検索キーワードを含む最小XML部分文書の抽出手法

文 連子

A Method for Extracting XML Sub-document Containing Search Keywords

Lianzi Wen

XMLデータは爆発的に増加しており，それに伴いサイズが巨大なXML文書も大量に増加している．サイズの巨大なXML文書の集合からユーザーの問合せにふさわしい情報を検索する際，検索結果として巨大なXML文書リストが返されても，返されたXML文書の中からさらにどの部分が問合せに適合しているのかを調べなくてはならず，それはユーザーにとって非常に煩雑な作業となる．本論文では，XML木構造を考慮せず，従来の検索エンジンと同様，問合せとしてキーワードだけを入力した際，すべてのキーワードを含む部分文書を簡単に抽出する手法を提案する．そして，最後に証明と評価実験を行い提案手法の有効性を検証する．

XML data is explosively increasing, and as a consequence, huge size of XML data also has been emerging. When a user search some information from huge size of XML data set, if the system just returns one huge size of XML data, they need to retrieve most appropriate part again by themselves, and it is very cumbersome. This paper, propose a simple method for extracting XML sub-documents containing all search keywords from an huge size of XML document when user input some search keywords as they do in the usual search engine. We show the effectiveness of the approach by experimental evaluations.

Key Words & Phrases：XML部分文書検索，木構造，キーワード検索
XML sub-document retrieval, Tree structure, Keyword search

1. はじめに

近年，XML［1］データは爆発的に増加し，それに伴いサイズの巨大なXML文書も大量に増加している．巨大なXMLデータの例として，Web上のオンライン百科事典であるWikipediaや，コンピューター科学分野の文献を保持するDigital Bibliography & Library Project(DBLP)［2］などが挙げられる．このような巨大なXML文書集合に対する検索は，ユーザーの問合せに対して1つの巨大なXML文書をそのまま返すと，ユーザーはさらにどの部分が問合せに適合しているかを調べなくてはならず，非常に煩雑な作業となる．また，それとは逆に，検索でヒットしたテキスト・ノード1つ1つが返された場合は，検索結果の粒度が小さく，ユーザーは，各々の検索結果同士の関係を自分で解釈しなくてはならない．そのため,ユーザーは問合せのすべてのキーワードを含むある程度まとまった部分文書を検索結果として返すことを望んでいる．

このXML文書検索の分野ではW3C勧告として公開されたXML問合せ言語XQueryなどの研究が行われている.しかし,これらのXML問合せ言語は,データベースの問合せ言語のSQLと同様,問合せを行うための専門知識や,検索対象のXML文書の構造を利用者があらかじめ把握し,検索時に構造を指定する必要があるため,利便性は決して高いとは言えない.またXQueryなどの問合せ言語を使用するためには,妥当なXML文書（Valid XML Document）※1でなくてはならない.しかしながら,整形式のXML文書（Well-formed XML Document）※2の方がより柔軟に利用できるため,現状,整形式 のXML文書がより多く存在している．さらにXML問合せ言語を利用するためには,XMLデータ形式を保ったままデータベースに格納する必要があるが,XMLデータ・サイズが巨大である場合,一般的に,XMLをいくつかのパートへの分割や,サイズを減らすために別の形式に変換してデータベースに格納するなどの処理を行う.そのため,現在は,サイズが巨大なXML文書に対する検索は,キーワード検索が主流となっている．

そこで本論文では，XML問合せ言語を利用せず，またXML木構造も考慮せず，従来の検索エンジンと同様，問合せとしてキーワードだけを入力した際，すべてのキーワードを

提出日:2012年5月8日　再提出日:2012年9月2日

※1 Valid XML document : 整形式XML文書 *2 としての条件を満たしていることに加えて，DTD，XML Schemaなど文書の論理的構造を規定する規則に準拠しているXML
※2 Well-formed XML document : すべての開始タグと終了タグが対になっているなど，表面的な形式が整えられているXML

含む最小部分文書を簡単に抽出する手法を提案する.

本論文の構成は次の通りである. 第 2 章で, 本提案の前提となる XML 関連技術について説明する. 第 3 章では関連研究について述べ, 第 4 章では提案手法について説明する. 第 5 章では提案手法の有効性について説明し, 第 6 章では評価実験について述べる. 最後に, 第 7 章において, まとめと今後の課題について述べる.

2. 基本的事項

2.1 XML データの木構造

任意の XML データは, 図 1 に示すように木構造としてモデル化可能である. ノードには,根ノード,要素ノード,属性ノード, テキスト・ノードがある. 根ノードは木の根であって, ある木構造に 1 つしかない仮想的なノードである. 要素ノードは各要素に対応し, 要素ノードあるいはテキスト・ノードを子ノードとして持つ. 属性ノードは,属性に対応し,属性名と属性値を持つ. テキスト・ノードは文字列に対応する.

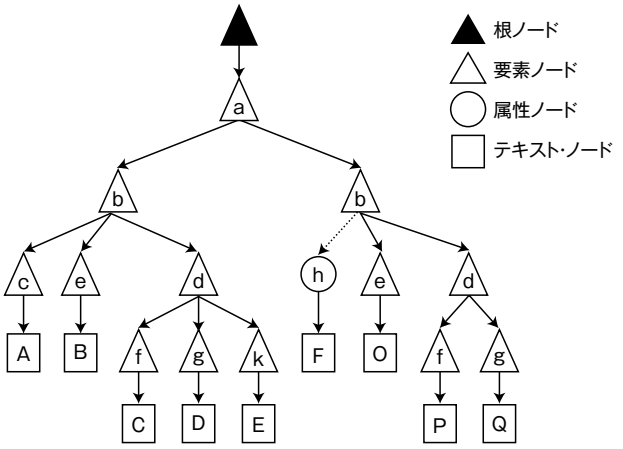


図1. XML木構造

2.2 最小 XML 部分文書

本論文では, 検索キーワードを含むテキスト・ノードの集合の最低共通親（Lowest Common Ancestors : LCAs）[3] [4] を根とする部分木, すなわち, LCA の開始タグと終了タグで囲まれた部分を最小 XML 部分文書と呼ぶ. 例えば, 図 1 において, 検索キーワード B と D で検索を行った際, これらのキーワードを含む 2 つのテキスト・ノードの LCA は, 要素ノード b である. 従って 2 つのキーワードを含む最小 XML 部分文書は <c>A</c><e>B</e><d><f>C</f><g>D</g><k>E</k></d> となる.

2.3 XML のシリアライズ化（Serialization）

本論文での XML シリアライズ化とは, 木構造になっている XML データを, 行きがけ順 (Pre-order)（あるいは帰りがけ順 (Post-order)）により木を巡回しながら, ノードを直線的な一列の文字列に変換することを言う. 例えば , 図 2 において, 最初の XML データを行きがけ順でシリアライズを行い abcd に変換することができる. また帰りがけ順のシリアライズでは bcda に変換できる. 本論文では, 以下シリアライズされた文字列 abcd, bcda を Sequence と呼ぶ.

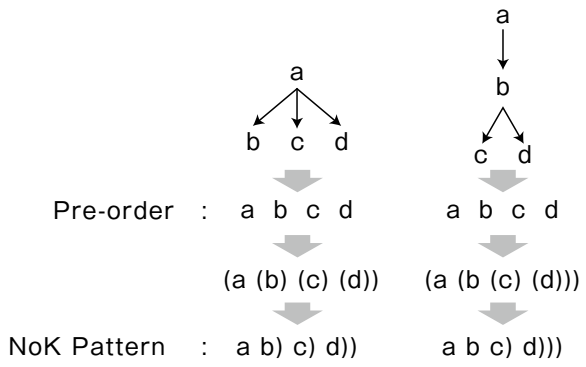


図2. NoK Pattern

XML のシリアライズは, XML データを簡潔にストレージに格納する際や,転送する場合に用いられることが多い.しかし, 一般的に, 単純にノードの要素でシリアライズしていくと, 木構造の情報は失われて行くため, この操作は不可逆である. 例えば, 図 2 の構造の異なる 2 つの XML の行きがけ順でのシリアライズの結果は共に abcd であることが分かる. そのため, 単純にシリアライズを行うだけなら XML の木構造情報が失われる.

従来手法では, これらの問題を解決するため, Ning Zhang 等［5］は NoK Pattern と呼ばれる手法を提案し, 木構造を保持しながらシリアライズを行っていた. 図 2 の最初の XML 木構造は文字列 “(a(b)(c)(d))” と表現できる. 最後の括弧 “)” は, a から始まる部分木の最後を示しており, 部分木の最初を示す開始括弧 “(” と対応する. NoK pattern では,すべての開始括弧を削除し,閉じ括弧のみ残し, “ab)c)d))” のように表示する. それは各々のノードが開始括弧の代わりとみなすことができるため, 開始括弧が削除されても木構造の情報は保持されている.

また, 以前筆者［6］は, 類似する XML データを特定するために, NoK Pattern を利用し, XML データをシリアライズしてから, それらの類似度を計算する研究を行っていた. このように XML データのシリアライズは, XML データの保管, 転送分野ではもちろん, ほかの様々な分野でも行われている.

3. 関連研究

これまでにも, XML データの部分文書を抽出する研究は盛んになされている. 三木等 [7] は,スーパーインポーズ･コードを, 階層構造を持つ XML データに適用し, 最小 XML 部分文書を抽出する手法を提案している. 彼らの研究では, テキスト・ノードの索引を工夫することによって, XML に対

してキーワード検索を行った際，どのようにそのキーワードを含むテキスト・ノードを効率よく特定するかについて焦点を当てている．しかし，それらのテキスト・ノードを含む最小部分木をどのように効率よく特定，抽出するかについては詳しく述べられていない．また彼らの研究では，XML データ構造を考慮した上でのキーワード検索であるのに対して，本論文は XML データの構造を考慮せずシリアライズされた文字列を対象とする点が異なる．

また，波多野等［8］は，キーワードを利用した XML 文書検索システムの実現のため，あらかじめ XML 文書を分割し，それらを検索結果の候補とする XML 文書分割手法を提案した．彼らの研究は，ある粒度決定法に基づき，あらかじめ XML 文書を分割するため，キーワードによる検索結果は，いつも決まった粒度の部分文書が返ってくる．それに対して，本論文では，キーワードによって，柔軟に検索結果の粒度が変わる手法を提案する．

4. 提案手法

提案手法の主な流れを図 3 に示す．

1）XML データをテキスト・ノードと要素ノードに分けてシリアライズを行う．シリアライズ 処理を行うと，XML データはテキスト・ノード / 要素ノードをつなげた 2 つの長い Sequence になる．
2）テキスト・ノードに対してプレイン・テキストのようにキーワード検索を行う (プレイン・テキストに対する検索は既存の検索技術をそのまま利用する)．そして検索されたテキスト・ノードから要素ノードの部分 Sequence を特定する．
3）要素ノードの部分 Sequence からキーワードを含む最小部分木の Sequence を特定する．（部分 Sequence と部分木の Sequence は概念が異なる．部分 Sequence は，長い Sequence の中から抽出した任意のサブ Sequence であるが，部分木の Sequence はその Sequence を木構造に戻した時に木構造として再現できる Sequence のことである）
4）最小部分木の Sequence から部分 XML 文書を再現し，ユーザーに提供する．

以下に提案手法の要素技術について述べる．

① XML データのシリアライズ

本論文では，Ning Zhang などが提案した NoK pattern をさらに発展させ，閉じ括弧の代わりにその閉じ括弧の数を数字として表示することにした．

NoK Pattern : a b) c) d))　　a b c) d)))

Advanced NoK Pattern : a b 1 c 1 d 2　　a b c 1 d 3

このシリアライズの手法を本論文では，Advanced NoK Pattern と呼ぶことにする．Advanced NoK Pattern は，詳細は後述するが，部分木を特定する数学的な計算の際に有用となる．閉じ括弧の代わりにその数を表示する数字を入れても，NoK Pattern と同様にシリアライズされた文字列から一意に木構造に戻すことは可能である．また閉じ括弧を使う時より，閉じ括弧の数だけの文字数がかかるのに対して，その数を数値で置き換えているため格納スペースを節約できる．

本論文で提案する Advanced NoK Pattern では，属性ノードも要素ノードと同様の方法でシリアライズを行っているが，要素ノードと区別するために，ノード名の前に特別な文字@を入れてシリアライズを行う（図 5）．

② Sequence から木構造再現

Advanced NoK Pattern から木構造に再現する方法を図 4 に示す．

Sequence を前から読み，要素ノードが来ると現在の位置から 1 つ下に下げ，その要素ノードを作成する．数字が

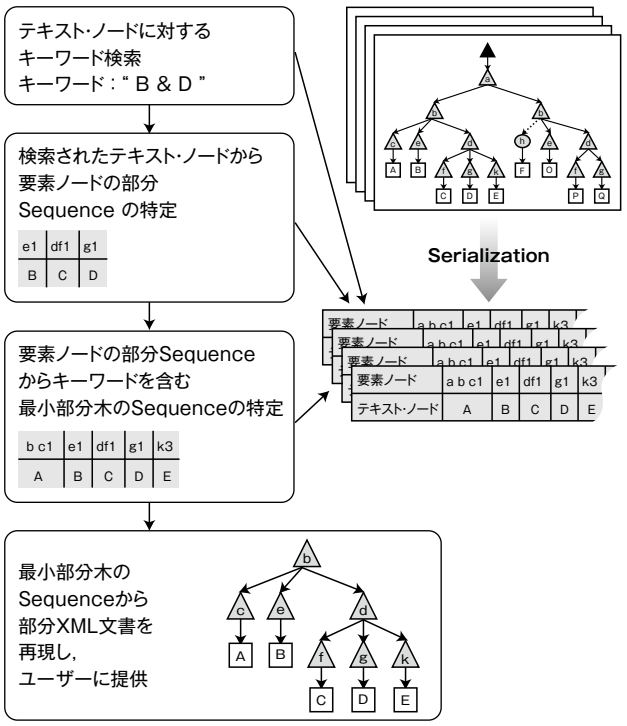


図3. 提案手法の概要

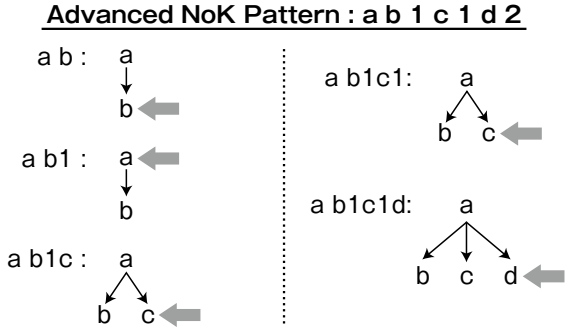


図4. XML木構造の再現の例

来た場合は，現在の位置からその数字の分上のノードに上っていく．

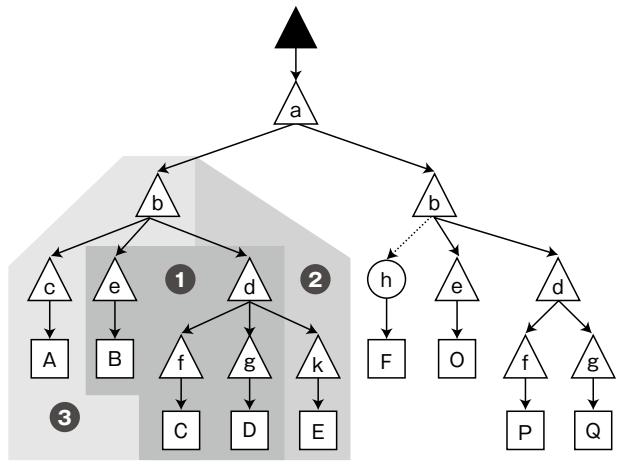
③ テキスト・ノードに対するキーワード検索

テキスト要素に対するキーワード検索は，従来技術を利用し，XML の構造を意識せずに検索を行い，検索されたテキスト・ノードからそれに対応する要素ノードの部分 Sequence を特定する．例えば，図５で，B と D でキーワード検索を行い，それぞれを含むテキスト・ノードが返ってきたとする．求められている部分木は，B が入っているテキスト・ノードのすぐ 1 つ上の要素ノード e を含まなくてはならない．また同じく D の 1 つすぐ上の要素ノード g も含まなくてはならない．そのため，今求めたい部分木は，必ず部分 Sequence “e1df1g”（図５の①）を含まなくてはならない．

④ 部分 Sequence から部分木の特定

部分木の Sequence 特定は以下の 3 つのステップで構成される：

1. テキスト・ノードによって特定された部分 Sequence（図５の①）を基点からの木の深さで数値化し，その途中計算結果の最大値を求める．
2. 部分木の右側の部分の Sequence（図５の②）を求める．
3. 部分木の左側の部分の Sequence（図５の③）を求める．



	3		1		2				
要素ノード	a b c1	e1	df1	g1	k3	b@h1	e1	df1	g4
テキスト･ノード	A	B	C	D	E	F	O	P	Q

図5. XMLデータとそのシリアライズの例

部分 Sequence を左から右に向かって足し算を行う．ここで，ノードの場合は，-1 を代入する．また，途中計算結果の最大値を覚えていく．　例えば，“e1df1g” では足し算の結果は -2 となり，途中の計算結果の最大値は 0 である．

	e	*1*	*d*	*f*	*1*	*g*	
	(-1)	*+ 1 +*	*(-1)*	*+ (-1)*	*+ 1 +*	*(-1)*	*= -2*
途中計算結果：	*-1*	*0*	*-1*	*-2*	*-1*	*-2*	

“e1df1g” までの最大値の値を求めた後，続けて Sequence の右に向かって，最大値より 1 つ大きい値になるまで足し算をしていく．e1df1g1k3 = 1（最大値 0 より 1 つ大きい）となるため，部分木の右側の Sequence（図５の②）を求められる（詳細は次節で述べる）．

次は部分木の左側の Sequence（図５の③）を求める．今回は，部分 Sequence “e1df1g” の e の前の文字列から左に向かって，上で求めた右側 Sequence の合計（e1df1g1k3 = 1）の反数となる -1 になるまで足し算する．bc1 = -1 になるので，“bc1” が左側の Sequence となる．そして 2 つの部分 Sequence をつなげた “bc1e1df1g1k3” は B と D を含む最小部分木の Sequence になる．そして求められた Sequence を木構造の XML データに再現してユーザーに提示する．

部分 Sequence から部分木の Sequence を抽出するアルゴリズムは，図６の Algorithm 1 で説明している．

3 つ以上のキーワード，テキスト・ノードを含む最小部分木を特定する場合も同様に上記の手法を適用できる．その際は，両側のテキスト・ノードからそれに対応する部分 Sequence を特定し，同じように部分木の Sequence を特定できる．

Algorithm 1. GetSubTreeSequence(*S*, *Tstart* , *Tend*)

```
Input: S, Tstart , Tend {Sequence, the start index of
       subsequence, the end index of subsequence}
Output: ss (Smallest sub tree Sequence).
Tmax= -10 ; Tsum = 0 ; ss = " " ;
for i = Tstart to Tend do
   if (S [i] == ELEMENT NODE) then
      Tsum += -1;
   else
      Tsum += Integer.parseInt(S [i] );
   if (Tmax< Tsum) then
      Tmax= Tsum;
   ss += S [i] ;
end for
for i = Tend +1 to Tsum == Tmax +1 do
   if (S [i] == ELEMENT NODE) then
      Tsum += -1;
   else
      Tsum += Integer.parseInt(S [i] );
   ss += S [i]
end for
Tsum2 = 0 ;
for i = Tstart -1 to Tsum2 == -(Tsum) do
   if (S [i] == ELEMENT NODE) then
      Tsum2 += -1;
   else
      Tsum2 += Integer.parseInt(S [i] );
   ss = S [i] + ss ;
end for
return ss ;
```

図6. 部分木のSequenceの抽出アルゴリズム

5. 提案手法の有効性

この章では，なぜ提案手法で，部分木の Sequence を特定できるかについて説明する．

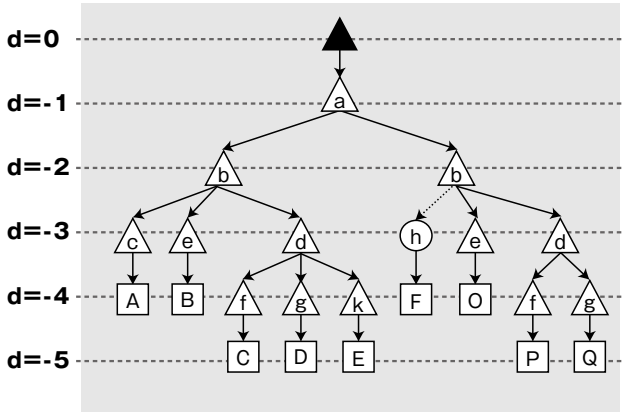


図7. XMLデータの深さ表現

XML データを木の深さで表現すると図 7 のようになる．

Advanced NoK Pattern でノードは探索経路が 1 つ深くなることを示し，数字は探索経路がその分浅くなることを示す．木の探索は，根ノードからその下のすべてのノード探索を終え，再び根ノードに戻って深さゼロになった際に終了となる．そのため，Advanced NoK Pattern の各々の要素を足していくと，その値は，その探索地点での深さを表している．また，最後まで足し算を行うと，深さゼロの場所に戻ってくるため，その計算結果はゼロになる．

式 1 での $\mathcal{N}$ は，Sequence においての各々の要素ノード，数字を指す．

式 1: $\mathcal{N}1 + \mathcal{N}2 + \mathcal{N}3 \ldots + \mathcal{N} = 0$

（$\mathcal{N}$ が要素ノードの時は -1 を代入，$\mathcal{N}$ が数字の場合はその数字を代入）

例：a+b+c+1+e+1+d+f+1+g+1+k+3
+b+@h+1+e+1+d+f+1+g+4 = 0（図 5）

部分木は，木構造の一部であり，それ自身も完全な木構造となっているため，上記のことは，部分木についても同様のことが言える．つまり長い Sequence からある部分木の Sequence を取り出して足し算すると必ずその結果はゼロになる．

式 2: $\mathcal{N}x+\ldots+\mathcal{N}m\text{-}1+\mathcal{N}m+\ldots+\mathcal{N}n+\mathcal{N}n+1+\ldots+\mathcal{N}y = 0$

例：b+c+1+e+1+d+f+1+g+1+k+3 = 0（図 5）

上の式 2 で，$\mathcal{N}m+\ldots+\mathcal{N}n$ はテキスト・ノード検索から特定した部分 Sequence とする．部分 Sequence の足し算の途中結果の最大値（$\mathcal{T}max$）は，木構造でいうと一番上のノードの深さである．部分 Sequence の右側に向かって，その最大値より 1 つ大きい数字になるまで足し算して行くこと（$\mathcal{N}m+\ldots+\mathcal{N}n + \mathcal{N}n+1+\ldots+\mathcal{N}y = \mathcal{T}max+1$）は，部分 Sequence $\mathcal{N}m+\ldots+\mathcal{N}n$ を含む親ノードの場所 $\mathcal{N}y$ を捜すことに対応する．

式 3: $\mathcal{N}m+\ldots+\mathcal{N}n+\mathcal{N}n+1+\ldots+\mathcal{N}y = \mathcal{T}max+1$

例：e+1+d+f+1+g+1+k+3 = 1（図 5）

式 2 より，部分木の左側の Sequence $\mathcal{N}x+\ldots+\mathcal{N}m\text{-}1$ に対しては，

式 4: $\mathcal{N}x+\ldots+\mathcal{N}m\text{-}1 = -(\mathcal{N}m+\ldots+\mathcal{N}n+\mathcal{N}n+1+\ldots+\mathcal{N}y)$

が成立する．

以上,この章では数学的な式を用いて提案手法がいつでも，どの XML についても成り立っていることを説明した．

6. 評価実験

提案手法の有用性，有効性を証明するため，Intel Core 2 Quad CPU, 2.67GHz, 8G メモリーの Windows 7（64bit）上で，JAVA 1.6 を使って実装を行い，評価実験を行った．

実験データとして , SIGMODRecord.xml［9］，DBLP.xml を用いた．まずこれらの XML データをシリアライズし，検索を高速にするために Lucene［10］が提供している API を使って，テキスト・ノード，要素ノードの Sequence に対して索引を作成した．各 XML データ・サイズとそのシリアライズに掛かった時間（処理時間①），要素ノードとテキスト・ノードの索引時間（処理時間②）は，表 1 の通りである．

XML データのシリアライズと索引は，ユーザーが XML データの格納時に行うので，時間が少し掛かっても利便性が損なわれることはない．

表1. XMLデータサイズと処理時間

XML データ	サイズ (KB)	処理時間① (sec)	処理時間② (sec)
DBLP.xml	1,088,968	1277.0	622.8
SIGMODRecord.xml	507	0.6	2.0

キーワード検索時，キーワードを含むテキスト・ノード同士の最大距離（シリアライズされた Sequence で，テキスト・ノードが互いに離れている距離）が大きいと，その 2 つのテキスト・ノードの LCA は根ノードになり，検索結果はその XML 全体文書となるため最小部分文書検索として意味がない．そのため，本実験ではテキスト・ノード同士の最大距離を設定し，このパラメーター以内に 2 つのテキスト・ノードが存在する場合のみ，最小部分文書を抽出するようにした．

```
<dblp>
   ・・・      ・・・      ・・・      ・・・
 <article mdate="2011-12-29" key="tr/trier/MI06-04"
                  publtype="informal publication">          Part 1
   <author>Sebastian Maneth</author>
   <author>Thomas Perst</author>
   <author>Helmut Seidl</author>
   <title>Exact XML Type Checking in Polynomial Time.</title>
   <journal>Universität Trier, Mathematik/Informatik,
                        Forschungsbericht</journal>
   <volume>06-04</volume>
   <year>2006</year>
 </article>
   ・・・      ・・・      ・・・      ・・・
 <phdthesis mdate="2012-04-18" key="phd/de/Weigel2006a">
   <author>Felix Weigel</author>
   <title>Structural summaries as a core technology for efficient
                        XML retrieval.</title>               Part 2
   <year>2006</year>
   <school>Ludwig Maximilians University Munich</school>
  <ee>http://edoc.ub.uni-muenchen.de/archive/00006259/</ee>
  <url>http://edoc.ub.uni-muenchen.de/archive/00006259/</url>
  <note type="urn">urn:nbn:de:bvb:19-62594</note>
  <note type="dnb">http://d-nb.info/982638604</note>
 </phdthesis>
   ・・・      ・・・      ・・・      ・・・
 <inproceedings mdate="2012-04-30" key="conf/hoti/MorrisTL09">
   <author>Gareth W. Morris</author>                         Part 3
   <author>David B. Thomas</author>
   <author>Wayne Luk</author>
   <title>FPGA Accelerated Low-Latency Market Data
                        Feed Processing.</title>
   <pages>83-89</pages>
   <year>2009</year>
   <booktitle>Hot Interconnects</booktitle>
   <ee>http://doi.ieeecomputersociety.org/10.1109/HOTI.2009.17
          </ee>
   <crossref>conf/hoti/2009</crossref>
   <url>db/conf/hoti/hoti2009.html#MorrisTL09</url>
 </inproceedings>
   ・・・      ・・・      ・・・      ・・・
</dblp>
```

図8. DBLP.xmlの例

DBLP.xml は，1 つの親要素ノード dbpl の下に多数の要素ノード article, book, mastersthesis, phdthesis, inproceedings などがある（図 8）．DBLP.xml では，これらの要素ノードを親とする部分文書を内容的にまとまった塊と言える．これらの部分文書で最初のテキスト・ノードから最後のテキスト・ノードまでの距離より少し大きい数字をパラメーターとする．例えば，図 8 で，要素ノード inproceedings を親とする部分文書で，最初のテキスト・ノード Gareth W. Morris から最後のテキスト・ノード db/conf/hoti/hoti2009.html#MorrisTL09 までの距離が 9 であるので，本実験では，10 を最大距離のパラメーターとして設定した．そしてキーワード "XML" と "Thomas" で検索した際，テキスト・ノードの同士の距離が 2 である Part1 のみを検索結果として返し，Part2 と Part3 にあるキーワードは，最大距離のパラメーターを越えているので，検索結果から外した．また SIGMODRecord.xml でも，内容的にまとまった部分文書の

```
<article>
   <title articleCode="182162">Performance
       enhancement through replication in an object-
       oriented DBMS</title>
   <authors>
      <author AuthorPosition="01">Eugene J.
          Shekita</author>
      <author AuthorPosition="02">Michael J.
          Carey</author>
   </authors>
</article>
<article>
    <title articleCode="192203">A performance evaluation
        of pointer-based joins</title>
   <authors>
      <author AuthorPosition="01">Eugene J.
          Shekita</author>
      <author AuthorPosition="02">Michael J.
          Carey</author>
    </authors>
</article>
```

図9. SIGMODRecord.xmlの例

テキスト・ノードの最大距離を確認し，同じくパラメーターを 10 とした．

テキスト・ノード同士の最大距離のパラメーターは，使用場面とその XML データ・タイプによってユーザーが設定できる．例えば，階層構造が深く，横にも多く延びている大きい XML 文書で，ユーザーが大きい部分文書も検索結果として必要とする時は，パラメーターを大きく設定し，また階層構造が浅く，小さい XML 文書に対し，小さい部分文書のみを求めるときは小さく設定を行うなどの調整が考えられる．

SIGMODRecord.xml に対してキーワード “Parallel

表2. キーワード検索時間

キーワード	処理時間 ① (msec)	処理時間 ② (msec)
“Parallel database”, “Michael J.Carey”	26	10
“Michael J. Carey”, “Eugene J. Shekita”, “Performance”	34	11
“Mandreoli”, “Wedemeijer”	27	11

SIGMODRecord.xmlに対する検索

キーワード	処理時間 ① (msec)	処理時間 ② (msec)
“Thomas Melia”, “DESPRIT”	319	12
“Robert”, “Al Davis”, “Nanophotonic”	296	11
“hypothesis”, “Data Analysis”	267	11

DBLP.xmlに対する検索

database”と“Michael J. Carey”で検索を行うと,Michael J. Carey が著者であり，タイトルが Parallel database を含むに論文を紹介する部分文書が，検索結果として返ってきた．また，キーワード “Michael J. Carey”，“Eugene J. Shekita” と “Performance” で検索した結果 Michael J. Carey と Eugene J.

Shekita が書いた Performance 関連の文献を紹介している部分が 2 つ検索された（図 9）．上で述べたキーワード以外にも表 2 で表示しているキーワードで検索してみた結果，ユーザーが望むような適切な部分文書が返ってくるのを確認した．また表 2 にテキスト・ノード Sequence に対するキーワード検索の処理時間（処理時間①），テキスト・ノードの検索結果から最小 XML 部分文書の Sequence を特定する処理時間（処理時間②）を表示した．表 2 より木構造の XML データにも関わらず，またデータ・サイズにも関係なく，最小 XML 部分文書を特定するにあまり時間が掛からないことが分かった．

7. おわりに

本論文では，XML 構造を意識せず，NoK パターンを発展させた方法で XML のシリアライズを行い，簡潔にすべてのキーワードを含む部分文書を特定する手法を提案した．また証明と評価実験を行い提案手法の有効性，有用性を示した．

この手法は，XML データを格納するための特別なデータベース，XML 特有のストア方法など，あらかじめ必要となるものはなく，妥当型，整形型に関わらず対応できる．また，テキスト・ノードに関するキーワード検索は，既存のプレイン・テキスト検索技術を利用できるため，木構造である XML 部分文書検索にも関わらず，簡単に実装できる．そのため XML データを取り扱う様々なアプリケーションの開発に簡単に取り入れることができるため，非常に有用と言える．

今後の課題としては以下のことが挙げられる．

第一にテキスト・ノード同士の最大距離のパラメーターの自動設定である．本評価実験では，ユーザーがデータ・タイプをみて，そのパラメーターを設定することを前提としているが，自動的に適切なパラメーターを設定する方法を加えるとより利便性が向上できる．

第二に評価実験の充実である．今回の評価実験では，2 つの XML データのみを使用していたが，より大規模かつ多様なタイプの XML データを基に評価実験を行う必要がある．

謝辞

本論文の執筆にあたり，多くの方にご協力していただき，心から感謝しております．特に，有益なご指摘を多くいただいた井上基晴さんと，日本語表現の修正にご協力いただいた同期の望月朝香さんに最大の感謝を申し上げます．

参考文献

［1］ W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition) W3C Recommendation 26 November 2008, http://www.w3.org/TR/REC-xml/
［2］ DBLP, http://dblp.uni-trier.de/xml/
［3］ Yunyao Li, Cong Yu, and H. V. Jagadish：Schema-Free xQuery, In VLDB, pp.72-83 (2004).
［4］ Yu Xu and Yannis Papakonstantinou：Efficient keyword search for smallest lcas in xml databases, In SIGMOD Conference, pp.537-538 (2005).
［5］ Ning Zhang, Varun Kacholia, and M. Tamer Ozsu：A succinct physical storage scheme for efficient evaluation of path queries in xml, In Proc. ICDE 2004, pp.54 (2004).
［6］ Lianzi Wen, Toshiyuki Amagasa and Hiroyuki Kitagawa：An Approach for XML Similarity Join using Tree Serialization, In Proc. 13th International Conference on Database Systems for Advanced Applications, (DASFAA 2008), LNCS 4947, pp.562-570, New Delhi, India, March 19-21 2008.
［7］ 三木 健士，横田 治夫：検索キーワードを含む最小 XML 部分文書抽出のための索引手法，DEWS2007，http://www.ieice.org/~de/DEWS/DEWS2007/pdf/c1-4.pdf
［8］ 波多野 賢治，絹谷 弘子，吉川 正俊，植村 俊亮：キーワードを利用した XML 文書検索ための検索結果粒度決定法，日本データベース学会 Letters. Vol2, http://www.dbsj.org/journal/vol2/no1/papers/hotano.pdf
［9］ ACM Special Interest Group on Management of Data : SigmodRecord.xml, http://www.sigmod.org/publications/sigmod-record/xml/SigmodRecord.xml/view
［10］ Lucene:Lucene Java Documentation, http://lucene.apache.org/core/old_versioned_docs/versions/3_0_2/index.html

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
ソフトウェア・エンジニア

文 連子 Lianzi Wen

［プロフィール］
2008 年，日本 IBM 入社．ソフトウェア開発研究所にて，e ディスカバリー製品の品質保証に従事．2010 年より e ディスカバリー製品の開発に従事．ドキュメントプロセッシング部分を担当．
WENL@jp.ibm.com