XML 文書検索のための類似度計算の効率化手法

森　康弘† 吉川　正俊†† 波多野賢治†††

† 名古屋大学 情報科学研究科
†† 名古屋大学 情報連携基盤センター
††† 奈良先端科学技術大学院大学 情報科学研究科
E-mail: †mori@dl.itc.nagoya-u.ac.jp, ††yosikawa@itc.nagoya-u.ac.jp, †††hatano@is.naist.ac.jp

あらまし　インターネット技術の発展に伴って XML の利用範囲は急速に拡大し, XML 文書とその一部分を検索することが可能なサーチエンジンへの関心が高まってきている. 特に, 利用者にとっての使いやすさから，キーワードを問合せとする XML サーチエンジンへの期待は大きい. しかし, 現在まで開発されてきた XML サーチエンジンは文書中に存在する膨大な数の要素ノードに対して類似度を逐次計算しているため, 検索に時間がかかるという問題がある. 本稿では, 検索結果の上位 k 件を高速に検索する手法を提案する. 本手法では XML 文書中のノード間における類似度の関係と文書構造から, 類似度計算を省略するしきい値を計算できる. このしきい値を利用して類似度を計算する要素の数を削減できるので, 実用的な検索速度の達成を期待できる. 実際に, 我々が開発している XML データベース XRel を使って本手法を実装し, その評価実験の結果について報告する.

キーワード　XML，情報検索，性能評価, top-k Query

An Efficient Retrieval Method for XML Search Engine

Yasuhiro MORI†, Masatoshi YOSHIKAWA††, and Kenji HATANO†††

† Graduate School of Information Science, Nagoya University
†† Information Technology Center, Nagoya University
††† Graduate School of Information Science, Nara Institute of Science and Technology
E-mail: †mori@dl.itc.nagoya-u.ac.jp, ††yosikawa@itc.nagoya-u.ac.jp, †††hatano@is.naist.ac.jp

Abstract　As Internet technologies develop, an XML search engine has attracted much attention. Especially, a keyword-based XML seach engine is expected by a naive user for its simple use. However, current XML search engines calculate similarities of a huge number of XML elements one by one; hence it takes very long time to retrieve results related to a keyword-based query. In this paper, we propose a top-k query method for speeding up the score calculation of XML search engines. By reflecting relationships of elements' similarities and an XML document structure, our method can compute the threshold for stopping score calculations, and make use of it to improve performance of XML search engines. We have implemented our method on our XML search engine, and examined both the query processing time and the number of elements with their similarities calculated to verify the effectiveness of our method

Key words　XML, Information Retrieval, Performance Evaluation, top-k Query

1. 背　　景

W3C(World Wide Web Consortium) から仕様が勧告された XML(Extensible Markup Language) [4] は，インターネット上の文書やデータを表現するメタ言語であり，SGML(Standard Generalized Markup Language) の設計思想を受け継ぐ形で設計された．XML 形式の文書の記述は，文書の可読性を上げたり，目的に応じたデータ構造の表現ができるため，近年になってさまざまな用途で使用され始めている．以上のような背景から XML で記述された文書が多くなることが予想されるため，XML サーチエンジンに対する性能向上への期待は大きい．

XML 文書を検索するための手法の中で主流となっているのが, XPath (XML Path Language) [5] や XQuery [3] のような XML 問合せ言語を用いた方法である. 実際，市販のデータベースシステムの検索機能に盛り込まれたり，盛んに研究が行われている．しかし，これらの XML 問合せ言語では検索したい XML 文書の文書構造をあらかじめ把握する必要があるので，利用者にとって使いやすいとはいえない. そこで，検索対象となる文

```
<article>
  <ti> XML Databases  </ti>
  <sec>
    <st> Storing XML documents  </st>
    <p>  The development of technologies… </p>
    <p>  In this section, we present several…</p>
    <ss>
      <st>  Query Languages </st>
      <p>   Proposal of query languages… </p>
      <p>   W3C has organized the XML… </p>
    </ss>
    <ss  no ="145">
      <st> Indices</st>
      <p>  Although the W3C XML … </p>
      <p>  Indices for XML should … </p>
    </ss>
  </sec>
</article>
```

図 1　XML 文書の例

Fig. 1　An XML Document Instance.

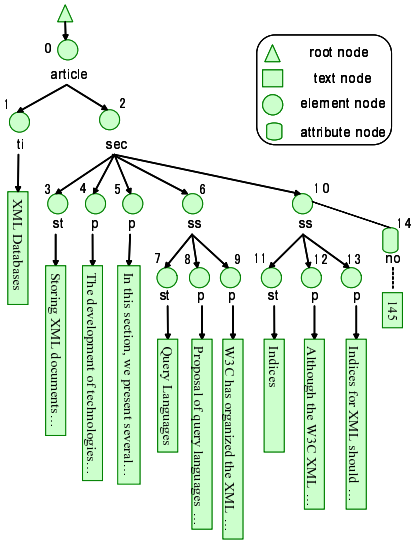

図 2　XML 文書の木構造

Fig. 2　A Tree Representation of an XML Document.

書構造や XML 問合せ言語の知識を必要とせずに，単純にキーワードを入力するだけの XML サーチエンジンが研究されてきた [7], [12].

しかし，これらの研究では，検索対象となる XML 文書の文書型定義 (DTD) や XML スキーマの存在を前提とするため，検索対象となる XML 文書の文書構造は限られる．また，すべての XML 文書ノードの類似度を計算するため，問合せ処理に非常に時間がかかるという問題点がある．特にネットワーク上にある膨大でさまざまな構造をもつ XML 文書を検索対象とするためには，実用的な速度で文書の構造に依存しないサーチエンジンが必要である．

我々は問合せキーワード入力型のインタフェースを備え，DTD や XML スキーマの存在を前提としない XML サーチエンジンを開発してきた [9]. このシステムでは，利用者がキーワードを入力するだけで問合せに相応した XML 文書の要素ノードを検索でき，問合せに対するふさわしさをもとにランキング化して利用者に提示する．さらに，統計量や文書の構造情報を利用して検索対象となる要素ノード数を削減し，検索速度の向上を実現している．しかし，それぞれの要素ノードに対して類似度を逐次計算するので，検索効率が悪く実用的な検索速度に至っていない．

そこで本稿では，キーワードを問合せとする XML サーチエンジンを高速化するため，検索結果となる任意の k 個の要素ノードを効率的に検索する top-k query 手法を提案する．Web 文書検索システムの多くの利用者はたいてい上位 10 件程度の検索結果しか見ないという調査 [2] があることから，近年 top-k query に関する研究が注目されてきている [8], [14]. しかし，XML 文書を対象とした top-k query の従来研究は，著者の知る限り特定の文書スキーマの存在を前提としている [10]. 本手法では，XML 文書の木構造とその中のノード間の類似度の関係から算出されるしきい値を使うため，特定のスキーマの存在を前提としない．このしきい値を使うことによって，余分な要素ノードの類似度計算をできる限り省略することを考える．さらに，本手法の有効性を実証するための評価実験について報告する．

2.　XML サーチエンジン

本研究は，XPath 1.0 [5] で定義されているデータモデルに基づいているため，本稿では XPath データモデルに準拠した用語を使用して議論を進める．

XPath データモデルは，XML 文書を木構造で表現できる．図 1 の XML 文書を木構造で表現した例を図 2 で示す．木構造の節点には，各ノードを表す文字が現れる順序に従って document order が割り当てられており，主に要素ノード，属性ノード，テキストノードの 3 種類のノードに区別される．

XPath データモデルに基づいた XML 文書のための検索モデルとして提案されたものに non-overlapping [6] リストモデルと proximal node [11] モデルがある．本稿で提案する検索モデルは，proximal node モデルに近い．よって，検索単位は要素ノードであり，図 2 の各接点に付けられた ID n を利用して **#n** と呼ぶ．

2.1　XML サーチエンジンの構成

我々が提案する XML サーチエンジンは図 3 のように，XML 文書から要素ノードを抽出する部分，要素ノードごとに特徴ベクトルを計算して索引ファイルをデータベースに構築する部分，要素ノードの統計量を分析する部分，要素ノードと問合せキーワードとの類似度を計算する部分，類似度をもとにランキング付きの k 個の要素ノードを表示する部分に分割される．

（1）　**XML 文書から要素ノードの抽出**

まず，XML 文書を XML プロセッサを用いて DOM 木を構築し，構築された DOM 木から要素ノードを抽出する．

（2）　**各要素ノードの索引の作成**

抽出された要素ノードに対して，不要語処理，接辞処理などの前処理を行った後，含まれている索引語数を数える．ここでの索引語とは，文書中に出現している単語に対して不要語処理，接辞処理などの前処理を行った後の文字列を指す．索引語の重み付け方法には，文献 [13] で用いた *tf-idf* 法を応用した手法を利用する．要素ノードの特徴ベクトルを作るために，まず XML 文書 D_i のそれぞれの要素ノード $e_{ij}(j = 1, 2 \ldots n_i)$ ごとに単語

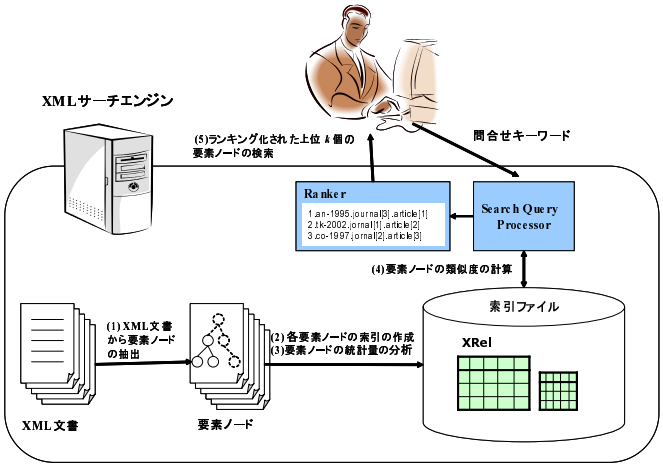

図 3　XML サーチエンジンの概略図

Fig. 3　Our XML Search Engine for XML Documents.

$t_k (k = 1, 2, \ldots, m)$ を抽出し，それぞれの出現頻度 $tf(e_{ij}, t_k)$ を求める．N_d を，すべての XML 文書の数とする．また，t_k が存在する XML 文書数 $idf(t_k)$ も求めると，e_{ij} の特徴ベクトル $F(e_{ij})$ は以下のように表現される．

$$F(e_{ij}) = (w_{t_1}^{e_{ij}}, w_{t_2}^{e_{ij}} \ldots w_{t_m}^{e_{ij}}) \tag{1}$$

$$w_{t_k}^{e_{ij}} = tf(e_{ij}, t_k) \cdot \log \frac{N_d}{idf(t_k)} \tag{2}$$

計算された特徴ベクトルから要素ノードごとに索引ファイルを構築する際，我々が開発している XML データベースシステムである XRel [15] を使用する．

（3） 要素ノードの統計量の分析

キーワードを問合せとする XML サーチエンジンの検索結果となりえない要素ノードを索引ファイルから取り除くため，波多野ら [9] の研究を参考にして要素ノードの統計量を用いた．本稿では，検索結果となりえない要素ノードを stop context と呼ぶことにする．

（4） 要素ノードの類似度の計算

問合せベクトルは，システム利用者が入力した単語が出現しているか否かで決定される．

$$q = (q_{t_1}, q_{t_2} \ldots q_{t_m}) \tag{3}$$

ただし，$q_{t_k} (k = 1, 2, \ldots, m)$ は，

$$q_{t_k} = \begin{cases} 1 & \text{問合せに } t_k \text{ が含まれている場合} \\ 0 & \text{問合せに } t_k \text{ が含まれていない場合} \end{cases} \tag{4}$$

要素ノード e_{ij} の特徴ベクトル $F(e_{ij})$ と問合せベクトル q 間の類似度は，ベクトル空間モデルに基づくと，両ベクトルの内積を e_{ij} における索引語の総出現回数 $tf(e_{ij}) = \sum_{k=1}^{m} tf(e_{ij}, t_k)$ で割った値で表現される．

$$sim(F(e_{ij}), q) = \frac{F(e_{ij}) \bullet q}{tf(e_{ij})} \tag{5}$$

（5） ランキング化された上位 k 個の要素ノードの表示

$sim(F(e_{ij}), q)$ の大きい要素ノードの順に，任意の特定の順位 (k 位) までの要素ノードをランキング化して表示する．

```
top-k has initialized
min-k :=0
  for scan candidates_i in lists of candidates
    threshold: = the best possible score of the rest of candidates
    min-k := the worst scores among current top-k results
    if threshold ≦ min-k
      then halt all score calculations
           in lists of candidates
    endif;
  endfor;
```

図 4　しきい値アルゴリズム

Fig. 4　The Threshold Algorithm.

3.　アプローチ

現在の XML サーチエンジンは膨大な文書集合からその構成要素である全部の要素ノードの類似度を逐次計算しているため，検索時間がかかり，実用的な検索速度に至っていない．そこで，任意の k 個の検索結果にならない要素ノードの類似度計算をできる限り省略する top-k query と呼ばれるアプローチをとる．次節では，top-k query を使うためのしきい値アルゴリズムを説明する．

3.1　しきい値アルゴリズム

最初に，ある問合せ q に対する上位 k 個の要素ノードが存在する状況を想定する．しきい値アルゴリズムでは，k 個の要素ノードの検索結果における類似度の最小値，すなわち k 番目の要素ノードの類似度を，*min-k* と定義する．さらに，まだ類似度が計算されていない要素ノードがとりうる類似度の最大値を *threshold* と定義する．いいかえれば，*threshold* は，類似度が計算されていない要素ノードの類似度計算を省略するか否かのしきい値といえる．すなわち，*threshold* と *min-k* を比較して *threshold* の方が大きければ，まだ類似度が計算されていない要素ノードのいずれかが検索結果となる可能性があるため，実際に類似度を計算して確かめる必要がある．しかし，*threshold* の方が小さければ，まだ類似度が計算されていない要素ノードが検索結果に入る可能性がないので，計算を省略できる．

本稿では，XML 文書の木構造とそのノード間における類似度の関係を利用して類似度が未計算のノードの類似度がとりうる最大値を算出し，それを類似度計算を省略するための *threshold* と設定する．

4.　構成ノード集合

本手法では，しきい値アルゴリズムで必要な *threshold* を求める際に，XML 文書の木構造によって規定される文書長の関係に注目する．ただし，XML サーチエンジンの中で，文書は索引化の際の前処理によって索引語の集合として扱っているため，文書長という概念を索引語数として扱う．索引語数に関する関係は特定の文書スキーマに依存しない．次節では索引語数に関する関係を扱うため「構成ノード集合」という概念を定義する．

4.1　構成ノード集合

類似度が計算済みのノードから計算されていない残りのノードの類似度の最大値を計算するため，以下のようなノード集合を定義する．

定義 1 (構成ノード集合)

ある *XML* 文書 D 中の識別子 m のノード D^m と，その子

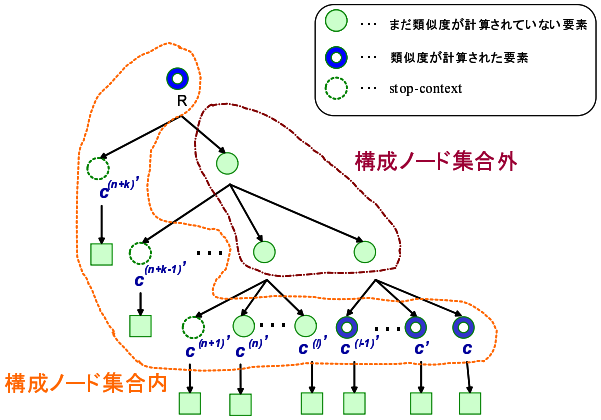

図 5　構成ノード集合
Fig. 5　Composed Node Set.

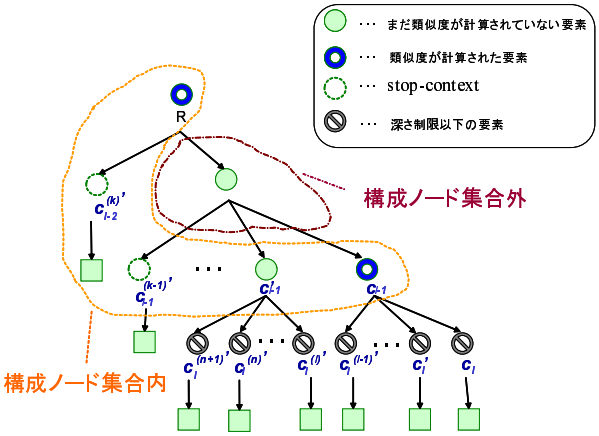

図 6　深さ制限のある構成ノード集合
Fig. 6　Composed Node Set Reflecting a Depth.

孫関係にあるノード集合 $c, c', \ldots, c^{(n+k)\prime}$ との間に $tf(D^m) = tf(c) + tf(c') + \cdots + tf(c^{(n+k)\prime})$ なる関係を持つノード集合を，**構成ノード集合**と呼ぶ．□

4.1.1　構成ノード集合の例

本稿で使用する構成ノード集合は，以下の三種類である．

- 親ノード P とその子ノード $c, c', \ldots, c^{(n+k)\prime}$.

XML の木構造にあるノードの中で，上位ノードは複数の下位ノードから構成される．特に，親ノードは子ノードが複数集まって構成される最小の単位ノードであり，$tf(P) = tf(c) + tf(c') + \cdots + tf(c^{(n+k)\prime})$. が成り立つ．

- 根ノード R と葉ノード $c, c', \ldots, c^{(n+k)\prime}$.

すべての XML ノードの起源は根ノードであるので，根ノードはその下位のすべてのノードの祖先ノードとみなせる．一方，葉ノードは XML の木構造にある最も下位の要素ノードであり，下位にテキストノード以外をもたない．以上の 2 点を合わせると，根ノードはすべての葉ノードが集まって構成されると言えるので，$tf(R) = tf(c) + tf(c') + \cdots + tf(c^{(n+k)\prime})$ が成り立つ．図 5 は, 根ノード R と葉ノード $c, c', \ldots, c^{(n+k)\prime}$ から構成される構成ノード集合とその外にあるノード集合を示している．

- 深さ制限がある根ノード R と葉ノード

図 6 は, 文書の深さに制限 $l-1$ を与えた場合の根ノード R と葉ノード $c_{l-2}, c_{l-1}', \ldots, c_{l-1}^{(k)\prime}$ から構成される構成ノード集合を示している. そえ字はそれぞれのノードの深さを示す. ある深さ l 以上のノードを考慮しないと仮定するので， $c_l, c_l', \ldots, c_l^{(n+k)\prime}$ は省略される．XML の木構造にある最も下位のノード集合 $c_{l-2}, c_{l-1}', \ldots, c_{l-1}^{(k)\prime}$ は, それ自身より下位にノードをもたないので, $tf(R) = tf(c_{l-2}) + tf(c_{l-1}') + \cdots + tf(c_{l-1}^{(k)\prime})$ が成り立つ．

5.　構成ノード集合を利用した *threshold* の計算

構成ノード集合を利用して，3 章で説明した *threshold*，つまり類似度計算が未計算な要素ノードにおける類似度の最大値を計算するため，以下の前定理を使用する．

前定理 5.1 (構成ノード集合にあるノードの類似度)
問合せとノードの類似度が式 *(5)* で与えられ，ある *XML* 文書 D 中の識別子 m のノード D^m と, その子孫関係にあるノード集合 $c, c', \ldots, c^{(n+k)\prime}$ が構成ノード集合を形成している場合, 類似度における関係は，以下のように与えられる．ただし, q は問合せベクトルとする．また，D^m と問合せベクトル q の類似度を $sim(D^m, q)$ と記述する．$tf(e_{ij})$ は，要素ノード e_{ij} における索引語の総出現回数である．

$$sim(D^m,q) = \frac{tf(c)}{tf(D^m)} \cdot sim(c,q) + \frac{tf(c')}{tf(D^m)} \cdot sim(c',q) + \cdots + \frac{tf(c^{(n+k)\prime})}{tf(D^m)} \cdot sim(c^{(n+k)\prime},q) \quad (6)$$

証明 5.1

$D^m, c, c', \ldots, c^{(n+k)\prime}$ は構成ノード集合であるので, $D^m = c + c' + \cdots + c^{(n+k)\prime}$ が成立するから

$$sim(D^m,q) = \frac{D^m \bullet q}{tf(D^m)} = \frac{(c + c' + \cdots + c^{(n+k)\prime}) \bullet q}{tf(D^m)}$$

$$= \frac{tf(c)}{tf(D^m)} \cdot \frac{(c \bullet q)}{tf(c)} + \frac{tf(c')}{tf(D^m)} \cdot \frac{(c' \bullet q)}{tf(c')} + \cdots + \frac{tf(c^{(n+k)\prime})}{tf(D^m)} \cdot \frac{(c^{(n+k)\prime} \bullet q)}{tf(c^{(n+k)\prime})}$$

$$= \frac{tf(c)}{tf(D^m)} \cdot sim(c,q) + \frac{tf(c')}{tf(D^m)} \cdot sim(c',q) + \cdots + \frac{tf(c^{(n+k)\prime})}{tf(D^m)} \cdot sim(c^{(n+k)\prime},q)$$

が成立する．□

5.1　構成ノード集合内における *threshold*

式 (6) を応用して，類似度が計算されていない構成ノード集合内にあるノードの類似度の最大値を求めるため，以下の式を提案する．

定理 5.11 (構成ノード集合内における *threshold*)
図 *5* に示される R を根ノードとする構成ノード集合の中で，索引語数の小さいノードの順に $c, c', \ldots, c^{(i-1)\prime}, c^{(i)\prime}, \ldots, c^{(n+k)\prime}$ とする. 構成ノード集合内で根ノード R と c から $c^{(i-1)\prime}$ までの類似度が計算されていると仮定する．また, ノード $c^{(n+1)\prime}, \ldots, c^{(n+k)\prime}$ は *stop context* と仮定する. すると, 類似度が計算されていない残りの子孫ノード $c^{(k)\prime} (i \leqq k \leqq n)$ の類似度について以下の式が成立する．

$$sim(c^{(k)\prime}, q) \leqq \frac{tf(R)}{tf(c^{(i)\prime})} \left\{ sim(R,q) - \sum_{j=0}^{i-1} \frac{tf(c^{(j)\prime})}{tf(R)} \cdot sim(c^{(j)\prime}, q) \right\} \quad (7)$$

証明 5.11

根ノード R と $c, c', \ldots, c^{(i-1)\prime}, c^{(i)\prime}, \ldots, c^{(n+k)\prime}$ は構成ノード集合なので，式 *(6)* を適用して，

$$sim(R,q) = \frac{tf(c)}{tf(R)} \cdot sim(c,q)$$
$$+ \frac{tf(c')}{tf(R)} \cdot sim(c',q) + \cdots + \frac{tf(c^{(i-1)'})}{tf(R)} \cdot sim(c^{(i-1)'},q)$$
$$+ \frac{tf(c^{(i)'})}{tf(R)} \cdot sim(c^{(i)'},q) + \cdots + \frac{tf(c^{(n)'})}{tf(R)} \cdot sim(c^{(n)'},q)$$
$$+ \frac{tf(c^{(n+1)'})}{tf(R)} \cdot sim(c^{(n+1)'},q) + \cdots + \frac{tf(c^{(n+k)'})}{tf(R)} \cdot sim(c^{(n+k)'},q)$$

従って，

$$sim(R,q) - \left\{\frac{tf(c)}{tf(R)} \cdot sim(c,q)\right.$$
$$\left. + \frac{tf(c')}{tf(R)} \cdot sim(c',q) + \cdots + \frac{tf(c^{(i-1)'})}{tf(R)} \cdot sim(c^{(i-1)'},q)\right\}$$
$$- \left\{\frac{tf(c^{(n+1)'})}{tf(R)} \cdot sim(c^{(n+1)'},q) + \cdots + \frac{tf(c^{(n+k)'})}{tf(R)} \cdot sim(c^{(n+k)'},q)\right\}$$
$$= \frac{tf(c^{(i)'})}{tf(R)} \cdot sim(c^{(i)'},q) + \cdots + \frac{tf(c^{(n)'})}{tf(R)} \cdot sim(c^{(n)'},q)$$

$0 \leqq \{sim(c^{(j)'},q) \mid n+1 \leqq j \leqq n+k\}$ より

$$sim(R,q)$$
$$- \left\{\frac{tf(c)}{tf(R)} \cdot sim(c,q) + \frac{tf(c')}{tf(R)} \cdot sim(c',q) + \cdots + \frac{tf(c^{(i-1)'})}{tf(R)} \cdot sim(c^{(i-1)'},q)\right\}$$
$$\geqq \frac{tf(c^{(i)'})}{tf(R)} \cdot sim(c^{(i)'},q) + \cdots + \frac{tf(c^{(n)'})}{tf(R)} \cdot sim(c^{(n)'},q)$$

索引語数に関する仮定より $tf(c^{(i)'}),\ldots,tf(c^{(n)'}) \geqq tf(c^{(i)'})$ が成り立つので

$$sim(R,q) - \left\{\frac{tf(c)}{tf(R)} \cdot sim(c,q)\right.$$
$$\left. + \frac{tf(c')}{tf(R)} \cdot sim(c',q) + \cdots + \frac{tf(c^{(i-1)'})}{tf(R)} \cdot sim(c^{(i-1)'},q)\right\}$$
$$\geqq \frac{tf(c^{(i)'})}{tf(R)} \cdot \left\{sim(c^{(i)'},q) + \cdots + sim(c^{(n)'},q)\right\}$$

ゆえに

$$\frac{tf(R)}{tf(c^{(i)'})} \cdot \left[sim(R,q) - \left\{\frac{tf(c)}{tf(R)} \cdot sim(c,q)\right.\right.$$
$$\left.\left. + \frac{tf(c')}{tf(R)} \cdot sim(c',q) + \cdots + \frac{tf(c^{(i-1)'})}{tf(R)} \cdot sim(c^{(i-1)'},q)\right\}\right]$$
$$\geqq sim(c^{(i)'},q) + \cdots + sim(c^{(n)'},q)$$

$0 \leqq \{sim(c^{(j)'},q) \mid i \leqq j \leqq n\}$ より，

$$\frac{tf(R)}{tf(c^{(i)'})} \cdot \left\{sim(R,q) - \sum_{j=0}^{i-1} \frac{tf(c^{(j)'})}{tf(R)} \cdot sim(c^{(j)'},q)\right\} \geqq sim(c^{(k)'},q)$$

ただし，$i \leqq k \leqq n$ である．よって，式 (7) が証明された． □

本稿では，式 (7) の右辺を，***max_leaf*** と定義する．*max_leaf* は構成ノード集合内で類似度が計算されていない要素ノードがとりうる類似度の最大値とみなせるため，構成ノード集合内の要素ノードの類似度計算を省略する *threshold* として使用できる．

5.2 構成ノード集合外における *threshold*

構成ノード集合内にあるすでに計算された類似度から，類似度が計算されていない構成ノード集合外にあるノードの類似度の最大値を求めるために以下の式を使用する．

定理 5.21 (構成ノード集合外における *threshold*)
図 5 に示される R を根ノードとする構成ノード集合 $c, c', \ldots, c^{(i-1)'}, c^{(i)'}, \ldots, c^{(n+k)'}$ のすべての類似度が計算されている，*stop context*，本手法によって類似度計算が省略されたかのいずれかの場合，構成ノード集合に含まれない，木構造の上位にある残りのノード集合 $p^{(j)'}(1 \leqq j \leqq u)$ の類似度に関して，以下の式が成り立つ．

$$sim(p^{(j)'},q) \leqq MAX(sim(c,q), sim(c',q), \ldots sim(c^{(n+k)'},q))$$

なお，u は構成ノード集合外にあるノードの数である．$\{p^{(j)'} \mid p^{(i)'} \notin R, c, c', \ldots, c^{(n+k)'}, tf(R) = tf(c) + tf(c') + \cdots + tf(c^{(n+k)'})\}$ である．

証明 5.21
図 5 に示される R を根ノードとする構成ノード集合の子孫ノード $c, c', \ldots, c^{(i-1)'}, c^{(i)'}, \ldots, c^{(n+k)'}$ のいずれかの子ノード $c^{(r)'}, c^{(r+1)'}, \ldots, c^{(r+m)'}(1 \leqq r \leqq r+m \leqq n+k)$ をもつ親ノード $p^{(j)'}(1 \leqq j \leqq u)$ を考える．また，親ノード $p^{(j)'}$ の子ノード $c^{(r)'}, c^{(r+1)'}, \ldots, c^{(r+m)'}$ と問合せ q との類似度の最大値 MAX$(sim(c^{(r)'},q), sim(c^{(r+1)'},q), \ldots sim(c^{(r+m)'},q))$ を $sim(c^{(l)'},q)(r \leqq l \leqq r+m)$ とする．親ノード $p^{(j)'}(1 \leqq j \leqq u)$ とその子ノード $c^{(r)'}, c^{(r+1)'}, \ldots, c^{(r+m)'}$ は構成ノード集合なので，式 (6) を適用して，

$$sim(p^{(j)},q)$$
$$= \frac{tf(c^{(r)'})}{tf(p^{(j)})} \cdot sim(c,q)$$
$$+ \frac{tf(c^{(r+1)'})}{tf(p^{(j)})} \cdot sim(c^{(r+1)'},q) + \cdots + \frac{tf(c^{(r+m)'})}{tf(p^{(j)})} \cdot sim(c^{(r+m)'},q)$$

仮定より，$sim(c^{(r)'},q), sim(c^{(r+1)'},q), \ldots sim(c^{(r+m)'},q)$ の最大値は $sim(c^{(l)'},q)$ なので

$$sim(p^{(j)},q)$$
$$\leqq \frac{tf(c^{(r)'})}{tf(p^{(j)})} \cdot sim(c^{(l)'},q)$$
$$+ \frac{tf(c^{(r+1)'})}{tf(p^{(j)})} \cdot sim(c^{(l)'},q) + \cdots + \frac{tf(c^{(r+m)'})}{tf(p^{(j)})} \cdot sim(c^{(l)'},q)$$
$$= \frac{tf(c^{(r)'}) + tf(c^{(r+1)'}) + \cdots + tf(c^{(r+m)'})}{tf(p^{(j)})} \cdot sim(c^{(l)'},q)$$

$tf(p^{(j)'}) = tf(c^{(r)'}) + tf(c^{(r+1)'}) + \cdots + tf(c^{(r+m)'})$ より

$$sim(p^{(j)},q) \leqq MAX(sim(c^{(r)'},q), sim(c^{(r+1)'},q), \ldots sim(c^{(r+m)'},q))$$

よって，親ノード $p^{(j)'}(1 \leqq j \leqq u)$ の類似度は子ノード $c^{(r)'}, c^{(r+1)'}, \ldots, c^{(r+m)'}$ の類似度の最大値を超えない。同様に他の親ノードやその祖先ノードに関しても同様である．以上より，$c, c', \ldots, c^{(i-1)'}, c^{(i)'}, \ldots, c^{(n+k)'}$ の類似度の最大値を，構成ノード集合に含まれない文書中の残りのノード集合 $p^{(j)'}$ の類似度は超

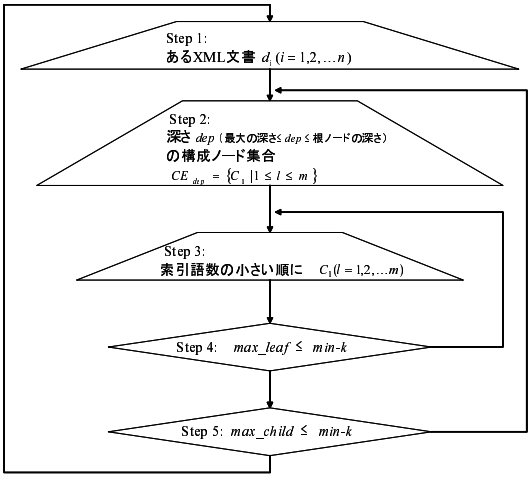

図 7　本手法の処理手順

Fig. 7　The Flow Processsing in our System.

```
Input: Query Keywords (key_1, key_2, ..., key_N)
Output: XML fragments e_j(j=1, 2 ... k)
Function: Search (key_1, key_2, ..., key_N, e_j)
min-k initialized
for(d_i(i=1,2,...,n))
  newdocument;
  maxdepth := the maximum depth of d_i
  Sim(R, q) := score of the root node R of d_i
  len_root := the token frequency of R
  for(dep, from maxdepth to the depth of its root node R)
    ratio * sim:=0
    for(C_l(l = 1, 2, ..., m) from Composed Node Set CE_dep)
      /* C_l is selected in ascending order of token frequency */
      len := token frequency of C_l
      if( (len_root / len) * (Sim(R, q) - ratio * sim) ≦ min-k )
      /* The left side of this equation corresponds to max_leaf */
        then goto newlevel;
      elseif
        then
        Sim(C_l,q) :=score calculatation of C_l
        ratio * sim + = (len / len_root) · Sim(C_l, q)
      endif;
    endfor;
    newlevel;
    if(every C_l 's similarity ≦ min-k)
    //* the above condition corresponds to computing max_child
      then goto newdocument;
    endif;
  endfor;
endfor;
```

図 8　XML 文書検索のしきい値アルゴリズム

Fig. 8　The Threshold Algorithm for XML Search Engines.

えない．定理 5.21 が証明された．　□

本稿では，定理 5.21 の右辺を，***max_child*** と定義する．*max_child* は，考慮している構成ノード集合の上位ノード，いいかえれば構成ノード集合の外にある要素ノードがもつ類似度の最大値とみなすことができる．よって，構成ノード集合の外にある要素ノードの類似度計算を省略する *threshold* として使用できる．

5.3　しきい値 *threshold* を利用した類似度計算の省略

定理 1 の max_leaf の値は，構成ノード集合内の計算されていない要素ノードの中で，索引語数のもっとも小さい要素ノードだけが 0 より大きい類似度をとる場合の値である．いいかえれば，構成ノード集合内で索引語数の最も小さな要素ノードは，もっとも大きな類似度をとる可能性がある．一方，定理 2 の *max_child* から，葉ノードの深さに近いほど類似度が最も大きい値をとることがわかる．

以上を考慮すると，葉ノードに近く索引語数が小さい要素ノードほど類似度がとりうる可能性が高い．よって，深さ制限の大きい構成ノード集合から，その中の要素ノードの索引語数が小さな順に $\max\_leaf$ を計算して，残りのノードの類似度計算を省略することをねらう．さらに，構成ノード集合内にランキングに入るような類似度の大きい要素ノードがない場合にも，構成ノード集合外，つまり，構成ノード集合の上位にある残りの要素ノードの類似度計算を省略することをねらう．

6.　XML 文書検索のしきい値アルゴリズム

本章では，前章の定理を利用した XML 文書検索のしきい値アルゴリズムを説明する．検索処理の手順を図 7 に，それを表現したアルゴリズムを図 8 に示す．

Step 1

文書毎に割り当てられたキーを利用して抽出したそれぞれの文書 $d_i(i = 1,2,\ldots,n)$ に対して文書の最大の深さ *maxdepth*，根ノード R の総索引語数 $tf(R)$ を抽出する．図 8 では，$tf(R)$ を *len_root* と呼んでいる．また，R の類似度 $Sim(R,q)$ を計算する．

Step 2

一文書の最大の深さ *maxdepth* から根ノードの深さまでのそれぞれの深さ *dep* に対して，根ノードに関する構成ノード集合 CE_{dep} を抽出する．

Step 3

抽出したそれぞれの構成ノード集合 CE_{dep} から，ノード $C_l(l = 1,2,\ldots,m)$ を抽出する．ノードは索引語数の小さい順にソートされて格納されている．また定理を利用して max_leaf を求めるため，それぞれ要素ノード C_l の索引語数 $tf(C_l)$ を抽出する．図 8 では，$tf(C_l)$ を *len* と呼んでいる．

Step 4

定理 1 より max_leaf を計算した値が，ランキング結果の最下位の類似度 *min-k* より大きければ，構成ノード集合内の要素ノードの中に検索結果となりうる可能性をもつノードがある．よって，C_l の類似度を計算し $ratio * sim$ を更新した後 Step 3 に戻り，構成ノード集合 CE_{dep} の中で次に索引語数の小さなノードを抽出する．反対に，max_leaf が *min-k* より小さければ，Step 3 の残りの要素ノードの類似度を計算する必要はないので，このステップを終了し Step 5 に入る．

Step 5

定理 2 より，Step 3 で計算した類似度のすべてが *min-k* より小さければ，構成ノード集合外の要素ノードの類似度は計算する必要はないので，Step 1 に入って新しい文書を考える．反対に，Step 3 で計算した類似度のうちひとつでも *min-k* より大きければ，構成ノード集合外の要素ノードの中に，検索結果となる要素ノードが存在する可能性があるため，Step 2 に戻って次の深さの構成ノード集合の処理を始める．

6.1　具体的な例

本節では，図 2 の木構造で表現できる図 1 の XML 文書に対して本手法を具体的に適用した例を，図 9，10，11，12 を使っ

て説明する．

最初に根ノード #0 の類似度を計算し，索引語数も求める．説明を簡単にするため，ノードの索引語数の小さい順に #1, #3, #4, …, #11, #12, #13 と仮定する.また，#1, #3, #7, #11 を *stop context* とする．

次に，計算した根ノードの類似度や索引語数の最小値を式 (7) に代入にして *max_leaf* を求める．*max_leaf* が *min-k* よりも小さければ残りのすべての要素ノードの類似度を計算する必要はない. しかし，ここでは *min-k* よりも大きい値であると仮定するので，構成ノード集合内にある要素ノードを索引語数の小さい順に抽出し，その索引語数と類似度を使って *max_leaf* を更新する. そして，残りの要素ノードの類似度を計算する必要があるか否かの判定に使用する．今回は#4，#5，#8 の類似度と索引語数を使って更新した *max_leaf* は,*min-k* よりも大きいと仮定するので，図 9 のように#4，#5，#8 の類似度が計算されている．

さらに#9 の索引語数を使って更新した *max_leaf* と *min-k* を比較すると，*min-k* の値の方が大きくなった．定理 5. 11 より，構成ノード集合の関係内の類似度が計算されていない#9, #12, #13 のノードは必ず検索結果ランキングの範囲には入らない．よって，実際に類似度を計算する必要がないので，#9, #12，#13 のノードの類似度計算を省略する (図 10).

さらに実際に計算した要素ノード#4，#5，#8 の類似度が *min-k* よりも小さければ，構成ノード集合の上位の要素ノード，すなわち構成集合外にある残りの要素ノードの類似度計算を省略できる．しかし，今回は #8 が *min-k* より大きいと仮定するので，深さ制限を変更して図 11 のような構成ノード集合の関係にあるノード集合を考える．図 9 と同様に要素ノードの索引語数の小さい順は#1，#3，#4，#5, #6，#10 と仮定する．#6 までの *max_leaf* は *min-k* よりも大きいので類似度を計算したが，#10 の索引語数を使って更新した *max_leaf* と *min-k* を比較すると，*min-k* の値の方が大きいので，定理 5. 11 より，#10 の要素ノードの類似度計算を省略する．

現在の深さ制限における構成ノード集合のなかで類似度を計算した要素ノード#4，#5，#6 のいずれも検索ランキングに入らなかった. すなわち，現在の構成ノード集合 *max_child* の値は *min-k* より小さいので，定理 5. 21 より，構成ノード集合の関係外にある#2 のノードは検索結果には入らないと分かる．よって，#2 のノードの類似度計算を省略する (図 12).

7. 性 能 評 価

7.1 実験と考察

本手法により要素ノードの類似度計算の回数を削減することで検索時間が短縮できることを実証するため，性能評価を行った. 実験で使用したデータは，XML 文書検索のための INEX テストコレクション [1] のうち，2001 年度の CG 論文集を使用した. 文書数は 83，XML 文書集合の大きさは約 2.62M バイトである. 統計量などの前処理を行った後の要素ノード数は 7240 である．

性能評価するために使用する問合せは，種類によって content-only(CO) トピックと content-and-structure(CAS) トピックに分けられる．本稿ではキーワードの問合せを扱っているので，評価実験には CO トピックを使用する．実際に使用した問合せは，2003 年度の INEX で使用されたトピック番号 118 の前処理済みの 10 個の問合せキーワードを使用した．

実験環境として，CPU は 1.20GHz Pentium M CPU，メモリは 760MB，OS は Windows XP，DBMS は Oracle 9i Release 1

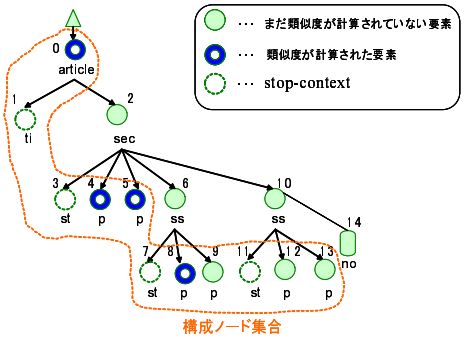

図 9　ある構成ノード集合における類似度計算

Fig. 9　The Score Calculations of a Given Composed Node Set.

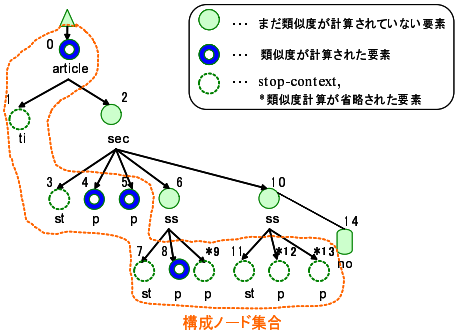

図 10　しきい値 *max_leaf* による類似度計算の省略

Fig. 10　The Score Calculations are Halted Using *max_leaf*.

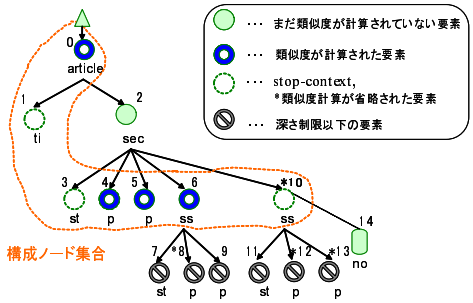

図 11　次の構成ノード集合における類似度計算

Fig. 11　The Score Calculations of the next Composed Node Set.

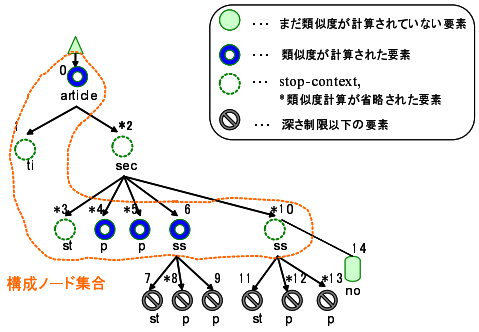

図 12　しきい値 *max_child* による類似度計算の省略

Fig. 12　The Score Calculations are Halted Using *max_child*.

を使用した. プログラミング言語は Java を用い，データベースとの接続には JDBC を使用した. また，図 8 の *maxdepth*，根ノードのキー，要素ノードの *len* などの情報は，XRel に格納されている. また構成ノード集合内のノードはあらかじめ索引語数の小さい順にソートされて格納されている.

本手法の比較対象として，要素ノードの類似度をひとつずつ計算した結果を，SQL の `ORDER BY` によって順序づけし，`ROWNUM` によって上位 *k* 個の要素ノードを抽出する手法をとった.

検索結果として抽出する要素ノード数 *k* を変化させたときに、比較対象と本手法を適用した場合の類似度を計算した要素数を図 13 に示す. また，図 14 には，それぞれの問合せ実行時間を示す. 問合せ実行時間は，問合せを 3 回行った平均値であり，Java の currentTimeMillis を使用して計測した.

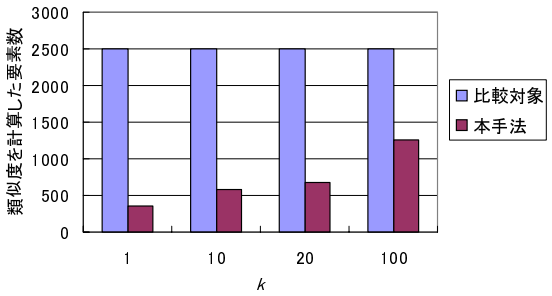

図 13　類似度を計算した要素数

Fig. 13　Number of Elements with Their Similarities Calculated.

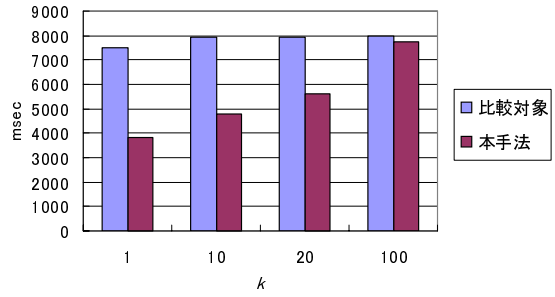

図 14　問合せ実行時間

Fig. 14　Query Processing Time.

図 13 より，比較対象よりも本手法の方が類似度を計算した要素の数が少ないことがわかる. 本手法によって，検索結果となりえない要素ノードの類似度計算の回数を削減できたことがわかる. また本手法では，*k* を小さくするほど類似度を計算する要素ノードの数を顕著に少なくできることが確認できる. 一方，比較対象の場合，上位 *k* 個の検索結果を抽出するのはすべての要素ノードの類似度を逐次計算した後であるため，*k* を変化させても変わらなかった.

図 14 では，比較対象よりも本手法の方が問合せ実行時間が小さいことがわかる. 図 13 で示した要素ノードの類似度計算の回数を削減した効果が，問合せ実行時間の短縮として現れていることが分かる. 特に *k* を小さくするほど類似度を計算する要素ノードの数が少ないので，問合せ実行時間も小さくなる. ただ，*k* が 100 の場合は比較対象よりもそれほど差が見られない. その原因として，現時点のシステムは Java の `PreparedStatement` による SQL を多数実行するなど，プログラムの最適化がまだ充分でないためだと考えられる. しかしながら，本手法によって上位 20 件程度を高速に検索できることを確認できたので，本手法の有効性を実証できたと考えている.

8.　まとめと今後の課題

本稿では，キーワードを問合せとする XML サーチエンジンの，膨大な数の要素ノードの類似度を逐次計算しているために検索コストが高いという問題に対して, 要素ノードの類似度計算の回数を削減するための手法を提案した. 実際に INEX テストコレクションの一部の XML 文書を用いた評価実験から，本手法によって類似度計算する要素の数を大幅に削減できるため, 問合せ実行時間の短縮に効果的であることを実証した.

今後の課題として, INEX テストコレクションのすべての XML 文書を使った詳細な実験で本手法の効果を実証することやプログラムの最適化などが挙げられる.

文　　献

[1] http://qmir.dcs.qmw.ac.uk/INEX/.

[2] B.Jansen. The Effect of Query Complexity on Web Searching Results. *Information Research*, Vol. 6, No. 1, 2000. http://InformationR.net/ir/6-1/paper87.html.

[3] S. Boag, D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie, J. Siméon. *XQuery 1.0: A Query Language for XML*. W3C Working Draft, October 2004. http://www.w3.org/TR/xquery.

[4] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, F. Yergeau. *Extensible Markup Language (XML) 1.0 (Third Edition)*. W3C Recommendation, February 2004. http://www.w3.org/TR/REC-xml.

[5] J. Clark and S. DeRose. *XML Path Language (XPath) Version 1.0*, Vol. 16. W3C Recommendation, November 1999. http://www.w3.org/TR/xpath.

[6] C. Clarke, G. Cormack, F. Burkowski. An Algebra for Structured Text Search and A Framework for its Implementation. *The Computer Journal*, Vol. 38, No. 1, pp. 43–56, 1995.

[7] S. Cohen, J. Mamou, Y.Kanza, and Y.Sagiv. XSEarch: A Semantic Search Engine for XML. *Proc. of the 29th International Conference on Very and Large Data Bases.*, pp. 45–56, September 2003. Morgan Kaufmanm.

[8] R. Fagin, A. Lotem, M. Naori. Optimal Aggregation Algorithms for Middleware. *Proc. of the 25th ACM Symposium on Principles of Database Systems.*, pp. 102–113, 2001.

[9] Kenji Hatano, Hiroko Kinutani, Toshiyuki Amagasa, Yasuhiro Mori, Masatoshi Yoshikawa, Shunsuke Uemura. Analyzing the Properties of XML Fragments Decomposed from the INEX Document Collection. *INEX2004*, 2004.

[10] A. Marian, S. Amer-Yahia, N. Koudas, D. Srivastava. Adaptive Processing of Top-*k* Queries in XML. *ICDE*, 2005.

[11] G. Navarro and R. Baeza-Yates. A Model to Query Document Databases by Content and Structure. *ACM Transactions on Information Systems*, Vol. 15, No. 4, pp. 400–435, October 1997.

[12] N.Fuhr, N.Gövert, and K.Großjohann. HyREX: Hypermedia Retrieval Engine for XML. *Proc. of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*, p. 449, August 2002. ACM Press.

[13] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Retrieval. *Information Processing Management*, Vol. 24, No. 5, pp. 513–523, 1988.

[14] M. Theobald, G. Weikum, R. Schenkel. Top-k Query Evaluation with Probabilistic Guarantees. *Proc. of the 30th International Conference on Very Large Data Bases.*, pp. 648–659, 2004.

[15] Masatoshi Yoshikawa, Toshiyuki Amagasa, Takeyuki Shimura, Shunsuke Uemura. XRel: A Path-Base Approach to Storage and Retrieval of XML Documents Using Relational Database. *ACM Transactions on Internet Technology*, Vol. 1, No. 1, pp. 110–141, 2001.