XMLデータにおける類似極大部分木の効率的な探索

寺島慎太郎† 天笠 俊之† 北川 博之†

† 筑波大学大学院システム工学研究科 〒305–8573 茨城県つくば市天王台 1–1–1
E-mail: †{shintaro,amagasa,kitagawa}@kde.cs.tsukuba.ac.jp

あらまし 近年，XML 形式で記述されたデータが爆発的に普及している．それに伴い，構造は異なるものの，類似する内容を一部共有している XML データ群が多く存在するようになった．本研究では二つの XML データに対し，その類似する内容を表すできるだけ大きな部分，すなわち最も大きな類似する部分木 (類似極大部分木) のペアを探索する効率的な手法を提案する．XML データの類似度は，テキストノードと木構造の両方を考慮する．また提案手法では類似度を再帰的に計算する．先に子ノードをルートとする部分木の類似度を計算し，その結果を，その親ノードをルートとする部分木の類似度計算に利用することで，計算量の削減を図っている．最後に実験によって提案手法の有用性を検証する．

キーワード XML，情報検索

A Scheme for Finding Maximal Similar Subtrees from XML Data

Shintaro TERAJIMA†, Toshiyuki AMAGASAI†, and Hiroyuki KITAGAWA†

† Graduate School of Systems and Information Engineering, University of Tsukuba
1–1–1 Tennodai, Tsukuba, Ibaraki 305–8573, Japan
E-mail: †{shintaro,amagasa,kitagawa}@kde.cs.tsukuba.ac.jp

Abstract Due to the recent popularization of XML data, there have been a lot of XML data that have different structures while present similar contents. In this paper, we propose an efficient method for finding pairs of maximal similar subtrees from a pair of XML data. In our approach, we consider both similarities with respect to text and structure. Specifically, we calculate similarities of XML subtrees in a bottom-up fashion. We calculate the similarity of a pair of subtrees in the light of the similarities of the subtrees so that we can save the computational time. At last, we show the feasibility of our method by some experiments.

Key words XML, information retrieval

1. まえがき

XML が W3C によって勧告されて以来，XML 形式のデータは，様々な分野で使用されるようになった．その急速な普及に伴い，構造が異なるものの，一部に類似する内容を含んでいる XML データ群が多く存在するようになった．図 1 は Web 上に公開されている XML データ「DBLP Bibliography」(左) と「ACM SIGMOD」(右) の一部である．この二つの XML は，著者とタイトルの順番が逆になっている，後者のみ URL の情報を含んでいるなど，構造が異なっている．しかし，著者とタイトルが一致していることから，この二つの XML は同じ論文を表していることが分かる．

このような異なる情報元に対する，類似する内容を表す部分の探索は，検索やクラスタリングなど，その情報元を有効活用する上で重要な技術である．本研究では，二つの XML データを対象とし，その中に含まれる類似する内容を表すできるだけ大きな部分を探索する手法を提案する．本研究ではこの「類似する内容を表すできるだけ大きな部分」を,「類似する最も大きな部分木 (類似極大部分木)」と呼ぶ．すなわち本手法では，この類似極大部分木のペアを探索することとなる．

例えば図 1 の XML は，前述の通り同じ論文を表すデータである．そのため類似極大部分木は <article>〜</article> タグに囲まれた部分 (四角で囲まれた部分) である．一方図 2 の XML は図 1 と同じ情報元の XML であるが，著者のみ一致しタイトルが異なることから，同じ著者が書いた別の論文であることが分かる．そのため類似極大部分木は <author>〜</author> タグに囲まれた部分となる．

類似する部分木を探索する手法として，対象となる部分木 (部分グラフ) の木編集距離を計算し，それを類似度として類似する部分木を探索する手法が既に研究されている [1] [2] (木編集距

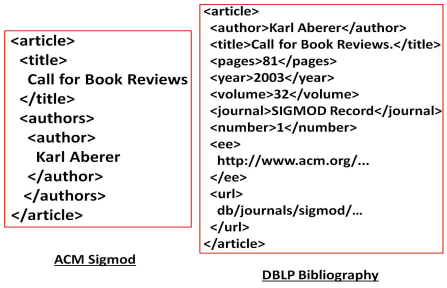


図 1　同じ論文を表す「ACM SIGMOD」「DBLP Bibliography」の XML の一部

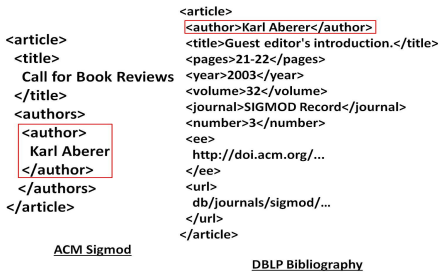


図 2　異なる論文を表す「ACM SIGMOD」「DBLP Bibliography」の XML の一部

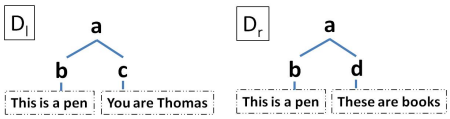


図 3　類似する XML の例

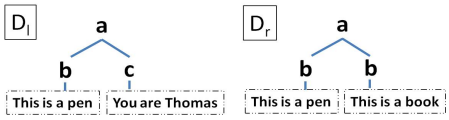


図 4　複数の部分木がマッチする XML の例

離とは，一方の木構造をもう一方のものと同じ形にするのに要する，最小の編集コストを指す)．これらの手法は，兄弟の順序を考慮する XML 木 (順序木) を対象としている．しかし実際の XML データは，兄弟の順序に意味を持たないことがある．例えば図 1 の XML の場合，<title>〜</title> で囲まれたタイトルを表す部分と <author>〜</author> で囲まれた著者を表す部分の順番を入れ替えたとしても，論文目録データとしてその内容が変化することはない．またこれらの既存手法では木構造に対する類似度のみを考慮しているが，実際の XML データの中にはその構造よりも，テキストノードの内容の方がより重要な意味を持つデータも存在する．本研究では兄弟の順序を考慮しない XML 木 (非順序木) を対象として，類似極大部分木を探索するより効率的な手法を提案する．また類似度に関しても，テキストに対する類似度と構造に対する類似度の両方を考慮していく．

2. 提 案 手 法

本稿における記号の定義は表 1 のとおりである．これ以外の記号については，その都度定義していく．

表 1　本稿における記号の定義

$D_l = (V_l, E_l, \lambda_l, r_l)$ $D_r = (V_r, E_r, \lambda_r, r_r)$	計算対象となる XML 木 ($V = V^T \cup V^I$: ノードの集合, E : エッジの集合, λ : 各ノードとそのラベルのマッピング, r : ルートノード)
V_l^T, V_r^T	テキストノードの集合
V_l^I, V_r^I	テキストノード以外のノード (内部ノード) の集合
$T_l = \langle t_l^1, t_l^2, ... \rangle$ $T_r = \langle t_r^1, t_r^2, ... \rangle$	V^T の後置順シーケンス
$I_l = \langle i_l^1, i_l^2, ... \rangle$ $I_r = \langle i_r^1, i_r^2, ... \rangle$	V^I の後置順シーケンス
$subtree(v)$	ノード v をルートとする部分木
$text(v)$	ノード v をルートとする部分木に含まれるテキストノードの重複しない単語集合
$size(v)$	ノード v をルートとする部分木のノード数
$children(v)$	ノード v の子ノードの集合
$descendant(v)$	ノード v の子孫ノードの集合
$parent(v)$	ノード v の親ノード
$height(v)$	ノード v の高さ
$label(v)$	ノード v のラベル (ノード名)

2.1 類似部分木，類似極大部分木の定義

それぞれノード $v_l \in V_l$， $v_r \in V_r$ をルートとする部分木ペアが次の条件を満たすとき，その部分木ペアを「類似部分木 (ペア)」と定義する．

（1） テキストノードに対する類似度 (テキスト類似度) が，あらかじめ与えられた閾値以上となる

（2） XML が持つ木構造に対する類似度 (構造類似度) が，あらかじめ与えられた閾値以上となる

（3） v_l の子ノード，v_r の子ノードをルートとする部分木 (子部分木) のいくつかが互いに類似していて，その部分木の v_l，v_r をルートとする部分木に対して占める割合 (類似する子部分木の占有率) が，あらかじめ与えられた閾値以上となる

そして類似部分木 (ペア) のうち，少なくともどちらか一方の部分木が他の類似部分木に含まれないものを「類似極大部分木 (ペア)」と定義する．本論文では今後，類似部分木ペアの探索をマッチング，類似部分木ペアそのものをマッチする (マッチングが成立する) 部分木ペアと呼ぶ．

2.2 類似度の定義

提案手法では，類似度を再帰的に定義している．すなわちあるノードをルートとする部分木 (親部分木) の類似度を，その子ノードをルートとする部分木 (子部分木) の類似度を用いて定義する．またマッチする相手のいない部分木は類似度を 0 とする．

2.2.1 テキスト類似度

テキストに対する類似の指標の一つとして，テキストを重複を許さない単語の集合とみなし，その集合の共起の度合いをそのテキストの類似度とする Jaccard 係数がある．本研究では Jaccard 係数に基づきテキスト類似度を評価する．ノード $v_l \in V_l$， $v_r \in V_r$ をルートとする部分木同士のテキスト類似度 $Sim_T(v_l, v_r)$ を次のように定義する．

$$Sim_T(v_l, v_r) = \frac{\text{(マッチする子部分木の) テキストノードの一致する単語数の和}}{\text{(全子部分木の) テキストノードの全単語数} - \text{テキストノードの一致する単語数の和}}$$

全単語数が部分木に含まれる全てのテキストノードを対象にするのに対し，一致する単語数の和はマッチングが成立している子部分木に含まれるテキストノードのみ考慮する (テキストノード単体の場合を除く)．この際先に類似度を計算した子部分木の一致する単語を記憶しておき，それを親部分木のテキスト類似度の計算に用いる．またマッチする相手のいない子部分木は，一致する単語がないとみなして計算する．

2.2.2 構造類似度

木構造に対する類似の指標の一つとして，1. 節で紹介した木編集距離がある [3]．本研究では木編集距離に基づき構造類似度を評価する．ノード $v_l \in V_l$， $v_r \in V_r$ をルートとする部分木同士の木編集距離 $Dist_S(v_l, v_r)$ および構造類似度 $Sim_S(v_l, v_r)$ は次のとおりである．

$Dist_S(v_l, v_r) =$
(マッチする子部分木同士の木編集距離の和) ＋
(マッチしない子部分木のノード数の和) ＋
(親ノード同士の編集距離)

$Sim_S(v_l, v_r) = 1 - \frac{Dist_S(v_l, v_r)}{\text{双方の子部分木のノード数の和}+1}$

ノード同士の編集距離は双方のノード名が同じ場合は 0，そうでない場合は 1 となる．部分木同士の木編集距離は，先に計算した子部分木の木編集距離を記憶しておき，それを親部分木の木編集距離の計算に用いる．またマッチする相手のいない子部分木は，部分木そのものを削除するコストを木編集距離とする．

2.3 類似度の計算手順

提案手法では，より葉に近い小さな部分木同士の類似度を先に計算する．そして，順により根に近い大きな部分木同士の類似度を計算していく中で，類似極大部分木の探索を行う．その際，親部分木の類似度計算に子部分木の類似度の計算結果を利用することで，計算量の削減を図っている．また子部分木にマッチする部分木が一つも存在しない親部分木は，類似度の計算を行わない．例えば図 3 の XML の場合、類似度の計算手順は次のようになる．

（1） テキストノードを部分木とみなし，全テキストノードペアに対して類似度を計算する．図 3 の場合，全テキストノードペアの類似度は次のようになる．

$sim_T(t_l^1, t_r^1) = \frac{4}{4+4-4} = 1.0$ $sim_T(t_l^1, t_r^2) = \frac{0}{4+3-0} = 0.0$
$sim_T(t_l^2, t_r^1) = \frac{0}{4+3-0} = 0.0$ $sim_T(t_l^2, t_r^2) = \frac{1}{3+3-1} = 0.2$

（2） マッチする部分木の親部分木同士の類似度を計算する．仮に t_l^1, t_r^1 のみがマッチする場合，$subtree(i_l^1)$ と $subtree(i_r^1)$ との類似度のみ計算する．$subtree(i_l^2)$ および $subtree(i_r^2)$ はマッチする相手がいないため，$subtree(i_l^2)$ と D_r の全ての部分木との類似度，および $subtree(i_r^2)$ と D_l の全ての部分木との類似度は 0 となる．

$sim_T(i_l^1, i_r^1) = \frac{4}{4+4-4} = 1.0$ $sim_S(i_l^1, i_r^1) = 1 - \frac{0+0+0}{1+1+1} = 1.0$

（3） (2) を繰り返す．図 3 の場合，$subtree(i_l^3)$ と $subtree(i_r^3)$ との類似度を計算する．

$sim_T(i_l^3, i_r^3) = \frac{4}{7+7-4} = 0.4$ $sim_S(i_l^3, i_r^3) = 1 - \frac{0+4+0}{4+4+1} = 0.56$

2.4 子部分木の選択

マッチングについて，一方の XML の一つの部分木に対し，もう一方の XML の複数の部分木がマッチする場合がある．これらの親部分木同士の類似度を計算する場合，マッチする子部分木のペアのテキスト，構造類似度の合計が最も大きくなるように，子部分木のペアの選択を行う．

例えば図 4 の XML において，$subtree(i_l^3)$ と $subtree(i_r^3)$ との類似度を計算する場合を想定する．マッチするテキストノードが t_l^1 と t_r^1，t_l^1 と t_r^2 のみの場合，$subtree(i_l^1)$ と $subtree(i_r^1)$，$subtree(i_l^1)$ と $subtree(i_r^2)$ との類似度は次のようになる．

$Sim_T(i_l^1, i_r^1) = \frac{4}{4+4-4} = 1.0$ $Sim_T(i_l^1, i_r^2) = \frac{3}{4+4-3} = 0.6$
$Sim_S(i_l^1, i_r^1) = Sim_S(i_l^1, i_r^2) = 1 - \frac{0+0+0}{1+1+1} = 1.0$

構造類似度は両ペアとも同じであるが，テキスト類似度は $subtree(i_l^1)$ と $subtree(i_r^1)$ の方が大きい．そのため $subtree(i_l^3)$ と $subtree(i_r^3)$ との類似度を計算するとき，$subtree(i_l^1)$ と $subtree(i_r^1)$ がマッチし，$subtree(i_r^2)$ はマッチする相手がいないものとみなす．$subtree(i_l^3)$ と $subtree(i_r^3)$ との類似度は次のようになる．

$Sim_T(i_l^3, i_r^3) = \frac{4}{7+5-4} = 0.5$
$Sim_S(i_l^3, i_r^3) = 1 - \frac{0+4+0}{4+4+1} = 0.56$

この子部分木の選択は，類似度に関する割当問題 (最大マッチング問題の特殊ケース) とみなすことができる．そのため Hungarian 法 [4] のような既存のアルゴリズムで解くことができる．

2.5 提案アルゴリズム

提案手法のアルゴリズムを図 5〜図 10 に示す．なおマッチする極大部分木ペア候補の集合を $M = \{..., (v_l, v_r, W, Sim_T, SizeDist, Sim_S), ...\}$ と表す．M で格納する情報は，対象部分木のルートノード v_l, v_r，一致するテキストノードの単語集合 W，テキスト類似度 Sim_T，木のサイズ - 木の編集距離 $SizeDist$ 及び構造類似度 Sim_S である．

proceeding searchMSS
input
D_l, D_r: 探索対象となる木
$\phi_t, \phi_s, \rho, \eta, \tau$:
テキスト類似度，構造類似度，類似する子部分木の占有率，木の高さ，構造類似度を反映する木のノード数の閾値
output M: 類似極大部分木ペア候補の集合

for each $t_l \in T_l$ **do**
 for each $t_r \in T_r$ **do**
  $Sim_T = textNodeSimilarity(t_l, t_r)$
  if $Sim_T \geqq \phi_t$ **then**
   $M \leftarrow M \cup \{(t_l, t_r, text(s) \cap text(t), Sim_T, 2, 0)\}$

for each $i_l \in I_l$ **do**
 $M \leftarrow M \cup findMatches(i_l, \phi_t, \phi_s, \rho, \tau)$

$M_{small} \leftarrow \{(i'_l, i'_r) | (i'_l, i'_r, w, x, y, z) \in M \wedge$
$(height(i'_l) < \eta \vee height(i'_r) < \eta)\}$
$M \leftarrow M - M_{small}$

return M

図 5 類似極大部分木の探索アルゴリズム

proceeding textNodeSimilarity
input t_l, t_r: 計算対象となるテキストノード
output Sim: テキスト類似度
return $\frac{|text(t_l) \cap text(t_r)|}{|text(t_l) \cup text(t_r)|}$

図 6 テキストノード単体のテキスト類似度計算アルゴリズム

proceeding textSimilarity
input
Max_l, Max_r:
最大マッチングとなる子部分木のルートノードの集合
$MEET$:
マッチする子部分木のテキストノードの一致する単語集合
output Sim: テキスト類似度

return $\frac{|MEET|}{|\bigcup_{i'_l \in Max_l} text(i'_l)|+|\bigcup_{i'_r \in Max_r} text(i'_r)|-|MEET|}$

図 7 部分木同士のテキスト類似度計算アルゴリズム

proceeding structureSize-Distance
input
i_l, i_r: 計算対象となる部分木のルートノード
M_{max}: マッチする子部分木ペアの最大マッチング
Max_l, Max_r:
最大マッチングとなる子部分木のルートノードの集合
output $SizeDist$: サイズ-木編集距離

return $\sum_{i'_l \in Max_l, i'_r \in Max_r, (i'_l, i'_r, x, sd, y, z) \in M_{max}} sd + 2 - NodeDist(i_l, i_r)$

図 8 (サイズ-木編集距離) の計算アルゴリズム

proceeding structureSimilarity
input
i_l, i_r: 計算対象となる部分木のルートノード
$SizeDist$: サイズ-木編集距離
output Sim: 編集類似度

$Dist = \sum_{i'_l \in children(i_l)} size(i'_l) + \sum_{i'_r \in children(i_r)} size(i'_r) - SizeDist + 2(NodeDist(i_l, i_r) - 1)$
return $1 - \frac{Dist}{\sum_{i'_l \in children(i_l)} size(i'_l) + \sum_{i'_r \in children(i_r)} size(i'_r) + 1}$

図 9 構造類似度の計算アルゴリズム

図 5 は類似極大部分木を探索するメインアルゴリズムである．最初に全テキストノードの類似度を計算する．その後 D_l を後置順に探索し，類似極大部分木ペアを探索していく．D_l を全て探索した後，類似極大部分木ペア候補から、部分木の高さが双方とも閾値以上となるペアを出力する．

図 6 および図 7 は，テキスト類似度を計算するアルゴリズムである．計算方法は 2.2.1 節のとおりである．

図 8 および図 9 は構造類似度を計算するアルゴリズムである．計算方法は 2.2.2 節のものと基本的に同じであるが，ノード v_l, v_r をルートとする部分木同士の木編集距離 $Dist_S(v_l, v_r)$ を次のように置き換えている (計算結果は同じである)．

$Dist_S(v_l, v_r) =$
(全子部分木のノード数の和) −
(マッチする子部分木のノード数の和) +
(マッチする子部分木同士の木編集距離の和) +
(親ノード同士の編集距離)

図 10 は，D_l 側の部分木との類似度計算対象となる D_r 側の部分木を探索し，その類似度が閾値を超えるか判別するアルゴリズムである．流れは次のとおりである．

（1） (2 行目) D_l 側の部分木のマッチしている全ての子部分木 (C_l と表す) を探索する．

（2） (3-6 行目) (1) の子部分木のマッチング先である，D_r 側の全ての子部分木 (C_r と表す) の全親ノードを探索する．

（3） (7-9 行目) D_l 側の対象ノード i_l をルートとする部分木と (2) の親ノード i_r をとルートとする D_r 側の部分木を類似度の計算対象とし，子部分木の選択を行う (M_{max} は子部分木ペアの類似度に対する最大マッチング，Max_l および Max_r はその D_l 側，D_r 側の子部分木のルートノードの集合を表す)．

（4） (10-13 行目) 類似する子部分木の占有率を計算する．

（5） (14-22 行目) (4) が閾値以上の場合，構造類似度を計算する．このとき，M 内の「木のサイズ - 木の編集距離

proceeding findMatches
input
i_l: D_l 側の対象ノード
$\phi_t, \phi_s, \rho, \tau$:
テキスト類似度，構造類似度，類似する子部分木の占有率，
構造類似度を反映する木のノード数の閾値
output M: 類似極大部分木ペア候補の集合

if $height(i_l) > 1$ **then**
$C_l \leftarrow \{i'_l | i'_l \in children(i_l) \wedge (i'_l, v, w, x, y, z) \in M\}$
for each $i'_l \in C_l$ **do**
$C_r \leftarrow \{i'_r | (i'_l, i'_r, w, x, y, z) \in M\}$
for each $i'_r \in C_r$ **do**
$i_r = parent(i'_r)$
$M_{max} \leftarrow MaximumMatch(C_l, C_r)$
$Max_l \leftarrow \{i'_l | (i'_l, v, w, x, y, z) \in M_{max}\}$
$Max_r \leftarrow \{i'_r | (v, i'_r, w, x, y, z) \in M_{max}\}$
if $\sum_{i'_l \in Max_l, i'_r \in Max_r, (i'_l, i'_r, w, x, y, z) \in M} size(i'_l)/size(i_l) \geqq \rho$ and
$\sum_{i'_l \in Max_l, i'_r \in Max_r, (i'_l, i'_r, w, x, y, z) \in M} size(i'_r)/size(i_r) \geqq \rho$ **then**
$SizeDist =$
$structureSizeDistance(i_l, i_r, M_{max}, Max_l, Max_r)$
$Empty_l \leftarrow$
$\{i'_l | i'_l \in Max_l \wedge height(i'_l) = 1 \wedge i'_l \, is\, not\, text\}$
$Empty_r \leftarrow$
$\{i'_r | i'_r \in Max_r \wedge height(i'_r) = 1 \wedge i'_r \, is\, not\, text\}$
for each $i'_l \in Empty_l, i'_r \in Empty_r$ **do**
if $label(i'_l) = label(i'_r)$ **then** $SizeDist = SizeDist + 2$
$Sim_S = structureSimilarity(i_l, i_r, SizeDist)$
if $size(i_l) < \tau$ or $size(i_r) < \tau$ or $Sim_S \geqq \phi_s$ **then**
$MEET \leftarrow \bigcup_{(v, w, W, x, y, z) \in M_{max}} W$
$Sim_T = textSimilarity(Max_l, Max_r, MEET)$
if $Sim_T \geqq \phi_t$ **then**
$M \leftarrow M \cup \{(i_l, i_r, MEET, Sim_T, SizeDist, Sim_S)\}$
$M_r \leftarrow M_r \cup i_r$

for each $i_r \in M_r$ **do**
$M_{descendant} \leftarrow \{(i'_l, i'_r) | i'_l \in descendants(i_l) \wedge i'_r \in descendants(i_r) \wedge (i'_l, i'_r, w, x, y, z) \in M\}$
$M \leftarrow M - M_{descendants}$

return M

図 10 部分木ペアが類似するかの判別アルゴリズム

($SizeDist$)」の情報を構造類似度の計算に利用する．また双方で一致する空ノード (テキストノードでないリーフノード) があった場合，構造類似度の計算の際に考慮する ($Empty_l$ および $Empty_r$ は D_l 側，D_r 側のリーフノードである子部分木の集合を表す).

（6） (23-25 行目) (5) が閾値以上の場合，テキスト類似度を計算する．但し部分木のノード数が閾値未満の場合，構造類似度は考慮しない (構造類似度が閾値未満でもマッチしているとみなす)．またこのとき，M 内の「一致するテキストノードの単語集合 (W)」の情報をテキスト類似度の計算に利用する ($MEET$ はマッチする子部分木に含まれる全テキストノードの一致する単語集合を表す).

（7） (26-28 行目) (6) が閾値以上の場合，類似極大部分木候補として計算結果を記憶する．

（8） (29-32 行目) 対象の部分木ペアがマッチする場合，その子孫ノードをルートとする部分木 (子孫部分木) を極大木候補から外す (M_r はマッチングが成立した D_r 側の部分木のルートノードを表す).

D_l 側の全部分木が D_r 側の多くとも一つの部分木とマッチする場合，時間計算量は $O(min(|D_l|^2, |D_r|^2) + min(height_l, height_r) \times min(|text_l|, |text_r|))$，空間計算量は $O(min(|leaf_l|, |leaf_r|))$ となる ($|D_l|, |D_r|$ は木のノード数，$height_l, height_r$ は木の高さの最大値，$|text_l|, |text_r|$ はテキストノード数，$|leaf_l|, |leaf_r|$ はリーフノード数を表す)．また D_l 側の全部分木が D_r 側の全ての部分木とマッチする場合 (最悪のケース)，時間計算量は $O(|D_l|^2 \times |D_r|^2 + min(height_l, height_r) \times |text_l| \times |text_r|)$，空間計算量は $O(|leaf_l| \times |leaf_r|)$ となる．

図 11 の XML に対して提案手法を当てはめた場合，計算の流れは次のようになる (構造類似度，テキスト類似度，類似する子部分木の占有率の閾値は共に 0.5，木の高さの閾値を 1，構造類似度を反映させる木のノード数の閾値を 0 とする．また v_l, v_r をルートとする部分木に対する類似する子部分木の占有率を $ratio(v_l), ratio(v_r)$ と表す).

（1） 全てのテキストノードのペアに対して類似度を計算する．マッチするテキストノードペアとその類似度は次のとおりである．

$Sim_T(t_l^1, t_r^1) = \frac{2}{2+2-2} = 1.0$ $Sim_T(t_l^2, t_r^3) = \frac{2}{3+2-2} = 0.67$
$Sim_T(t_l^3, t_r^2) = \frac{2}{2+2-2} = 1.0$

現地点での類似極大部分木候補は $(t_l^1, t_r^1), (t_l^2, t_r^3), (t_l^3, t_r^2)$ をルートとする部分木である．

（2） D_l 側のノード i_l^1 をルートとする部分木と，その子部分木のマッチング先である D_r 側の部分木の親ノード i_r^1 をルートとする部分木が計算対象となる．類似する子部分木の占有率および類似度は次のとおりである．

$ratio(i_l^1) = ratio(i_r^1) = \frac{1}{2} = 0.5$
$Sim_T(i_l^1, i_r^1) = \frac{2}{2+2-2} = 1.0$ $Sim_S(i_l^1, i_r^1) = 1 - \frac{0+0+0}{1+1+1} = 1.0$

全て閾値以上となるので，$subtree(i_l^1)$ と $subtree(i_r^1)$ がマッチする．このとき $subtree(i_l^1)$ および $subtree(i_r^1)$ の子孫である t_l^1 および t_r^1 が候補から外れる． 現地点での類似極大部分木候補は $(i_l^1, i_r^1), (t_l^2, t_r^3), (t_l^3, t_r^2)$ をルートとする部分木である．

（3） (2) と同様にして，$subtree(i_l^2)$ と $subtree(i_r^3)$ が計算対象となる．類似する子部分木の占有率および類似度は次のとおりである．

$ratio(i_l^2) = ratio(i_r^3) = \frac{1}{2} = 0.5$
$Sim_T(i_l^2, i_r^3) = \frac{2}{3+2-2} = 0.67$
$Sim_S(i_l^2, i_r^3) = 1 - \frac{0+0+0}{1+1+1} = 1.0$

全て閾値以上となるので，$subtree(i_l^2)$ と $subtree(i_r^3)$ がマッチし，t_l^2 および t_r^3 が候補から外れる．現地点での類似極大部分木候補は $(i_l^1, i_r^1), (i_l^2, i_r^3), (t_l^3, t_r^2)$ をルートとする部分木である．

（4） $subtree(i_l^3)$ と $subtree(i_r^4)$ が計算対象となる．このときマッチする子部分木ペアは $subtree(i_l^1), subtree(i_r^1)$ および $subtree(i_l^2), subtree(i_r^3)$ である．類似する子部分木の占有率および類似度は次のとおりである．

$ratio(i_l^3) = \frac{4}{5} = 0.8$ $ratio(i_r^4) = \frac{4}{7} = 0.57$
$Sim_T(i_l^3, i_r^4) = \frac{4}{5+6-4} = 0.57$
$Sim_S(i_l^3, i_r^4) = 1 - \frac{0+2+1}{4+6+1} = 0.73$

全て閾値以上となるので，$subtree(i_l^3)$ および $subtree(i_r^4)$ がマッチし，$subtree(i_l^1)$ と $subtree(i_r^1)$，および $subtree(i_l^2)$ と $subtree(i_r^3)$ が候補から外れる．現地点での類似極大部分木候補は $(i_l^3, i_r^4), (t_l^3, t_r^2)$ をルートとする部分木である．

（5） $subtree(i_l^4)$ および $subtree(i_r^2)$ が計算対象となる．類似する子部分木の占有率および類似度は次のとおりである．

$ratio(i_l^4) = ratio(i_r^2) = \frac{1}{2} = 0.5$
$Sim_T(i_l^4, i_r^2) = \frac{2}{2+2-2} = 1.0$ $Sim_S(i_l^4, i_r^2) = 1 - \frac{0+0+0}{1+1+1} = 1.0$

全て閾値以上となるので，$subtree(i_l^4)$ および $subtree(i_r^2)$ がマッチし，t_l^3 および t_r^2 が候補から外れる．現地点での類似極大部分木候補は $(i_l^3, i_r^4), (i_l^4, i_r^2)$ をルートとする部分木である．

（6） $subtree(i_l^5)$ のマッチする子部分木は $subtree(i_l^3)$ と $subtree(i_l^4)$ である．しかし $subtree(i_l^3)$ のマッチング先のルートノードである i_r^4 には親ノードが存在しない．したがって計算対象となるのは，$subtree(i_l^4)$ のマッチング先の親ノードをルートとする部分木である $subtree(i_r^4)$ のみとなる．このときマッチする子部分木ペアは $subtree(i_l^4), subtree(i_r^2)$ のみである．類似する子部分木の占有率および類似度は次のとおりである．

$ratio(i_l^5) = \frac{2}{8} = 0.25$ $ratio(i_r^4) = \frac{2}{7} = 0.29$
$Sim_T(i_l^5, i_r^4) = \frac{2}{7+6-2} = 0.18$
$Sim_S(i_l^5, i_r^4) = 1 - \frac{0+9+0}{7+6+1} = 0.36$

全て閾値未満なので，$subtree(i_l^5)$ と $subtree(i_r^4)$ は類似極大部分木候補とはならない．最終的に類似極大部分木ペアは $(i_l^3, i_r^4), (i_l^4, i_r^2)$ をルートとする部分木である．

3. 実　　装

3.1 XML データの格納方法

提案手法を実装したシステムでは，巨大な XML データを扱うことを想定し，XML データを関係データベース (RDB) のテーブルの形に変換して扱う．XML を直接読み込むと，大量のメモリが必要となるので，巨大な XML データを取り扱うことが困難なためである．テーブルには各ノード毎に，対象ノードの DeweyID [5]，ノード名，ノードの根からの高さ，対応ノー

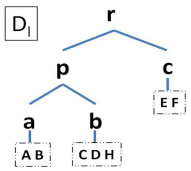
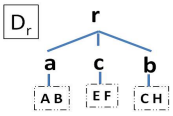


図 11　探索対象となる XML



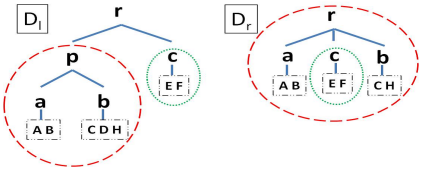


図 12　図 11 の類似極大部分木

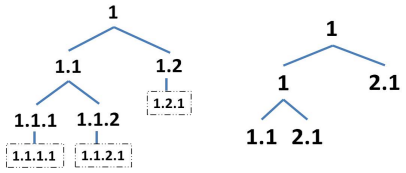


図 13　DeweyID(左) および Patricia 木 (右) の例

ドをルートとする部分木に含まれる全テキストノードの単語集合，対応するノードの子孫ノード数の情報を格納する．

3.2　類似極大部分木候補の格納方法

本システムでは，類似極大部分木候補に関する情報を格納するデータ構造として，DeweyID を基準とした Patricia 木 [6] を用いることで，処理時間の削減を図っている．

DeweyID は，木構造内のあるノードを一意に表す手法である．前置順で何番目の子ノードなのかを整数で表し，それをルートノードから自分のノードまで順に繋いだ配列でノードを表す．また Patricia 木は，文字列を格納するための木構造を持つデータ構造である．共通するオブジェクトの接頭部を根のラベルとし，共通しない接尾部を葉のラベルとすることで，共通部分を持つデータ集合を効率よく扱うことができる．図 13(左) は図 11 の D_l の各ノードに対応する DeweyID を表したもの，図 13(右) は図 13(左) の DeweyID を格納した Patricia 木である．

本システムではマッチングが成立した部分木ペアに関する情報を格納する際，一方の XML の DeweyID を基準とした Patricia 木に格納する．例えばある部分木ペアがマッチする場合，一方の部分木のルートノードに対応する DeweyID を格納している Patricia 木のノードの部分に，部分木ペアに関する情報を入れる．これにより，部分木に含まれるマッチする全子部分木の探索等の効率が向上する．

4.　評 価 実 験

本実験では提案手法の有用性を，ベンチマークデータおよび実データを用いて検証する．ベンチマークデータを使う実験で使用したのは Intel Core 2 Duo(1.2GHz) の CPU と 2GB のメインメモリを持つ WindowsVista マシンで，Java SE 1.6.0_20 で実装した．実データを使う実験で使用したのは Dual Core AMD Opteron(2.4GHz) の CPU と 16GB のメインメモリを持つ SunOS 5.10 マシンで，Java SE 1.5.0_15 で実装した．また RDBMS は PostgreSQL 8.4 を用いた．

4.1　提案手法の規模耐性の検証

本節では，規模耐性の観点から提案手法の有用性を検証する．

4.1.1　類似部分木のサイズとパラメータによる処理時間への影響

類似極大部分木のサイズや数，類似度の閾値による，処理時間が受ける影響を検証する．データの生成には XML 用ベンチマークツール「XMark」(A)(注1)，「XBench」(B)(注2) を利用し，A の特定の部分木を B に挿入する (B')．B に挿入した A の部分木を類似極大部分木とみなし，A と B' に対して類似極大部分木の探索を行う．特に表記がない場合，類似する子部分木の占有率，テキスト，構造類似度の閾値は共に 0.5，木の高さの閾値を 3，構造類似度を考慮する木のノード数の閾値を 3 に設定する．なお構造類似度を考慮する木のノード数が閾値未満の部分木は，その構造類似度が閾値未満でも類似しているとみなす．さらに XML を 3.1 節で説明したとおり RDB に格納し，全テキストノード同士の類似度の計算が完了している前提で実験を行う．

図 14 は，1,000KB の XMark データと，全体のサイズを 1,000KB に固定した上で，挿入する部分木のサイズを 100KB〜500KB の間で変更したデータに対して，類似極大部分木の探索を行ったときの処理時間である．この結果から，類似部分木のサイズが大きいほど，処理時間が長くなることが分かった．

図 15 は，1,000KB の XMark データと，全体のサイズを 1,000KB，挿入する部分木のサイズを 200KB に固定し，その部分木を一分割 (200KB)，二分割 (100KB*2)，四分割 (50KB*4) して挿入したデータに対して，類似極大部分木の探索を行ったときの処理時間である．この結果から，処理時間は主に類似部分木の総サイズに影響を受け，それらがどの程度分割しているかには影響を受けないことが分かった．これはマッチする部分木の合計サイズが同じ上でその部分木の数が増えたとしても，分割によってその部分木ペア一つ当たりのサイズは減るためだと考えられる．

図 16 は，テキスト，構造類似度，類似する子部分木の占有率の閾値をそれぞれ (0.3, 0.3, 0.3), (0.5, 0.5, 0.5), (0.7, 0.7, 0.5), (0.9, 0.9, 0.5) に設定した上で，類似極大部分木の探索を行ったときの処理時間である．なお本実験は，1,000KB の XMark データと，全体のサイズを 1,000KB，挿入する部分木のサイズを 300KB にしたデータを利用している．この結果から，閾値が低いほど，処理時間が長くなることが分かった．これは閾値が低いほど，計算対象となる部分木ペアの数が増えるからだと考えられる．

4.1.2　複数の部分木がマッチする場合の処理時間

2.5 節で説明したとおり，双方の XML 木に含まれる全部分木同士がマッチする場合，提案手法の計算量は最悪となる．この場合の処理時間の検証を行う．

図 17 に示すとおりにデータを作成し，同じデータ同士で類似極大部分木の探索を行った時の処理時間を計測する．このとき木の高さの閾値，構造類似度を考慮する木のノード数の閾値を 0 に設定する．他の条件は 4.1.1 節のものと同じである．

(注1)：http://www.xml-benchmark.org/
(注2)：http://se.uwaterloo.ca/~ddbms/projects/xbench/Publications.html

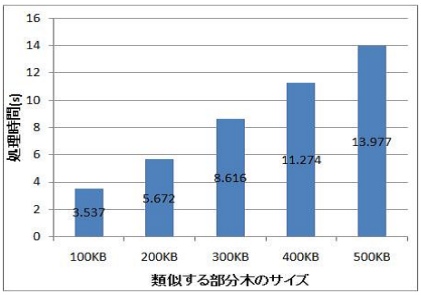


図 14　類似部分木のサイズによる処理時間の違い

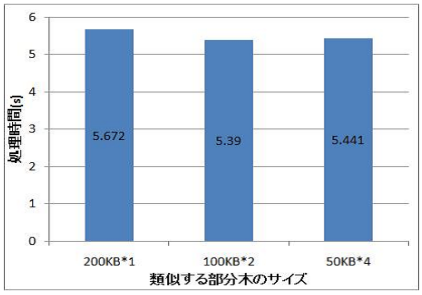


図 15　類似部分木の数による処理時間の違い

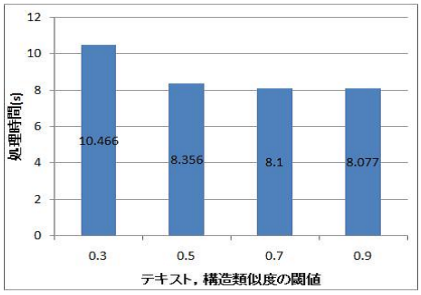


図 16　閾値による処理時間の違い

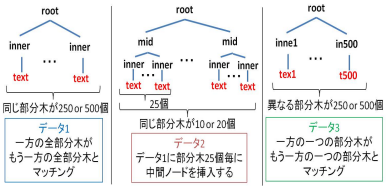


図 17　使用するデータの種類

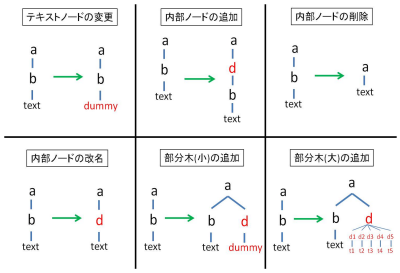


図 18　与えるノイズの種類

図 19　抽出できた類似極大部分木の例

表 2　各データに対する処理時間 (秒)

データの種類/部分木数	250×250	250×500	500×500
データ 1	189.5	1513	3103
データ 2	3.152	7.653	13.87
データ 3	0.062	0.058	0.100

実験結果は表 2 のとおりである．この結果から，一方のある部分木にもう一方の多数の部分木がマッチする場合，処理時間が大幅に増加することが分かった．これは子部分木の選択（割当問題）が，子部分木数の 3 乗に比例する計算量を要するのが原因である．

4.2　提案手法の精度の検証

本節では，精度の観点から提案手法の有用性を検証する．

4.2.1　ノイズのあるデータに対する精度

実際の XML データの場合，双方で同じ内容を表していても，テキストの表記が異なっていたり，内部ノードが追加，削除されていたりと，その構造が異なっていることがある．このような構造の違いをノイズとみなし，提案手法のノイズに対する影響を検証する．200KB の XMark データ A と，データ全体のサイズを 200KB，XBench データに挿入する A の部分木のサイズを 80KB を固定し，その挿入する部分木にノイズを与えたデータ B'_N との類似極大部分木の探索結果を，ノイズを与えなかった場合の結果と比較し，その再現性を算出する．再現性の定義は次のとおりである．

$$\text{再現性} = \frac{\text{ノイズを与えた場合と与えない場合の両方で抽出された部分木ペア数}}{\text{ノイズを与えない場合に抽出された部分木ペア数}}$$

なおノイズは，挿入する部分木に含まれるテキストノードまたは内部ノードの 5〜30 %をランダムに選択して付与する．与えるノイズの種類は図 18 のとおりである．各種閾値や他の条件は 4.1.1 節と同様である．

表 3　各ノイズに対する再現性

ノイズの種類/ノイズの割合	5 %	10 %	20 %	30 %
テキストノードの変更	1	0.95	0.75	0.55
内部ノードの追加	0.75	0.525	0.25	0.075
内部ノードの削除	0.75	0.7	0.325	0.225
内部ノードの変更	1	1	1	1
部分木 (小) の追加	1	1	1	0.975
部分木 (大) の追加	0.825	0.6	0.275	0.125

各ノイズに対する再現性を表 3 に示す．この結果から，内部ノードの追加，削除に影響を受けやすいことが分かった．これは内部ノードが追加，削除されることで，計算対象となる部分木が大きく変化するからだと考えられる．その一方，内部ノードの変更の影響を受けにくいことが分かった．これは大きい部分木の場合，ノード一つ当りの構造類似度への影響が小さく，部分木のマッチング状況があまり変化しないからだと考えられる．また小さい部分木も，構造類似度を考慮する最小ノード数に対する閾値を用意することで，変更の影響を受けにくくなっている．さらに部分木の追加の場合，追加する部分木のサイズによって影響の度合が大きく変化することが分かった．これは部分木のサイズが大きいほど，テキスト，構造類似度への影響が大きくなるからだと考えられる．

表 4 は，再現性の計算式の分子に，子孫部分木が抽出された場合の数も含めた場合の結果である．この結果から，本来抽出されるべき部分木が抽出されない場合でも，中に含まれる子孫部分木は高い水準で抽出できることが分かった．すなわち提案手法は，ノイズを含んだ XML データに対しても，類似する部分を抽出すること自体は十分可能であると言える．

4.2.2　提案手法の類似度と本来の類似度との差異

提案手法のテキスト類似度および構造類似度は，類似度が閾

表 4　各ノイズに対する子孫部分木を考慮した再現性

ノイズの種類/ノイズの割合	5 %	10 %	20 %	30 %
テキストノードの変更	1	0.975	0.825	0.775
内部ノードの追加	0.975	1	0.925	0.975
内部ノードの削除	0.975	1	0.9	0.8
内部ノードの変更	1	1	1	1
部分木 (小) の追加	1	1	1	1
部分木 (大) の追加	1	0.825	0.825	0.75

値を超えない子部分木に関して，本来は類似度が 0 以上であっても 0 とみなす定義である．そのため，全子部分木の本来の類似度を考慮した場合の類似度と異なる場合がある．この類似度を本来の類似度と呼び，提案手法の類似度との違いを検証する．3 組の 50KB の XMark データ $A^{1,2,3}$ と，$A^{1,2,3}$ にノイズとして，テキスト，内部ノードの 30 %に対し，図 18 のテキストノードの変更，内部ノードの削除，部分木 (小) の追加したデータ $B_N^{1,2,3}$ を用意する．そのデータに対し，提案手法を用いた場合と本来の類似度を用いた場合とで，探索結果の違いを比較し，その再現性を算出する．再現性の定義は次のとおりである．

$$\text{再現性} = \frac{\text{提案手法の類似度と本来の類似度が共に閾値を超える極大部分木ペア数}}{\text{本来の類似度が閾値を超える極大部分木ペア数}}$$

テキスト，構造類似度の閾値は 0.3，0.5，0.7，0.9 に変化させる．部分木ペアの類似度が閾値以上ならば，その親部分木同士の類似度を計算する際に，その部分木ペアの計算結果を考慮に入れることになる．また類似する子部分木の占有率の閾値を 0.3，木の高さの閾値を 2，構造類似度を考慮する木のノード数の閾値を 3 で固定する．他の条件は 4. 1. 1 節と同様である．

表 5　各閾値に対する再現性

テキスト，構造類似度の閾値	0.3	0.5	0.7	0.9
再現性	0.424	0.577	0.725	0.889

各閾値に対する再現性を表 5 に示す．なお結果は 3 組のデータの結果をまとめたものである．この結果から提案手法のテキスト類似度および構造類似度は，特に類似度が低い部分ペアに対して本来の類似度より低くなる場合があることが分かった．この理由は前述の通り，類似度が閾値を超えない子部分木の類似度を全て 0 とみなしているからである．特に類似度が低い場合，類似度が閾値を超えない (マッチしていない) 子部分木が多いので，その分差がつきやすい．しかし全子部分木の本来の類似度を考慮すると，計算量が膨大になってしまう．具体的には 2. 5 節に示した提案手法の最悪のケースと同等の計算量となる．そのため類似する子部分木の占有率を計算し，それを類似部分木の定義に加えている．

4. 2. 3　実データに対する精度

より現実に即した精度を検証するため，実在するデータを用いた場合の精度を検証する．1. 節で紹介した XML データ「ACM SIGMOD」,「DBLP Bibliography」に対して提案手法で類似極大部分木の探索を行い，実際にどのようなデータが抽出できるか観察する．また本実験では,「DBLP Bibliography」をルートノード直下から 4 つに分け，並列計算を行っている．「ACM SIGMOD」のノード数は 14,978,「DBLP Bibliography」のノード数は 22,708,715 なので，類似する子部分木の占有率の閾値が 6.5×10^{-4} 未満でないと,「DBLP Bibliography」全体が類似部分木とはなりえないため，分割して並列処理を行うことが可能である．またテキスト類似度を計算する際，可読文字以外の記号は無視し，大文字と小文字の区別をせずに計算する．他の条件は 4. 1. 1 節と同様である．

例えばテキスト，構造類似度の閾値を 0.9，他の閾値を 4. 2. 2 節と同様に設定した場合，抽出できた類似極大部分木ペアの数は 3,954,216 となった．図 19 は，抽出できた類似極大部分木をまとめたものの一例である．同じ種類の四角で囲まれた部分同士がテキスト類似度 1.0，構造類似度 1.0 の類似極大部分木ペアとなっている．

5.　まとめと今後の課題

本研究では二つの XML データに対する，類似極大部分木ペアの探索手法を提案した．兄弟の順番を考慮しない XML データを対象とし，XML のテキストノードに対する類似度と木構造に対する類似度を両方考慮する．また子ノードをルートとする部分木の類似度を先に計算し，その結果を利用して，再帰的に親ノードをルートとする部分木の類似度を計算することで，計算量の削減を図っている．また実験によって，処理時間や精度の面において，提案手法が類似極大部分木の探索に有用であることを確認した．

今後の課題としては，更なる処理時間の削減や Jaccard 係数や木編集距離以外の類似度への対応，内部ノードが追加，削除されたデータや同じ部分木が多数存在するデータに対する対応方法の検討などが挙げられる．

謝辞　本研究の一部は科学研究費補助金特定領域研究 (♯21013004) および若手研究 B(♯21700093) による．

文　献

[1] N. Augsten, D. Barbosa, M. Boehlen, and T. Palpanas, “Tasm: Top-k approximate subtree matching,” International Conference on Data Engineering, pp.353–364, March 2010.

[2] J.T.L. Wang, B.A. Shapiro, D. Shasha, K. Zhang, and K.M. Currey, “An algorithm for finding the largest approximately common substructures of two trees,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.20, pp.889–895, Aug. 1998.

[3] P. Bille, “A survey on tree edit distance and related problems,” Theoretical Computer Science, vol.337, pp.217–239, June 2005.

[4] H.W. Kuhn, “The hungarian method for the assignment problem,” Naval Research Logistics Quarterly, vol.2, pp.83–97, March 1955.

[5] I. Tatarinov, S.D. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang, “Storing and querying ordered xml using a relational database system,” Special Interest Group on Management Of Data, pp.204–215, June 2002.

[6] D.R. Morrison, “Patricia - practical algorithm to retrieve information coded in alphanumeric,” Journal of the ACM, vol.15, pp.514–534, Oct. 1968.