オントロジとパスの類似度を用いたXMLデータの曖昧検索

文　連子† 天笠　俊之†,†† 北川　博之†,††

† 筑波大学 システム情報工学研究科 〒305-8573 茨城県つくば市天王台 1-1-1
†† 筑波大学 計算科学研究センター
E-mail: moon@kde.cs.tsukuba.ac.jp, {amagasa,kitagawa}@cs.tsukuba.ac.jp

あらまし 類似した内容を持つものの異なった構造を持つ XML データに対して XPath 問合せを適用する手法について議論する．現在，XML データは爆発的に増加し，その結果，類似した内容を異なるタグ名，異なる構造で記述している XML データが大量に出現している．類似した内容を持つそれらのデータは，その構造を知らない限り検索できない．そこで本研究では，1) オントロジを使って類似した概念を表す XPath 式中のタグ名を書き換え，2) 編集距離にもとづく類似度を利用して，類似している XML データに対して XPath 式を適用する手法を提案する．また，その有効性を実験により示す．

キーワード XML データ，XPath，オントロジ，編集距離，編集類似度

Proximity Search of XML Data using Ontology and XPath Edit Similarity

Lianzi WEN†, Toshiyuki AMAGASA†,††, and Hiroyuki KITAGAWA†,††

† Graduate School of Systems and Information Engineering, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
†† Center for Computational Sciences, University of Tsukuba
E-mail: moon@kde.cs.tsukuba.ac.jp, {amagasa,kitagawa}@cs.tsukuba.ac.jp

Abstract In this paper, we discuss a technique for applying XPath queries to XML data which have similar contents with different structures using ontology and XPath edit similarity. XML data is explosively increasing, and as a consequence, a large amount of XML data, in which similar contents are described with different tag names and structures, have been emerging. In such a situation, one cannot write a query against such XML data unless he/she knows the structure of the data. In this research, we discuss the following techniques to cope with this problem: 1) we attempt to expand an XPath query by replacing a tag name in the query with similar ones by using ontology; and 2) we try to apply XPath queries to XML data with different structures by using edit similarity, which can be computed by the edit distance between two distinct path expressions. We show the effectiveness of the approach by experimental evaluations.

Key words XML data, XPath, Ontology, Edit distance, Edit similarity

1. はじめに

XML (Extensible Markup Language) [1] は，1998 年 2 月に World Wide Web Consortium(W3C) で標準化されて以来，さまざまな応用で利用されるようになっている．XML はデータ記述のためのメタ言語であり，基本的な構成要素である要素を階層的に記述することで，任意の木構造のデータを表現することができる．その応用範囲は急速に広がっており，ビジネスデータ，科学データ，ウェブ文書，ログなど数多くの応用で利用されるようになっている．

XML データを利用する際，データ中の興味のある部分を特定する仕組みが必要になる．このための汎用言語として XPath [2] が 1999 年に W3C 勧告として公開された．XPath は，XML 検索言語 XQuery [3] や変換言語 XSLT [4] など，XML に関連するほとんど全ての規格において標準的に用いられている重要な言語である．

XML データは爆発的に増加しつつあり，その結果，類似した内容を異なるタグ名，異なる構造で記述している XML デー

```
<articles>
   <article volume=“ 20 ”>
      <title>XML</title>
      <author>Max</author>
   </article>
   <article volume=“ 21 ”>
      <title>XML Join</title>
      <author>Jane</author>
   </article>
</articles>
```

図 1　XML データ

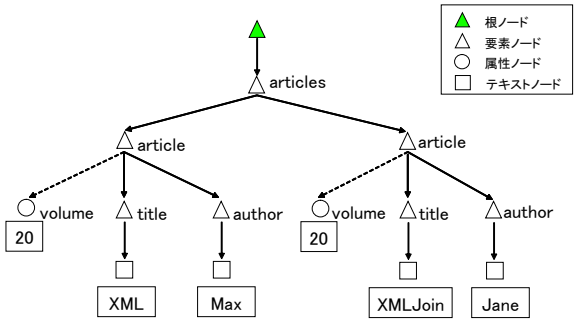

図 2　XML 木構造

タが大量に出現している．例えば，電子図書館における XML で記述された論文データベースを考えると，ACM, IEEE, 情報処理学会，電子情報通信学会など，多くの学会で類似したサービスを行なっているが，使用しているタグ名や XML データのタグの構造などが異なっている．利用者は，自分が知っている XML データに対する XPath 問合せ式を書くことはできるが，他の類似した内容を持つデータに対する問合せ式は，その構造を知ってからでないと書くことができない．この問題は，1) 類似した概念を表現するのに異なるタグ名を使っていることと，2) 内容は類似しているが構造が異なるという二つの問題に分けて考えることができる．

そこで本研究では，既知の XML データに対する XPath 問合せ式を，構造を知らない XML データに対して適用する手法を提案する．具体的には，1) オントロジを使って概念が類似しているタグ名に XPath 式を書き換える，2) 編集距離にもとづく編集類似度を利用して，問合せパス式を構造が類似している XML データに対しても適用するという二点について検討する．また，大量の XML データに対する処理を可能にするため，SSJoin [15] に基づく処理手法についても議論する．

本論文の構成は次の通りである．第 2 章では，前提となる XML 関連技術について説明する．第 3 章では関連研究について述べ，第 4 章では提案手法について説明する．第 5 章ではシステム概略について説明し，第 6 章では評価実験について述べる．最後に第 7 章において，まとめと今後の課題について述べる．

2. 基本的事項

2.1 XML データ

任意の XML データは木構造としてモデル化される．図 1 の XML 文書を木構造でモデル化すると図 2 のようになる．ノードには，根ノード，要素ノード，属性ノード，テキストノードがある．根ノードは木の根であって，ある木構造に一つしかない仮想的なノードである．要素ノードは各要素に対応し，要素ノードあるいはテキストノードを子ノードとして持つ．属性ノードは，属性に対応し，属性名と属性値を持つ．テキストノードは文字列に対応する．

2.2 XPath

XPath 式は，XML データ中のノードを指し示すための言語であり，検索言語としても用いられる．利用者は，XPath を使って XML データ中の任意のノードを指し示して，そのデータにアクセスすることが可能となる．

XPath によるノード特定の方法の基本はロケーションパス（location path）である．木の根を起点とするかどうかによって，

- 絶対ロケーションパス
- 相対ロケーションパス

に分類される．ロケーションパスは，ロケーションステップ（location step）と呼ばれる基本単位を “/” でつなぎ合わせて表現される．

ロケーションパス＝/ロケーションステップ/ロケーションステップ/...

ロケーションステップは，軸（axis），ノードテスト（node test），述語（predicate）の 3 つ部分から構成される．軸は，起点からノードをたどる方向を決める指定で，XPath では child, descendant, parent, ancestor など 13 の軸がある．ノードテストは，軸で決まった方向の中でどのようなノードを選択するのかを指定するもので，特定の要素名やキーワードを書く．述語は，ノードテストの後に角括弧でくくって記述した式て，ノードテストで指定されたノード集合を絞り込むことができる．図 2 において，「根ノードの子ノードの “articles” 要素のさらに子ノードの “article” 要素」を指定したいときは，“/child::articles/child::article” と記述する．XPath では，軸を表すキーワードに対して child:: が省略できるので，“/articles/article” のように指定できる．また，「根ノードの子孫ノードの “title” 要素」は，“/descendant-or-self::node()/child::title” と記述し，省略すると “//title” となり，図 2 にあるすべての “title” 要素を指定することが可能である．

3. 関 連 研 究

Liang 等 [11] は，類似した内容で異なる構造を持つ二つの XML データの類似結合 (similarity join) を行う手法を提案している．まず，双方の XML データから意味的にまとまった部分を抽出する．それぞれの部分 XML データに関して，そこに含まれるテキストノードとパス式の類似度を計算し，ある閾値を超える類似度を持つペア同士を結合する．

類似度を判定する具体的な手順は以下の通りである: 1) ある XML データが持っているテキストノード数と，その XML

データと類似度を判定するもう一つの XML データが共通に持っているテキストノード数でテキストノードの類似度を計算する．2) 同じテキストノードを持つパス式で，根ノードから葉までのノード数と，共通する要素数でパス式の類似度を計算する．3) テキストノードとパス式の類似度を利用し，二つの XML データの類似度を計算する．計算される類似度が与えられた閾値 τ より大きければ，二つの XML データが類似していると判定する．

Zhang 等 [12] は木編集距離 (Tree Edit Distance;TED) を利用し，二つの XML データの類似度を計算する手法を提案している．木編集距離とは，木構造に対してノードの挿入，削除，置換などの操作を施し，目的とする構造を得るために必要な最小の操作数のことである．この木編集距離を適用するため，まず XML データから木構造を求め，ある木構造から，目的とする XML データの木構造に変換するときの木編集距離で類似度を判定する．木編集距離は一般的に計算量が膨大になるため，大量の XML データには適用できない問題がある．

Amer-Yahia 等 [13] は木パターンを利用し，その木パターンに近い XML データを検索する手法を提案している．与えられた木パターンには，ノード間の親子関係，先祖と子孫関係が示されていて，ノードとエッジにはスコアが付いている．その木パターンを変更することによって，それにマッチする XML データを検索でき，さらに木パターンに付いているスコアによって検索結果のスコアも計算できる．よって，与えられた木パターンに一番近い答え K 個を検索できる．

これらの研究は，基本的に部分 XML データ同士の類似度を求める方法について議論している．また，要素名の類似度やパス式の類似度といった要素を考慮していない．これに対して我々は，XPath を用いた XML データの検索において，要素名やパス式の類似度を考慮した検索方式を提案しようとする点が異なる．

関係表同士の類似結合に関する研究には WHIRL [14] がある．WHIRL では，ベクトル空間モデルを使って，テキストの類似度を計り，類似結合を行っている．

4. 提 案 手 法

類似した内容を持ち，異なった構造を持つ XML データを検索するため，本研究では以下の二つの方式を組み合わせる．

（1） 類似した要素名に対するマッチングを行なうために，オントロジを用いて要素名と意味の近い要素名を取得する．これにより，意味的に近い内容を表していると期待される要素名に関する検索が可能となる．

例えば，同じ XML 論文データにおいても，あるデータでは論文を表しているのにタグ名 “paper” を使っているとする．このとき，全ての論文に関する情報は XPath 式 “//paper” で取得できる．ところが，別のデータでは “article” を使っている．この場合 “//paper” では何も返ってこない．“article” と “paper” は語は異なるが類似した概念を表しているので，これらをマッチングしたい．このためにはオントロジを使う．

（2） 類似したパス式に対する検索を可能にするために，パス式同士のマッチングに編集類似度を適用する．これは，編集距離に基づいて計算される類似度である．編集類似度がある程度大きいパス式については，同様の内容を保持していることが期待される．

図 3 オントロジの例

例えば，あるデータでは “/article/author” で，論文の著者名を表しているとする．このとき，全ての論文の著者名に関する情報は XPath 式 “/article/author” で取得できる．ところが，別のデータでは論文の著者名を表しているのに “/article/authors/author” を使っている場合，“/article/author” では何も返ってこない．同じ内容を表している類似したパス式は，要素 “authors” を挿入したり，削除したりすると同じパス式になる．これらのマッチングには編集類似度を使う．

以下では，それぞれの方式について具体的に述べる．

4.1 オントロジを利用した類似要素名検索

まず，オントロジを利用して類似した要素名を得る方法について述べる．オントロジとは，与えられた問題領域内の語彙や概念集合を整理，分類したものである．オントロジは一般的にグラフ構造として与えられ，ある語彙（概念）に類似した語彙（概念）はそのグラフにおける近傍を探索することに他ならない．本研究では，オントロジは非巡回有向グラフであるものとする．

オントロジにおける類似語彙の探索を行なうため，本研究ではグラフ理論における最小共通先祖 (Least Common Ancestor; LCA) からの距離を用いる．LCA とは，その名が示す通り二つのノードが共通に持っている最も近い先祖ノードのことである．二つの語彙から LCA までの距離が近ければ，それらの語彙はより類似していると考えられる．例えば，図 3 において “report” と “diploma” の LCA は “document” であり，“diploma” と “diary” の LCA は “writing” である．例えば，LCA までの最大距離が閾値 2 以下であれば類似しているとするなら，二つのノード “report” と “diploma” は類似しているが，“diploma” と “diary” は類似していないと判定する．

4.1.1 オントロジについて

本研究の目的を考えるとき，用いることのできるオントロジには以下の二通りが考えられる．

1) シソーラスなど，既存のオントロジを利用する．

2) 利用する XML スキーマが複数存在し，それらがあらかじめ知られている場合，スキーマに含まれるマークアップ語彙に関するオントロジを個別に作成する．

1) は，マークアップ語彙に “title” や “author” など一般的な単語を用いている場合，未知のスキーマに対しても有効であ

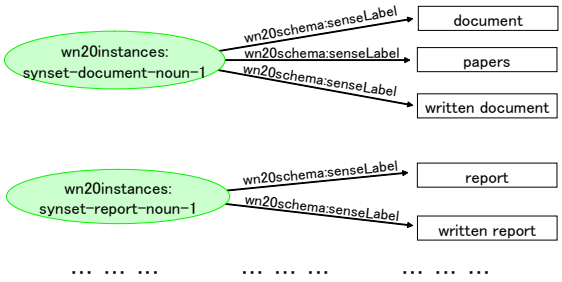

図 4　wordnet-senselabels.rdf

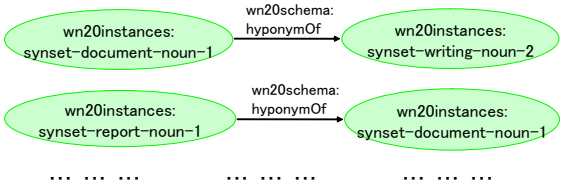

図 5　wordnet-hyponym.rdf

る点が利点である．しかし，省略形や記号列などが用いられている場合には有効でない．逆に，2) は未知のスキーマに対処できないが，特定の応用向きにカスタマイズされたオントロジを利用した処理が可能である．ただし，巨大なスキーマに対しては作業量が増えてしまう．応用の性質などを考慮し，両者を適宜選択，組合せて用いることが重要である．

本研究では，オントロジとして WordNet [5] を利用する．WordNet は英語のシソーラス辞書である．その構成は synset と呼ばれる同義語の集合を基本とし，synset 間の語彙の関係を記述することで，概念体系を記述している．WordNet を機械処理するため，いくつかプロジェクトが存在するが，ここでは RDF/OWL Representation of WordNet [5] を利用する．これは WordNet の語彙体系をメタデータ記述言語である RDF (Resource Description Work) [9] で記述したものである．

RDF/OWL Representation of WordNet はいくつかのファイルから構成されているが，本研究では synset と具体的な語彙の関係を記述した wordnet-senselabels.rdf，synset 間の上位，下位の関係を記述した wordnet-hyponym.rdf を利用する．RDF 形式の二つのファイルに対して SPARQL 問合せを記述することで，ある語彙に類似した語彙を検索することができる．具体的なデータの内容については，次節で述べる．

4.1.2　RDF 問合せ言語 SPARQL

本研究では，類似語彙の検索に SPARQL(Simple Protocol And RDF Query Language) [6] を用いる．SPARQL は，RDF のための問合せ言語である．SPARQL では，グラフのパターンを検索キーとして与えると，それにマッチする部分グラフが返される．例えば，wordnet-senselabels.rdf と wordnet-hyponym.rdf の一部をグラフ表示すると図 4 と図 5 のようになる．このとき類似した語彙は，SPARQL を利用して以下のように検索できる．

（1）　与えられた語彙 w を含んでいる synset S_w を wordnet-senselabels.rdf で検索する．

```
SELECT  ?Sw
WHERE { ?Sw  wn20schema:senseLabel  w }
```

（2）　synset S_w と LCA への最大距離がある値以下であるすべての，synset S_{wi} を wordnet-hyponym.rdf を使って検索する．

```
SELECT  ?Sw1 ?Sw2 … ?Swi
WHERE { Sw    wn20schema:hyponymOf ?Sw1.
        ?Sw1  wn20schema:hyponymOf ?Sw2.
        ?Sw3  wn20schema:hyponymOf ?Sw1.
          …    …    …
      }
```

（3）　synset S_w と synset S_{wi} の同義語グループのすべての語彙 W_i を wordnet-senselabels.rdf を使って検索する．検索された語彙は w と類似した語彙である．

```
SELECT  ?W1 ?W2 … ?Wi
WHERE { Sw    wn20schema:senseLabel  ?W1.
        Sw1   wn20schema:senseLabel  ?W2.
        Sw3   wn20schema:senseLabel  ?W3.
          …    …    …
      }
```

提案手法では，入力された問合せパス式に対して，各ロケーションステップの要素名と類似している要素名を SPARQL で検索し，XPath 式の該当する要素名を書き換えたパス式を検索式に加える．

4.2　編集類似度を利用したパス式のマッチング

まず，編集距離 (Edit Distance) について説明する．編集距離は二つの文字列がどの程度異なっているかを示す数値である．具体的には，文字の挿入，削除，置換によって，一つの文字列を別の文字列に変形するのに必要な手順の最小回数として与えられる．すなわち編集距離が大きいほど類似度が低いと考えられる．一般的には，変換操作ごとに異なるコストを与える重み付き編集距離，操作に置換だけを許した ハミング距離，挿入と削除だけを許したインデル距離などのバリエーションがあるが，本研究では，挿入，削除，置換のコストがすべて 1 であるレーベンシュタイン距離を利用する．

編集類似度 (Edit Similarity) [15] は，編集距離から求められる類似度である．二つの文字列 σ_1 と σ_2 の編集距離を $ED(\sigma_1, \sigma 2)$ としたとき，二つの文字列の編集類似度 $ES(\sigma_1, \sigma_2)$ は，以下のように計算できる．

$$ES(\sigma_1, \sigma_2) = 1.0 - \frac{ED(\sigma_1, \sigma_2)}{max(|\sigma_1, \sigma_2|)} \qquad (1)$$

二つの文字列が類似するほど編集類似度も高い値となる．

パス式に対して編集距離と編集類似度を適用するためには，パス式に含まれている要素名を一つのアルファベットとみなせばよい．例えば，二つのパス式，

p_1 = “/articles/article/author ”
p_2 = “/article/authors/author ”

の編集距離は以下のようになる．

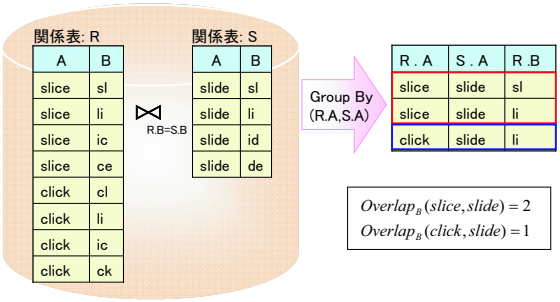

図 6　SSJoin の概要

p_1	/articles	/article		/author
p_2		/article	/authors	/author
コスト	1	0	1	0

パス式 p_1 と p_2 の編集距離は，2 になり，編集類似度は，$ES(p_1, p_2) = 1.0 - \frac{2}{3} = 0.33$ になる．

問合せパス式が入力されると XML データに含まれるそれぞれのパス式との編集距離を計算し，そこから編集類似度を計算する．計算された編集類似度が閾値 α 以上であれば類似していると判定する．しかしながら，XML データには大量のパス式が含まれるため，いちいち編集距離を計算し，編集類似度を計算するのは計算量が膨大になってしまう．

4.2.1　SSJoin

この問題に対応するため，本研究では SSJoin [15] を利用する．SSJoin とは，関係データベースの機能を利用して，ある文字列に対して大量の文字列から編集距離が与えられた閾値以下の文字列を検索することができる方法である．SSJoin の概要は以下の通りである．まず，あらかじめ全ての文字列に対して，q-gram を作成し，図 6 のようにデータベース中の関係表 R に格納しておく．q-gram は，文字列の長さ q の部分文字列全体の集合で，例えば，“slice” の 2-gram は (‘sl’, ‘li’, ‘ic’, ‘ce’) となる．

問合せ文字列が与えられると，同様にその q-gram を計算し，関係データベース中の関係表 S に格納する．二つの関係表 R と S に対して，$R.B = S.B$ の条件で結合演算を行なう．その結合演算の結果を文字列 $R.A$，$S.A$ によってグループにしたとき，その要素数を $Overlap_B(R.A, S.A)$ とする．$Overlap_B(R.A, S.A)$，二つの文字列の長さ，パラメータ q によって編集距離の取りうる最小値が，以下の式で計算できる．

$$ED(R.A, S.A) \geqq \frac{max(|R.A|, |S.A|) - q + 1}{q} - \frac{Overlap_B(R.A, S.A)}{q} \quad (2)$$

上で計算した編集距離 $ED(R.A, S.A)$ の最小値を使って，編集類似度を計算する．

$$ES(R.A, S.A) = 1.0 - \frac{ED(R.A, S.A)}{max(|R.A|, |S.A|)} \quad (3)$$

計算された編集類似度 $ES(R.A, S.A)$ が与えられた閾値 α より大きければ，候補のペアとする．

ただし，結果の中にも実際には編集類似度が α より小さなもの (false drop) が含まれている可能性がある．候補ペアのそれぞれに対して，今度は実際に編集距離を計算し，真の編集類似度を計算する．すべてのペアに対して編集距離を計算すると計算量が膨大になるが，SSJoin によって，関係データベースの機能を利用して候補数を絞り込むことで，計算量を抑えることができる．

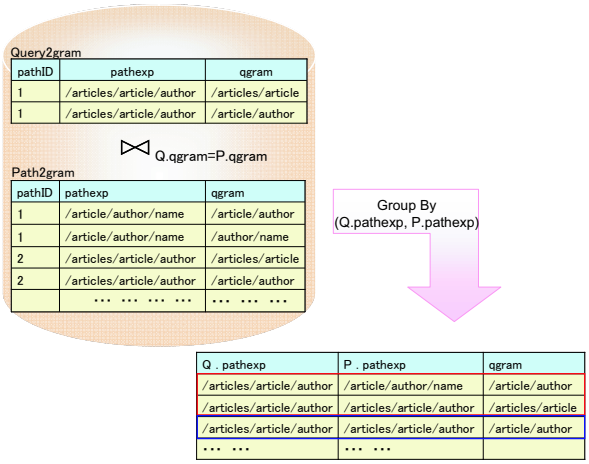

図 7　SSJoin を利用した XPath 式のマッチング

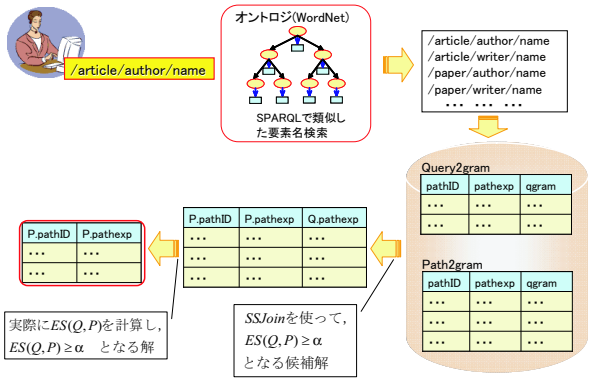

図 8　システムの概要

4.2.2　SSJoin の適用

SSJoin を本研究に適用するには次のような手順になる (図 7)．1) データベースに格納されているすべての XML データのあらゆるパス式を抽出し，あらかじめ q-gram を計算し関係表 path2gram に格納する．2) 問合せパス式が入力されると，そのパス式の q-gram を別途関係表 query2gram に格納する．3) 二つの関係表 path2gram と query2gram で SSJoin を行ない，問合せパス式と編集距離が α 以上であると思われる候補解を得る．4) 候補解にのぞれぞれについて，問合せパス式との厳密な編集距離と編集類似度を計算し，真の解を求める．SSJoin を利用することで，問合せパス式に類似したパス式を効率良く検索することができる．

5.　システムの概要

図 8 に提案システムの概要を示す．データベースに格納されている XML データのすべてのパス式とその q-gram は，あらかじめ関係表 path2gram に格納しておく．

利用者からの，問合せパス式が入力されると，その問合せパ

ス式 ($q = p_1/p_2/\ldots/p_n$) を要素名で分割する．各々の要素名について，SPARQL を用いて WordNet から類似語彙を検索する．検索した類似語彙については，それをそのまま用いることはせず，システム内に存在する全ての要素名とマッチングを行ない，検索結果として含まれる可能性のない要素名はあらかじめ除外しておく．それぞれのパスステップ (p_i) に関して類似語彙 (P_i) が検索されると，その直積：

$$P_1 \times P_2 \times \ldots \times P_n$$

が候補パス集合となる．

候補パス集合について q-gram を計算し関係表 query2gram に格納し，query2gram と path2gram に対して SSJoin を行い，編集類似度が α 以上のパス式を求める．利用者からの問合せパス式は，求められた類似したパス式に変換される．

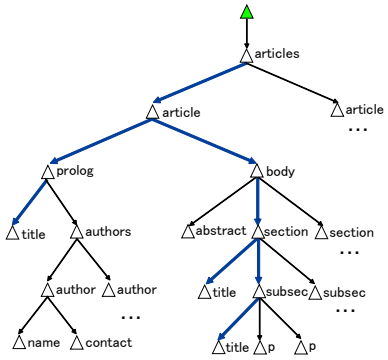

図 9　XML データ

ここで，“//title” のような descendant 軸を含む問合せの場合は，予め該当するすべてのパス式に展開してから処理を行なう．例えば，利用者が図 9 の XML データを念頭に，“//title” で検索を行おうとしているとする．このときは，まず “//title” 問合せパス式から，マッチするパス式である

- /article/prolog/title
- /article/body/section/title
- /article/body/section/subsec/title

を求める．こうして得られた各々のパス式で検索を行う．

6. 実　　験

提案手法の妥当性と処理時間を検証するために，実験を行った．

6.1 実験環境

用いた計算機のスペックを表 1 に示す．

表 1　実験環境

CPU	Intel(R) Xeon(TM) 3.0GHz × 4
OS	Red Hat Enterprise Linux 4.0
メモリ	6GB
JAVA	J2SE 1.5
DB	PostgreSQL 8.1.0

実験データとして，SIGMODRecord.xml [7]，DBLP.xml [8]，XML データ生成プログラム XBench [10] を使って生成した article1-26.xml と Wikipedia の abstract.xml を使った．各 XML データサイズと，抽出されたパス式の数は，表 2 の通りである．

表 2　実験データ

XML データ	サイズ (KB)	パス数
SigmodRecord.xml	464	12
DBLP.xml	357284	164
article1-26.xml	10608	38
abstract.xml	266108	10

6.2 実験方法

SIGMODRecord.xml の構造を既知として，他の XML データに対する問合せを行なう．構造を知らない 3 つの XML データは，プログラムを使って予めパス式を抽出し，1-gram と 2-gram に分割して，関係データベースに格納する．また，その際 3 つの XML データに使われているタグ名も予め，関係データベースに格納する．本実験では，LCA までの最大距離を，1 と 2 に設定したときに各々の場合について実験を行なった．問合せパス式として，表 3 の SIGMODRecord.xml のパス式を使う．

表 3　SIGMODRecord.xml のパス式

p_1	/SigmodRecord/issues/issue/articles/article/authors/author
p_2	/SigmodRecord/issues/issue
p_3	/SigmodRecord/issues/issue/number
p_4	/SigmodRecord/issues/issue/volume
p_5	/SigmodRecord/issues/issue/articles/article/title
p_6	/SigmodRecord/issues/issue/articles/article

6.3 実験結果

6.3.1 検索結果の妥当性

提案手法では，ある問合せパス式に対して類似するパス式を検索すると，結果に同一のパス式が複数回現れることがある．その場合，最も高い類似度のみを残し，他は結果から削除することとした．

表 4 に問合せパス式 p_1 で検索された検索結果を示す．

表 4　問合せパス式 p_1 から検索されたパス式

pathID	pathexp	ES
184	/dblp/article/author	0.285
7	/article/prolog/authors/author	0.285
96	/dblp/book/author	0.285
103	/dblp/book/series	0.142
73	/dblp/article/journal	0.142
77	/dblp/article/month	0.142
69	/dblp/mastersthesis/author	0.142
…	… … …	…

表から分かるように，pathID が 103, 73, 77 のパス式は，検索結果にふさわしくない．これは，編集類似度を適用しているために，パス式の末尾の要素名が問合せパス式の末尾の要素名と完全に違うものも同等に検索されるからである．しかし，パス式では末尾の要素名が利用者の検索意図を表しており，もっとも重要である．

これに対応するため，末尾の要素名に着目する．入力された問合せパス式の末尾の要素名とその類似語のみを末尾に持つものだけを残すようにフィルターを掛け，末尾の要素名が類似するパス式のみを検索結果とする．

表 5-10 に，LCA までの最大距離を 2 と設定したとき，検索された上位 5 個のパス式を示す．

LCA までの最大距離を 3 以上に設定すると，元の要素名と類似度が低い要素名も取得される．例えば，“paper” に対して最大距離を 3 に設定すると “article” 以外にも “creation”,“prose” など元の要素名と類似度が低いものまで取得される。よって，LCA までの距離を 1 あるいは 2 とした．

表より利用者が指定した問合せパス式に近いパス式が検索されていることが分かる．

表 5　問合せパス式 p_1 から検索されたパス式

pathID	pathexp	ES
184	/dblp/article/author	0.285
7	/article/prolog/authors/author	0.285
96	/dblp/book/author	0.285
69	/dblp/mastersthesis/author	0.142
62	/dblp/phdthesis/author	0.142

表 6　問合せパス式 p_2 から検索されたパス式

pathID	pathexp	ES
103	/dblp/book/series	0.333
67	/dblp/phdthesis/series	0.333
40	/dblp/proceedings/series	0.333
73	/dblp/article/journal	0.333
195	/dblp/proceedings/journal	0.333

表 7　問合せパス式 p_3 から検索されたパス式

pathID	pathexp	ES
125	/dblp/article/number	0.25
186	/dblp/phdthesis/number	0.25
158	/dblp/proceedings/number	0.25
202	/dblp/inproceedings/number	0.25
187	/dblp/book/month	0.250

表 8　問合せパス式 p_4 から検索されたパス式

pathID	pathexp	ES
89	/dblp/book/volume	0.25
176	/dblp/article/volume	0.25
196	/dblp/proceedings/volume	0.25
103	/dblp/book/series	0.25
67	/dblp/phdthesis/series	0.25

表 9　問合せパス式 p_5 から検索されたパス式

pathID	pathexp	ES
131	/dblp/article/title	0.333
120	/dblp/mastersthesis/title	0.333
71	/dblp/book/title	0.333
5	/article/prolog/title	0.166
219	/feed/doc/title	0.166

表 10　問合せパス式 p_6 から検索されたパス式

pathID	pathexp	ES
55	/dblp/article	0.2
195	/dblp/proceedings/journal	0.2
73	/dblp/article/journal	0.2
33	/article/epilog	0.2
38	/article/prolog/genre	0.2

表 11　処理時間の比較 [msec]

問合せ	ES	EDSIM	SSJ(1)	SSJ(2)
p_1	0.1	78974	24422	3737
	0.2	67825	13816	2971
p_2	0.1	9496	1186	-
	0.2	9516	1139	-
	0.3	9451	1096	-
p_3	0.1	47843	4522	-
	0.2	47721	4503	-
	0.3	45880	2501	-
	0.4	46042	2506	-
	0.5	45874	2497	-
p_4	0.1	62205	6382	-
	0.2	62231	6356	-
	0.3	59831	3799	-
	0.4	59820	3800	-
	0.5	59841	3798	-
p_5	0.1	616521	146868	2804
	0.2	570458	102485	2383
	0.3	570300	102435	2382
p_6	0.1	70597	12020	-
	0.2	70475	11808	-
	0.3	67612	8867	-
	0.4	67667	8865	-

6.3.2　処 理 時 間

表 11 に，SIGMODRecord.xml の問合せ式（LCA までの最大距離 = 2）で検索したとき，以下の三つのパターンにおける処理時間を示す．

- EDSIM　：SSJoin を使わない場合
- SSJ(1)　：SSJoin(1-gram) を使った場合
- SSJ(2)　：SSJoin(2-garam) を使った場合

結果より，SSJoin を使ってない場合は，指定した類似度に関係なく処理時間が掛かることが分かった．しかし，1-gram と 2-gram の SSJoin は，指定した類似度を使って先にフィルターを行なうので，類似度が高いほど処理時間が速い．問合せ式 p_1 と p_5 の処理時間を見ると，1-gram の SSJoin を使っている場合より，2-gram を使っている場合のほうが処理時間が速い．しかし，他の問合せ式 (p_2, p_3, p_4, p_6, p_7) では，2-gram の

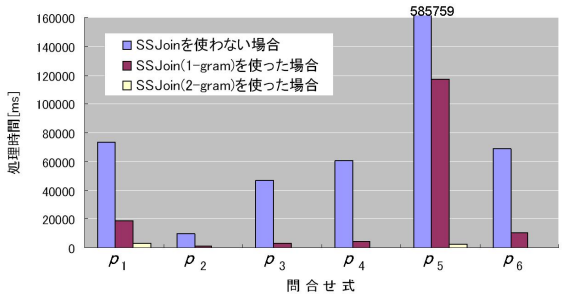

図 10　実験結果（LCA までの最大距離 = 2）

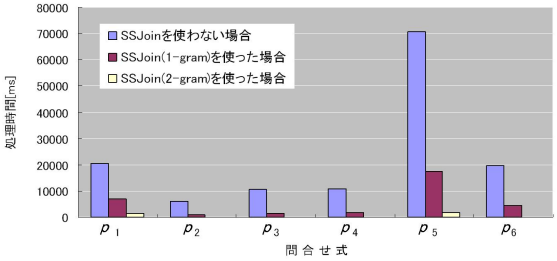

図 11　実験結果（LCA までの最大距離 = 1）

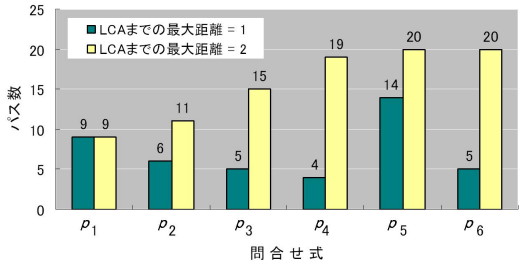

図 12　実験結果（検索されたパス数）

SSJoin を使うと検索できない．それは，本実験で扱っているパス式が短いことと，パス式間の類似度が低すぎるためである．

表 11 で示した処理時間から，各々の問合せ式の平均処理時間を計算しグラフにすると，図 10 のようになる．問合せパス式 p_5 は，オントロジを使って変換されたパス式が多くなっているので，SSJoin を使ってない場合は大量のパス式の編集距離を計算しなければならない．そのため処理時間がかなり掛かっているが，SSJoin によって処理が効率にされている．

LCA までの最大距離を 1 に設定したとき，各々の問合せパス式の平均処理時間をグラフにすると図 11 になる．この場合においても，SSJoin を使わないときが一番処理時間がかかり，2-gram の SSJoin を使ったときが最も速い．図 10 と比べると，図 11 の処理時間が全体的に速い．図 12 に，LCA までの最大距離が 1 と 2 に設定したとき，検索されたパス数を示す．LCA までの最大距離の値を減らすと，検索されるパス数も減少することが観察される．

選択されたパス式の妥当性の検証については今後の課題である．

7. まとめと今後の課題

本研究では，オントロジと編集距離にもとづいた編集類似度を使って既知の XML データに対する XPath 問合せを，構造を知らない XML データに対して検索する手法を提案した．また，q-gram のパラメータ q と，LCA までの最大距離の値を変化しながら，実験を行い提案手法の有効性を示した．

今後は，類似したパス式の検索に掛かる処理時間の短縮，およびパス式の変換に重み付き編集距離を導入することなども考えられる．

謝　　辞

本研究の一部は，科学研究費補助金特定領域研究 (#18049005)，萌芽研究 (#18650018)，若手研究 (B)(#17700110) の支援により行われた．

文　　献

[1] World Wide Web consortium: Extensible Markup Language (XML) 1.0 (Fourth Edition), http://www.w3.org/TR/REC-xml. W3CRecommendation 16 August 2006, edited in place 29 September 2006.

[2] World Wide Web consortium: XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/1999/REC-xpath-19991116. W3C Recommendation 16 November 1999.

[3] World Wide Web consortium: XML Query Language (XQuery) Version 1.0, http://www.w3.org/TR/2006/PR-xquery-20061121. W3C Proposed Recommendation 21 November 2006.

[4] World Wide Web consortium: XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/1999/REC-xslt-19991116. W3C Recommendation 16 November 1999.

[5] WordNet Available at http://www.w3.org/2006/03/wn/wn20/.

[6] World Wide Web consortium: SPARQL Query Language for RDF, http://www.w3.org/TR/2006/WD-rdf-sparql-query-20061004/. W3C Working Draft 4 October 2006.

[7] ACM SIGMOD Record in XML. Available at http://www.acm.org/sigmod/record/xml/.

[8] XML Version of DBLP. Available at http://dblp.uni-trier.de/xml/.

[9] World Wide Web consortium: Resource Description Framework (RDF), http://www.w3.org/RDF/.

[10] XBench - A Family of Benchmarks for XML DBMSs, http://db.uwaterloo.ca/ddbms/projects/xbench/.

[11] W. Liang and H. Yokota: “A Path-sequence Based Method for Solving the One-to-multiple Matching Problem in Leaf-Clustering Based Approximate XML Join Algorithms”, DEWS2006 4A-i10S.

[12] K. Zhang and D. Shasha: “Tree Pattern Matching, Pattern Matching Algorithms”, chaper 11, Oxford University Press(1997).

[13] S. Amer-Yahia, S. Cho, and D. Srivastava: “Tree Pattern Relaxation”, In Proc. of the EDBT Conf., 2002.

[14] William W. Cohen: “Data integration using similarity joins and a word-based information representation language”, ACM Trans. Inf. Syst. 18(3): 288-321 (2000)

[15] S. Chaudhuri, V. Ganti and R. Kaushik: “A Primitive Operator for Similarity Joins in Data Cleaning”, Proceedings of the 22nd International Conference on Data Engineering (ICDE), 2006.