社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－DBS－140 (I) (27)
2006／7／12

内容と構造を指定したXML文書検索

寺田 憲正† 清水 敏之†† 吉川 正俊††

† 名古屋大学大学院 情報科学研究科
†† 京都大学大学院 情報学研究科
E-mail: †terada@dl.itc.nagoya-u.ac.jp, ††shimizu@soc.i.kyoto-u.ac.jp,
†††yoshikawa@i.kyoto-u.ac.jp

あらまし XML (Extensible Markup Language) は文書やデータの構造，意味を記述するためのマークアップ言語の一つであり，現在利用が広がっている．本研究では，学術論文などの文書志向のXML文書を対象とし，精度と速度の面から，検索性能を向上させる手法を提案する．特に，検索語による内容の条件と構造の条件を両方指定した検索の性能を向上させる．精度に関しては，複数の条件が指定された問合せを処理するために，各条件の評価を統合する方法を考案した．また，速度に関しては，関係データベースを使って，XMLデータベースを実装し，速度を向上させるスキーマを考案した．INEXテストコレクションを用いた実験を行い，検索速度については従来手法に比べ，内容と構造を指定した問合せにおいて，平均26.4倍の高速化を達成すると共に複数条件の評価を統合する関数の性能を確認した．

Content-And-Structure XML Document Retrieval

Norimasa TERADA†, Toshiyuki SHIMIZU††, and Masatoshi YOSHIKAWA††

† Graduate School of Information Science, Nagoya University
†† Graduate School of Informatics, Kyoto University
E-mail: †terada@dl.itc.nagoya-u.ac.jp, ††shimizu@soc.i.kyoto-u.ac.jp,
†††yoshikawa@i.kyoto-u.ac.jp

Abstract XML is one of markup languages to describe documents, data structure and meanings. Today, XML is more and more widely used. In our study, we set document-centric XML documents as a target, for example academic papers, and we propose the approach that improves the performance of retrieval from the aspect of precision and speed. Especially, our study improves the performance of retrieval that identifies the condition both about content by keywords and structure. In terms of precision, we considered the approach that integrates scores of each conditions to process the query that identifies multiple conditions. In terms of speed, we implemented XML database by using relational database, and we considered the schema that improves speed. We experimented with INEX test collection, and we achieved about 26.4 times speed-up on the average to the query that identifies content and structure. Furthermore we confirmed the performance of functions that integrate scores of multiple conditions.

1. はじめに

XML (Extensible Markup Language) は文書やデータの構造，意味を記述するためのマークアップ言語の一つである．XMLではタグの名前を自由に決められるため，様々な分野において利用されている．XML文書は大きく二つの種類に分けられる．注文情報や科学技術データのようなデータ中心のXML文書と論文やWebサイ

トのような文書中心の XML 文書である．データ中心の XML 文書を検索する場合，検索の答は明確に定義できることが多い．文書は XPath などの XML 問合せ言語によって検索され，その結果は問合せ条件と合致したものになる．一方，文書中心の XML 文書を検索する場合，問合せは検索語によって与えられ，検索の答は問合せとの類似度のような尺度とともに返されるべきである．この場合，XML 問合せ言語による問合せは必ずしも有効ではなく，情報検索の手法を用いて検索を行うことが有効となる．

そこで本研究では XML によって構造化された文書の情報検索を扱う．関係データベースにもとづく XML データベース XRel [1] に検索語による検索機能を付加し，情報検索の手法による検索に対応した．XRel に格納された文書は部分文書の単位で検索を行うことが可能であり，検索結果は検索要求との関連が高い順に順位付けされて出力される．検索語による検索はベクトル空間モデルに基づいて実装した．

構造化された文書の検索を行う場合，一つの文書が複数の部分文書を含むため，膨大な検索対象を持つことになる．全ての部分文書ベクトルの情報を持つことはデータの肥大化につながり，精度/再現率の低下や検索速度の低下を招く．

文献 [2] では，文書志向の XML 文書の中でも，部分文書単位でデータ志向であるか文書志向であるかという性質を持つことに注目し，XML 文書から得られるピリオド率，異なり語数の統計量を用いて，論文の表題や著者のように検索語による検索対象として不適切なデータ指向の部分文書を削除することによって検索の精度と速度を向上した．

しかし，XML の構造化されているという利点を活かして，検索語だけでなく，構造も指定した検索も行いたいという要求が考えられる．文献 [2] では検索語のみによる検索を対象としており，構造を指定した問合せについては考えていない．

本研究では，XPath を拡張して，検索語による内容の条件と XML 木上の経路による構造の条件を指定した問合せを対象に考える．XPath に about 関数を導入し，これにより内容と構造の条件を指定する．この関数は二つの引数をとり，第一引数は相対経路，第二引数は検索語である．第一引数の相対経路を持つ部分文書に対して，検索語に対して関連の強い順に順位付けて返す関数である．そのような拡張された問合せの中で，特に複数の条件を指定した問合せの処理方法が問題になる．複数の条件を指定した問合せの例として以下のようなものが挙げられる．

//article[about(.,"XML")]//sec[about(.,"database")]

この問合せは XML に関連する論文の中で，database に関連する節を探す問合せである．この問合せでは"論文全体が XML に関連している"という条件と"その論文の節が database に関連している"という二つの条件を指定することになる．各条件の評価を行い，それを問合せ全体に対してどのように評価を行うかは検索性能を左右するため重要である．

そのため，本研究では問合せに存在する複数の条件に対する評価値を統合する方法を文献 [5] を参考に統合関数を考え，どの関数が有効であるかを確かめるために比較実験を行った．

また文献 [2] は検索語のみによる問合せを対象としており，内容と構造を指定した問合せの処理について考えていない．そこで検索対象を削除することなく，検索速度を向上させる手法を提案する．

具体的には，膨大な数になる部分文書ベクトルを格納した関係表を語の種類によって分けることで高速化を狙う．問合せ時にはユーザが入力した検索語に対応するベクトルの要素だけを参照することで検索速度を向上する．この手法は検索語のみによる検索に対しても有効である．構造を指定した検索に対しては，文献 [1] の関係スキーマを拡張し，索引をはることによって構造指定の条件判定を効率化する．

以下，第 2. 節では XML データベースシステム XRel [1] を紹介し，XRel を拡張し，ベクトル空間モデルに基づいて構築した XML 情報検索システムについて説明する．第 3. 節では複数の条件が指定された問合せに対して，各条件の評価を統合する手法について述べる．第 4. 節では検索速度を向上するための関係スキーマについて述べる．第 5. 節では構造を指定した検索に対して，提案した評価値を統合する関数が検索精度に与える影響を比較実験から調査した．それと同時に，第 4. 節で述べた検索速度を向上する手法の効果を実験で確認した．最後に第 6. 節でまとめと今後の課題について述べる．

2. XML 情報検索システム

この節では，ベクトル空間モデルによる XML 文書の検索とその実装について述べる．本研究では，関係データベースに基づいた XML データベースシステム XRel を利用している．部分文書のベクトルを関係表に格納し，検索語や経路を含む内容と構造を指定した問合せを関係データベースの問合せ言語である SQL に変換することで実装している．

2.1 XRel

XRel は関係データベースに基づく XML データベー

スシステムである．XRel は関係データベースの構造に XML の文書構造を写像するアプローチをとっており，XML 文書を分解して固定した関係スキーマを持つ四つの表に格納する．関係スキーマの構造は XML 文書スキーマや，文書の論理構造に現れる要素型に依存しないため，文書の論理構造の変化が関係スキーマに影響を及ぼすことがない．XML 文書中の各節点は XML 木中での位置を表すラベルと XML 木における根からの経路によって管理される．

利用者から検索語や XML 問合せ言語による問合せが発行されると，XRel はそれらの問合せを関係データベースに対する問合せ言語である SQL 文に変換し，SQL 問合せとして実行する．XRel では XML 問合せ言語 XPath に対応している．システム側は利用者や利用システムに対して XML 文書が関係データベースによって管理されていることを隠し，あたかも実際に XML 文書が格納されているかのように扱う．利用者や応用システム側から検索語や XML 問合せ言語による問合せが発行された場合，システム側が自動的に SQL に変換して SQL 問合せを実行する．

XML 文書を格納するための表として Element, Attribute, Text, Path の四つを定義する．

- Element (docID, nodeID, pathID, start, end)
- Attribute (docID, nodeID, pathID, start, end, value)
- Text (docID, nodeID, pathID, start, end, value)
- Path (pathID, pathexp)

Element, Attribute, Text, Path はそれぞれ要素ノード，属性ノード，テキストノード，XML 文書に現れる全ての経路を格納する表である．表の各属性について説明する．

- docID: XML 文書を識別するための ID
- nodeID: ノードを一意に識別するための ID
- pathID: 経路を識別するための ID
- start: 各ノードの開始タグのバイト位置
- end: 各ノードの終了タグのバイト位置
- value: 属性値または要素内の文字列
- pathexp: 実際の経路表現

```
<a b="Attr"><c>Text 1</c><c>Text 2</c></a>
```

図 1　XML 文書の例

XML 文書の格納例を挙げる．図 1 の XML 文書を木構造で表したものが図 2 である．図 2 の XML 文書を格納すると表 1 のようになる．

2.2　ベクトル空間モデルによる部分文書検索

XRel ではベクトル空間モデルを関係データベースに基づいて実装することで，検索語入力による XML 文書の検索に対応している．XML 文書中の各部分文書の文書ベクトルを格納するための表の関係スキーマを以下に記す．

- Token (docID, nodeID, token, tfipf)

表 Token は XML 文書に登場する各単語 (token 属性) と単語が属する部分文書の ID(docID 属性と nodeID 属性)，単語の重み (tfipf 属性) を属性として持つ．同一の部分文書 ID を持つトークンの集合が部分文書の文書ベクトルを表す．文書ベクトルの要素は重みがゼロでないもののみを格納している．

単語の重みは TF-IPF を文書長に応じて正規化した指標を用いる．正規化については [4] を参考に行った．TF-IPF は TF(Term Frequency) と IPF(Inversed Path Frequency) [3] の積である．TF はある部分文書における単語の出現頻度を表す．IPF は XML 文書構造を考慮に入れた単語の特定性を表す尺度である．情報検索の分野で広く使われている IDF(Inversed Document Frequency) が文書集合の中での単語の特定性を表す尺度であるのに対し，IPF はある経路における部分文書集合の中での単語の特定性を表す尺度である．ある部分文書が索引語 k に対して持つベクトルの重みは以下の式 (1) で与える．

$$weight = \frac{ntf}{ndl} \times ipf \qquad (1)$$

TF に対して，対数を 2 回とって弱めたのが ntf であり，式 (2) で表せる．

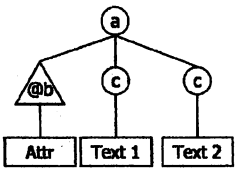

図 2　XML 木の例

表 1　図 2 の XML 文書を格納した例

Element

docID	nodeID	pathID	start	end
1	1	1	1	47
1	3	3	15	28
1	4	3	31	44

Path

pathID	pathexp
1	#/a
2	#/a#/@b
3	#/a#/c

Attribute

docID	nodeID	pathID	start	end	value
1	2	2	4	11	Attr

Text

docID	nodeID	pathID	start	end	value
1	3	3	18	23	Text 1
1	4	3	34	39	Text 2

$$ntf = 1 + ln(1 + ln(tf)) \quad (2)$$

N_p をある経路 p における全部分文書の数，$df_p(t)$ を経路 p における索引語 t が存在する部分文書の数とすると，経路 p における単語 t の ipf は式 (3) で計算される．

$$ipf = ln\frac{N_p + 1}{df_p} \quad (3)$$

ndl は正規化された文書長であり，式 (4) で表せる．

$$\left((1-s) + s \times \frac{dl}{avgdl_p}\right) \times (1 + ln(avgdl_p)) \quad (4)$$

この式での s はパラメータであり，0 以上 1 以下の値をとる．本研究では s=0.2 として実装した．dl は対象部分文書が持つ索引語数を表す．$avgdl_p$ は対象部分文書が持つ経路 p における全部分文書の索引語数の平均である．

この計算を各部分文書が持つ全ての索引語に対してそれぞれ行い，Token 表の行として格納する．そのため，Token 表の行数は膨大になる．問合せを処理する際には，入力された検索語と合致する Token 表の行を探す必要がある．一般に関係データベースの検索速度は表の行数に依存するため，この処理はコストが非常に大きい．これに対する改善策は第 4. 節で述べる．

2.3 問合せ変換

XRel では検索語による XML 文書の検索に対応すると同時に XML 木中での経路を決定する XPath のような問合せにも対応している．入力された検索語や経路情報をシステムが自動的に関係データベースに対する問合せ言語 SQL に変換し，関係データベースへの問合せを実行する．これにより，利用者やアプリケーションは表の形で格納された XML 文書をあたかも実際の XML 文書が存在するかのように扱うことができる．本節では検索語による問合せと XPath による問合せから SQL 問合せへの変換について記す．

2.3.1 検索語による問合せの変換

検索語 "keyword_1, keyword_2, ..., keyword_n" は以下のような SQL 文に変換される．

```
SELECT * FROM (
  SELECT t.docID, t.nodeID, e.start, e,end,
         SUM(t.tfipf) score
  FROM   Token t, Element e
  WHERE  t.docID  = e.documentID
  AND    t.nodeID = e.nodeID
  AND    t.token in ('keyword_1', 'keyword_2',
                                 ..., 'keyword_n')
  GROUP BY t.docID, t.nodeID, e.start, e.end ) r
ORDER BY r.score DESC
```

"keyword_1, keyword_2, ..., keyword_n" の TF-IPF の和を各部分文書に対して計算し，部分文書の順位付けを行っている．検索語ベクトルと部分文書ベクトルの類似度は内積によって計算する．TF-IPF の和は検索語集合内に出現する単語の重みを 1，出現しない単語の重みを 0 とした検索語ベクトルと，各部分文書ベクトルとの内積にあたる．

ここで出現する Token 表は行数が膨大であり，その中から検索語に合致する行を探し出すコストが大きくなり，結果として検索速度の低下を招く．

2.3.2 XPath 問合せの変換

Path 表を用いることにより XPath のような経路情報を含む問合せに対応することができる．以下では XPath から SQL への変換について述べる．

単純な経路式"/a/c"を変換した SQL 文を以下に記す．Path 表から経路に対応する経路 ID を求め，Element 表と結合することによりその経路上の要素を指定する．

```
SELECT e.docID, e.nodeID, e.start, e,end
FROM   Element e, Path p
WHERE  e.pathID  = p.pathID
AND    p.pathexp like '#/a#/c'
```

子孫要素を指定する//を含む経路式"//a//c"を変換した SQL を以下に記す．Path 表に対する文字列照合により該当する経路の経路 ID を求め，Element 表と結合することによりその経路上の要素を指定する．この場合，正規表現の処理を Path 表に対して行うことになる．正規表現の条件検索はコストの大きい処理である．また Element 表の行数が膨大であるため，Element 表と Path 表の結合もコストの大きい処理になる．

```
SELECT e.docID, e.nodeID, e.start, e,end
FROM   Element e, Path p
WHERE  e.pathID  = p.pathID
AND    p.pathexp like '%#/a%#/c'
```

2.3.3 内容と構造を指定した問合せの変換

検索語による内容の条件と経路による構造の条件の両方を指定した問合せから変換される SQL 文は，先に述べた検索語から変換される SQL 文に構造の条件を追加することで作成できる．

例えば，以下の問合せから変換される SQL 文を考える．

//a//c[about(.,"keyword_1, keyword_2,..., keyword_n")]

この問合せは//a//c という経路を持つ部分文書に対して，検索語"keyword_1, keyword_2,...,keyword_n" に対して，関連の高い部分文書から順に並べた結果を求める問合せである．

この問合せから変換される SQL 文を以下に示す．

```
SELECT * FROM (
```

```
SELECT t.docID, t.nodeID, e.start, e,end,
       SUM(t.tfipf) score
FROM   Token t, Element e, Path p
WHERE  t.docID  = e.documentID
AND    t.nodeID = e.nodeID
AND    e.pathID = p.pathID
AND    t.token in ('keyword_1', 'keyword_2',
                          ..., 'keyword_n')
AND    p.pathexp like '%#/a%#/c'
GROUP BY t.docID, t.nodeID, e.start, e.end ) r
ORDER BY r.score DESC
```

3. 複数の条件に対する評価値の統合

複数の条件を指定した問合せに対して，各部分文書に評価値を与え，その値が高い順にユーザの検索要求を満たしているとして出力したい．複数の条件を指定した問合せとしては例えば以下のようなものが挙げられる．

//article[about(.,"XML")]//sec[about(.,"database")]

この問合せの意味は以下のようになり，複数の条件を指定していることが分かる．

- **出力結果:** 論文の節
- **条件**

（1） 論文全体が検索語"XML"に関連している

（2） 出力する節が検索語"database"に関連している

このような問合せの検索結果として，条件 (1) と条件 (2) を両方満たすものが望ましい．したがって，出力結果となる各部分文書の問合せに対する評価値を計算するためには，各条件に対する評価値を計算し，それらを統合する必要がある．

本節では，複数の条件を指定した問合せに対して，各条件の評価値をどのように統合するかについて述べる．評価値の統合方法により，検索性能は左右されるため重要である．

まず，各条件の評価値を計算する．条件 (1) に対しては，その部分文書が論文全体を表しているという構造の条件を満たす中で，検索語"XML"に対する評価値を計算する．この値を s_1 とする．条件 (2) に対しては，その部分文書が節を表しているという構造の条件を満たす中で，検索語"database"に対する評価値を計算する．この値を s_2 とする．

次に，計算された評価値を統合する方法について説明する．最終的に出力するのは論文の節であるため，各節に対して，その節が属する論文全体の条件 (1) に対する評価値 s_1 と，その節自身の条件 (2) に対する評価値 s_2 を統合して，その節が問合せに対して持つ評価値とする．

そのときに，s_1 と s_2 をどのように統合するかが問題となる．本研究では，以下に挙げる三つの関数を考え，実験を行った．全体的な評価値を S とする．

- **相加平均**

$$S = \frac{s_1 + s_2}{2} \tag{5}$$

この評価値を用いることで，一つの条件に対する評価値が高くても，S の値はほとんど変化がない．

- **相乗平均**

$$S = \sqrt{s_1 \times s_2} \tag{6}$$

相加平均と同様，一つの条件に対する評価値が高くても，S の値はほとんど変化がない．また，s_1, s_2 は，正の値であるため，相加平均より相乗平均は必ず小さい．

- **調和平均**

$$S = \frac{2}{\frac{1}{s_1} + \frac{1}{s_2}} \tag{7}$$

相加平均，相乗平均の場合と同様，ただ一つの条件に対する評価値が高くても，S の値はほとんど変化がない．s_1, s_2 のいずれかが 0 のときは S は 0 であるとして計算した．

4. 検索速度を向上する関係スキーマ

この節では，XML 文書検索を高速に行うため，検索速度を向上する手法について説明する．

4.1 Token 表の分割

検索語を指定した検索において，ボトルネックになるのは膨大な数の部分文書ベクトルから検索語に該当するベクトルの要素を探し出す処理である．文献 [2] では検索対象を削除することによって速度を向上した．

しかし，内容と構造を指定した検索においては，検索対象の削除は精度を低下させる恐れがある．そこで，部分文書ベクトルは削除せず，語の種類によって Token 表を分けることで問合せ時の検索対象を減らす．それにより，検索速度の向上を目指す．

検索語 "keyword_1, keyword_2, ..., keyword_n" が入力された場合を考える．そのときの SQL 文を以下に示す．語によって Token 表が分割されているため，語 keyword_i(i=1,2,...,n) に関する部分文書ベクトルのみを格納した表 Token_keyword_i(i=1,2,...,n) を参照するだけで，表の中から検索語に合致するベクトルの要素であるか否かを判定する必要はない．また語によって分けられたそれぞれの表の行数は元の Token 表に比べて非常に小さい．これにより，検索速度の向上が期待できる．

```
SELECT * FROM (
  SELECT t.docID, t.nodeID, e.start, e,end,
```

```
        SUM(t.tfipf) score
  FROM  (SELECT * FROM Token_keyword_1
         UNION
         SELECT * FROM Token_keyword_2
         UNION
         ...
         SELECT * FROM Token_keyword_n) t,
         Element e
  WHERE  t.docID  = e.docID
  AND    t.nodeID = e.nodeID
  GROUP BY t.docID, t.nodeID, e.start, e.end ) r
ORDER BY r.score DESC
```

4.2 Element 表の拡張

構造を指定した検索において，検索速度の低下を招く原因は二つ考えられる．まず，一つは Element 表と Path 表の結合である．関係データベースにおいて結合という処理はコストが大きい．もう一つは，"//"を含む問合せの時に必要になる正規表現の処理である．

この問題に対処するために，Path 表が持つ経路の情報を Element 表に含めることで問合せ時の結合を減らす．具体的には，Element 表の pathID に対応する Path 表の pathexp を Element 表に格納することで，Element 表のみで構造に関する条件の判定が行えるようにして，高速化をはかる．また"//"を含むような経路の指定において，末尾に注目する．例えば，"//sec"のような構造の条件を満たす部分文書のタグ名は sec である．この事実を利用して，構造の条件に関する検索の高速化をはかる．具体的には，Element 表にタグ名という属性 (tag) を追加して，その属性に対する索引をはる．表 2 に，拡張した Element 表を示す．

表 2 拡張した Element 表

docID	nodeID	pathexp	tag	start	end
1	1	#/a	a	1	47
1	3	#/a#/@b	@b	15	28
1	4	#/a#/c	c	31	44

拡張した Element 表を用いた構造に関する問合せ"//a//c"から変換される SQL 問合せを以下に示す．Element 表の拡張によって結合のない SQL 問合せに変換できている．また"//a//c"という経路の条件を満たす部分文書のタグは必ず c であることからタグ名が c であるという条件を SQL 問合せに追加している．この条件判定は正規表現よりコストが小さく，索引を用いることでさらに高速化が狙える．

```
SELECT e.docID, e.nodeID, e.start, e,end
FROM   Element e
WHERE  e.pathexp like '%#/a%#/c'
AND    e.tag like 'c'
```

5. 実　　験

テストコレクションとして INEX-1.9 を使用して実験を行った．本節では，その結果について述べる．

5.1 INEX

INEX(INitiative for the Evaluation of XML retrieval) [6] は 2002 年 4 月に発足した XML 部分文書検索のための国際プロジェクトである．INEX テストコレクションは大きく三つの部分に分かれている．すなわち，検索対象の文書群 INEX document collection, 検索の問合せ集合 INEX topics, 問合せに対する解答部分文書集合とその評価 INEX relevant assessments である．

INEX document collection は IEEE の様々なトランザクションの論文を XML 形式で構造化したものから構成されている．INEX-1.9 は 1995 年から 2004 年の論文から構成されており，論文件数は 16,819 件，総サイズは 764MB，総部分文書数は約 1,600 万件，部分文書ベクトルの要素数は約 1 億 2,800 万件である．

INEX2005 の Ad-hoc Retrieval では三つのサブタスクを定義している．

（1）CO: Content-oriented XML retrieval using content only conditions.

（2）CO+S: Content-oriented XML retrieval using additional structural hints.

（3）CAS: Content-oriented XML retrieval based on content-and-structure queries.

CO は検索システムの利用者が文書構造についての知識を必要としない検索を前提としており，検索語集合を入力とする．CO+S は検索システムの利用者が文書構造に関するヒントを追加できる場合を考える．入力は検索語集合に加え，文書構造に関するヒントが追加される．CAS は検索の答となる構造を指定する target element と検索条件として構造を指定する support element があり，それぞれを厳密な制約と考える場合とあいまいな制約として考える場合が存在する．本研究の現時点では，target element, support element の両方に対する構造の条件を厳密な制約と考える SSCAS サブタスクを対象としている．

CO, CO+S サブタスクでは利用者の検索行動を仮定して Focused, Through, FetchBrowse と呼ばれる三つの検索方策を定義している．Focused では要素のオーバーラップを許さず，先祖子孫関係にあるノードのうち関連度の高い一つの要素のみを答とする．Through では要素のオーバーラップを許し，全ての要素に対して関連度の高い順に順位付けを行う．FetchBrowse では検索を二段階に分けて行う．初めに論文単位で関連の高い順に順位

付けを行い，次にその論文内での関連の高い部分文書を特定する．INEX topics では後述する CO サブタスクと CO+S サブタスクで共通したものが 40 個，CAS サブタスクで 47 個用意されている．

INEX relevant assessments は解答部分文書集合の relevance をトピックの内容がどの程度カバーされているかを評価する exhaustivity とトピックの内容にどの程度類似しているのかを評価する specificity の二次元で表現している．exhaustiveness は 2(highly exhaustive), 1(somewhat exhaustive), 0(not), ?(too small) の四段階の値を取り，specificity は 0 から 1 の間の連続した値を取る．また，ある問合せに対して relevant な要素が存在する場合，その祖先要素は relevant であるとしている．

以下に述べる評価実験では，CPU:Intel Xeon 3.80 GHz(2 CPU), Memory: 8.0 GB, OS: Miracle Linux 3.0, DBMS: Oracle 10g Release1 という実験環境で行った．

5.2 精度の評価実験

本節では，内容と構造を指定した問合せにおいて，複数の条件をどのように統合するかが検索精度を左右することを示す．先に提案した相加平均，相乗平均，調和平均を統合関数としてそれぞれ用いた場合に，検索精度がどのようになるか比較実験を行った．

target element, support element がともに構造の条件を厳密な制約として考えた SSCAS サブタスクを問合せの対象として実験を行った．解答文書が用意されている SSCAS サブタスクの問合せは 8 個あり，その 8 個の問合せに対する平均の精度/再現率グラフを図 3 に示す．

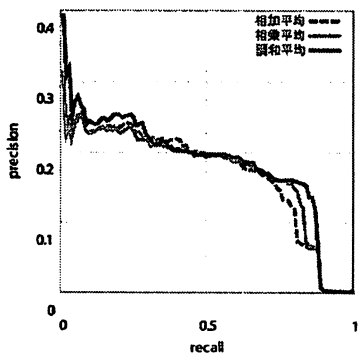

図 3 精度の実験結果

横軸に再現率，縦軸に精度をとっている．したがって，グラフの左側ほどランキングが上位である部分文書に対する精度をプロットしている．このグラフからは，調和平均，相乗平均，相加平均の順に精度が高いことが分かる．このことから，複数の条件に対する評価値の統合関数が検索精度を左右することが確認できた．

さらに，同じ問合せに対して，他の研究との精度比較を行ったグラフが図 4 にある．グラフの細い線が他の研究グループのシステムが出した実験結果を表している．本研究は，他の研究と比べて上位に位置する高い精度での検索が可能であることが分かる．

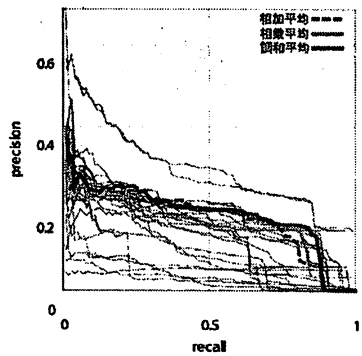

図 4 他の研究との精度の比較

5.3 速度の評価実験

本節では提案した手法が検索速度の向上に対して有効であることを実験から確認する．検索語のみによる問合せである INEX の CO の問合せ 40 個と検索語と構造の両方を指定した INEX の CAS の問合せ 26 個を対象として行った．問合せの評価尺度は実行時間である．Token 表を語の種類によって分け，Element 表を拡張した本手法適用後と手法適用前で比較を行う．CO と CAS の問合せ実行時間の結果をそれぞれ図 5 と図 6 に示す．また，両方の図とも縦軸は問合せ実行時間を底を 10 とした対数軸でとっている．CO,CAS いずれの問合せにおいても本手法を適用した方が問合せ実行時間が短いことが確認できる．CO,CAS で平均ではそれぞれ約 15.1 倍，約 26.4 倍の速度向上ができた．CAS の中には Q10 のように問合せ実行時間が 1 分以上かかったものがある．これは構造の指定が"//bdy//*"である問合せで，タグ名に関する条件を使う事ができないため，1 分以上かかったと考えられる．

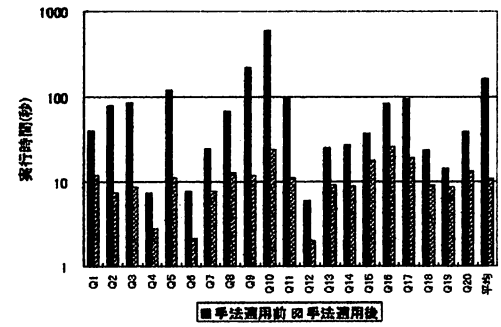

図 5 CO の検索速度の実験結果

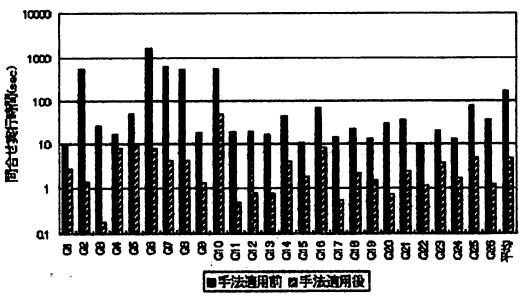

図 6 CAS の検索速度の実験結果

6. ま と め

本研究では，内容と構造を指定した検索に対応するための手法について提案した．中でも，複数の条件を指定した問合せに対して，各条件の評価値をどのように統合するかが検索精度を向上するために重要であると考えた．そのため，複数の統合関数に対して比較実験を行うことで，統合関数が検索精度に影響を与えることを確認した．また，今回の実験では統合関数として調和平均を用いると精度が良いことを確認した．

速度に関しても，Token 表を分けることと，Element 表を拡張して結合の数を減らし，タグ名属性の追加により，経路表現の条件による選択を高速化する手法を提案した．さらに，実験から提案した手法により，検索語のみを指定した問合せで平均約 15.1 倍，内容と構造を指定した問合せにおいて，平均で約 26.4 倍の速度向上ができたことを確認した．

今後の課題としては以下のことが考えられる．

- 他の統合関数での実験

本研究では，統合関数として相加平均，相乗平均，調和平均を対象として実験を行った．しかし，今後は他の統合関数を用いてさらに実験を行うことで，より適切な統合関数を探したいと考えている．例えば，その関数の一つとして PRO 関数を考えている．

全体での評価値を S, 各条件の評価値を s_i とする．このとき，PRO 関数は以下の式で表される．

$$S = 1 - (1 - s_1)(1 - s_2) \cdots (1 - s_n)$$

この式では，$s_i (i = 1, 2, ..., n)$ は 0 以上 1 以下の値であることを前提としている．そのため，PRO 関数を統合関数として用いる時は，評価値のとりうる最低の値を 0，最高の値を 1 にするよう正規化を行う必要がある．

- あいまいな構造の指定をする検索

文書志向の XML 文書においては，構造を指定した検索において構造に関する条件よりも検索語との関連度の方が重要な場合がある．厳密には構造の条件を満たしていないが，検索語との関連は強い部分文書を回答とすることで，あいまいな構造の指定をする検索に対応したいと考えている．

文 献

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

[2] Kei Fujimoto, Toshiyuki Shimizu, Kenji Hatano, Toshiyuki Amagasa, Hiroko Kinutani, Masatoshi Yoshikawa: "An Implementation of High-Speed and High-Precision XML Information Retrieval System on Relational Databases," Initiative for the Evaluation of XML Retrieval 2005 (INEX2005)

[3] Torsten Grabs, and Hans-Jorg Schek: ETH Zurich at INEX: Flexible Information Retrieval from XML with PowerDB-XML, INEX Workshop 2002, pp.141-188 (2002).

[4] F. Liu, C. Yu, W. Meng, A. Chowdhury: Effective Keyword Search in Relational Databases, Proc. of the 2006 ACM SIGMOD international conference on Management of data, Chicago, USA, June 2006.

[5] 鈴木 優，波多野 賢治，吉川 正俊，植村 俊亮: 複数のメディアで構成された電子文書の検索手法，情報処理学会論文誌: データベース，Vol.42, No.SIG10 (TOD 11), pp.11-21(2001)

[6] INitative for the Evaluation of XML Retrieval (INEX) 2005. http://inex.is.informatik.uni-duisburg.de/2005/