DEWS2006 6B-i10

XML 情報検索システムとその高速化に関する研究

藤本　圭[†]　清水　敏之[†]　寺田　憲正[†]　波多野賢治[††]　鈴木　優[†††]
天笠　俊之[††††]　絹谷　弘子[†††††]　吉川　正俊[†]

† 名古屋大学 情報科学研究科
†† 奈良先端科学技術大学院大学 情報科学研究科
††† 立命館大学 情報理工学部
†††† 筑波大学 システム情報工学研究科
††††† お茶の水女子大学 総合情報処理センター
E-mail: {†fujimoto, shimizu, terada}@dl.itc.nagoya-u.ac.jp, ††hatano@is.naist.jp †††suzuki@ics.ritsumei.ac.jp
††††amagasa@cs.tsukuba.ac.jp †††††kinutani@edu.cc.ocha.ac.jp †yosikawa@is.nagoya-u.ac.jp

あらまし　本論文では XRel を基礎として開発を進めている XML 情報検索システムについて述べる．この XML 情報検索システムはベクトル空間モデルに基づいて実装されており，検索語による検索に対応している．XRel は我々が開発している関係データベースに基づく XML データベースである．XML 文書の検索を行う場合，一つの XML 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．全ての部分文書のデータを保持することはデータの肥大化につながり，検索精度や検索速度の低下を招く．既存の手法では検索対象はシステムを構築する際に手動で設定していたが，システム構築時の負担が大きくなる．そこで本論文では検索語を用いた XML 文書検索において検索対象として適切な XML 部分文書をピリオド率，異なり語数，経験則を用いて選択する手法を提案する．開発した XML 情報検索システムを用いた評価実験の結果，提案手法により検索精度と検索速度が同時に改善されることが確認できた．

キーワード　XML, 情報検索, 問合せ処理, XML データベース

A Study on XML Information Retrieval System and Its Speeding Up

Kei FUJIMOTO[†], Toshiyuki SHIMIZU[†], Norimasa TERADA[†], Kenji HATANO[††], Yu SUZUKI[†††], Toshiyuki AMAGASA[††††], Hiroko KINUTANI[†††††], and Masatoshi YOSHIKAWA[†]

† Graduate School of Information Science, Nagoya University
†† Graduate School of Information Science, Nara Institute of Science and Technology
††† College of Information Science and Technology, Ritsumeikan University
†††† Graduate School of Systems and Information Engineering, University of Tsukuba
††††† Information Media and Education Square, Ochanomizu University
E-mail: {†fujimoto, shimizu, terada}@dl.itc.nagoya-u.ac.jp, ††hatano@is.naist.jp †††suzuki@ics.ritsumei.ac.jp
††††amagasa@cs.tsukuba.ac.jp †††††kinutani@edu.cc.ocha.ac.jp †yosikawa@is.nagoya-u.ac.jp

Abstract　This paper describes an XML information retrieval system that we have developed. It is based on a vector space model, and implemented on top of XRel, a relational XML database system that has been developed in our research group. When a query is processed, a large number of fragments are retrieved, because a single XML document usually contains many XML fragments. Keeping all XML fragments degrades precision and increases query processing time, because some XML fragments are not appropriate as a query target. In existing methods, retrieval targets are manually selected by human experts when an XML collection is stored in the system. Such manual selection is not feasible when many kinds of XML documents are stored in the system. To cope with the problem we propose a method for automatically selecting document-centric fragments by introducing three measurements, namely, period ratio, number of different words, and empirical rules. By deleting inappropriate data-centric fragments from results of keyword query, we can improve the accuracy and performance of our system. Through performance evaluations, we confirmed the improvement of retrieval precision and query processing speed.

Key words　XML, Information Retrieval, Query Processing, XML Database

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．記述された 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，検索の答えは明確に定義できることが多い．文書は 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，検索の答えは問合せとの類似度のような尺度と共に返されることが多い．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な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．我々が開発を進めている，関係データベースにもとづく XML データベース XRel [4] 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な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もとづいて実装した．

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と考えられる．例えば，論文を構造化した文書において，表題や著者といった部分文書はデータ志向であり，それ単体では検索語による検索結果として不適切である．一方，章や節，段落といった部分文書は文書志向であり，検索語による検索が有効である．

構造化された文書の検索を行なう場合，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．従来の研究では検索語による検索対象として適切である文書志向の部分文書の選択は手動で行われており，システム構築時の負担が大きかった．そこで，本稿では XML 文書から得られる統計値をもとに検索語による検索対象として不適切であるデータ志向の部分文書を排除し，文書志向の部分文書を自動的に選択する手法を提案する．統計値としてピリオド率，異なり語数を用い，さらに図や表を削除するといった経験則を用いて文書志向の部分文書を選択する．

以下，第 2. 節では関連研究について述べる．第 3. 節では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記述し，第 4. 節では文書志向部分文書の選択手法について述べる．第 5. 節では手法の適用例について記す．第 5. 3 節では文書検索の高速化手法についての性能評価を行なう．第 6. 節ではまとめと今後の課題について述べる．

2. 関連研究

1. 節で述べたとおり，XML 文書にはデータ志向の XML 文書と文書志向の XML 文書が存在する．XML 文書に対する検索は当初はデータ志向の XML 文書を対象としており，XQuery [2] や XPath [3] などの XML 問合せ言語を利用したデータベース的アプローチによる検索が主流であった．しかし，文書志向の XML 文書が広がるにつれ，データベース的アプローチによる検索では XML 部分文書を効果的に検索することができなくなってきた．そのような背景の下，XML 文書の検索に対し情報検索技術，特に検索語による検索技術の適用が求められており，そのような技術を適用した XML 文書検索に関する研究が行われている [7].

Hayashi らの研究 [8] では XML 文書検索では文書構造と文書内容に基づく順序付け機能が重要であるとし，検索対象となる部分文書をあらかじめ設定した上で索引を生成し，構造指定を伴った重みつき検索語による検索を実現している．Amer-Yahia の研究では，現在，主流となりつつある XML 問合せ言語 XQuery に情報検索的な機能を付加し，XML 文書検索を効果的に行えるように TeXQeury [9] や FlexPath [10] を提案している．

これらの研究は XML 部分文書と検索語による問合せとの類似度を計算する枠組みを提供しているに過ぎない．検索対象とする部分文書はシステムの設計者が経験則に従って決定するか，その設定がなければ全ての部分文書を検索対象とすることになるが，以下のような問題点が存在する．経験則により検索対象とする部分文書を決定する場合，検索対象となる文書群の文書型定義が複数にまたがるとシステム設計者の負担は膨大なものとなり，本来，検索対象とすべき部分文書を検索対象から除外してしまう危険性がある．検索対象とする部分文書を絞り込まない場合，データの肥大化による検索精度の低下や問合せ処理速度の低下が発生する．これらは検索語による XML 部分文書検索において，どのような粒度の部分文書を検索対象とすべきかを決定する仕組みが現状では定まっていないことによる．

そこで本論分では従来この分野で行われてきた研究では議論されなかった，検索語による検索対象として適切な XML 部分文書を決定する方法を提案する．本手法では検索語による XML 部分文書検索の解答として適切な部分文書を XML 部分文書から得られる統計値により機械的に決定する．これにより検索対象とすべき部分文書が半自動的に決まるため，検索語による XML 部分文書検索システムの設計が容易となる．

また従来の情報検索分野においては検索精度の向上が重要視されてきたが，膨大な数の XML 部分文書を検索する場合には検索精度と同時に検索速度も追求しなければならない．本手法においては検索対象として不適切な部分文書を除外するため，検索精度の向上や問合せ処理速度の向上が期待される．

3. 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．

3.1 XRel

XRel は我々が開発している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

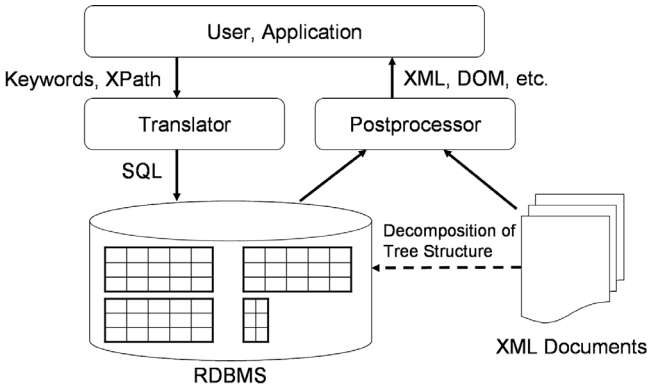

図 1　XRel アーキテクチャの概要

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 は検索語や XML 問合せ言語による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 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．XRel のアーキテクチャのうち問合せインタフェースの部分の概要を図 1 に示す．図 1 において，システム側は利用者や応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 (documentID, nodeID, pathID, start, end)
キーは (documentID, nodeID)

Attribute (documentID, nodeID, pathID,
                              start, end, value)
キーは (documentID, nodeID)

Text (documentID, nodeID, pathID,
                              start, end, value)
キーは (documentID, nodeID)

Path (pathID, pathexp)
キーは (documentID, nodeID)
```

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．

documentID	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に付与した通し番号.
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	Attribute では属性値を，Text では要素内の文字列をそれぞれ格納する.
pathexp	実際の経路表現を格納する.

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

図 2　XML 文書の例

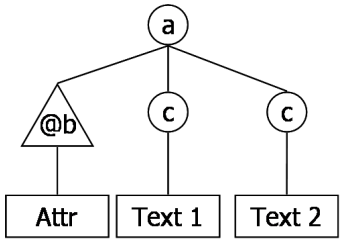

図 3　XML 木の例

表 1　図 3 の XML 文書を格納した例

Element

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

Attribute

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

Text

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

Path

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

次に XML 文書の格納例を挙げる．図 2 の XML 文書を木構造で表したものが図 3 である．図 3 の XML 文書を格納すると表 1 のようになる．

3.2　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(documentID, nodeID, token, tfipf)
キーは (documentID, nodeID, token)
```

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(documentID 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 を用いる．TF-IPF は TF(Term Frequency) と IPF(Inverse Path Frequency) [5] の積である．TF はある部分文書における単語の出現頻度である．文書長に応じて正規化を行なう．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(Inverse 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．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 は以下の式で計算される．

$$ipf_p(t) = \log \frac{N_p}{df_p(t)} + 1$$

3.3 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 木中での経路を指定する 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．

3.3.1 検索語による問合せの変換

入力された検索語 `keyword 1,keyword 2,...,keyword n` は以下のような SQL に変換される．

```
SELECT * FROM (
  SELECT t.documentID, t.nodeID,
         e.start, e,end, SUM(t.tfipf) score
  FROM   Token t, Element e
  WHERE  t.documentID  = e.documentID
  AND    t.nodeID = e.nodeID
  AND    t.token in ('keyword 1', 'keyword 2',
                     ..., 'keyword n')
  GROUP BY t.documentID, t.nodeID,
           e.start, e.end ) r
ORDER BY r.score DESC
```

各部分文書における `keyword 1,keyword 2,...,keyword n` の TF-IPF の和を計算し，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．

3.3.2 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 を以下に示す．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.documentID, 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 “//c” を変換した SQL を以下に示す．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．

```
SELECT e.documentID, e.nodeID, e.start, e.end
FROM   Element e, Path p
WHERE  e.pathID = p.pathID
AND    p.pathexp LIKE '%/c'
```

述部において条件が指定された場合，XPath で記述された条件は SQL の WHERE 句における条件に適切に変換される．“/a[@b=”ATTR”]” を変換した SQL を以下に示す．[@b=”ATTR”] は Attribute 表を用いた適切な条件に変換されている．

```
SELECT e.documentID, e.nodeID, e.start, e.end
FROM   Element e, Path p1, Attribute a, Path p2
WHERE  e.pathID = p1.pathID
AND    p1.pathexp LIKE '/a'
AND    a.pathID = p2,pathID
AND    p2.pathexp LIKE '/a/@b'
AND    a.value = 'Attr'
AND    e.start < a.start
AND    e.end   > a.end
```

4. 文書志向部分文書の選択手法

XML 部分文書を検索対象とする場合，その部分文書数は膨大なものとなる．そのため，データの肥大化による検索精度の低下や問合せ処理速度の低下が発生する．そこで検索語による XML 部分文書検索の解答として適切な部分文書を XML 部分文書から得られる統計値により機械的に決定する．

構造化された文書においては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．例えば，論文を構造化した文書において，表題や著者といった部分文書はデータ志向であり，それ単体では検索語による検索結果として不適切である．一方，章や節，段落といった部分文書は文書志向であり，検索語による検索が有効である．したがって，データ志向の部分文書を検索語による検索対象として不適切であるとし，文書志向の部分文書を検索語による検索対象として適切であるとする．

そこで，XML 文書群中の全ての部分文書の中から検索語による検索対象とすべき文書志向の部分文書を選択する手法について述べる．経験則によってあらかじめ検索対象となる部分文書を設定する手法では，検索語による検索対象とすべき多くの部分文書を検索対象から外してしまう危険性をはらんでいる．そこで，本研究では文書志向部分文書の選択を統計量に基づき，検索語による検索対象として不適切な部分文書（以下，不要部分文書と記述する.）を削除することにより行なう．削除は以下の三段階で行なう．

（1） ピリオド率を用いたスキーマレベルでの不要部分文書の削除

（2） 異なり語数を用いたインスタンスレベルでの不要部分文書の削除

（3） 経験則による不要部分文書の削除

不要部分文書の削除により，保持するデータ量の削減，処理

データ量の削減による問合せ処理速度の向上，不要部分文書の削除による精度/再現率の向上が期待される．以下，各段階について詳しく述べる．

4.1 ピリオド率による削除

一般的に文章の末尾は “.” または “?” または “!” で終了するため，内容が “.” または “?” または “!” で終了する部分文書を文書志向の部分文書であると判断する．ただし，インスタンスレベルでは文書志向の部分文書であるにもかかわらず文末が “.” または “?” または “!” で終了しないものや，データ志向の部分文書であるにもかかわらず文末が “.” または “?” または “!” で終了するものが存在する．そこで，文書志向かどうかの判断の単位をタグ名ごととする．あるタグ名の文書集合に対し “.” または “?” または “!” で終了する部分文書の割合が多ければ，そのタグ名を持つ部分文書は文書志向であると判断する．

タグ名ごとにそのタグ名を持つ要素が文書志向かどうかを判断するためにピリオド率を用いる．ピリオド率は次のように定義する．あるタグ名 t の要素集合の数を D_t とする，D_t に含まれる要素のうち，葉ノードに関しては文末が “.” または “?” または “!” で終了するような内容を持つ要素の集合の数，内部ノードに関しては前述の葉ノードを子孫要素として一つ以上持つ要素の集合の数を N_t とすると，タグ名 t の部分文書のピリオド率 r_t は以下の式で計算できる．

$$r_t = \frac{N_t}{D_t}$$

ピリオド率が低い部分文書集合はデータ志向であるとみなし，検索語による検索対象である部分文書集合から削除する．具体的には Element 表と Token 表から該当する行を削除する．削除のためのしきい値は文書集合から得られた統計値により決定する．

4.2 異なり語数による削除

ピリオド率による削除によりデータ志向が強いタグ名の部分文書が削除され，文書志向が強い，あるいは文書志向とデータ志向が混ざっているタグ名の部分文書が残る．文書志向とデータ志向が混ざっているタグ名の部分文書に対して，インスタンス単位で文書志向かどうかの判断を行う．

ここでは部分文書に含まれる単語のうち，異なる単語の数（以下，異なり語数と記述する.）を用いる．異なり語数が少ない部分文書はデータ志向であるとみなし，検索語による検索対象である部分文書集合から削除する．具体的には Element 表と Token 表から該当する行を削除する．削除のためのしきい値は文書集合から得られた統計値により決定する．

4.3 経験則による削除

以上の処理を行った後，文書志向の強い部分文書が残る．しかし文書志向の強い部分文書でも検索対象として不適切なものが存在する．

例えば，表内に文章が書かれていた場合，それは文書志向が強いものとして判断されるが，表あるいは表の一つの項目のみが検索結果として表示されることは検索結果として不適切である．表はその説明を行う文章と共に出力されるべきである．

そこで経験則から削除が妥当だと考えられる部分文書を削除する．これは情報検索におけるストップワードの概念を語から部分文書が属する経路に置き換えたものである．このような経路を Path 表から削除し，その経路に存在する部分文書を Element 表から，その部分文書に属する語を Token 表から削除する．

5. 実　　験

本節では実験に使用するテストコレクションに対し，本手法を適用した場合について述べる．実験で使用するテストコレクションは INEX-1.9 とする．

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 relevance assesments 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 形式で構造化してものから構成されている．INEX-1.8 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 1600 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 は検索の答えとなる文書構造を明示的に指定する．検索の答え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．

CO, CO+S サブタスクでは利用者の検索行動を仮定して Focussed, Through, FetchBrowse と呼ばれる三つの検索方策を定義している．Foccused 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えとする．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なう．FetchBrowse では検索を二段階に分けて行なう．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 は後述する CO サブタスクと CO+S サブタス

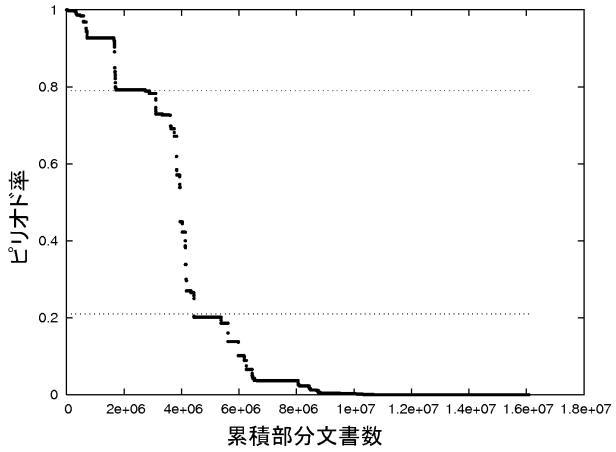

図 4　ピリオド率の推移

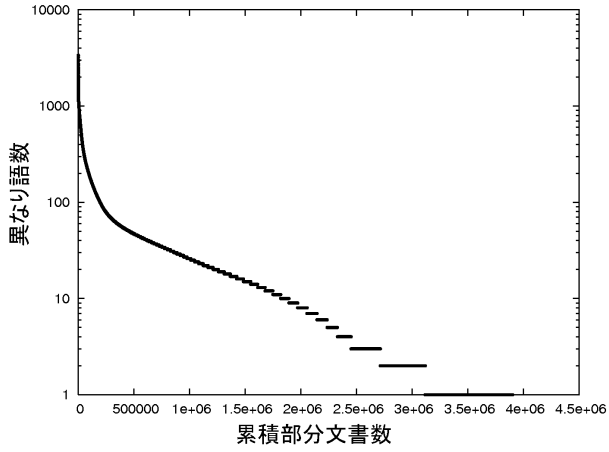

図 5　異なり語数とその異なり語数を持つ部分文書の数の関係

クで共通したものが 40 個，CAS サブタスクで 40 個用意されている．

INEX relevance assesments 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 exhaustiveness 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．

5.2　INEX-1.9 に対する手法の適用

本節では INEX-1.9 テストコレクションに対して，ピリオド率を用いたスキーマレベルでの不要部分文書の削除，異なり語数を用いたインスタンスレベルでの不要部分文書の削除，経験則による不要部分文書の削除を適用した場合について記述する．

5.2.1　ピリオド率による削除の適用

INEX-1.9 テストコレクションに対しピリオド率の計算を行った．各部分文書をピリオド率の降順に並べた際のピリオド率の推移を表したグラフを図 4 に示す．

図 4 から部分文書は大きく三つのグループに分かれていることがわかる．ピリオド率が 1 から約 0.8 の値を取る文書的な性質を示す部分文書のグループ，ピリオド率が約 0.2 から 0 の値を取るデータ的な性質を示す部分文書のグループ，ピリオド率が約 0.8 から約 0.2 の値を取る両者の中間的な性質を示す部分文書のグループである．

文書的な性質を示す部分文書のグループと両者の中間的な性質を示す部分文書のグループを残し，データ的な性質を示す部分文書のグループを削除した．削除のためのしきい値はピリオド率 0.21 である．これにより約 1600 万件存在した部分文書は約 440 万件に減少した．

5.2.2　異なり語数による削除の適用

ピリオド率による削除を行った部分文書集合に対し異なり語数の計算を行った．異なり語数と，その異なり語数を持つ部分文書の数の関係を示すグラフを図 5 に示す．

図 5 から部分文書は大きく二つののグループに分かれていることがわかる．少数の異なり語を持つ大多数の部分文書のグループと多数の異なり語を持つ少数の部分文書のグループである．

表 2　各手法による削除部分文書数

手法	削除部分文書数
ピリオド率単体	11,645,300
異なり語数単体	15,220,278
経験則単体	2,780,569
ピリオド率と異なり語数の重複	11,641,769
異なり語数と経験則の重複	2,761,245
経験則とピリオド率の重複	2,473,584
全手法の重複	2,471,852

少数の異なり語を持つ大多数の部分文書のグループを削除した．削除のためのしきい値は異なり語数 30 である．これにより約 440 件存在した部分文書は約 85 万件に減少した．

5.2.3　経験則による削除の適用

ピリオド率と異なり語数を利用して削除を行った部分文書集合の中には文書的な性質を持つが，検索語による検索結果としては不適切なものも存在する．例えば表の一つの項目などである．

ここで、図，表，数式にあたる部分文書とその子孫部分文書を削除することとする．図，表，数式を内に含む部分文書は削除されないことに注意する．これにより約 85 万件存在した部分文書は約 84 万件に減少した．

以上の三段階の削除を実行することにより，約 1600 万件存在した部分文書は約 84 万件になった．

各手法を単体で適用したときの削除部分文書数と，各手法で削除する部分文書のうち重複するものの数は表 2 の通りである．また，削除された部分文書の合計数は 15,241,401 である．

5.3　評価実験

本章では統計量に基づく文書志向部分文書の選択手法の性能評価を行なう．手法適用前の文書ベクトルと手法適用後の文書ベクトルを用いて実験を行い，精度/再現率と検索速度の比較を行なう．

使用するテストコレクションは INEX-1.9 とする．性能評価

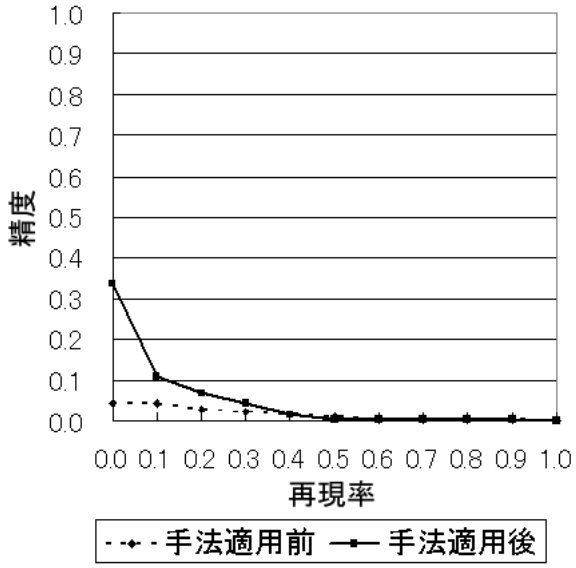

図 6　検索精度の比較

は CO.Through タスクにより行なう．構築した XML 情報検索システムに対して検索語集合の入力により検索を行い，全部分文書に対して順位付けを行なうことにより CO.Through タスクを実行する．

実験環境は次の通りである．CPU: Intel Xeon 2GHz (Dual), Memory: 2048 MB, OS: Asianux 1.0, DBMS: Oracle 10g Release 1.

5.3.1　検索精度の評価

手法適用前の文書ベクトルと手法適用後の文書ベクトルを用いて精度/再現率を比較し，手法適用後に精度/再現率が向上することを確認する．順位付けられた部分文書の上位 1500 件に対して精度/再現率を計算した．これは INEX にて定められていることによる．高速化手法適用前の文書ベクトルと高速化手法適用後の文書ベクトルをそれぞれ用いた場合の精度/再現率を図 6 に示す.

図 6 から手法適用前の部分文書群に対する検索精度が低いことが分かる．これは検索語による検索対象として不適切な文書を多く保持しているためである．一方，手法適用後の部分文書群に対する検索精度が再現率が高い部分，すなわち検索結果の上位においてで顕著に向上していることが確認できる．本手法により検索語による検索対象として適切な部分文書を選択できたことが分かる．

5.3.2　問合せ処理速度の評価

手法適用前の文書ベクトルと手法適用後の文書ベクトルを用いて問合せ処理時間を比較し，手法適用後に問合せ処理速度が向上するを確認する．高速化手法適用前の文書ベクトルと高速化手法適用後の文書ベクトルをそれぞれ用いた場合の問合せ処理時間を図 7 と図 8 に示す.

図 7 と図 8 から全ての問合せにおいて，手法適用前の部分文書群よりも手法適用後の部分文書群に対する検索速度が向上していることが確認できた．最大で約 60 倍に速度が向上している．検索対象とする部分文書の減少により問合せ処理が効率化されたことが確認できた.

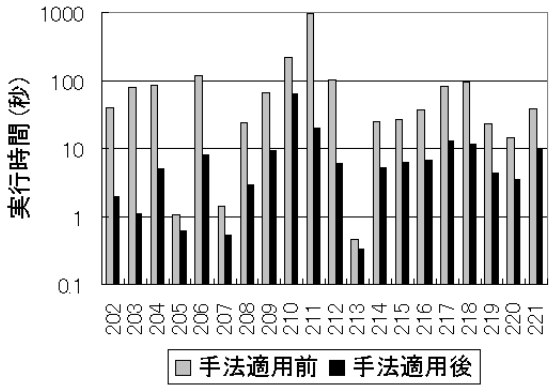

図 7　トピック 202 からトピック 221 の問合せ処理時間の比較

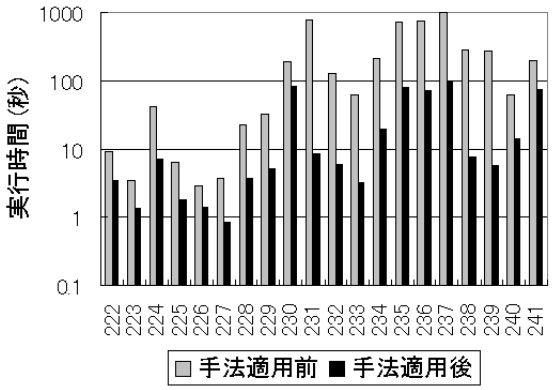

図 8　トピック 222 からトピック 241 の問合せ処理時間の比較

5.3.3　削除判定率

前述の実験結果から本手法が検索精度の向上と検索速度の向上に有効であることが確認できたが，削除した文書中に解答となる部分文書が含まれている可能性が残されている．そこで，削除した部分文書中に解答となる部分文書がどれだけ含まれているかを調べ，解答部分文書のうち本手法により削除された部分文書の割合を計算した．この割合を削除判定率と呼ぶこととする．削除判定率 D は以下の式で計算できる.

$$D = \frac{\text{削除された解答部分文書の数}}{\text{全解答部分文書の数}}$$

INEX-1.9 に対する削除判定率は表 3 の通りである．表 3 からトピック 207, 222, 232, 233, 234, 241 において特に多くの Exhaustivity が 2 である部分文書が削除されていることが分かる．これらの文書ではイタリック体やリストの項目といった，本研究においてデータ志向であるとみなした部分文書の Exhaustivity を 2 としている．本手法ではデータ志向の部分文書は網羅性が低いとみなし，検索対象から削除したため，このような結果になったと考えられる.

6. おわりに

本稿では XML 情報検索システムの実装と検索語による検索対象として適切な部分文書の選択手法について述べた.

表 3　削除判定率

	Exhaustivity		
トピック	2	1	2&1
202	0.000	0.214	0.077
203	0.000	0.144	0.140
205	N/A	0.215	0.215
206	0.133	0.347	0.262
207	0.600	0.326	0.441
208	0.070	0.199	0.185
209	0.000	0.053	0.048
210	0.047	0.171	0.153
212	0.000	0.813	0.705
213	0.000	0.262	0.234
216	0.060	0.129	0.109
217	0.000	0.048	0.037
218	0.000	0.253	0.252
219	0.000	0.132	0.121
221	0.049	0.316	0.186
222	0.450	0.154	0.284
223	0.094	0.080	0.084
227	0.022	0.067	0.059
228	0.000	0.340	0.328
229	0.000	0.078	0.067
230	0.091	0.036	0.048
232	0.744	0.450	0.575
233	0.150	0.231	0.196
234	0.224	0.718	0.483
235	0.090	0.199	0.160
236	0.029	0.278	0.263
237	0.010	0.269	0.246
239	0.105	0.042	0.050
241	0.339	0.278	0.324
Total	0.215	0.348	0.314

XRel は関係データベースに基づく XML データベースであり，固定したスキーマを持つ四つの表に XML 文書を格納する．さらに文書ベクトルを格納することにより，検索語による XML 文書検索に対応している．XML により構造化された文書は部分文書単位で検索を行なうことができる．

部分文書を検索対象とする場合，全ての部分文書のデータを保持すると，データの肥大化により検索精度と検索速度が低下する．部分文書集合の統計量を用いて検索語による検索対象として適切な部分文書を選択し，検索対象として不適切な文書を除く手法を提案した．提案手法を用いることにより検索精度と検索速度の向上が確認できた．

今後の課題には次のようなものが挙げられる．

- 単語の重み付けに文書構造の情報を積極的に取り入れることで，精度/再現率の一層の改善を目指す．

本稿では単語の重み付けに TF-IPF を利用した．伝統的な手法により正規化した TF 値は文書長が短いほど値が大きくなる．したがって部分文書群に対して TF 値を計算すると文書長の短いデータ志向の部分文書の TF 値が高くなってしまうと言う問題がある．XML 文書の文書構造の情報を考慮に入れた重み付け手法を考案する必要がある．

- 検索システムの利用者から検索結果とする経路が指定された場合について，その順位付け手法と効率的な問合せ処理を考案する．

本稿では検索システムの利用者が文書構造に対する知識を持っていない場合について考え，問合せが検索語集合であることを前提とした．検索精度と検索速度の向上を達成するために検索語による文書検索の特性からデータ志向の部分文書を検索対象から除いた．しかし，検索条件としてデータ志向の部分文書が指定されることがあり得る．例えば “/article[title=”A Study on XML Document Retrieval and its Application System”]” という XPath 問合せである．このように経路が指定された場合についても部分文書の順位付け手法と効率的な問合せ処理手法を考案する必要がある．

文　献

[1] W3C. Extensible Markup Language (XML). 1996-2003. http://www.w3.org/XML/

[2] W3C. XQuery 1.0: An XML Query Language. 2004. http://www.w3.org/TR/xquery/

[3] W3C. XML Path Language (XPath) Version 1.0. 1999. http://www.w3.org/TR/xpath/

[4] Masatoshi Yoshikawa, Toshiyuki Amagasa, Takeyuki Shimura, and 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).

[5] Torsten Grabs, and Hans-Jorg Schek: ETH Zurich at INEX: Flexible Information Retrieval from XML with PowerDB-XML, INEX Workshop 2002, pp. 141-148 (2002).

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

[7] Kenji Hatano, Hiroko Kinutani, Toshiyuki Amagasa, Yasuhiro Mori, Masatoshi Yoshikawa, and Shunsuke Uemura: Analyzing the Properties of XML Fragments decomposed from the INEX Document Collection, Advances in XML Information Retrieval, Lecture Notes in Computer Science, Volume 3493, pp. 168-182, Springer-Verlag, May 2005.

[8] Yoshihiko Hayashi, Junji Tomita, and Genichiro Kikui: Searching Text-rich XML Documents with Relevance Ranking, http://www.haifa.il.ibm.com /sigir00-xml/final-papers/Hayashi/hayashi.html, In ACM SIGIR 2000 Workshop on XML and Information Retrieval (2000).

[9] Sihem Amer-Yahia, Chavdar Botev, and Jayavel Shanmugasundaram: TeXQuery: A Full-Text SEarch Extension to XQuery, Proc. of 13th International World Wide Web Conference, ACM Press, pp. 538-594 (2004).

[10] Sihem Amer-Yahia, Laks V. S. Lakshmanan, and Shashank Pandit: FleXPath: Flexible Structure and Full-Text Querying for XML, Proc. of the 2003 ACM SIGMOD International Conference on Management of Data, ACM Press, pp. 83-94 (2004).