# 部分文書クラスタリングを用いたXML情報検索の効率化

吉井 迪利† 宮崎 純† 波多野 賢治‡ 加藤 博一†

†奈良先端科学技術大学院大学 情報科学研究科 〒630-0192 奈良県生駒市高山町 8916-5
‡同志社大学 文化情報学部 〒610-0394 京都府京田辺市多々羅都谷 1-3
E-mail: †(michitoshi-y, miyazaki, kato)@is.naist.jp, ‡khatano@mail.doshisha.ac.jp

**あらまし** 本研究ではXML文書の部分文書をクラスタリングすることにより、その検索性能を上げることを目指す。近年、大容量のXML文書、例えばWikipedia等が増えてきている。これらのXML文書を上手く利用するために、文書の内容だけでなく、XMLの文書構造も考慮した部分文書に関する様々な検索手法が考えられている。しかしその問題点として、部分文書を対象にすると検索対象が膨大になってしまう点が挙げられる。そこで本手法により、部分文書検索におけるクラスタリングの有効性を示す。

**キーワード** クラスタリング、XML、情報検索

# A Study on XML Information Retrieval Using Partial Document Clustering

Michitoshi Yoshii† Jun Miyazaki† Kenji Hatano‡ Hirokazu Kato†

† Nara Institute of Science and Technology Graduate School of Information Science
8916-5 Takayama, Ikoma, Nara 630-0192
‡ Doshisha University Faculty of Culture and Information Science
1-3 Tatara Miyakodani, Kyotanabe City, Kyoto 610-0394
E-mail: †(michitoshi-y, miyazaki, kato)@is.naist.jp, ‡khatano@mail.doshisha.ac.jp

**Abstract:** We aim at reducing the processing cost in XML information retrieval that can search XML partial documents as well as whole documents by clustering XML partial documents. ecently, various kinds of XML documents are increasing. In particular, very large XML documents, such as Wikipedia, become the target of XML information retrieval. In order to improve the precision of retrieved results, various methods have been proposed, taking account of the structure of XML documents as well as their contents. However, if we consider partial documents, the processing cost would be high because the search space becomes huge. To cope with this problem, we adopt a document clustering method to XML partial documents and combine it with our XML search system to reduce the processing cost. Therefore, we explore an effect of document clustering method in XML information retrieval.

**キーワード** Clustering、XML、Information Retrieval

## 1. はじめに

近年、タグを自由に決定でき、柔軟な文書構造をとることが出来るXML文書が増えてきている。その中でも特に大規模のXML文書の利用方法に関する研究が注目されている。大規模XML文書の一例として、Wikipedia[1]の文書データがある。

XML文書が大規模になるにつれ、全文書に対する検索などの処理コストが非常に高くなることが問題とされている。また、XML文書は構造化文書であることから、文書の内容(Content)のみならず、文書の構造(Structure)も考慮した部分文書単位での検索処理を行うことが可能である。本研究においては、1つのXML文書に含まれる部分文書も検索対象として考える。部分文書単位での検索を行うことにより、そのXML文書を利用するユーザの情報要求に対する適合部分を抽出することが可能となる。しかしその反面、部分文書単位で検索処理を行うと検索対象がさらに膨大になり、従来のXML文書検索よりも処理コストが増大する原因になってしまう。

そこで、本研究では部分文書単位のXML文書を文書の内容でクラスタリングすることにより、検索対象となる部分文書の数を制限し、検索にかかる処理コストを軽減させる。その結果として検索精度の低下を抑え、速度面を向上させることから、検索性能の向上を目指す。以降で、部分文書検索におけるクラスタリングの有効性を示していく。

## 2. 準備と関連研究

まず、本論文で使用する用語や関連する研究などを紹介する。

はじめに、今回利用するデータセットは、INEX2007[2]のデータコレクションである Wikipedia の XML 文書データを使用する。文書に関するパラメタは表 1 のようになっており、XML 木の構造例として図 1 のようなものがある。

XML 文書検索を行う手法には様々なアプローチが存在しているが、本手法では XRel[3]を利用し関係データベース上に XML 文書を格納する。

表 1. XML 文書(Wikipedia)に関するパラメタ

| | |
|---|---|
| データ容量 | 4.38GB |
| 文書数 | 659,388 |
| 部分文書数 | 52,562,497 |
| 文書内総単語数 | 518,106,956 |
| 文書内の単語種類数 | 2,043,112 |

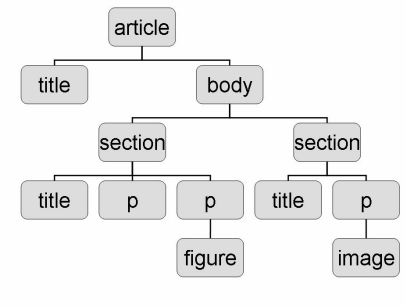


図 1． Wikipedia の XML 木例

次に XML 文書に対する問い合わせに関して説明する。XML 文書問い合わせの方法には 2 種類ある。1 つはキーワードから検索を行い、部分文書集合を出力する CO(Content Only)問い合わせ、もう 1 つはキーワードと文書構造から検索を行い、部分文書集合を出力する CAS(Content And Structure)問い合わせがある。その中でも CAS 問い合わせでは XPath の拡張である NEXI[4]を用いる。本論文では、図 2 (INEX2007 CAS query 415)のような CAS 問い合わせのみについて考える。

しかしながら、NEXI の問い合わせ形式を関係データベースに格納された XML 文書にそのまま適用することは不可能であるため、NEXI 問い合わせを SQL に変換する必要がある。清水ら[5]は、XRel を利用して XML 文書を関係データベースに格納し、NESI 問い合わせを SQL に変換する手法を提案している。
この研究の特長として、特に CO 問い合わせに対して高い精度が出ているが、問題点として部分文書ごとの単語を登録してあるテーブルのサイズが非常に大きく、問い合わせ時に検索対象が多くなってしまう点がある。

## 3. 提案手法

従来手法の問題点として、単語テーブルのサイズが非常に大きく、検索対象となるデータ量が多くなってしまうことがある。その解決策として本研究では部分文書に対してクラスタリングを行う。クラスタリングを行う理由として、次の 2 点がある。1 つはキーワードなどの検索要素から検索対象とするクラスタを決定することで、検索対象を減らすことが出来る点である。もう 1 つの点として、同じクラスタ内には、部分文書の特徴が類似したものが格納されているため、検索対象を減らしても検索精度が落ちにくい点である。クラスタリングの結果として、部分文書に対する問い合わせにおいて検索速度を向上させ、かつ検索精度を出来るだけ落とさないようにすることが可能であると考える。

ここで、クラスタリングのために部分文書の特徴量を与える必要があるが、XML 文書検索においても通常の文書検索と同様な手法が利用可能であることから、ベクトル空間モデル[6]を用いて各部分文書を文書ベクトル化することが出来る。

しかし、全ての文書内の語を使って文書ベクトルを作成しようとすると次元数が非常に大きなものとなってしまう。次元数が大きくなると、クラスタリングの際の計算量が増えるだけでなく、“次元の呪い[7]”と呼ばれる現象が起こり、文書ベクトルがクラスタを形成しない可能性がある。そこで我々は、全部分文書において索引語に対して重み付けを行い、その重みが全部分文書中において特に大きいものを利用して文書ベクトルを作成し、さらにクラスタリングを行う。これにより、特徴的な語の多いクラスタと特徴的な語の含まれていないクラスタに分けることが可能となっている。また、n 語以下の文書をクラスタリングの対象から外す事で、より情報量の多い文書が検索結果に反映されるようにしている。

索引語の重み付け手法として一般的によく知られている tf (term frequency)と idf (inverse document

```
//article[about(.,space history)]//section[about(.,astronaut cosmonaut engineer)]
```

図 2． CAS 問い合わせの例

frequency)[6]を用いる方法が有名であるが、近年の研究から XML 文書においては、その構造を有効利用した ipf (inverse path frequency)[8]という指標がある。これは文書の根ノードからのパスを基にして、文書を分けてそのパスにおける語の特異性を求める手法であり、XML 文書中においては tf-idf を用いるより tf-ipf を利用した方が良いとされている。このことから、本手法においては tf-ipf を利用して重みを計算している。そして全文書中における tf-ipf 値が上位 k 種類の単語を文書ベクトルの要素とすることで、クラスタリングのための文書ベクトルの次元を制限している。こうして各部分文書について作成された文書ベクトルを基にしてクラスタリングを行う。クラスタリングアルゴリズムは凝集型階層的クラスタリング手法の１つである、セントロイド法(重心法)[9]を用いている。セントロイド法は、クラスタ内に含まれる各ベクトルの距離を計算し、その重心から一定の距離にあるものを１つのクラスタとして考える方法である。しかし、データ容量の観点から、一度に全てのベクトル間距離を計算しようとするとメモリ領域にデータが格納出来ないため、今回はある一定数の部分文書ごとにクラスタリングを行い、その結果得られたクラスタセントロイドを用いてクラスタを凝集させる方法を取る。この方法では、一定数の部分文書間のみでしか文書ベクトルの距離を比較しないので、クラスタリング結果の精度が落ちる可能性があるが、読み込む文書数を一定にすることにより、メモリ領域に格納するデータ量を制限することが可能になっている。

以上のようにして作成されたクラスタ集合を実際に関係データベース上に格納された XML 文書に対し適用するが、その前に、クラスタ情報を格納していない形式である XRel を用いた XML 文書の格納方式について説明する。XRel を用いて XML 文書を格納すると、部分文書情報は表 ELEMENT に、パス情報は表 PATH に、文書内の単語情報は表 TERM に格納される。

- DOCUMENT (DOCID, FILE)
- ELEMENT (DOCID, NODEID, PATHID, ST, ED)
- PATH (PATHID, PATHEXP)
- TERM (TERM, DOCID, NODEID, TFIPF)

各部分文書は基本的に DOCID、NODEID の２つのキーを用いることにより識別することが可能となっている。また、表 ELEMENT 内のキーである ST、ED は XML 文書内における開始バイト、終了バイト位置であり、これら２つの値を比較することにより、部分文書同士の親子関係を求めることが出来るため、構造結合(Structural join)[10]を行う際に使用される。経路情報は PATH、PATHEXP の２つのキーに格納されるが、実際にアクセスに用いるものは PATH、問い合わせの際に使用するものが PATHEXP である。

そこで我々は、クラスタを考慮した部分文書情報と、各文書内に含まれている単語情報を格納するテーブルを以下のように作成した。

- CLUSTER_T (CLUSTERID, DOCID、NODEID, PATH, PATHEXP, ST, ED)
- CLUSTER_WORD (CLUSTERID, WORD, DOCID, NODEID, TFIPF)
- DOCUMENT (DOCID, FILEPATH)

表 CLUSTER_T には部分文書の基本情報、つまり表 ELEMENT と表 PATH を合わせた情報にクラスタ ID を付加している。単語情報を CLUSTER_WORD に格納してある。これは表 TERM にクラスタ ID を付加したものであると言える。表 DOCUMENT は、実際に記述されている XML ファイルとの対応付けのために用いられる。

新たに作成した表の具体的な値は表２．のとおりである。そしてこれら３つのテーブルに対して問い合わせを行う。

問い合わせについて、まずその概念図を図３に示す。図３からも分かるように、まず問い合わせのキーワード情報に対して、スコアが上位のクラスタを c 個選択し、そのクラスタに含まれる部分文書集合のみに対して問い合わせのキーワード、経路情報に適合する部分文書を抽出し、スコア順に出力する。

表２．テーブル情報

| CLUSTER_T | | | |
|---|---|---|---|
| CLUSTERID | DOCID | NODEID | PATH |
| PATHEXP | ST | ED | |
| 0 | 100 | 0 | /article[1] |
| #/article | 40 | 7174 | |

| CLUSTER_WORD | | | | |
|---|---|---|---|---|
| CLUSTERID | WORD | DOCID | NODEID | TFIPF |
| 0 | amount | 100 | 0 | 0.4960527 |

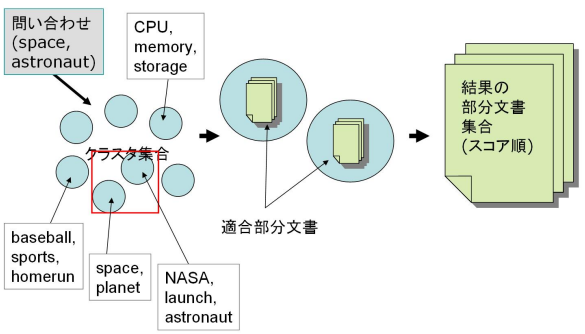


図３．クラスタを考慮した問い合わせ概念図

この概念に基づいて、関係データベース上に格納された部分文書集合に対する問い合わせの流れは以下のようになっている。

1. 表 CLUSTER_WORD より該当する語を含むスコアが上位 c 個のクラスタを選択する
2. 表 CLUSTER_T より 1 で求められたクラスタ内から、該当するパス表現に一致する部分文書集合を選択する
3. 構造結合が必要なものは結合し、最終的にスコアが高い順に出力する

これを踏まえ、実際の問い合わせ例の SQL を図 4 に示す。まず SQL 中の 9-14, 25-30 行目によって問い合わせに対して該当する語が含まれるスコアが上位 10 個のクラスタを選択している。その情報から、6-20, 22-37 行目によって対象となるクラスタ内から、該当するパス表現に一致する部分文書集合を選択する。この問い合わせでは構造結合が必要であるため、38-41 行目の条件を使用して構造結合を行う。そして最終的に得られた結果からスコアが高い順に、INEX により定められた件数である上位 1500 件を出力する。

```
1  SELECT *
2  FROM (
3   SELECT d.filepath, r1.path,
4      SUM(r0.score+r1.score) AS score
5   FROM document d, (
6    SELECT c.docid, c.nodeid, c.path,
7        c.st, c.ed, SUM(cw.tfipf) AS score
8     FROM cluster_t c, cluster_word cw, (
9      SELECT clusterid, SUM(tfipf) AS score
10     FROM cluster_word
11     WHERE word IN ( 'histori' , 'space' )
12     AND ROWNUM BETWEEN 1 AND 10
13     GROUP BY clusterid
14     ORDER BY score DESC) c0
15    WHERE cw.word IN ( 'histori' , 'space' )
16    AND c.docid = cw.docid
17    AND c.nodeid = c.nodeid
18    AND c.clusterid IN c0.clusterid
19    AND c.pathexp LIKE '#%/article'
20    GROUP BY c.docid, c.nodeid, c.path, c.st, c.ed
21    ) r0, (
22   SELECT c.docid, c.nodeid, c.path,
23       c.st, c.ed, SUM(cw.tfipf) AS score
24    FROM cluster_t c, cluster_word cw, (
25     SELECT clusterid, SUM(tfipf) AS score
26     FROM cluster_word
27     WHERE word IN ( 'histori' , 'space' )
28     AND ROWNUM BETWEEN 1 AND 10
29     GROUP BY clusterid
30     ORDER BY score DESC) c0
31     WHERE cw.word IN ( 'histori' , 'space' )
32     AND c.docid = cw.docid
33     AND c.nodeid = c.nodeid
34     AND c.clusterid IN c0.clusterid
35     AND c.pathexp LIKE '#%/article'
36     GROUP BY c.docid, c.nodeid, c.path, c.st, c.ed
37     ) r1
38   WHERE d.docid = r1.docid
39   AND r0.docid = r1.docid
40   AND r0.st <= r1.st
41   AND r1.ed <= r0.ed
42   GROUP BY d.filepath, r1.path
43   ORDER BY score DESC
44  )
45  WHERE ROWNUM BETWEEN 1 AND 1500;
```

図 4．SQL 問い合わせ例（図 2 の問い合わせに対応）

また、本手法を行う上で、以下のような決定すべきパラメタが存在している。

- クラスタ半径の最大値
- 文書長の最小値
- ベクトルの要素とする語の種類数
- 問い合わせの際に取得するクラスタ数

次節において、実際にクラスタリングの有効性を確認する。

## 4. 評価実験

まず、実験に使用する XML 文書は第 2 節で述べたように、Wikipedia の文書集合を使用する。問い合わせ文は INEX 2007 で使用された CAS 問い合わせ（NEXI により記述）の中から 10 種類を使用する。これを第 3 節の手法により SQL に変換したものを使用する。実験環境は表 3 のとおりである。

表 3．実験環境

| DB サーバ | |
|---|---|
| CPU | SUN SPARC III (1.2GHz, 8CPU) |
| メモリ | 32.0GB |
| OS | Solalis 9 |
| RDBMS | Oracle 10g Release10 |

各部分文書はあらかじめクラスタリングされたものを関係データベース上に格納する。ちなみにデータベース中の索引語データは、ストップワードを排除し、ステミング処理を行った上で格納されている。

今回クラスタリングのために設定したパラメタは表 4 のとおりである。なお、クラスタリングプログラムには大島らにより公開されている SlothLib[11]を用いて作成した。

表４．クラスタリング用パラメタ一覧

| | |
|---|---|
| クラスタ半径最大値 | 35.0 |
| 文書長の最小値 (n) | 10 |
| ベクトルの要素とする語の種類 (k) | 1,000 |
| 問い合わせ時取得クラスタ数 (c) | 10 |
| 読み込み部分文書数の平均 | 7,500 |

このパラメタにより実際にクラスタリングを行ったところ、全部分文書を 200 クラスタ程度まで絞り込むことが出来た。

クラスタリングの効果を示すための比較対象として、クラスタリングを行っていないもの(従来手法)、DOCID の剰余を用いて単純に分割したもの(文書内容を考慮していない部分文書分割手法)の２種類を用意する。

なお単純分割したものは、提案手法と同じクラスタ数になるように分割数を設定している。

これら３種類のデータセットに対して前節で述べた SQL 問い合わせを行い、INEX より提供されている評価ツールである、EvalJ[12]を使用し評価を行う。

EvalJ により得られる情報検索システムの評価尺度の１つである適合率-再現率[6]をグラフに表したものを図５に示す。横軸が再現率、縦軸が適合率を示す。再現率とは、全適合文書のうち、検索された適合文書数の割合であり、適合率は検索された文書のうち、適合文書が含まれる割合を指す。この適合率-再現率グラフは、INEX の問い合わせ集合の中から選んだ 10 種類の問い合わせ結果の平均値を示している。

このグラフを見ると、提案手法のデータセットと文書内容を考慮せずに分割したデータセットと比べた場合、明らかに精度面に差が出たことが分かる。また、提案手法は従来手法と比較した場合、大きな精度低下が見られなかったことが分かる。提案手法の特徴として、再現率が低いところでは提案手法の方が適合率の面で高い結果が得られたという点である。これは、クラスタ内に適合文書集合が集中していることが原因と考えられ、従来手法と比べると検索結果の上位に適合文書が含まれる割合が高いことを示している。

次に、クラスタリングに用いたパラメタを変化させて、同様の精度評価を行う。まず、同じ問い合わせ文において、問い合わせ時の取得クラスタ数(c)の値を 1,10,20,30,40,50 と変化させた。その結果、単純分割したデータセットでは c の値を変化させてもほとんど精度向上は見られなかったが、提案手法では図６に示すとおり、上位 20 クラスタを超えてから大幅な精度向上が見られた。

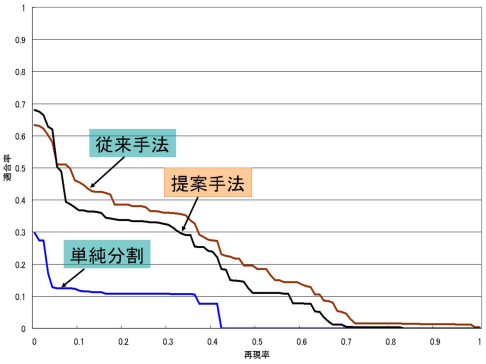


図５．実験により得られた再現率-適合率曲線

このことから、クラスタリングにより、この場合では上位 10 から 20 クラスタ中に適合文書集合が存在していたことが分かる。しかし、スコアが上位でも適合文書集合が該当クラスタに含まれない点から、本手法におけるクラスタリングは文書内容のみでしか考慮されていないため、文書内容と経路情報を問う CAS 問い合わせにおいては不適であると考えられる。

続いて、文書長の最小値(n)を 5, 10 と変化させた場合についても行う。その結果を図７に示す。この結果から、文書長の最小値を小さくした場合、精度の低下が見られた。その理由として、文書長の短い部分文書を検索対象として含んだ結果、適合文書でないものが多く増えた点が挙げられる。その結果、文書長の最小値が小さなものの方が検索精度の低下を引き起こしたと考えられる。

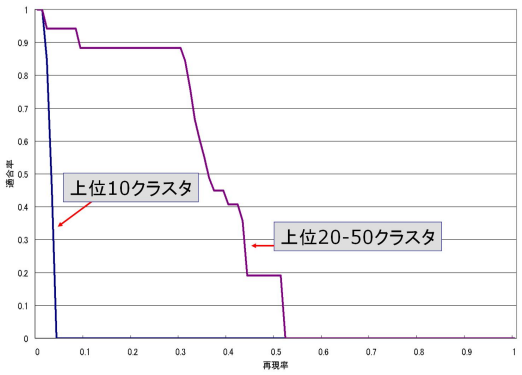


図６．提案手法のパラメタ変化時(取得クラスタ数)再現率-適合率曲線

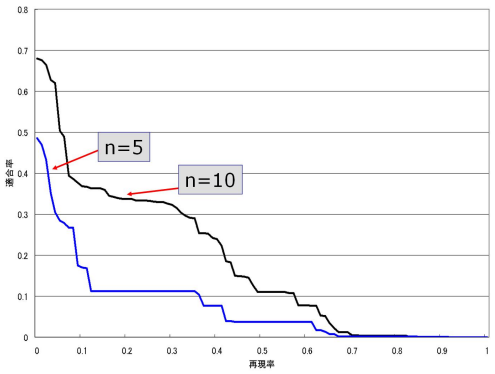


図７．提案手法のパラメタ変化時(文書長の最小値)
再現率-適合率曲線

また、従来手法と提案手法の速度面についても比較を行った。この実験では、精度面での評価と同じ問い合わせ文を用いて、実行速度の比較を行っている。なお、提案手法は、CLUSTER_T、CLUSTER_WORD 共に CLUSTERID ごとに実体化ビュー(Materialized View)を作成し、ネストされた SQL 文を分割し perl スクリプト上で統合して問い合わせを行っている。以下の図８に従来手法と提案手法の実行時間の比較グラフを示す。横軸が問い合わせ番号、縦軸が実行時間(秒)となっている。

グラフからも分かるように、提案手法では多くの問い合わせにおいて速度向上が見られ、向上が見られたものでは約 30%から 90%、全体平均では 30%程の速度面での向上がされていることが観察出来る。

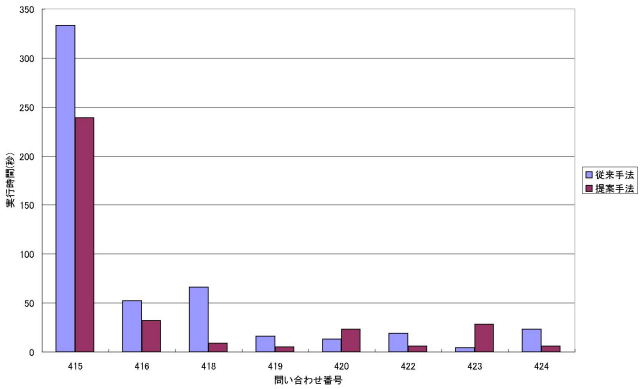


図８．実行時間の比較

## 5. 結論と今後の課題

本論文では、大規模の XML 文書の部分文書をクラスタリングし、検索性能を上げることを目指した。XML 文書は関係データベース上に格納し、NEXI で与えられる CAS 問い合わせを SQL に変換した。各部分文書は tf-ipf 値の高い順にソートしたときの上位に来る語を使い文書ベクトルの次元数を制限した。また、一定数の部分文書ごとに読み込むことにより、クラスタリング時、メモリに格納するデータ量を制限することにより全部分文書に対するクラスタリングを行った。関係データベース上に格納された XML 文書に対して、クラスタリング結果を反映させるために新たにテーブルを作成し、問い合わせに使われる SQL 文を変更するための方法を提案した。

そしてクラスタリングの有効性を調べるために、クラスタリングを行っていないデータセット、文書内容を考慮せずに分割したデータセットと提案手法を精度面で比較し、提案手法を用いることにより部分文書集合を分割した際の精度低下が抑えられ、適合率が低い点では提案手法の方が優れた結果を出したことを観察した。そして、提案手法のパラメタ変化により、精度面の向上、低下を招くことを観察した。また、従来手法と提案手法を速度面で比較したところ、提案手法は従来手法に比べて速度面での向上が見られた。

今後の課題として、より様々なパラメタ環境において評価実験を行い、その結果からクラスタリングのためのパラメタ決定則を考案することや、キーワードと経路情報を考慮したクラスタリング手法の提案、その他のクラスタリング手法との性能比較などが考えられる。

## 6. 謝辞

本研究の一部は、科学技術振興機 CREST「情報社会を支える新しい高性能情報処理技術」、ならびに科研費補助金特定領域研究「情報爆発時代に向けた新しい IT 基盤技術の研究」(課題番号 19024058)による。ここに記して謝意を表します。

## 参　考　文　献

[1] Wikipedia: http://www.wikipedia.org

[2] INEX: INitiative for the Evaluation of XML Retrieval (2007). http://inex.is.informatik.uni-duisburg.de/2007/

[3] M. Yoshikawa, T. Amagasa, T. Shimura, S. Uemura : “XRel: A Path-Based Approach to Storage and Retrieval of XML Documents using Relatioinal Databases”, ACM Transactions on Internet Technology, Vol. 1, No.1, pp.110-141, 2001.

[4] A. Trotman, B. Sigrubjoernsson: “Narrowed Extended XPath I (NEXI)”, Advances in XML Information Retrieval, Vol. 3493/2005, pp.16-40.

[5] 清水敏之, 寺田憲正, 吉川正俊: “関係データベースを用いた XML 情報検索システムの開発”, 情報処理学会論文誌:データベース, Vol. 48, No. SIG11, pp.224-234, 2007.

[6] 北研二, 津田和彦, 獅々堀正幹: “情報検索アルゴリズム”, 共立出版

[7] 石井 健一郎, 上田 修功, 前田 英作, 村瀬 洋: “わかりやすいパターン認識”, オーム社 (1998).

[8] K. Hatano, T.Simizu, J. Miyazaki, Y. Suzuki, H.Kinutani, M. Yoshikawa: “Ranking and Presenting Search Results in an RDB-based XML Search Engine”, Pre-Proceedings of INEX 2007 Workshop, pp.156-169, 2007.

[9] B.S.Everitt: “Cluster Analysis”, Edward Arnold, third edition (1993).

[10] S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, D. Srivastava, and Y. Wu, “Structural joins: A primitive for efficient XML query pattern matching”, Proc. ICDE, p.141, 2002.

[11] 大島裕明, 中村聡史, 田中克己: “SlothLib: Web 研究のためのプログラミングライブラリ”, DEWS2007 C7-10.

[12] EvalJ: INEX Evaluation Package. http://evalj.sourceforge.net/