

# Similarity Metric for XML Documents

Zhongping Zhang<sup>1,2</sup> Rong Li<sup>1</sup> Shunliang Cao<sup>1,3</sup> Yangyong Zhu<sup>1</sup>

<sup>1</sup>(Fudan University, ShangHai China 200433) <sup>2</sup>(Yanshan University, HeBei China 066004)

<sup>3</sup>(NingBo University, ZheJiang China 315211) (<http://www.dmgroun.org.cn>)  
{011021396,011021394,011021393}@fudan.edu.cn, zhuyangyong@etang.com

## Abstract

Since XML documents can be represented as trees, Based on traditional tree edit distance, this paper presents structural similarity metric for XML documents ,which is based on edge constraint, path constraint, and inclusive path constraint, and similarity metric based on machine learning with node costs. It extends scope for searching XML documents, and improves recall and precision for searching XML documents.

## 1 Introduction

XML(eXtensible Markup Language) is standard for data representation and exchange on the Web, which promulgated by W3C(Word Wide Web Consortium) on February 1998 [Bray *et al.*, 1998]. XML describes emphasize particularly on structural information for self-description, extensibility, analysis, and open, which makes content description and display usage apart[Bosak *et al.*, 1998]. People can read document easily, parse syntax, and use DTD(Document Type Definition)[Fan and Libkin, 2002][Arenas and Libkin, 2002]. It is an important cause that XML becomes standard language on the Web world. There are generally lots of XML documents without DTDs, especially documents generated from HTML. But existing DTDs have been worthy of searching and processing yet. The similarity for XML data enlarges query scope. XML data is tree model unlike in relational database where atomic object is tuple. The tuple distance can be defined accurately on numerical value type, character type etc., but in XML model, the distance metric must be defined on the new atomic object: XML tree model. Due to the inherent structural complexity and ill-defined “closeness” in XML model, however, devising an intuitive metric is not a trivial task.

The rest of the paper is organized as follows: section 2 introduces related works, section 3 defines basic conceptions, definitions, and notations of XML documents. Section 4 presents set metric ways based on edge constraint and path constraint. Section 5 gives a linear metric way based on inclusive path constraint. Section 6 represents a distance metric based on costs. Section 7 presents an algorithm based on machine learning with node costs. We conclude with summary and discussion of future work in section 8.

## 2 Related Work

E.W.Myers uses the LCS(Longest Common Subsequence) algorithm to computer two plain text files in [E.W.Myers, 1986], which can not be generalized to handle structured data because it don't understand the hierarchical structure information contained in such data set. Chawathe *et al.* also presents a heuristic algorithm, MH-Diff, to detect change in unordered structured documents in [S.Chawathe and H.Garcia-Molina, 1996]. However, the worst case of algorithm is in  $O(n^3)$ , and tests very small documents. XMLTreeDiff [Curbera and D.A.Epstein, 1999] computers the difference between two XML documents using DOMHash. However, it uses conflicts with cost model employed by Zhang and Shasha's algorithm, and may not generate an optimal result. General tree edit distance problem for unordered trees is NP-hard [Mani *et al.*, 2001] [Tai, 1979][Barnard *et al.*, 1995], works in [Zhang and Shasha, 1989][Chawathe *et al.*, 1995] instead try to find an efficient algorithm for limited case such as ordered/binary trees or trees with additional constraints. The best known algorithm for computing tree edit distance between two ordered trees is by Zhang and Shasha with the time complexity of roughly  $O(n^4)$ , where n is the number of the nodes in a tree. Wang *et al.* [Wang *et al.*, 2003] argue that an unordered model (only ancestor relationships are significant) is more suitable for most database application, but using an unordered model, change detection is substantially harder than using the ordered model. The XML data model is naturally propitious to tree edit distance because tree structure can be obtained from XML documents easily. This paper represents similarity metric ways for XML documents, based on traditional tree edit distance and deep study on XML documents data with constraints.

## 3 Basic Conceptions and Notations

**Definition 1** A DTD(Document Type Definition)[Fan and Libkin, 2002][Arenas and Libkin, 2002] is defined to be  $D = (E, A, P, R, r)$ , where:

- $D$  is a name of document type definition, symbolic tuple semantics;
- $E$  is a finite set of element types;
- $A$  is a finite set of attributes, disjoint from  $E$ ;
- $P$  is a mapping from  $E$  to element type

definitions: for each  $\tau \in E$ ,  $P(\tau)$  is a regular expression  $\alpha$  defined as follows:

$$\alpha ::= S \mid \varepsilon \mid \tau' \mid \alpha \mid \alpha \mid \alpha, \alpha \mid \alpha^*$$

where  $S$  denotes string type,  $\tau' \in E$ ,  $\varepsilon$  is the empty word, and “|”, “,” and “\*” denote union, concatenation, and the Kleene closure, respectively;

- $R$  is a mapping from  $E$  to  $\rho(A)$ , the power-set of  $A$ ; if  $@l \in R(\tau)$  then we say  $@l$  is defined for  $\tau$ ;
- $r \in E$  and is called the element type of the root. Without loss of generality, we assume that  $r$  does not occur in  $P(\tau)$  for any  $\tau \in E$ .

**Definition 2** A **XML Document Tree** is defined to be a tree  $T=(V, lab, ele, att, val, root)$ , where:

- $T$  is a name of XML Document Tree, symbolic tuple semantics;
- $V$  is a finite set of nodes;
- $lab$  is a mapping from  $V$  to  $E \cup A \cup \{S\}$ , for each node  $v \in V$ , if  $lab(v)=\tau$  and  $\tau \in E$ , then we say  $v$  is an element node; if  $lab(v) \in A$ , then we say  $v$  is a attribute node; if  $lab(v)=S$ , then we say  $v$  is a text node;
- $ele$  is a partial function from  $V$  to  $V^*$  such that for any  $v \in V$ ,  $ele(v)=[v_1, \dots, v_n]$  and  $lab(v_i) \in P(\tau)$ , where  $i \in [1, n]$ ;
- $att$  is a partial function from  $V$  to  $A$  such that for any  $v \in V$  and  $@l \in A$ ,  $att(v, @l)$  is defined iff  $lab(v)=\tau, \tau \in E$  and  $@l \in R(\tau)$ ;
- $val$  is a partial function from  $V$  to string  $S$  such that for any  $v \in V$ ,  $val(v)$  is defined iff  $lab(v)=S$  or  $lab(v) \in A$ .

$root \in V, lab(root)=r$  called the root of tree  $T$ .

**Definition 3** An **Ordered Labeled Tree**  $T$  is a rooted tree in which the children of each node are ordered and labeled. If any node  $v$  has  $k$  children then these children are uniquely identified, left to right, as  $v_1, v_2, \dots, v_k$ .  $l(T)$  is a label of the root node of  $T$ .

**Definition 4** **Node Level** is the number of paths(or edges) from the root to self-node, also named node depth. Node level of the root node is 0.

**Definition 5** **Node Branch** is the number of children of a node.

**Definition 6** **Node Base** is the number of maximum possible children of node that can be

obtained from the DTD schema [Thompson *et al.*, 2001][Biron and Malhotra, 2001].

## 4 Set Metric Method

**Definition 7 (Edge Constraint)** Given an ordered labeled tree  $T$ ,  $u \rightarrow v$  is an edge constraint denoted a node  $u$  abut to  $v$ . If  $u=u'$  and  $v=v'$ , then we say that  $u \rightarrow v$  and  $u' \rightarrow v'$  are matched. Let  $X$  and  $Y$  be XML document trees,  $e^X$  and  $e^Y$  be the set of edge constraints in  $X$  and  $Y$ , respectively.

**Example 1** Figure 1 denotes three XML document trees to *reference* of GenBank(database of gene).

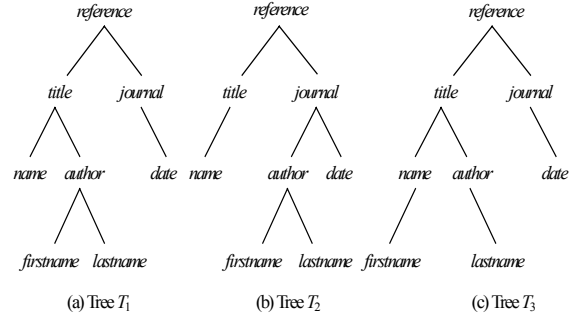


Figure 1 Example of similar XML documents trees

In order to write expediently, we abbreviate each element name.  $r, t, j, n, a, d, f$ , and  $l$  are abbreviation of *reference, title, journal, name, author, date, firstname, and lastname*, respectively. We can obtain the following sets from definition 7.

$$e^{T_1} = \{r \rightarrow t, r \rightarrow j, t \rightarrow n, t \rightarrow a, j \rightarrow d, a \rightarrow f, a \rightarrow l\};$$

$$e^{T_2} = \{r \rightarrow t, r \rightarrow j, t \rightarrow n, j \rightarrow a, j \rightarrow d, a \rightarrow f, a \rightarrow l\};$$

$$e^{T_3} = \{r \rightarrow t, r \rightarrow j, t \rightarrow n, t \rightarrow a, j \rightarrow d, n \rightarrow f, a \rightarrow l\}.$$

If edge constraints match more between two XML document trees, they are more similar. So similarities among XML documents based on edge constraints can be defined as follows:

$$sim_e(X, Y) = \frac{|e^X \cap e^Y|}{|e^X \cup e^Y|} \quad (1)$$

where,  $|e^X \cap e^Y|$  denotes the number of edge constraints of XML document tree  $X$  and  $Y$  matched, i.e., intersection of edge constraints;  $|e^X \cup e^Y|$  denotes union of edge constraints.

We can get similarity of edge constraints for three document trees of Example 1 based on Equation 1 as follows:

$$sim_e(T_1, T_2) = \frac{|e^{T_1} \cap e^{T_2}|}{|e^{T_1} \cup e^{T_2}|} = \frac{6}{8}$$

$$sim_e(T_1, T_3) = \frac{|e^{T_1} \cap e^{T_3}|}{|e^{T_1} \cup e^{T_3}|} = \frac{6}{8}$$

$$sim_e(T_2, T_3) = \frac{|e^{T_2} \cap e^{T_3}|}{|e^{T_2} \cup e^{T_3}|} = \frac{5}{9}$$

Clearly,  $sim_e(T_1, T_2)$  and  $sim_e(T_1, T_3)$  are

<sup>1</sup> Part of this work was supported by Major Project of Shanghai Community of Science and Technology, China(No. 02DJ14013).

same, one can not determine which is more similar to  $T_1$ . This is because Equation 1 does not consider the structural characteristics of XML document trees, i.e., level relationship between node and edge, that is to say path constraint problem.

**Definition 8 (Path Constraint)** Given an ordered labeled tree  $T$ ,  $v_1 \rightarrow v_2, v_2 \rightarrow v_3, \dots, v_{n-1} \rightarrow v_n$  is a path constraint denoted consecutive edge constraints from the root node to the leaf node. Where  $v_1$  is the root node,  $v_n$  is the leaf node, shortening is  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n$ . If  $v_1 = v'_1, v_2 = v'_2, \dots, v_n = v'_n$ , then we say that  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n$  and  $v'_1 \rightarrow v'_2 \rightarrow \dots \rightarrow v'_n$  are matched. Let  $X$  and  $Y$  be XML document trees,  $p^X$  and  $p^Y$  be the set of path constraints in  $X$  and  $Y$ , respectively.

Similarly, if path constraints match more between two XML document trees, they become more similar. So similarities among XML documents based on path constraints can be defined as follows:

$$sim_p(X, Y) = \frac{|p^X \cap p^Y|}{|p^X \cup p^Y|} \quad (2)$$

where,  $|p^X \cap p^Y|$  denotes the number of path constraints of XML document tree  $X$  and  $Y$  matched, i.e., intersection of path constraints;  $|p^X \cup p^Y|$  denotes union of path constraints.

**Example 2** Consider three trees  $T_1, T_2$ , and  $T_3$  in Figure 1 again. The sets of path constraints as follows:

$$\begin{aligned} p^{T_1} &= \{r \rightarrow t \rightarrow n, r \rightarrow t \rightarrow a \rightarrow f, r \rightarrow t \rightarrow a \rightarrow l, r \rightarrow j \rightarrow d\}; \\ p^{T_2} &= \{r \rightarrow t \rightarrow n, r \rightarrow j \rightarrow a \rightarrow f, r \rightarrow j \rightarrow a \rightarrow l, r \rightarrow j \rightarrow d\}; \\ p^{T_3} &= \{r \rightarrow t \rightarrow n \rightarrow f, r \rightarrow t \rightarrow a \rightarrow l, r \rightarrow j \rightarrow d\}. \end{aligned}$$

We can get similarity of path constraints for three document trees in Figure 1 based on Equation 2 as follows:

$$\begin{aligned} sim_p(T_1, T_2) &= \frac{|p^{T_1} \cap p^{T_2}|}{|p^{T_1} \cup p^{T_2}|} = \frac{2}{6} \\ sim_p(T_1, T_3) &= \frac{|p^{T_1} \cap p^{T_3}|}{|p^{T_1} \cup p^{T_3}|} = \frac{2}{5} \\ sim_p(T_2, T_3) &= \frac{|p^{T_2} \cap p^{T_3}|}{|p^{T_2} \cup p^{T_3}|} = \frac{1}{6} \end{aligned}$$

$sim_p(T_1, T_2) < sim_p(T_1, T_3)$  shows that  $T_3$  is more similar to  $T_1$  than  $T_2$ . Clearly, path constraint is superior to edge constraint. Edge constraint keeps parent-child relationship, but that path constraint treats only the whole root-to-leaf path as an atomic unit, so it exists inclusive path problem. For instance, path constraint  $r \rightarrow t \rightarrow n \rightarrow f$  in  $T_3$  contains path constraint  $r \rightarrow t \rightarrow n$  in  $T_1$ . That is to say that two path constraints are same as from the first node *reference* to the third node *name*. But both exist essential difference, they have different depths.

## 5 Linear Metric Method

**Definition 9 (Inclusive Path Constraint)** Given two path constraints  $p: v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_m \rightarrow \dots \rightarrow v_n$  and  $p': v'_1 \rightarrow v'_2 \rightarrow \dots \rightarrow v'_m$ . If  $v_1 = v'_1, v_2 = v'_2, \dots, v_m = v'_m$  and  $n \geq m$ , then we say that path constraint  $p$  includes path constraint  $p'$ , denoted  $p' \subseteq p$ .

Traditional similarity maps strings to a linear space in a manner that measures similarity clustering lexicographic order. We use it for reference to measure similarity because elements of XML documents are parsed strings. Let us briefly review the description in [Jagadish *et al.*, 2000].

Let the size of the alphabet be  $\alpha$ , with an established lexicographic order on the symbols is the alphabet. Choose an integer  $\beta > 2\alpha$ . Let a string of length  $n$  be  $S_1 S_2 \dots S_n$ , with each symbol  $S_i$  mapped to an integer  $t_i$  between 1 and  $\alpha$ . The string as a whole is then mapped to  $t_1 / \beta + t_2 / \beta^2 + \dots + t_n / \beta^n$ . Generally let  $\beta = 2\alpha + 1$ .

According to above description, XML document tree is mapped as follows:

(1). Given document tree  $X$ ,  $S^X$  denotes set of all nodes label,  $t_X(l)$  is a mapping function that returns an established order value of node label;

(2). Path constraint  $p_i = v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n$  maps into some rational number:

$$Q_X(p_i) = t_X(v_1) / \beta + t_X(v_2) / \beta^2 + \dots + t_X(v_n) / \beta^n$$

where  $\beta = 2\alpha + 1$ ,  $\alpha$  is the number of node label.

After each path of trees is mapped to numeric space, one can define the similarity of two trees  $X$  and  $Y$  in terms of average pair-wise Euclidean distance as follows:

$$sim_{inc}(X, Y) = 1 - \sqrt{\frac{\sum_{i=1}^m |Q_X(p_i) - Q_Y(p_i)|^2}{m}} \quad (3)$$

where  $\sqrt{\frac{\sum_{i=1}^m |Q_X(p_i) - Q_Y(p_i)|^2}{m}}$  is Euclidean distance,

$p_i$  is path constraint, Equation 3 shows that if linear error is smaller, then the similarity is greater.

**Example 3** Consider Figure 1 again. The set of node label  $S = \{author, date, firstname, journal, lastname, name, reference, title\}$ . We use abbreviation for convenience yet. Then  $t(a)=1, t(d)=2, t(f)=3, t(j)=4, t(l)=5, t(n)=6, t(r)=7, t(t)=8, \alpha=8, \beta=2\alpha+1=17$ . Path constraints are still as follows:

$$\begin{aligned} p^{T_1} &= \{r \rightarrow t \rightarrow n, r \rightarrow t \rightarrow a \rightarrow f, r \rightarrow t \rightarrow a \rightarrow l, r \rightarrow j \rightarrow d\}; \\ p^{T_2} &= \{r \rightarrow t \rightarrow n, r \rightarrow j \rightarrow a \rightarrow f, r \rightarrow j \rightarrow a \rightarrow l, r \rightarrow j \rightarrow d\}; \\ p^{T_3} &= \{r \rightarrow t \rightarrow n \rightarrow f, r \rightarrow t \rightarrow a \rightarrow l, r \rightarrow j \rightarrow d\}. \end{aligned}$$

Then,

$$\begin{aligned} Q_{T_1}(r \rightarrow t \rightarrow n) &= t(r) / \beta + t(t) / \beta^2 + t(n) / \beta^3 \\ &= 7/17 + 8/17^2 + 6/17^3 \end{aligned}$$

$$\begin{aligned}
Q_{T_1}(r \rightarrow t \rightarrow a \rightarrow f) &= t(r)/\beta + t(t)/\beta^2 + t(a)/\beta^3 \\
&\quad + t(f)/\beta^4 = 7/17 + 8/17^2 + 1/17^3 + 3/17^4 \\
Q_{T_1}(r \rightarrow t \rightarrow a \rightarrow l) &= t(r)/\beta + t(t)/\beta^2 + t(a)/\beta^3 \\
&\quad + t(l)/\beta^4 = 7/17 + 8/17^2 + 1/17^3 + 5/17^4 \\
Q_{T_1}(r \rightarrow j \rightarrow d) &= t(r)/\beta + t(j)/\beta^2 + t(d)/\beta^3 \\
&= 6/17 + 4/17^2 + 2/17^3
\end{aligned}$$

Likewise,

$$\begin{aligned}
Q_{T_2}(r \rightarrow t \rightarrow n) &= 7/17 + 8/17^2 + 6/17^3 \\
Q_{T_2}(r \rightarrow j \rightarrow a \rightarrow f) &= 7/17 + 4/17^2 + 1/17^3 + 3/17^4 \\
Q_{T_2}(r \rightarrow j \rightarrow a \rightarrow l) &= 7/17 + 4/17^2 + 1/17^3 + 5/17^4 \\
Q_{T_2}(r \rightarrow j \rightarrow d) &= 6/17 + 4/17^2 + 2/17^3
\end{aligned}$$

We can figure out  $sim_{inc}(T_1, T_2)$  based on Equation 3. But  $sim_{inc}(X, Y)$  is limited of equivalent path constraints between two documents.

## 6 Costs Metric Method

Generally, the problem of computing the distance between two trees is based on ordered labeled trees. When dealing with unordered trees, the problem is known to be NP-complete [Zhang and Shasha, 1989]. Thus people focused on finding restricted cases such as ordered or degree-2 tree with which a notion of distance is meaningful for an application under consideration. The traditional edit distance between two strings is the minimum number of edit operations (e.g., insert, delete, or update) required to transform from one string to the other. The problem of computing the distance between two trees is a generalization of the problem of computing the distance between two strings to labeled trees. With regard to query, it is compute distance between query tree and data trees, i.e., the cost problem of edit operations transformed from source tree to target tree.

Zhang and Shasha algorithm does not consider cost of all the operators. Although to assign identical cost to all operations, that is,  $cost(insert)=cost(delete)=cost(update)=\sigma$ , or different cost, that is,  $cost(insert)=\alpha$ ,  $cost(delete)=\beta$ ,  $cost(update)=\gamma$ , does not effect correctness of algorithm, but identical cost does not consider semantic constraints. It is prodigious impact on XML model to insert a node or to delete node near the root node and the leaf node. The operation near the root node is more important than which near the leaf nodes. Thus edit distance based on variable cost is further incarnate similarity for XML documents, however, complexity is higher.

The distance between two object trees  $X$  and  $Y$  is defined as the summation of the minimal operation costs that are needed in order to convert the source tree  $X$  to the target tree  $Y$ :

$$d_{io}(X, Y) = \sum_{i,j} cost(op_i(v_j)) \quad (4)$$

Where the operator  $op_i$  is applied to the node  $v_j$  of the source tree  $X$ . Further, let  $\alpha(op_i)$  and  $\theta(v_j)$  be the characteristics of the operator  $op_i$  and the node  $v_j$ , respectively. Then, Equation 4 can be rephrased to:

$$\cos t(op_i(v_j)) = \frac{1}{\Delta} (\sum_{i} w_i \alpha_i(op_i) + \sum_{j} w_j \theta_j(v_j)) \quad (5)$$

Where  $w_i$  and  $w_j$  are weight factors,  $\Delta$  is a normalization factor. Most of the existing works using tree edit distance assume the cost of operation to be a constant  $\alpha$ . On the contrary, the Equation 5 provides a generic method to assign arbitrary costs to operators differently using their characteristics. There are various types of operators and node characteristics that may affect the cost assignment. We illustrate them below:

### (1) Operator type cost

We assign variable costs based on the operator, see to Figure 2. The distance based on *update* operation between query tree and target tree is  $\sigma$ , that is,  $dist(Q, T_1) = cost(update(OLAP \rightarrow datamining)) = \sigma$ , the distance based on *insert* operation between query tree and target tree is  $\delta$ , that is,  $dist(Q, T_2) = cost(insert(firstname)) = \delta$ .

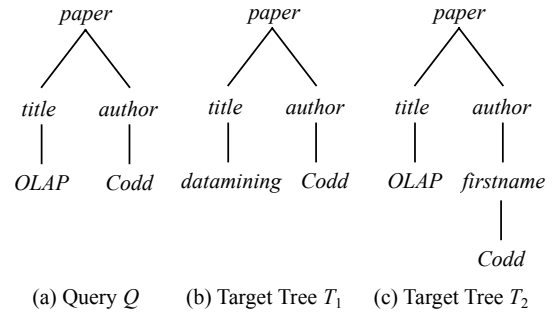


Figure 2 Example of variable operator costs

### (2) Node level cost

The cost for an operation can be dependent on the level of the node that the operation is applied. The intuitive idea of this variation is that editing the root node and editing the leaf node may yield significantly different impact. Assuming the operation near the root node is more important than the ones further down from the root node, the following formula can be used to assign a variable cost:

$$cost(v) = \frac{1}{(1 + level(v))^k}$$

where  $k \geq 1$  is relevant factor. This formula assigns  $cost(v)=1$  when the root node is edited. It shows that cost is up to maximal value, i.e., it leads the whole tree to change from top to leaf node when the root

node is edited.

Figure 3 is the example of variable level costs assignment. For convenience, let relevant factor  $k=2$ , then  $cost(cite)=1/2^2=0.25$  in Figure3(b). Similarly,  $cost(cite)=1/4^2=0.0625$  in Figure3(c). Thus, target tree  $T_2$  is “better” than target tree  $T_1$ .

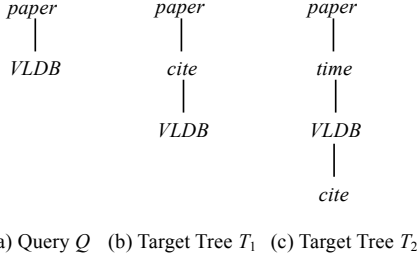


Figure 3 Example of variable level costs

### (3) Node branch cost

The number of children of parent node effect efficiency of tree operation directivity. Based on definition 5 and definition 6, we can define the formula to measure node branch cost as follows:

$$cost(v) = b(v) = \frac{\text{number of children of } v}{\text{base of } v}$$

For instance, consider the following simple XML DTD:

```
<!ELEMENT conference (code?, paper|(title, cite?))>
<!ELEMENT paper (title, location?)>
```

Then, base of *conference* is 3, and base of *paper* is 2. XML document trees accorded with above DTD are seen to Figure 4.

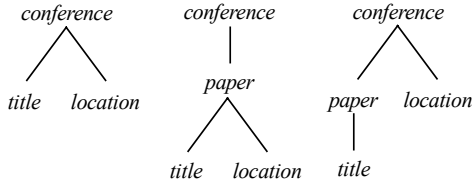


Figure 4 Example of variable node branch costs

Then,  $b(paper)=2/2=1$  in Figure 4(b). Similarly,  $b(paper)=1/2=0.5$  in Figure 4(c). Thus editing the *paper* node in target tree  $T_1$  is more expensive than which in target tree  $T_2$ .

### (4) Semantic interpretation

The semantic of node in the context may affect the cost of operation. For example, consider Figure 5, where two target trees  $T_1$  and  $T_2$  have the same cost based on node type, node level, and node branching. In terms of the context, *title* in  $T_1$  describes official, and *title* in  $T_2$  describes article. These subtle differences just materialize by data mining. Confidences and supports [Han and Kamber, 2001] of *title* in  $T_1$  and  $T_2$  are different, i.e., complexities are different. These make costs different. If complexities are higher, then costs are higher, if complexities are

lower, then costs are lower. There exists indiscernible relationship between document query and semantic. These are correlative with applicative fields. All problems need to be studied further.

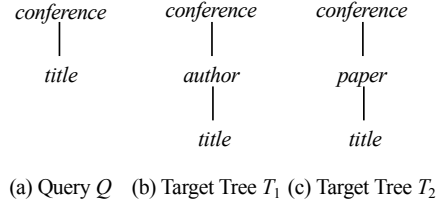


Figure 5 Example of document tree semantic

## 7 Machine learning cost algorithm

Consider the following three operation types and three node characteristics in the Equation 4:

①  $o_1$ : insert operation type; ②  $o_2$ : delete operation type; ③  $o_3$ : update operation type; ④  $\theta_1$ : node level; ⑤  $\theta_2$ : node branch; ⑥  $\theta_3$ : semantic interpretation.

Let  $o_1(\text{insert})=0.4$ ,  $o_2(\text{delete})=0.4$ ,  $o_3(\text{update})=0.6$ ,  $\theta_1(v)=0.5$ ,  $\theta_2(v)=0.5$ ,  $\theta_3(v)=0$ ,  $w_{o_1}=w_{o_2}=w_{o_3}=w_{\theta_1}=w_{\theta_2}=w_{\theta_3}=1$  and  $\Delta=6$ , so the distance between  $X$  and  $Y$  can be computed as follows:

$$\begin{aligned} d_{io}(X, Y) &= \sum_{i,j} cost(op_i(v_j)) \\ &= \frac{1}{\Delta} \left( \sum_{i=1}^3 w_{o_i} o_i(op_i) + \sum_{j=1}^3 w_{\theta_j} \theta_j(v_j) \right) \quad (6) \\ &= \frac{1}{6} (1 \times 0.4 + 1 \times 0.4 + 1 \times 0.6 + 1 \times 0.5 + 1 \times 0.5 + 1 \times 0) \\ &= 0.4 \end{aligned}$$

i.e., the distance between  $X$  and  $Y$  is 0.4. If we assign uniform costs to each operation in Equation 4, i.e.,  $cost(\text{insert})=cost(\text{delete})=cost(\text{update})=1$ , then  $d_{io}(X, Y)=cost(op_i(v_j))=1$ , which is quite far from 0.4. It says the importance to assign unequal costs. Thereby, in order to obtain the furthest suitable costs in the given application field, the machine learning technology should be used [Mitchell, 2002]. The cost-measuring algorithm based on machine learning is given as following.

- First of all, assume there exists  $n$  training set instances  $I_1, \dots, I_n$ , and each instance  $I_i (1 \leq i \leq n)$  has 3-tuple:
  - $Q_i$ : query asked;
  - $A_i$ : approximate answer;
  - $L_i = \begin{cases} \text{Yes} & A_i \text{ is relevant to } Q_i \\ \text{No} & \text{otherwise} \end{cases}$
- There exists objective function  $u$  and its approximate function  $\hat{u}$ , extracts  $\theta_i$  (operation types and node characteristics) from XML documents and

calculates in application field.

Initialize objective function  $u(I_i)$ , e.g.,

$$u(I_i) = \begin{cases} +1 & L_i = Yes \\ -1 & L_i = No \end{cases}$$

calculate the approximate function:  
 $\hat{u}(I_i) = w_1\theta_1 + w_2\theta_2 + \dots + w_k\theta_k$ .

**Algorithm SWC** (Similarity metric for XML documents based on Weight Costs)

**Input:** XML document trees  $X, Y$

**Output:** Similarity  $d_{io}(X, Y)$  of XML document trees  $X, Y$

**SWC** ( $X, Y$ )

- (1) Initialize  $w_i, \theta_i, \eta$  and  $u(I_i)$ ;
- (2) While the terminating condition is not satisfied {
- (3) For each  $I_i$  {
- (4)  $\hat{u}(I_i) = w_1\theta_1 + w_2\theta_2 + \dots + w_k\theta_k$ ;
- (5)  $\Delta u_i = u(I_i) - \hat{u}(I_i)$ ;
- (6)  $\hat{w}_i = w_i + \eta \times \Delta u_i \times \theta_i$ ; /\*  $\eta$  is used to moderate the size of weight, so called learning factor \*/
- (7)  $\Delta w_i = \hat{w}_i - w_i$ ;
- (8) };
- (9)  $d_{io}(X, Y) = \frac{1}{k} \sum_i^k w_i \theta_i$ ;
- (10) Return  $d_{io}(X, Y)$ .

Terminating condition: Training stops when

- $\Delta w_i$  is 0, or less than some specified threshold, or
- More than the cycle times which was preassigned.

When the error  $\Delta w_i$  is 0, it means there is not any weight to be updated, and when the error  $\Delta w_i$  is a very small positive number, each weight will be increased to the corresponding ratio of characteristics, thus advance the value of  $\hat{u}(I_i)$ , and reduce the error. When there exist enough training sets, the weights  $\hat{w}_i$  will be changed continuously and eventually be converged. Once the weights are converged, the Equation 6 will satisfy non-uniform costs' effect.

### Algorithm analysis

The validity of the algorithm can be proved easily by the content in the above sections.

The terminability of the algorithm can be determined by the cycle of **While** and **For**. Because the number of training set is  $n$ , the cycle **For** is terminable. The Terminating condition of **While** in the second step of the algorithm is determined by  $\Delta w_i$  and the cycle times which are preassigned.  $\eta$  is learning factor, commonly be the very small positive number, which used to moderate the size of the weight and make  $\Delta w_i$  becomes zero or less than

some specified threshold. So the **While** cycle is terminable and the whole algorithm is terminable too.

The time complexity of the algorithm is mainly determined by the cycle of **While** and **For**. Let the maximum of  $\Delta w_i$  and the cycle times preassigned is  $m$ , the number of training sets is  $n$ . The time complexity of the whole algorithm is  $O(m \times n)$ .

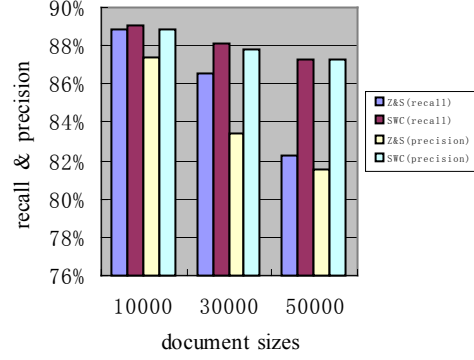


Figure 6 Performance analysis

We realize Z&S algorithm [Zhang and Shasha, 1989] and SWC algorithm presented in this paper based on documents data in GenBank. GenBank is the NIH genetic sequence database, an annotated collection of all publicly available DNA sequences. A new release is made every two months. GenBank is part of the International Nucleotide Sequence Database Collaboration, which is comprised of the DNA DataBank of Japan (DDBJ), the European Molecular Biology Laboratory (EMBL), and GenBank at the National Center for Biotechnology Information. Each GenBank entry includes a concise description of the sequence, the scientific name and taxonomy of the source organism, and a table of features that identifies coding regions and other sites of biological significance, such as transcription units, sites of mutations or modifications, and repeats. Protein translations for coding regions are included in the feature table. Bibliographic references are included along with a link to the Medline unique identifier for all published sequences. Performance of algorithms is shown in Figure 6. It concludes that recall and precision descend when document sizes increase. However, ranges of recall and precision of SWC are lesser than those of Z&S. That is, SWC algorithm improves recall and precision based on similarity of XML documents.

## 8 Conclusions and future works

In this paper we mainly discussed similarity metric for XML documents based on sets and costs, which could offer more abundant method to the similarity metric for XML documents and the distance metric of clustered XML documents based on tree edit distance [Nierman and Jagadish, 2002] and facilitate recall and precision for XML documents. The essence of

similarity for XML documents is to support the query on XML documents and find out all candidate documents according to the similarity metric, in which we can choose the candidates that are most close to or next close to the source document and order them in turn, etc. Various measurements have various results in different application fields and conditions related to the material application. But the semantic constraints play an important role in the measurement. The semantic has certain subjectivity besides the objectivity constraints. So the semantic constraint is very complex which should be studied further. Since DTD defined well the schema of XML document, the document accorded with DTD should have some similarity to the DTD itself. So we can consider from the view point of the measurement of the similarity between document and DTD when we depict the similarity between the documents. But because the element type defined by DTD has the selectivity (i.e.?) and repeatability (i.e.\*) characteristics, we should consider the operation of ? and \* when we construct a DTD tree. We will study on the comparability issue between the document and DTD in the future.

## References

- [Arenas and Libkin, 2002] M. Arenas and L. Libkin. A Normal Form for XML Documents. Symposium on Principles of Database Systems (PODS'02), Madison, Wisconsin, U.S.A. ACM press, 2002: 85-96.
- [Barnard *et al.*, 1995] D.T.Barnard, G.Larke, and N.Duncan. Tree-to-Tree Correction for Document Trees. Technical report, Queen's Univ. Computer Science Dept., Jan. 1995.
- [Biron and Malhotra, 2001] P.V.Biron and A.Malhotra(Eds). "XML Schema Part 2:Datatypes". W3C Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
- [Bosak *et al.*, 1998] J. Bosak, T. Bray, D. Connolly, and, E. Maler, etc. Sperberg-McQueen, L. Wood, and J. Clark. W3C XML Specification DTD. <http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm>. June 1998
- [Bray *et al.*, 1998] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 Specification, w3c.org, available: <http://www.w3.org/TR/1998/REC-xml-19980210>. 1998
- [Chawathe *et al.*, 1995] S.S.Chawathe, A.Rajaman, H.Garcia-Molina, and J.Widom. Change Detection in Hierarchically Structure Information. In ACM SIGMOD, Montreal, Quebec, Canada, Jun. 1996.
- [Curbera and D.A.Epstein, 1999] Curbera and D.A.Epstein. Fast Difference and Update of XML Documents. *XTech'99*, San Jose, March 1999.
- [E.W.Myers, 1986] E.W.Myers. An O(ND) Difference Algorithm and Its Variations. *Algorithmica*, 1(2):251-266,1986.
- [Fan and Libkin, 2002] W. Fan and L. Libkin. On XML Integrity Constraints in the Presence of DTDs, Journal of the ACM (JACM), 2002, Volume 49, Issue 3: 368- 406.
- [Han and Kamber, 2001] Jiawei Han, Micheline Kamber. DATA MINING CONCEPTS and TECHNIQUES. (gravure), 2001.
- [Jagadish *et al.*, 2000] H.V.Jagadish, N.Koudas, and D.Srivatava. On Effective Multi-Dimensional Indexing for Strings. In *ACM SIGMOD*, Dallas, TX, May 2000.
- [Mani *et al.*, 2001] M.Mani, D.Lee, and M.Murata. Normal Forms for Regular Tree Grammars. Technical report, UCLA Computer Science Dept., 2001.
- [Mitchell, 2002] T.M.Mitchell. "*Machine Learning*". McGraw-Hill Co., 1997.
- [Nierman and Jagadish, 2002] A.Nierman and H.V.Jagadish. Evaluating Structural Similarity in XML Documents. In *Int'l Workshop on the Web and Databases(WebDB)*, Madison, WI, Jun. 2002.
- [S.Chawathe and H.Garcia-Molina, 1996] S.Chawathe and H.Garcia-Molina. Meaningful Change Detection in Structured Data. *Proceeding of the ACM SIGMOD International Conference on Management of Data*, June 1996.
- [Tai, 1979] K.Tai. The Tree-to-Tree Correction Problem. *J.ACM*, 26(3):422-433,1979.
- [Thompson *et al.*, 2001] H.S.Thompson, D.Beech, M.Maloney, and N.Mendelsohn(Eds). "XML Schema Part 1:Structures". W3C Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>.
- [Wang *et al.*, 2003] Yuan Wang, David J.DeWitt, and Jin-Yi Cai. X-Diff: An Effective Change Detection Algorithm for XML Documents. ICDE 2003.
- [Zhang and Shasha, 1989] K.Zhang and D.Shasha. Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. *SIAM J. Comput.*, 18(6)1245-1262, DEC. 1989.