データベースシステム　122－47
(2000. 7. 27)

構造化文書をランキング可能な全文検索システム

富田 準二, 菊井 玄一郎, 林 良彦
NTT サイバースペース研究所
〒239-0847 神奈川県横須賀市光の丘 1-1
tomita@isl.ntt.co.jp, kikui@nttlny.isl.ntt.co.jp, hayashi@nttnly.isl.ntt.co.jp

あらまし　近年，Webコンテンツの記述の枠組みとしてXMLが注目を集めている．一般的なWebコンテンツとしてのXML文書の検索においては，文書構造と文書内のテキストに記述された内容に基づく検索結果のランキング機能を実現することが重要である．

本稿では，まず，WebコンテンツとしてのXML文書を検索対象とする全文検索システムへの要求条件を明らかにする．次に，この要求条件を満たすものとして，フォーマットファイルに記述された1) 検索対象とするXML文書の部分構造，2) インデクスファイルの形式，3) 単語の抽出方法の指定に基づき，インデクスファイルを作成する全文検索システムを提案する．

キーワード　全文検索システム，構造化文書，XML，ランキング検索，フォーマットファイル，Web

Search system for ranking structured documents

Junji Tomita, Genichiro Kikui and Yoshihiko Hayashi
NTT Cyber Space Laboratories
1-1 Hikarinooka Yokosuka-Shi Kanagawa 239-0847 Japan
tomita@isl.ntt.co.jp, kikui@nttlny.isl.ntt.co.jp, hayashi@nttnly.isl.ntt.co.jp

Abstract　Recently XML has been considered as an emerging standard for describing resources on the Web. It is indispensable for next generation Web search systems to rank XML documents based on their relevance to the query and structural constraints.

This paper describes requirements for search systems, which handle XML documents describing resources on the Web. It then describes a new search system, which makes index files according to a format file specifying 1)the sub-structure of XML documents, 2)the type of index files and 3)the schema of extracting terms to satisfy the requirements.

key words　search system, structured documents, XML, ranking, format file, Web

1 はじめに

現在，Webコンテンツを取得する多くの場面で，HTML文書を検索対象とする全文検索システムが利用されている．HTMLはブラウザでの表示を制御するための言語であり，文書構造等を適切に記述することができない．そのため，Webコンテンツの記述の新しい枠組みとしてXML[1]が注目を集めている[2]．このような，一般的なWebコンテンツとしてのXML文書を検索対象とする場合，XML文書が持つことのできる文書構造と，テキスト部分に記述された内容に基づき検索結果をランキングする機能を実現することが重要である．

XML文書の検索に関する研究は，データベースの分野を中心として様々なものが行なわれてきている．XML文書を解析して得られる木構造の，それぞれのノードをオブジェクト関係データベースに格納する方法[3]や，RDBにXML文書の持つ木構造を格納する方法[4]が提案されている．また，W3C等には，XQL[1]，XML-QL[2]等の検索言語が提案されている．しかし，これらの研究はいずれも，検索式に構造情報を厳密に記述することを想定し，主にデータ指向の文書(Data-oriented documents)[5]を検索対象とし，検索結果のランキングについては言及していない．

また，全文検索システムをベースとした研究として[6][7]がある．これらの研究はテキストの内容を格納するインデクスと構造用のインデクスを併用することによって構造情報を検索要求の中で指定することができる．これらの研究は主に人間が可読な文書(Human-readable documents)[5]を対象とした研究であるが，Webコンテンツの記述に用いられるようなXML文書の持つ特徴を十分に考慮していない．このようなXML文書を検索対象とする場合には，データ指向の文書(Data-oriented documents)と人間が可読な文書(Human-readale documents)の両方の性質を持った混合文書(Mixed-model documents)[5]を考慮する必要がある．

本稿では，一般的なWebコンテンツとしてのXML文書を検索対象とする場合に，考慮しなければならない要求条件について述べ，それらの要求条件を満たす全文検索システムを提案する．実際にプロトタイプを作成し大規模な文書集合に対して本検索システムが有効に働くことを示す．

[1]http://www.w3.org/TandS/QL/QL98/pp/xql.html
[2]http://www.w3.org/TR/NOTE-xml-ql

2 全文検索システムへの要求条件

一般的なWebコンテンツとしてのXML文書は以下のような特徴を持つ．このような特徴を持つ文書の例を図1に示す．

- 主にテキストで記述
  現在，Webコンテンツの多くは，主にテキストで記述され，それに表示用のタグが付加されたものである．同様にWebコンテンツがXMLで記述されたとしても，文書の大部分はテキストを中心としたものとなる．

- 詳細な構造を指定した検索が困難
  XML文書は，DTDが存在する場合でも要素の省略，繰り返し，順序の入れ替わりが起こる．また，DTDが存在しない文書を検索対象とする場合もある．そのため，詳細な文書構造までを指定した検索を行なうことは困難である．

- テキスト以外の型の情報の埋め込み
  Webコンテンツには，id情報，作成日，制御情報等の様々なテキスト以外の型の情報が含まれる場合もある．

- 複数の言語が混在
  現在，Webは世界的に普及し様々な言語で記述されている．また，XMLでは標準の文字コードにUnicodeを採用している．このような状況から1つの文書内に様々な言語が混在する場合もある．

このような特徴を持つXML文書を検索対象とする上で考慮しなければならない点を以下に述べる．

2.1 サブ構造も含めた文字列検索

検索対象となるXML文書の集合は様々な構造を持つため，詳細な構造までを指定した検索を行なうことは難しい．そのため，ある構造を指定した時に，そのサブ構造をまとめて検索できることが望ましい[8]．例えば，検索者が，「<TITLE>内に'検索'が含まれる文書」を検索したいとする．図1の文書では，'検索'が<TITLE>に入れ子になっている<KEYWORD>内に存在する．この場合，<TITLE>の中の<KEYWORD>というサブ構造も含めて検索を行ない，ユーザの要求にこの文書がヒットする必要がある．

```
<DOC>
<LANG> ja </LANG>
<DATE> 19990118 </DATE>
<ENCODING> euc-jp </ENCODING>
<TITLE> 情報 <KEYWORD> 検索 </KEYWORD> システムの新バージョン開発 </TITLE>
<TEXT>
<SECTION name=''概要''>
我々は <KEYWORD> 自然言語処理技術 </KEYWORD> を利用した新しい情報検索システ
ム <LINK href="http://titan.isl.ntt.co.jp/">TITAN</LINK> を開発した.
この検索システムは.....
<ESECTION name=''Abstract''>
We have developed a new search engine called
<LINK href="http://titan.isl.ntt.co.jp">TITAN</LINK>. It uses some
<KEYWORD>natual language processing techiniques</KEYWORD>.
.................
</ESECTION>
<SECTION name=''動作環境''>
TITAN の動作環境は以下の通りである.
使用マシン
<SPEC><CPU><NAME>risc</NAME><SPEED>200<SPEED></CPU>
<MEMORY> 512Mb </MEMORY></SPEC>
</SECTION>
</TEXT>
</DOC>
```

図 1: XML による Web コンテンツの記述

2.2 ランキング機能

大量の Web コンテンツを対象とする全文検索システムにとって，検索結果のランキングは必須の機能である．現在，多くの全文検索システムでは，単語の重要度を基にランキングを行なっている．この重要度の計算には，tf*idf 法 [9] のように単語の出現頻度を基にした方法が用いられることが少なくない．このような出現頻度を適切に計算するためは，検索対象とする領域を定め，その領域内で出現頻度を計算する必要がある．これを実現するためには，以下の 2 点を考慮する必要がある．

- 関係のない部分構造を除く．
  例えば，“euc-jp” が多くの文書の<ENCODING>(その文書の文字コードを指定) 内で出現したとする．この場合の idf 法による”euc-jp” の重要度は小さな値となる[3]．しかし，<TEXT>(本文) 内で”euc-jp” が現れた場合には文書を特徴づける単語となり，重要な単語として取り扱う必要がある．このように，文書全体から単語の出現頻度を計算するのではなく，必要な部分構造を選択して計算できる必要がある．

- 複数の部分構造をまとめる．
  例えば，文書のタイトルと本文を同一に扱って検索を行ないたいとする．この場合，タイトルと本文から別々に単語の出現頻度を計算するのではなく，これらをまとめて出現頻度を計算できることが望ましい．なぜなら，タイトルと本文を別々に扱うと，タイトル内で出現する一般的な単語に対して，過度の重要度を与えてしまうからである[4]．このように複数の部分構造をまとめて単語の出現頻度を計算できる必要がある．

このように，単語の重要度を適切に計算するためには，XML 文書のどの部分構造を用いるのか，どの部分構造とどの部分構造をまとめるのかを制御できる必要がある．

[3]idf 法では，少ない文書に出現する単語程，文書を特徴付けるのに相応しいと考え，その重要度を大きくする．

[4]タイトルは本文と比べて短いため，一般的な語も少数の文書にしか出現しないため，idf 法による重要度が大きくなる

2.3 テキスト以外の型の情報への対応

図 1には，<DATE> や<CPU>/<SPEED> 等，テキスト以外の型の情報が含まれている．このような場合，例えば，<DATE> の内容は通常のテキスト検索だけでなく，ある値(日付)よりも大きい(新しい)ものを取得するといった数値レンジ検索ができる必要がある．このようにすべての情報を通常のテキストと同じように検索するのではなく，対象となる情報の型に即した様々な方法で検索ができる必要がある．

2.4 単語抽出処理

「単語」を基にする全文検索システムでは文書中の文字列に何らかの処理を行なうことによって「単語」を抽出し，これをインデクスファイルに登録している．例えば，日本語の形態素解析や英語の stemming[9] である．これらの単語抽出処理は，インデクス対象となる言語や情報の型によって異なる処理となるため，単語抽出処理を変更しながらインデクスの作成ができる必要がある．

3 構造化文書全文検索システムの設計

第 2 節で述べた要求条件を満たし，大規模の文書を検索可能とするために，我々は，以下のような方針を取った．

1. インデクス作成者が，1) 検索対象とする XML 文書の部分構造 2) インデクスファイルの形式，3) 単語抽出処理を指定するフォーマットファイルを作成する．
2. インデクサはそのフォーマットファイルに基づき複数のインデクスファイルを作成し，それぞれに id を付与する．
3. 検索者は，検索式にこの id をキーワードと一緒に指定する．
4. 検索サーバはこの id を基に適切なインデクスファイルを用いて検索を行ない結果を検索者に返す．

この方針に基づく検索システムの全体像を図 2に示す．それぞれのモジュールの詳細については第 4.1 節で述べる．

このようにインデクスファイルに id を付与し検索時に指定する方法はフィールド検索 [10] と呼ばれ，既に大規模の検索システムで実証済みであり，実装も容易である．ただし，この方針ではフォーマットファイル記述時に，どの部分構造を検索対象とするのかを決定する必要がある．本来は，XQL 等のように検索時に文書内の任意の部分構造を指定できることが望ましいが，1) 第 2 節で述べた要求条件を検索時に解決するのは困難であること，2)DTD や検索対象の文書集合を基に，任意の部分構造を検索時に指定可能なフォーマットファイルを半自動的に生成することも原理的には可能であることから，上記の方針を採用した．以下，フォーマットファイルの詳細と本システムの検索機能について述べる．

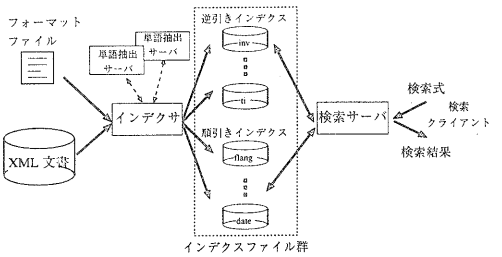

図 2: 検索システムの構成

3.1 フォーマットファイル

フォーマットファイルの例を図 3に示す．フォーマットファイルは XML で記述される．フォーマットファイルは以下の部分から成る．

- INDEX_FILE_DEFINITION
  どのような形式のインデクスファイルを作成するのかを指定する．それぞれの<INDEX> 要素で規定された形式のインデクスファイルが作成され，それぞれに id が付与される．
- ANALYZER_DEFINITION
  どのような単語抽出処理を行なうのかを指定する．それぞれの<ANALYZER> は，単語抽出処理を行なうサーバと結び付けられ，それぞれの処理方法に id が付与される．
- FIELD_DEFINITION
  XML 文書のどの部分構造を検索可能とするのかを指定する．<INDEX> 要素や<ANALYZER> 要素に付与された id を用いて，どのインデクスファイルに単語を登録するのか，どのような単語抽出処理を行なうのかを指定する．

```
<INDEX_FORMAT>
<INDEX_FILE_DEFINITION>
<INDEX id="inv" style="INVERTED" type="TEXT"/>
<INDEX id="ti" style="INVERTED" type="TEXT"/>
<INDEX id="lang" style="INVERTED" type="TEXT"/>
<INDEX id="flang" style="SEQUENTIAL" type="TEXT"/>
<INDEX id="url" style="SEQUENTIAL" type="TEXT"/>
<INDEX id="cpu_speed" style="SEQUENTIAL" type="NUM"/>
<INDEX id="date" style="SEQUENTIAL" type="NUM"/>
</INDEX_FILE_DEFINITION>

<ANALYZER_DEFINITION>
<ANALYZER aid="ja" host="localhost" prot="10002"/>
<ANALYZER aid="eu" host="localhost" port="10001"/>
</ANALYZER_DEFINITION>

<FIELD_DEFINITION>
<FIELD fid="title" target="//TITLE" index_ids="ti inv"/>
<FIELD fid="ja_text" target="//SECTION" ignores="//SPEC" index_ids="inv" analyzer="ja"/ >
<FIELD fid="eu_text" target="//ESECTION" index_ids="inv" analyzer="eu"/>
<FIELD fid="text_lang" target="//LANG" index_ids="lang flang"/>
<FIELD fid="ref" target="//SECTION//LINK/@href" index_ids="url"/>
<FIELD fid="f_date" target="//DATE" index_ids="date"/>
<FIELD fid="speed" target="//CPU/SPEED" index_ids="cpu_speed"/>
</FIELD_DEFINITION>
</INDEX_FORMAT>
```

図 3: フォーマットファイル

それぞれの詳細な指定方法について述べる．

3.1.1 INDEX_FILE_DEFINITION

<INDEX> 要素の style 属性と type 属性を用いてどのような形式のインデクスファイルを作成するのかを指定する．現在，style 属性には，INVERTED と SEQUENTIAL が指定可能であり，逆引きインデクス (inverted file) と順引きインデクスの 2 種類のインデクスファイルを作成することができる．

逆引きインデクス　逆引きインデクスは，現在多くの検索システムで利用されているものであり，以下の形式をしている．'+' は 1 回以上の繰り返しを表す．

キー：単語
値　：(文書 ID，単語の重要度)+

逆引きインデクスを用いると，単語が与えられた場合にその単語が出現する文書の集合を高速に取得することができる．

順引きインデクス　逆引きインデクスは，高速な検索を実現できる反面，以下の 2 つの問題がある．

- フィルタ検索，再ランキング検索の実現が困難
  フィルタ検索とはヒットした文書の中で特定の条件を満たす文書だけを取得する検索方法であり，再ランキング検索とは，特定の条件を満たす文書のランクを上位に修正する検索方法である．例えば，フィルタ検索は “UNIX” を含むという検索条件にヒットした文書の中で日本語で書かれた文書 (**<LANG>** が ja) だけを取り出すというものであり，再ランキング検索は，このような文書を検索結果の上位にランクするというものである．逆引きインデクスの構造から明らかなように，逆引きインデクスを用いてこれらの検索を行なおうとしても，対象となる文書セットの中に日本語で書かれた文書が多数ある場合には，大量の文書を処理する必要があり効率が悪い．そのため，フィルタ検索，再ランキング検索を実現するのに逆引きインデクスは向いて

いない．

- 数値レンジ検索の実現が困難
  数値レンジ検索とは指定された数字よりも大きい (or 小さい or 等しい) 数字を含む文書を取得するという検索方法である．逆引きインデクスの構造から明らかなように，逆引きインデクスを用いて数値レンジ検索を実現することは困難である．

これらの問題を解決するために，以下の順引きインデクスを用いる．

キー：文書 ID
値　：(単語，単語の重要度)*

順引きインデクスは，文書 ID をキーとし，その文書に出現する単語と単語の重要度を引くことができる．詳しくは第 3.2.2 節で述べるが，このファイルを用いると通常検索の結果の上位 n 件に対するフィルタ検索や再ランキング検索を容易に実現することができる．また，単語の部分に数値 (type 属性を NUM とする) を登録すればフィルタ検索や再ランキング検索の条件に数値レンジ検索を取り入れることもできる．このように style 属性と type 属性を用いて，利用方法や登録する情報の型に即したインデクスファイルを作成することができる．

3.1.2 ANALYZER_DEFINITION

<ANALYZER> 要素に，単語抽出処理を行なう単語抽出サーバのホスト名 (host 属性) とポート番号 (port 属性) を指定する．単語抽出サーバは，文字列を受け取り，インデクスファイルに登録する単語をその文字列から作成する．例えば，aid="ja" によって指定される単語抽出サーバは日本語の文字列を入力とし，その形態素解析結果を返す．

3.1.3 FIELD_DEFINITION

<FIELD> 要素の，target 属性に XML 文書のどの部分構造を検索可能とするのかを指定する．target 属性の値は，XPath[5]の省略シンタックスに対応している．現在は，'/'(子要素), '//'(子孫), '@'(属性) の結合によってできる表現だけをサポートしている．

XML 文書を解析し，target 属性で指定された部分構造のサブ構造までを対象としてインデクスを作成する．例えば，target="//TILTE" という指定では，図 1の<TITLE> のサブ構造である<KEYWORD> 内の単語も一緒にインデクスされる．そのため，ユーザは，<TITLE> のサブ構造までを意識せずに検索を行なうことができる．<KEYWORD> までを意識した検索を行ないたい場合には target="//TITLE/KEYWORD" と指定すれば良い．ignores 属性には target 属性の値のサブ構造の内，一緒にインデクスを行ないたくないものを指定する (target="//SECTION" ignores="//SPEC")

index_ids 属性には，単語を登録するインデクスファイルの id を指定する．この id は<INDEX> の id 属性で指定されているものでなければならない．index_ids 属性には複数の id を指定することができ，また，異なる<FIELD> に同じ id を指定することもできる (例 inv 等)．このように 1 つ部分構造から複数のインデクスファイルを作成したり，複数の部分構造から 1 つのインデクスファイルを作成することが可能である．この柔軟な指定方法によって単語の出現頻度を適切に計算することが可能となる．

また，analyzer 属性には，<ANALYZER> 要素の aid 属性の値のどれかを指定する．この指定によって 1 つの文書が複数の言語で記述されている場合でも，なんらかのタグによって部分構造毎に使用言語を特定できる場合には，単語抽出処理を変更しながらインデクスの作成ができる．

3.2 検索機能

多くの全文検索システムと同様に，本検索システムでは，ユーザがキーワードを含む簡単な検索式を指定することによって，その検索式に合致する文書をランキングした結果を返す．検索式は，単語，ブール演算子，構造の指定，フィルタ (再ランキング) 検索の指定からなる．

3.2.1 通常検索

通常検索はフィルタ (再ランキング) 検索の指定を含まない検索であり，逆引きインデクスを用いて処理される．例えば，<TITLE> という構造に “検索” を含み，本文にシステムを含む文書を検索したい場合，以下のように検索式を指定する．

```
システム and ti=(検索)
```

ここで，ti という構造の指定は，図 3のフォーマットファイルによって<TITLE> という部分構造を表す

[5] http://www.w3.org/TR/1999/REC-xpath-19991116

ように関連づけられている．このような検索式が与えられると，それぞれの単語の重要度とブール演算子から評価点(適合度)を計算し，この値の降順にソートして検索結果を作成する．

3.2.2 フィルタ検索，再ランキング検索

フィルタ検索，再ランキング検索は順引きインデクスを用いた検索であり，以下の処理を行なう．

フィルタ検索(filter=()) フィルタ検索の条件部分の評価点が0の場合，文書全体の評価点を0とする．

再ランキング検索(rerank=()) 再ランキング検索の条件部分の評価点を通常検索の評価点に加算したものを文書全体の評価点としてランキングを修正する．

例えば，以下の検索式は，UNIXを含む文書の内，`//<CPU>/<SPEED>`に1000以上の数字を含む文書だけが検索される．

```
UNIX filter=(cpu_speed>1000)
```

また，

```
UNIX rerank=(flang=(ja))
```

という検索式は，UNIXを含む文書の中で評価点の大きい上位n件に対してflangにjaを含む文書のランクをあげるという処理となる．ここでcpu_speedやflangは，図3のフォーマットファイルによって，それぞれ，`<LANG>`や`<CPU>/<SPEED>`を表すように関連づけられている．

このように順引きインデクスを用いるとフィルタ検索，再ランキング検索，数値レンジ検索といった逆引きインデクスだけでは処理が困難な検索を実現できる．

4 実装および評価

4.1 システム構成

第3節で述べた全文検索システムのプロトタイプを作成した．本検索システムは，図2に示したように，前処理を行なうインデクサと検索時の処理を行なう検索サーバからなる．本システムでは大規模の文書集合を対象とするためXML文書の解析にSAXパーザ[6]を用いた．SAXパーザは要素の開始，終了，文字列の出現といったイベントに応じて処理が順次呼び出されるため，文書の大きさに制約を受けず，高速に処理が可能である．インデクサは以下の処理を行なう．

1. フォーマットファイルを入力する．
   フォーマットファイルの`<INDEX>`要素と同じ数だけ空のインデクスファイルを作成する．
2. XML文書の集合を入力する．
3. SAXの”要素イベント”が起きたら`<FIELD>`のtarget属性で与えられたパスをチェックする．
4. 登録対象要素ならそれ以降の文字列をインデクスに登録する．この際，
   - index_idsで指定されたすべてのインデクスに単語を登録する．
   - analyzer属性が指定された場合には，その値に応じた単語抽出サーバと通信を行ない，文字列を単語に分割したものをインデクスに登録する．

検索サーバは，以下の処理を行なう．

1. 単語，and, or, 構造の指定, filter, rerankを含む検索式を検索クライアントから受け取る．
2. 検索式を解析し，第3.2節で述べた方法でそれぞれの文書の評価点を計算する．
3. 評価点の降順に文書をソートし検索結果とする．
4. 検索結果を検索クライアントに返す．

4.2 評価

本検索システムはSAXパーザにXML4C 3.0.1[7]を用い，C++(SUN WorkShop5.0)言語で実装を行ない，Sun Ultra Enterprise450(2 × Ultra SPARC-II 296MHz，メモリ 512Mb)，Solaris2.6上で動作している．

検索サーバの検索速度は従来からあるフィールド検索と同程度であり，インデクス作成時間はフィールド検索のインデクス作成時間にSAXパーズ処理時間を加えたものである．そこで，SAXパーズ処理時間を単体で測定した．Web上から集めた約36,000件のHTML文書(主に日本語 283Mb)をXML文書に変

[6] http://www.megginson.com/SAX/index.html

[7] http://www.trl.ibm.co.jp/alphaWorks/xml4c/xml4c.htm

換することによって評価セットを作成した．XML4CのSAXパーザが，この評価セットのすべての文書を解析するのにかかる時間は約10分間であった．このようにSAXパーズ処理にかかる時間はほんの僅かである．

また，上記の評価セットから本検索システムを用いて作成したインデクスファイルのサイズは49.8Mbであり，これは元の文書の約18%と小さい．以上のことから，本手法に基づき大規模のXML文書集合を対象とした全文検索システムを構築することは可能であると考えられる．

5 結論および今後の課題

本稿では，一般的なWebコンテンツとしてのXMLを再考し，このようなXML文書を対象とした全文検索システムへの要求条件を明らかにした．その中で，サブ構造も含めた文字列検索，正確なランキング，テキスト以外の型の情報への対応，単語抽出処理の選択が必要であることを述べた．次に，このような要求条件を満たすものとして「フォーマットファイルに記述された1) 検索対象とするXML文書の部分構造，2) インデクスファイルの形式，3) 単語抽出処理方法の指定に基づきインデクスファイルを複数作成し，それぞれのインデクスファイルにidをふり，検索時にこれらのidを指定する．」という方針に基づく全文検索エンジンを提案した．本フォーマットファイルの記述方法は非常に柔軟性の高いものであり，例えば，INDEX_FILE_DEFINITIONに新しいインデクスの形式(style属性やtype属性)を導入することによって，RDB等との連携も可能である．また，中国語や韓国語といった様々な言語の文字列から単語抽出を行なう単語抽出サーバを作成し，フォーマットファイルのANALYZER_DEFINITIONに指定するだけで，容易にこれらの言語にも拡張することができる．

本システムの大きな制限として，検索時に検索対象とする部分構造を指定できないということがあげられる．フォーマットファイルの半自動生成や，[7]で提案されている構造インデクスを併用することによって，この制限を解決することが重要である．また，データベースや単語抽出サーバとのプロトコルを明確にすることができれば，今後，その適用範囲は広がるものと考えられる．

参考文献

[1] Bray, T., Paoli, J. and Sperberg-McQueen, C.: Extensible Markup Language (XML) 1.0 (1998). http://www.w3.org/TR/1998/REC-xml-19980210.

[2] 浦本直彦: 小特集「XML: インターネット上での情報の記述と交換」にあたって，人工知能学会誌, Vol. 13, No. 4, p. 506 (1998).

[3] 吉川正俊, 志村壮是, 植村俊亮: オブジェクト関係データベースを用いたXML文書の格納と検索，情報処理学会論文誌, Vol. 40, No. SIG6, pp. 115–131 (1999).

[4] 金政泰彦, 久保田和己, 石川博: XML問い合わせ処理システム(xQues)のデータ格納管理，信学技報 DE99-50, pp. 117–122 (1999).

[5] Fankhauser, P., Marchiori, M. and Robie, J.: XML Query Requirement W3C Working Draft 31 January 2000 (2000). http://www.w3.org/TR/xmlquery-req.

[6] 多田勝己, 岡本卓哉, 菅谷奈津子, 加藤寛次, 川下靖司: 構造化文書対応全文検索システム Bibliotheca2 TextSearchの開発(3)，情報処理学会第55回(平成9年後期)全国大会，pp. 3–111–3–112 (1997).

[7] 井形伸之, 難波功: 大規模な構造化文書データベースにおけるインデクシングと検索の手法，情処研報情報学基礎 57-2, pp. 9–16 (2000).

[8] Florescu, D., Kossmann, D. and Manolescu, I.: Integrating keyword search into XML query processing, *9th International WWW Conference Proceedings*, pp. 119–135 (2000).

[9] Frakes, W. B. and Baeza-Yates, R.(eds.): *Information Retrieval Data Structures & Algorithms*, Prentice Hall (1992).

[10] Baeza-Yates, R. and Ribeiro-Neto, B.: *Modern Information Retrieval*, Addison-Wesley (1999).