3

| Conditional Probability |
|---|
| $P(ac_{**}\|ac_{1*}, ac_{2*})$ |
| $P(ac_{i*}\|ac_{i1}, ac_{i2}, ac_{i3})$ |

(d) CP4

**Table 1: Sample conditional probabilities**

### 2.2.3 Computing Probabilities

Using the network of Fig. 2 we can now compute the probability $P(mv_{11})$ of trees $U$ and $U'$ being duplicates. In fact, we can compute the probability of any comparable pair of nodes, one from each tree, being duplicates and we can use any *a priori* knowledge that we might have on any other pair of nodes being duplicates. For instance, if we know that nodes $ac_2$ (from tree $U$) and $ac_3$ (from tree $U'$) are duplicates, we can compute the probability of the trees being duplicates as $P(mv_{11}|ac_{23})$. Or we can compute the probability of the cast in two movies being equal if the directors are the same, i.e. $P(cst_{11}|dr_{11})$.

We are mainly interested in the probability of the root nodes being duplicates $P(mv_{11})$, which can be interpreted as a similarity value between the two XML objects. To obtain this probability, we first need to define the *prior probabilities* associated to the leaf nodes of the network and the *conditional probabilities* associated to the inner nodes of the network.

In the network in Fig. 2, we need to define the prior probabilities of values being duplicates in the context of their parent XML node, e.g., $P(mv_{11}[year])$, $P(mv_{11}[title])$, $P(dr_{11}[value])$, and $P(ac_{ij}[value])$.

We further need to define four types of conditional probabilities: (1) the probability of the values of the nodes being duplicates, given that each individual pair of values contains duplicates; (2) the probability of the children nodes being duplicates, given that each individual pair of children are duplicates; (3) the probability of two nodes being duplicates given that their values and their children are duplicates; and (4) the probability of a set of nodes of the same type being duplicates given that each pair of individual nodes in the set are duplicates. In our example, these four types of conditional probabilities (denoted CP1 through CP4) correspond to the respective probabilities listed in Tab. 1(a) through (d).

In this section, we provided an intuitive overview of how we use a Bayesian network for XML duplicate detection. The following two sections provide the technical details for constructing the Bayesian network and computing the probabilities, respectively.

# 3. BAYESIAN NETWORK CONSTRUCTION

Formally, an *XML tree* is defined as a triple $U = (t, V, C)$, where

- $t$ is a root tag label, e.g., for tree $U$ in Fig. 1, $t = mv_1$.
- $V$ is a set of (attribute,value) pairs. If the node itself has a value, we can consider it as a special (attribute,value) pair. For tree $U$ in Fig. 1, we have $V = \{(year, \text{'1983'}), (title, \text{'Pros and Cons'})\}$.
- $C$ is a set of XML trees, i.e., the sub-trees of $U$. For tree $U$ in Fig. 1, $C$ contains subtrees rooted at $dr$ and $cst$. These subtrees are again each described by a triple.

We say that two XML trees are *duplicates* if their root nodes are duplicates.

**Algorithm 1**: ***BNGen(XTreeSet* U*, XTreeSet* U′*)***

```
   Input: U = {(t1, V1, C1), (t2, V2, C2), ... },
          U' = {(t'1, V'1, C'1), (t'2, V'2, C'2), ... }
   Output: A directed graph G = (N, E)
   /* -------------- Initialization --------------   */
   /* Root node tags of all XML trees in U and U'    */
1  S ← {t1, t2, ...};
2  S' ← {t'1, t'2, ...};
   /* Tags in S and S' representing real-world type r */
3  S_r = {t_i ∈ S | T_{t_i} = r};
4  S'_r = {t'_i ∈ S' | T_{t'_i} = r};
   /* -------------- BN Construction --------------   */
5  foreach type r ∈ S ∪ S' do
       /* Nodes with single occurrence               */
6      if |S_r| ≤ 1 and |S'_r| ≤ 1 then
7          Insert into N a node t_ii;
8          if V_i ∪ V'_i ≠ ∅ then
9              Insert into N a node V_{t_ii};
10             Insert into E an edge from this node to node t_ii;
11         if C_i ∪ C'_i ≠ ∅ then
12             Insert into N a node C_{t_ii};
13             Insert into E an edge from this node to node t_ii;
14         if node V_{t_ii} was created then
15             foreach attribute a ∈ V_i ∪ V'_i do
16                 Create a node t_ii[a];
17                 Insert an edge from this node to node V_{t_ii};
18         if node C_{t_ii} was created then
19             G' = (N', E') ← BNGen(C_i, C'_i);
20             foreach node n ∈ N' do
21                 Insert n into N;
22             foreach edge e ∈ E' do
23                 Insert e into E;
24             foreach node n ∈ N' without outgoing edges do
25                 Insert an edge in E from n to node C_{t_ii};
       /* Nodes with multiple occurrences            */
26     else if S_r or S'_r contain more than one tag each then
27         Insert into N a node t_**;
28         foreach tag t_i ∈ S_r do
29             Insert into N a node t_i*;
30             Insert into E an edge from this node to node t_**;
31             foreach tag t'_j ∈ S'_r do
32                 Insert into N a node t_ij;
33                 Insert into E an edge from this node to node
                   t_i*;
34         foreach newly created node t_ij do
35             Similar to processing of node t_ii (lines 8-25),
               second subscript i is replaced by j...
```

To every node with a given tag, we can associate a real-world type. We define $\mathcal{T}_t$ as the real-world type associated with the nodes of tag $t$. We define the type of an XML tree as the type of its root node. We assume that two trees can only be duplicates if they are of the same type. Also, two nodes can be compared only if they are of the same type. In our example, the real-world types are $\mathcal{T}_{mv} = movie$, $\mathcal{T}_{dr} = director$, $\mathcal{T}_{cst} = cast$, and $\mathcal{T}_{ac} = actor$. For simplicity, in the subsequent definitions we assume that nodes with the same real world type also have the same tag. That is, a relabeling step has preceded the construction of the BN.

In Algorithm 1, we provide pseudo-code for constructing the Bayesian network, as outlined in Sec. 2, from the structure of the XML trees being compared. The algorithm takes as input two sets of XML trees $U$ and $U'$, from which root node tag names of a given real-world type $r$ are extracted (lines 1-4). When processing nodes of real-world type $r$, we distinguish between node types occurring at most once (l.6) and node-types with multiple occurrences (l.26). The difference is that in the first case, only a single node $V_{ii}$ and a single node $C_{ii}$ needs to be constructed, whereas in the latter case, we need to construct a node $t_{ij}$ for every combination of nodes $t_i$ and $t_j$ of same type, as well as nodes $t_{**}$, $t_{i*}$. When applying this function to the XML trees $U$ and $U'$ of Fig. 1, we obtain the graph in Fig. 2.

# 4. DEFINING THE PROBABILITIES

To complete the Bayesian network constructed as described in the previous section, we need to define the prior probabilities associated to the network's leaf nodes and the conditional probabilities associated to the network's inner nodes, as illustrated in Sec. 2.2.3. In both cases, our model takes an epistemological view of the problem, where probabilities are interpreted as degrees of *belief* that can be specified independently of experimentation.

In the following, for simplicity, we will use the notation $P(x)$ to mean $\mathrm{P}(x = 1)$. We will assume that all probabilities $P(x = 0)$ are defined as $P(x = 0) = 1 - P(x = 1)$.

## 4.1 Prior Probabilities

In our model, prior probabilities represent the likelihood that two values in the XML trees are the same. They are associated with nodes labeled as $t_{ij}[a]$, where $a$ is an attribute name. For instance, in the case of the network in Fig. 2, we need to define the prior probability $P(mv_{11}[year])$ of both years in the $mv$ XML nodes being the same.

These probabilities can be defined based on the similarity between values. For instance, the probability of the title attributes in two movies being the same can be a string similarity between both titles. If we normalize this similarity to a value between 0 and 1, we can apply it as a probability value. However, it is sometimes not possible, or not efficient, to measure the similarity between two attribute values. In this case, we define the probability as a small constant, representing the (small) possibility of any two values being duplicates before we observe them.

Thus, we define

$$P(t_{ij}[a]) = \begin{cases} sim(V_i[a], V_j[a]) & \text{if similarity was measured} \\ k_a & \text{otherwise} \end{cases} \quad (1)$$

where $V_i[a]$ is the value of attribute $a$ of the $i-th$ node with tag $t$ in the XML tree, $sim(\cdot)$ is a similarity function,

normalized to fit between 0 and 1, and $k_a$ is a small constant, representing the probability of two values of attribute $a$ being similar. We name constant $k_a$ the *default probability*.

For instance, for the year attribute in the *mv* nodes, we can define $sim(y_1, y_2) = 1$ if $y_1 = y_2$, and $sim(y_1, y_2) = 0$ otherwise. The default probability $k_{year}$ can be derived from the distribution of years in the database, or simply be set to a small number.

## 4.2 Conditional Probabilities

As discussed in Sec. 2.2.3, four types of conditional probabilities (CP1 through CP4) are present in our model. In this section, we discuss how we compute each type of conditional probability.

**Conditional Probability CP1.** CP1 denotes the probability of the values of the XML nodes being duplicates given that their attribute values are duplicates. Formally, this corresponds to $P(V_{t_{ij}}|t_{ij}[a_1], t_{ij}[a_2], \dots)$ and is a function of the values of the Bayesian network nodes $t_{ij}[a_1], t_{ij}[a_2], \dots$ This function can be arbitrarily defined, as long as it conforms to the axioms of probability.

We therefore propose to define this probability as follows: (a) if all attribute values are duplicates, we consider the XML node values as duplicates; (b) if none of the attribute values are duplicates, we consider the XML node values as non-duplicates; (c) if some of the attribute values are duplicates, we determine that the probability of the XML nodes being duplicates equals a given value, $w_a$. This value represents the importance of the corresponding attribute in determining if the nodes are duplicates.

This definition is represented in Eq. (2).

$$P(V_{t_{ij}}|t_{ij}[a_1], t_{ij}[a_2], \dots, t_{ij}[a_n]) = \sum_{1 \le k \le n | t_{ij}[a_k]=1} w_{a_k} \quad (2)$$

subject to $\sum_{1 \le k \le n} w_{a_k} = 1$.

For instance, in the network of Fig. 2, we could use $w_{year} = 0.1$ and $w_{title} = 0.9$, meaning that if the years are equal, there is a 10% chance of the nodes being duplicates, but if the titles are equal, then there is a 90% chance of the nodes being duplicates. These values can be learned from data or empirically determined.

**Conditional Probability CP2.** The probability of considering children nodes as duplicates, given that each pair of comparable children nodes is a duplicate, is denoted CP2 and formally defined as $P(C_{t_{ij}}|t^1_{ij}, t^2_{ij}, \dots)$. Intuitively, it makes sense to say that two nodes are duplicates only if all of their child nodes are also duplicates. However, it may be the case that, for instance, the XML tree is incomplete or contains erroneous information. Thus, we relax this assumption and state that the more child nodes in both trees are duplicates, the higher the probability that the parent nodes are duplicates. This is represented by Eq. (3).

$$P(C_{t_{ij}}|t^1_{ij}, t^2_{ij}, \dots, t^n_{ij}) = \frac{1}{n} \times \sum_{k=1}^{n} t^k_{ij} \quad (3)$$

According to Eq. (3), the probability is directly proportional to the number of child nodes that are duplicates.

For instance, in the network of Fig. 2, the probability $P(C_{mv_{11}}|dr_{11}, cst_{11})$ is defined as $P(C_{mv_{11}}|dr_{11}, cst_{11}) = (dr_{11} + cst_{11})/2$.

**Conditional Probability CP3.** To define the probabilities of two nodes being duplicates given that their values and their children are duplicates, i.e., $P(t_{ij}|V_{t_{ij}}, C_{t_{ij}})$, we consider the nodes as duplicates if both their values and their children are duplicates. Thus, the probability is defined as in Eq. (4).

$$P(t_{ij}|V_{t_{ij}}, C_{t_{ij}}) = \begin{cases} 1 & \text{iff } V_{t_{ij}} = C_{t_{ij}} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

**Conditional Probability CP4.** Finally, we define the probabilities of a set of nodes of the same type being duplicates given that each pair of individual nodes in the set are duplicates, i.e., $P(t_{**}|t_{1*}, t_{2*}, \dots)$ and $P(t_{i*}|t_{i1}, t_{i2}, \dots)$. We start by defining the probability $P(t_{**}|t_{1*}, t_{2*}, \dots)$, of the set of nodes being duplicates given that each of its nodes is a duplicate. As before, we assume that the more nodes are duplicates, the higher the probability that the whole set of nodes is a duplicate. The probability can thus be defined as shown in Eq. (5).

$$P(t_{**}|t_{1*}, t_{2*}, \dots, t_{n*}) = \frac{1}{n} \sum_{k=1}^{n} t_{k*} \quad (5)$$

We can now define the probability $P(t_{i*}|t_{i1}, t_{i2}, \dots)$, which reflects the fact that a node in an XML tree is a duplicate if it is a duplicate of at least one node of the same type in the other XML tree. This is represented in Eq. (6).

$$P(t_{i*}|t_{i1}, t_{i2}, \dots, t_{in}) = \begin{cases} 1 & \text{iff } \exists_j \,|\, t_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

### 4.2.1 Final Probability

Once all prior and conditional probabilities are defined, the Bayesian network can be used to compute the probability of two XML trees being duplicates, i.e. $P(t_{11})$, where $t$ is the tag for the root node of both trees. This can be achieved by any probability propagation algorithm, such as those described in [18].

We can illustrate the probability computation for the network of Fig. 2, where we wish to compute the probability of XML trees $U$ and $U'$ being duplicates, i.e., $P(mv_{11})$. According to the network, and applying Eq. (4), the probability is defined as:

$$P(mv_{11}) = \sum_{V_{mv_{11}}, C_{mv_{11}}} P(mv_{11}|V_{mv_{11}}, C_{mv_{11}}) P(V_{mv_{11}}) P(C_{mv_{11}}) = P(V_{mv_{11}}) \times P(C_{mv_{11}}) \quad (7)$$

Similarly, by applying Eq. (2), probability $P(V_{mv_{11}})$ is defined as:

$$P(V_{mv_{11}}) = w_{year} P(mv_{11}[year]) + w_{title} P(mv_{11} title) \quad (8)$$

As for probability $P(C_{mv_{11}})$, according to Eq. (3), we have:

$$P(C_{mv_{11}}) = \frac{P(dr_{11}) + P(cst_{11})}{2} \quad (9)$$

We now proceed by computing probability $P(dr_{11})$, using Eqs. (4) and (2), as follows:

$$P(dr_{11}) = w_{value} P(dr_{11}[value]) = P(dr_{11}[value]) \quad (10)$$

since $w_{value} = 1$, according to Eq. (2).

Similarly, for $P(cst_{11})$, using Eqs. (3) and (5):

$$P(cst_{11}) = P(ac_{**}) = \frac{P(ac_{1*}) + P(ac_{2*})}{2} \quad (11)$$

Using Eqs (6) and (2) we can compute probability $P(ac_{1*})$ as:

$$P(ac_{1*}) = 1 - \prod_{i=1}^{3}(1 - P(ac_{1i}[value])) \quad (12)$$

A similar equation can be obtained for $P(ac_{2*})$.

Finally, joining Eqs. (7) through (12), we have:

$$P(mv_{11}) = \Big(w_{year}P(mv_{11}[year]) + w_{title}P(mv_{11}[title])\Big) \times \Big(P(dr_{11}[value]) + \big(1 - \prod_{i=1}^{3}(1 - P(ac_{1i}[value])) + 1 - \prod_{i=1}^{3}(1 - P(ac_{2i}[value]))\big) \times \frac{1}{2}\Big) \times \frac{1}{2} \quad (13)$$

### 4.2.2 Efficiency and Limitations

It can be easily seen that, if the XML trees being compared do not have multiple nodes of the same type, the procedure proposed in Sec. 3 is linear in the number of nodes of the largest tree, since a constant number of nodes in the network is created for each pair of nodes in the XML trees. Nevertheless, if multiple nodes of the same type do occur, the resulting procedure could be of in the order of $O(n \times n')$, where $n$ and $n'$ are the number of nodes in each tree being compared. However, this would only occur in the worst case, if every type in the XML trees (except for the root nodes) would allow repetition of its elements. Since this is a scalability issue, it is, for now, outside the scope of this paper, although it shall be explored in future work. Furthermore, we note that, in practice, the model construction procedure is used only once. Since we assume that all objects in the dataset comply to the same schema, the model will be the same for all objects, having only a small change when computing the probabilities for repeating nodes of the same type, accounting for the different number of nodes that can occur.

Finally, since the Bayesian network built has no cycles, computing the probabilities is linear in the number of nodes in the network. As can be seen by Eqs. (1) through (6), all probabilities can be computed in linear or constant time (if we account for the cost of computing the similarity between XML node values as constant).

## 5. EVALUATION

We now evaluate our similarity measure developed for XML duplicate detection in terms of effectiveness on both artificial and real world data. Our evaluation covers (i) the impact of default probability on effectiveness, (ii) the impact of data quality on effectiveness, i.e., how errors (e.g., typos or missing data) and error frequencies affect the result, and (iii) a comparison to DogmatiX [26], a state of the art similarity measure for XML duplicate detection.

### 5.1 Data Sets

To test our approach, we used the following three datasets.

**Artificial Data.** The first data set, named the *IMDB* dataset, consists of 4,288 distinct objects representing movies, extracted from the Internet Movie Database[1]. For the experiments, a set of 4,288 artificially generated duplicate objects were added to the collection[2]. Each generated object is a duplicate of one of the initial database, and may contain the following errors: (i) typographical errors, (ii) missing data (e.g., a missing title), and (iii) duplicate erroneous data (e.g., the same movie containing two different titles). Unless stated otherwise, typographical errors, missing data, and duplicate erroneous data occur with 20%, 10%, and 8% probability, respectively.

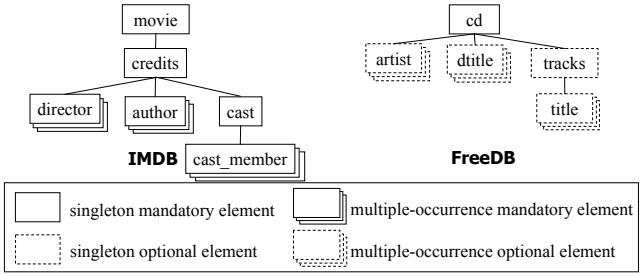


**Figure 3: IMDB and FreeDB XML schemas**

**Real-World Data.** The two remaining data sets consist of real world data. The *IMDB+FilmDienst* dataset was generated by integrating 500 movie objects extracted form IMDB and the same 500 movies extracted from FilmDienst[3]. No artificial data was generated.

The *FreeDB* dataset contains 10,000 distinct objects, representing CDs, extracted from the FreeDB database[4]. With a lot of manual effort, we managed to identify all duplicates within this data set. We found 300 duplicate CD pairs (each pair has been classified as duplicate or non-duplicate in a double-blind process). According to statistics we collected from users performing the manual classification, 17.3% of duplicate classifications are due to missing data, 12.1% to typographical errors, and 11.8% to duplicate erroneous data. The remaining classifications are due to other types of errors we identified, but are not listed here for brevity.

Fig. 3 shows the schemas for the above datasets. The schema we considered for the IMDB and IMDB+FilmDienst datasets is labeled simply as IMDB.

### 5.2 Experimental Setup

The experiments performed on all datasets aim at determining the effectiveness of the BN model we propose in determining if a given pair of XML objects are duplicates. The network constructed for both collections is parameterized as described in Sec. 4. Only objects whose duplicate probability is above or equal to 0.6 are considered duplicates. This threshold was set according to previous empirical results.

We consider all values as textual strings and use $p = 1 - ed(V_1, V_2)/max(|V_1|, |V_2|)$ as prior probability, where $ed(V_1, V_2)$ is the edit distance between string values $V_1$ and $V_2$, and $|V_i|$ is the length of string $V_i$.

To measure effectiveness, we use the commonly used *precision* and *recall*. Precision measures the percentage of correctly identified duplicates contained over the total set of objects determined as duplicates by the system. Recall mea-

[1] http://www.imdb.com

[2] All duplicate data was generated using the Dirty XML Generator available at http://www.hpi.uni-potsdam.de/~naumann/projekte/dirtyxml.

[3] http://film-dienst.kim-info.de/

[4] http://www.freedb.org/

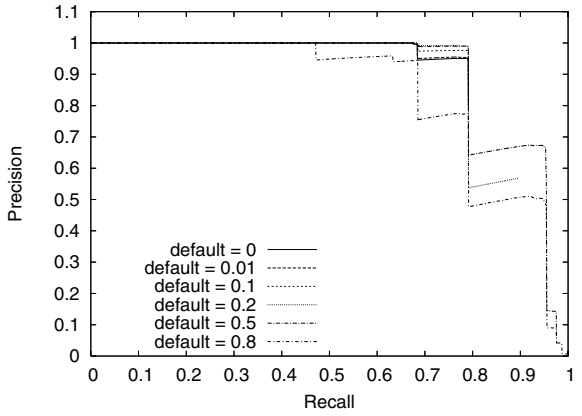


**Figure 4: Effectiveness for varying default prob. $k_a$**

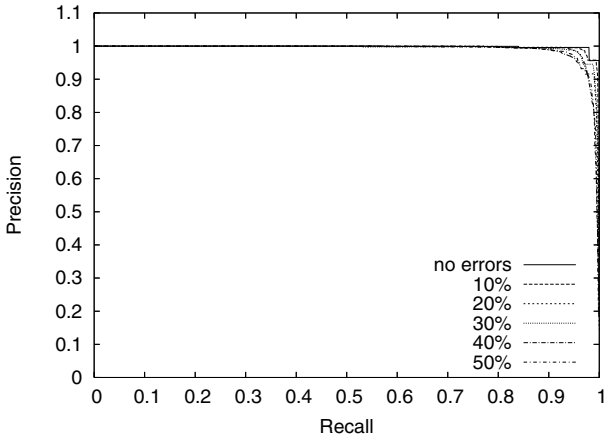


**Figure 5: Precision and recall values for different probabilities of occurrence of typographical errors.**

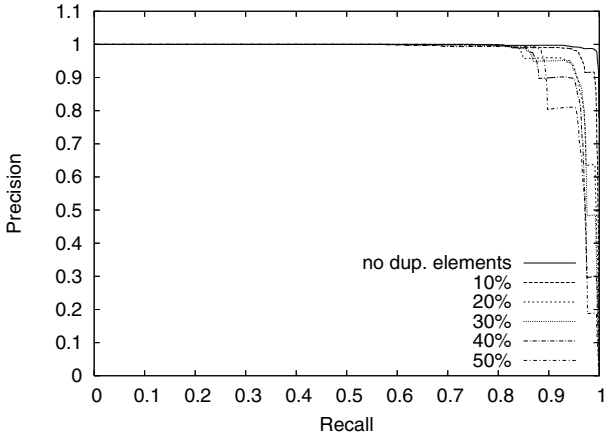


**Figure 6: Precision and recall values for different amounts of duplicate erroneous data.**

sures the percentage of duplicates correctly identified by the system over the total set of duplicate objects.

## 5.3 Experiments

We now discuss experiments we performed to evaluate the effectiveness of our approach.

**Experiment 1.** The first set of experiments aims at examining the impact of the choice of a default probability on the performance of the model. To this end, we vary the default probability $k_a$ (see Eq. 2) between 0 and 0.8. Fig. 4 shows precision and recall for varying default probabilities on the IMDB data set.

**Discussion.** Clearly, the choice of a default probability affects effectiveness. For values below 0.2, objects with missing data are often considered as non-duplicates, because the model behaves as if the missing elements were completely different. This has a clear impact on recall, which reaches a maximum of 79%. For values above 0.5, the opposite occurs. In this case, the model is too permissive, assuming a high similarity for elements that are missing. Thus, although there is a visible increase in recall, precision drops rapidly. Best results are achieved for values between 0.2 and 0.5. In the following experiments, we use default probability value of 0.5, which maximizes both precision and recall.

**Experiment 2.** The next series of experiments were performed to determine the impact of the quality of the data being processed on the performance of the model. To this end, each error probability was varied between 0% and 50% when generating duplicates in the IMDB dataset, while maintaining the remaining at a fixed value. Figs. 5, 6, and 7 show results for varying the probability of typographical errors, duplicate erroneous data, and missing data, respectively.

**Discussion.** In Fig. 5, we observe that precision suffers very little from the increase in typographical errors, maintaining values close to 100%, even at high recall. The model is, therefore, capable of dealing adequately with the occurrence of errors in the textual data fields. This can also be observed when the amount of duplicate erroneous data increases, as shown in Fig. 6. Although the effect on precision is slightly higher, the model is still able to maintain 99% precision for recall values up to 84%.

On the other hand, varying the amount of missing data has a much higher impact on the quality of the results, as shown in Fig. 7. This is particularly evident in precision, which drops sharply when data becomes incomplete. When only 10% of the XML elements are missing, precision is still above to 95% for recall values of about 74%. However, for 20% missing data this can only be achieved for recall values below 55%. Nevertheless, for low recall values, precision still remains quite high, even when 50% of the data is missing.

**Experiment 3.** To provide a better perspective on the results obtained by our proposal, the last set of experiments compares the effectiveness of our approach to another XML similarity measure specifically developed for XML duplicate detection. More specifically, we compare to the state-of-the-art XML duplicate detection similarity measure of *DogmatiX* [26]. DogmatiX uses two different similarity measures: the simple normalized edit distance, also used by the BN model and the edit distance adjusted by the Inverse Document Frequency (IDF) of the words in the values being compared. Using IDF assures that words that occur in many objects are given less weight, since they contribute less to distinguish between the objects. More details can be found in [26].

We compare our approach to DogmatiX on all three data sets in terms of effectiveness. On the IMDB data set, we

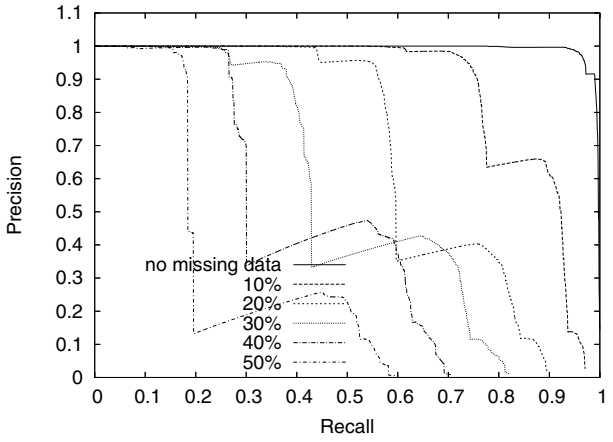


**Figure 7: Precision and recall values for different amounts of missing data.**

perform the same experiments as described in Experiment 2, but only show results where differences are significant, i.e., results for varying missing data (see Fig. 8). On the two real-world data sets, we compare our method to DogmatiX using IDF, as well as to DogmatiX without IDF. Because our model does not use IDF, comparing to the latter variant is more significant. Results on the IMDB+FilmDienst and the FreeDB datasets are reported in Figs. 9 and 10, respectively. All algorithms use a similarity threshold of 0.6 to distinguish between duplicates and non-duplicates.

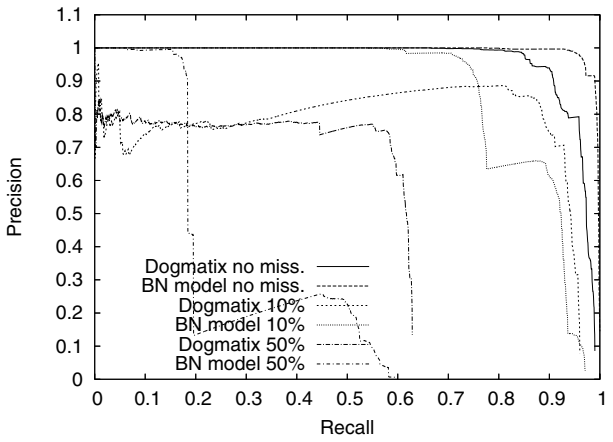


**Figure 8: Comparison between the Dogmatix system and the BN model, for varying amounts of missing data.**

**Discussion.** As can be observed in Fig. 8, when no data is missing, both systems show a similar performance. Nevertheless, for DogmatiX, precision drops below 90% when recall reaches about 91%, whereas for the BN model this drop only occurs for recall values higher than 99%. When 10% of the data is missing, the BN model shows a consistent improvement over DogmatiX. Only for recall higher than 76% does DogmatiX present higher precision values. Further, we can see that the BN model has a much more regular curve, indicating that it is more effective at avoiding false duplicates in the highest ranked results. On the other hand, DogmatiX is more resilient to the increase in missing data. At 50% missing data, although there is a clear decrease in the final recall, precision values do not degrade much, whereas for the BN model, precision drops rapidly for recall values as low as 17%.

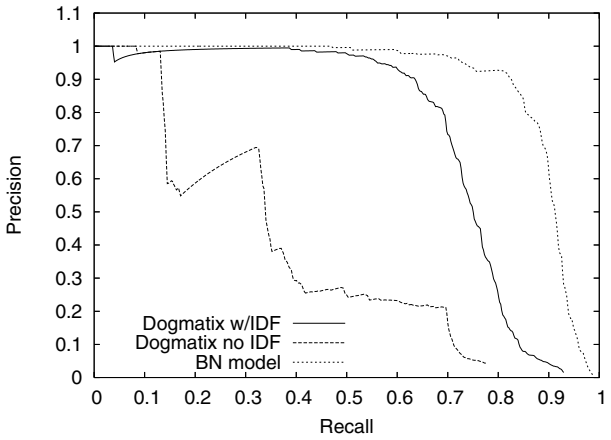


**Figure 9: Precision and recall values on the integrated IMDB+FilmDienst database.**

Considering the results on the integrated IMDB+FilmDienst dataset shown in Fig. 9, we see that the BN model is very effective in detecting duplicates in this collection. Precision remains above 92% and only shows a significant drop when recall reaches about 80%, whereas DogmatiX, with the help of IDF is able to maintain a similar precision only up to 60% recall. Without using IDF, DogmatiX obtains very poor results. Note that the results shown in Fig. 8 only consider the DogmatiX variant that uses IDF.

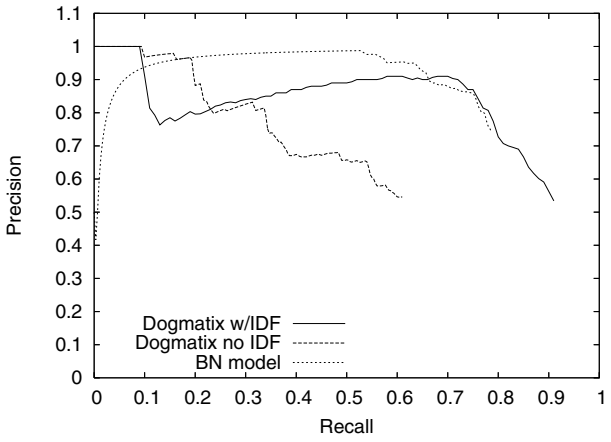


**Figure 10: Precision and recall values on the FreeDB database.**

This is also supported by the results on the 10.000 CDs from FreeDB, shown in Fig. 10, where DogmatiX using IDF significantly outperforms DogmatiX not using IDF. Overall, DogmatiX using IDF shows the best results, being able to retrieve 91% of the known duplicate items. In contrast, the highest recall achieved by the BN model was 79%. On the

other hand, the BN model shows higher overall precision values, always above 90% for recall levels from 7% to 66%. The fact that this collection contains many dummy values for many of the fields in the XML objects (e.g., "track1", "track2", ..., in the CD track titles, or "various artists" in the artists' names), explains the false positives among the highest ranked results. It also explains why a similarity measure that integrates IDF can provide more accurate results.

Throughout our experiments, we see that our BN approach to similarity measurement is very effective in detecting duplicates in XML data. However, it can still incorporate further improvements, such as the use of IDF to further improve effectiveness. Before we conclude and discuss future research directions, we provide an overview of related work.

# 6. RELATED WORK

The approach taken in this work is closely related to previous applications of Bayesian networks in the field of Information Retrieval (IR). Bayesian network models were first introduced in IR by Turtle and Croft in [24], with the purpose of ranking search results. In their model, document terms, documents and user queries are seen as events and are represented as nodes in the network. The model takes the viewpoint that the observation of a document induces belief on its set of index terms, and that specification of such terms induces belief in a user query or information need. Later, a second model was proposed by Ribeiro-Neto and Muntz in [21], demonstrating that BNs can be effectively used to combine different types of information to further improve search results. More recently, Acid et al. [1] presented a third model whose network topology is defined in such way that an exact propagation algorithm can be used to efficiently compute the relevance probabilities of documents.

Bayesian networks have also been applied to other IR problems ranging from relevance feedback [11] to document clustering and classification [9]. In this paper, we apply Bayesian networks to effectively detect duplicates in hierarchical and semi-structured XML data.

Duplicate detection, originally defined by Newcombe et al. [17] and formalized by Fellegi and Sunter [10] has been studied extensively under various aspects. Broadly speaking, research in duplicate detection falls into two categories: techniques to improve *effectiveness* and techniques to improve *scalability* both in space (data size processed) and in time.

In Tab. 2, we summarize some duplicate detection methods, classifying them along two dimensions, namely *data* and *approach*. For data, we distinguish between (i) data in a single relational *table* (ii) *tree* data, and (iii) data represented as a *graph*. The second dimension discerns between three approaches used for duplicate detection: (i) machine *learning*, where models and similarity measures are learned, (ii) the use of *clustering* techniques, and (iii) *iterative* algorithms that iteratively detect pairs of duplicates, which are aggregated to clusters. In Tab. 2, we also show whether an article mainly focuses on scalability ($S$) or effectiveness ($E$).

| | Table | Tree | Graph |
|---|---|---|---|
| **Learning** | Bil.03[5]($E$)<br>Sar.02[22]($E$) | | Sin.05[23]($E$)<br>Bhat.06 [4]($E$) |
| **Clustering** | Chau.05[6]($E$,$S$) | | Kala.06[14]($E$, $S$)<br>Yin06[29]($E$,$S$) |
| **Iterative** | Her.95[12]($S$)<br>Mon.97[16]($S$)<br>Jin03[13]($E$,$S$) | Ana.02[2]($E$,$S$)<br>Puh.06[19]($S$)<br>Mil.06 [15]($E$)<br>Weis04 [25]($E$,$T$)<br>Weis05 [26]($E$) | Dong05[8]($E$)<br>Weis06[27]($E$,$S$)<br>Bhat.04[3]($E$) |

**Table 2: Summary of duplicate detection approaches focusing on scalability (S) or effectiveness (E)**

The work presented in this paper can be classified as an approach to *effectively* detect duplicates in *tree* data. Although learning can be used to improve the probability model, it is not required and is not the focus of this article so we classify it as an iterative algorithm. The novelty compared to other similarity measures considering relationships (i.e., applying on tree and graph data) is that it considers duplicate probability of descendants, instead of the duplicate status of direct neighbors (children) only.

The only similarity measure we are aware of that also considers the complete sub-structure and uses more than just the duplicate status of descendants is the structure-aware similarity measure proposed by Milano et al. [15]. To compare two candidate XML elements rooted at $c$ and $c'$, a maximum overlay between the two trees $T(c)$ and $T(c')$ is computed. In this overlay, two non-leaf nodes can be matched iff they are ancestors of two leaves that are matched. Once a maximal overlay has been determined, its cost is computed. The cost of a pair of leaf nodes in the overlay is defined by a string distance function, e.g., the string edit distance. For non-leaf nodes, the cost is set to 0. The total cost of the overlay is the sum of costs of node pairs and is equal to the distance between two XML trees rooted at $c$ and $c'$. Our similarity measure is more flexible as it leverages the assumption that XML elements can only be duplicates if their XML path from the root is equal. Another limitation of the structure aware similarity measure is that the weight of non-leaf nodes is 0, so the final result essentially depends on the similarity of value nodes. This is not the case for our similarity measure, where probabilities are also computed for non-leaf nodes.

# 7. CONCLUSION AND OUTLOOK

This paper presents a novel method for XML duplicate detection. Using a Bayesian network model, we are able to accurately determine the probability of two XML objects in a given database being duplicates. This model is derived from the structure of the XML objects being compared and all probabilities are computed taking into account not only the values contained in the objects but also their internal structure.

The Bayesian network model we propose not only provides a formal basis for the duplicate detection procedure, but also requires very little parameterization. In fact, all results reported, using three distinct datasets, were obtained using the same configuration, described in Sec. 3, and experiments only required setting a single parameter—the default probability. Nevertheless, the model also provides great flexibility in its configuration, allowing the use of different similarity measures for the field values and different conditional probabilities to combine the similarity probabilities of the XML elements.

Experiments performed on both artificial and real-world data show that the proposed model is able to achieve highly accurate results, yielding both high precision and recall in all cases. To further validate these results, we compared

our approach to a state-of-the-art XML duplicate detection system, DogmatiX. The BN model has consistently shown better results when DogmatiX did not use IDF to improve effectiveness.

The success achieved through the experiments and the flexibility provided by the proposed solution leaves open room for much future work. Among other tasks, we intend to study the use of IDF correction for textual values, the use of domain dependent similarity measures for prior probabilities, extend the BN model construction algorithm to compare XML objects with different structures, experiment with more collections and different network configurations, and apply machine learning methods to derive the conditional probabilities, based on existing data. Another issue we intend to consider is that of scalability, both in space and in time. To improve runtime, we can devise blocking techniques to avoid computing probabilities. Scaling to large amounts of data, and thus large Bayesian networks, requires using external memory and also needs to be studied in the future.

## Acknowledgements

This work was partially supported by FCT project IR-BASE (ref. POSC/EIA/58194/2004), by INRIA Futurs project GEMO, and by GRICES project XClean.

## 8. REFERENCES

[1] S. Acid, L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. An information retrieval model based on simple bayesian networks. *International Journal of Intelligent Systems*, 18(2):251–265, Jan. 2003.

[2] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *Conference on Very Large Databases (VLDB)*, pages 586–597, Hong Kong, China, 2002.

[3] I. Bhattachary, L. Getoor, and L. Licamele. Query-time entity resolution (poster). In *Conference on Knowledge Discovery and Data Mining (KDD)*, pages 529–534, Philadelphia, PA, 2006.

[4] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *Conference on Data Mining (SDM)*, Bethesda, MD, 2006.

[5] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Conference on Knowledge Discovery and Data Mining (KDD)*, pages 39–48, Washington, DC, 2003.

[6] S. Chaudhuri, V. Ganti, and R. Motwani. Robust identification of fuzzy duplicates. In *International Conference on Data Engineering (ICDE)*, pages 865–876, Tokyo, Japan, 2005.

[7] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Conference on Knowledge Discovery and Data Mining (KDD)*, pages 475–480, Edmonton, Alberta, Canada, 2002.

[8] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *Conference on the Management of Data (SIGMOD)*, pages 85–96, Baltimore, MD, 2005.

[9] S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th International Conference on Information and Knowledge Management CIKM 98*, pages 148–155, Bethesda, MD, USA, Nov. 1998.

[10] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 1969.

[11] D. Haines and W. B. Croft. Relevance feedback and inference networks. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2–11, Pittsburgh, PA, USA, June 1993.

[12] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *Conference on the Management of Data (SIGMOD)*, pages 127–138, San Jose, CA, 1995.

[13] L. Jin, C. Li, and S. Mehrotra. Efficient record linkage in large data sets. In *Conference on Database Systems for Advanced Applications (DASFAA)*, Kyoto, Japan, 2003.

[14] D. V. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Transactions on Database Systems (TODS)*, 31(2):716–767, 2006.

[15] D. Milano, M. Scannapieco, and T. Catarci. Structure aware xml object identification. In *VLDB Workshop on Clean Databases (CleanDB)*, Seoul, Korea, 2006.

[16] A. E. Monge and C. P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, Tucson, AZ, 1997.

[17] H. Newcombe, J. Kennedy, S. Axford, and A. James. Automatic linkage of vital records. *Science 130*, (3381):954–959, 1959.

[18] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of plausible inference*. Morgan Kaufmann Publishers, 2nd edition, 1988.

[19] S. Puhlmann, M. Weis, and F. Naumann. Xml duplicate detection using sorted neigborhoods. In *Conference on Extending Database Technology (EDBT)*, pages 773–791, Munich, Germany, 2006.

[20] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23:3–13, 2000.

[21] B. Ribeiro-Neto and R. Muntz. A belief network model for IR. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260, Zurich, Switzerland, Aug. 1996.

[22] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Conference on Knowledge Discovery and Data Mining (KDD)*, pages 269–278, Edmonton, Alberta, 2002.

[23] P. Singla and P. Domingos. Object identification with attribute-mediated dependences. In *Conference on Principals and Practice of Knowledge Discovery in Databases (PKDD)*, pages 297–308, Porto, Portugal, 2005.

[24] H. Turtle and W. B. Croft. Inference networks for document retrieval. In *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1–24, Brussels, Belgium, Sept. 1990.

[25] M. Weis and F. Naumann. Duplicate detection in xml. In *SIGMOD Workshop on Information Quality in Information Systems (IQIS)*, pages 10–19, Paris, France, 2004.

[26] M. Weis and F. Naumann. Dogmatix tracks down duplicates in xml. In *Conference on the Management of Data (SIGMOD)*, pages 431–442, Baltimore, MD, 2005.

[27] M. Weis and F. Naumann. Detecting duplicates in complex xml data. In *International Conference on Data Engineering (ICDE)*, Atlanta, GA, 2006.

[28] W. E. Winkler. Overview of record linkage and current research directions. Technical report, U. S. Bureau of the Census, 2006.

[29] X. Yin, J. Han, and P. S. Yu. LinkClus: Efficient clustering via heterogeneous semantic links. In *Conference on Very Large Databases (VLDB)*, pages 427–438, Seoul, Korea, 2006.