XML キーワード検索における要素の特性を考慮した検索結果の構築

田邊　翼† 　清水 敏之† 　吉川 正俊†

†京都大学大学院情報学研究科 〒606-8501 京都市左京区吉田本町 36-1

E-mail: †tanabe@db.soc.i.kyoto-u.ac.jp, {tshimizu, yoshikawa}@i.kyoto-u.ac.jp

あらまし　本稿では，XML キーワード検索において要素の特性に着目し，特性に応じて扱いを変えて検索結果を構築する手法の提案を行う．XML は様々な用途で使われており，大きくデータ指向 XML と文書指向 XML に分けることができると考えられる．XML 文書に対するキーワード検索の研究では，LCA に基づく取得を行う手法が提案されている一方で，情報検索技術を XML 文書に適用し，XML 文書の粒度に着目した取得を行う XML 情報検索の研究がある．前者は主にデータ指向 XML を対象とした手法であり，後者は文書指向 XML を対象とした手法と考えられるが，一般的な文書ではデータ指向の部分と文書指向の部分が混在しているため，既存研究では適切な結果を返すことができない場合がある．我々は，データ指向の部分と文書指向の部分が混在するような場合にも適用できる XML キーワード検索手法を提案する．

キーワード　XML キーワード検索，LCA，XML 情報検索

1. はじめに

近年，構造化された文書が多く存在し，中でも XML は構造化されたデータおよび文書に対する標準的な記述フォーマットの一つとして普及している．XML はデータ表現やインターネット上のデータ交換にもよく用いられており，XML 文書を効率的に検索する技術は非常に重要なものとなってきている．

XML に対する検索の一つとして，XPath などの XML 特有の問合せ言語を用いるものがある．しかし，このような XML 特有の問合せ言語を利用するためには，文書の論理構造についての予備知識や問合せ言語の記述方法を知っておく必要があるため，そのような知識を持たない多くの一般利用者にとっては利用しづらいものである．それに対して，Web 検索で一般的であるキーワード検索のように，対象となる XML 文書集合から利用者が問合せに用いた問合せ語を含む部分文書のみを取得する XML キーワード検索がある．XML キーワード検索は，事前に検索対象とする文書の論理構造についての知識や複雑な問合せ言語を必要とせず，多くの利用者にとって利用しやすい検索手法であると思われる．

我々は，XML を検索する際に，対象の XML がデータ指向 XML か文書指向 XML かを意識することが重要であると考えた．データ指向 XML としては，例えば論文書誌情報を扱っている DBLP[2]などが挙げられ，文書指向 XML としては，例えば本文を含んだ論文自体や Wikipedia XML Corpus[3]などが挙げられる．Sujeet[1]は，文書指向 XML を対象に効果的な部分木を取得する手法を提案しており，データ指向 XML とは異なり文書指向 XML では要素の粒度が発生するために，ただ単に問合せ語が含まれるかどうかを調べるだけでは効果的な結果は得られないことに着目している．この研究からも対象の XML がデータ指向 XML か文書指向 XML であるかを意識することの重要性が分かる．

XML に対するキーワード検索の技術では，LCA(Lowest Common Ancestor)を用いた手法[4, 8, 9, 10, 11]や，IR(Information Retrieval)技術を利用したもの[12, 15, 16]が挙げられる．前者は主にデータ指向 XML を対象とした手法であり，後者は文書指向 XML を対象とした手法と考えられる．しかし，一般的な文書ではデータ指向の部分と文書指向の部分が混在しているため，既存研究では適切な結果を返すことができない場合がある．そのため，我々は要素の特性に着目し，データ指向の部分か文書指向の部分かによって扱いを変えて検索結果を構築すべきだと考えた．以降では，XML 中のデータ指向の部分に対応する要素をデータ指向要素，文書指向の部分に対応する要素を文書指向要素と呼ぶこととする．また，LCA を用いた XML キーワード検索を LCA 型検索と呼び，IR 技術を用いたものを IR 型検索と呼ぶこととする．

LCA 型検索では，利用者による問合せの検索結果としてノードの特定が行われる．しかし，ノードを検索結果として返すことは，そのノードをルートノードとする部分木全体を返すことと考えることができ，文書指向の部分が混在している場合には余分な情報を与えすぎてしまい，利用者の負担にも繋がる．一方，IR 型 XML キーワード検索では，要素の粒度に着目して検索結果を取得するが，我々はデータ指向要素に関しては粒度問題がないものとして扱うべきだと考えており，適切な検索結果を得るためには工夫が必要である．

我々はまずデータ指向要素と文書指向要素を分類し，それぞれの要素において LCA 型検索と IR 型検索

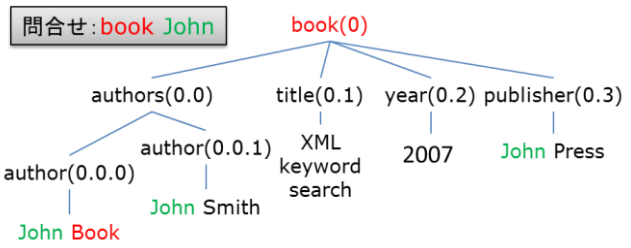


図 1　XML 木に対しての LCA

を適用した後に検索結果を構築することを考えている．我々は，LCA 型検索における手法と IR 型検索における手法を組合せて利用することによって，データ指向の部分と文書指向の部分が混在するような場合にも適用できる XML キーワード検索を対象にした手法を提案する．

2. 関連研究

LCA 型検索では，入力されたキーワードの全てを含む要素である LCA(Lowest Common Ancestor)を求めるが，LCA に関する研究についても，LCA の中から特定のノードのみを取得する SLCA(Smallest LCA)[5]や ELCA(Exclusive LCA)[6]といった手法などが提案されている．SLCA は，子孫に各問合せ語が少なくとも一回以上出現する部分木を含まない最小のノードであり，ELCA は子孫の中で既に各問合せ語が少なくとも一回以上出現している部分木を取り除いた後に，子孫に各問合せ語が少なくとも一回以上出現するノードである．

ここで，図 1 においてノード ID として先行研究[10]などでも用いられている Dewey ID を用いており，本論文では要素名と Dewey ID を用いて要素を示すこととする．例えば authors(0.0)は Dewey ID が 0.0 である authors 要素を示す．図 1 に示すように，問合せである book と John を含む LCA として author(0.0.0)，author(0.0.0)以下にある Book と author(0.0.1)以下にある John の LCA として authors(0.0), author(0.0.0)以下にある John と book(0)の LCA として book(0)，といったように図 1 では三つの LCA が考えられる．この例において，その特性より SLCA は author(0.0.0)となり，ELCA は author(0.0.0)，book(0)となる．ELCA について，まず author(0.0.0)が ELCA となり，それ以外の組合せとして publisher(0.3)以下の John と book(0)との LCA である book(0)が ELCA となっている．

Rui[8] らの研究では，このような ELCA を取得する高速なアルゴリズムを提案している．Zhifeng[11] らの研究では，特に SLCA に着目したものとなっており，利用者の意図を推定し，XML に適用できる TF(term frequency)と DF(document frequency)を定義することによって新たなランキング手法を提案している．また，Ziyang[9, 10] らの研究では，入力されたキーワードやそのキーワードがマッチする場所などから利用者がどのような要素を求めているのかを推定する手法を提案している．Umaporn[14]らの研究では，同じ結果を取得するために異なる語を用いている問題や，どのノードを返し，どれだけの情報が含まれるべきかといった問題を XSemantic と呼ばれるキーワードに基づいた XML の意味的検索を提案することによって対処している．

一方，IR 型検索として，Hatano[15]らの研究では，XML の内容と構造を分析することによって自動で適切な検索結果の単位を見つけ出す手法を提案している．Fujimoto[16]らの研究では，要素名，ピリオドの割合，異なる語の数などに着目することによって自動的に文書指向要素を選択し，適切でないデータ指向要素を取り除くことによって検索を効率化する手法の提案を行っている．Martin[12] らの研究では，TopX と呼ばれる XML 文書のための top-k 検索手法を提案している．この手法は XML 情報検索技術を利用したものであり，マッチする部分を明示した問合せを用いている．

我々は，データ指向の部分と文書指向の部分が混在しているような XML に対しても，適切な結果を利用者に提示する手法を提案する．

3. データ指向要素と文書指向要素の混在

既存研究における XML キーワード検索は，暗黙的にデータ指向 XML に特化したものや文書指向 XML に特化したものであると我々は考えているが，一般的な文書ではデータ指向の部分と文書指向の部分が混在するようなものとなっている場合が多いと考えられる．そのため，既存研究をそのまま適用するだけでは利用者が望む結果を得ることは難しい場合がある．

適切な結果を取得するためには，まずデータ指向要素と文書指向要素を判別する必要があると思われる．本論文では，手動で分類するなどにより，あらかじめデータ指向要素と文書指向要素について判別がついているものとして議論を進める．

また，対象とする XML 文書によっては，文書中に文の一部を強調するタグなどの体裁に関するタグが含まれている場合があるが，そのようなタグは徳田らの研究[17]を利用することで判別し取り除くことができると考えられ，本研究で対象とする XML は論理的な構造に関するタグのみを含むものとする．

3.1. LCA 型検索の適用

データ指向要素と文書指向要素が混在した XML に対して，従来の LCA 型検索を適用すると，利用者による問合せの結果としてノードの特定が行われる．そのため，この検索結果として返されるものは，特定され

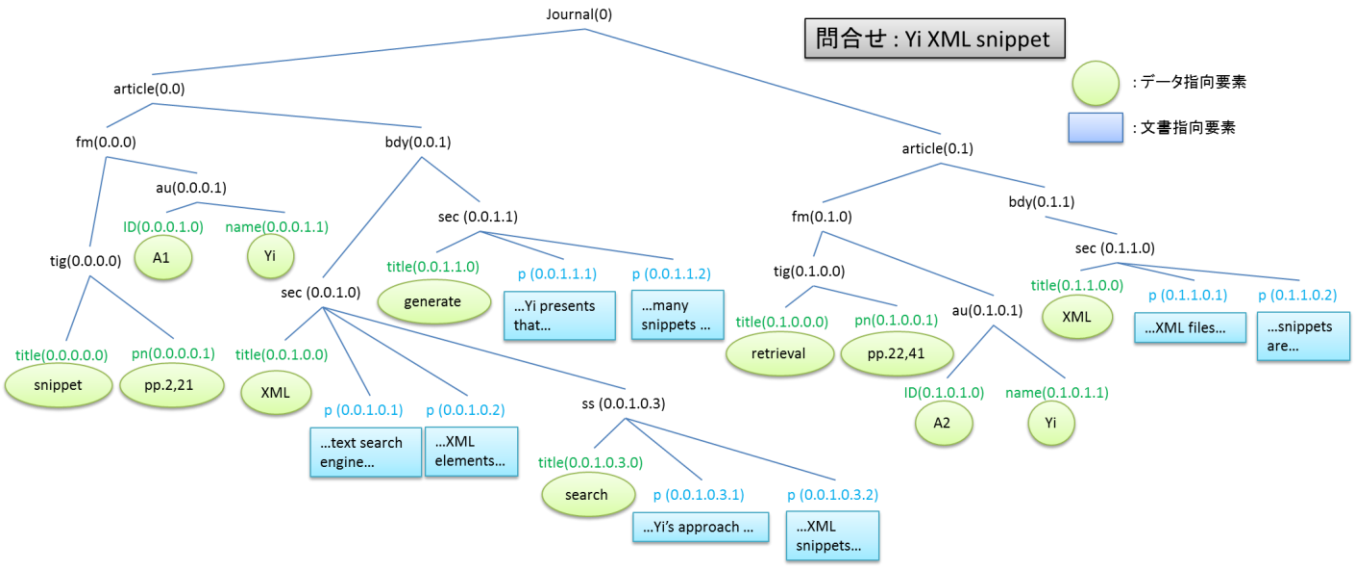
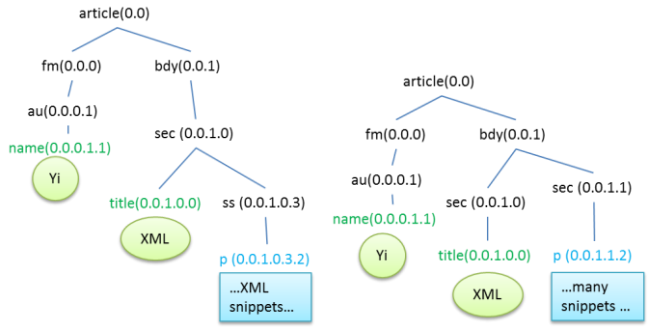


図 2　論文の XML 木



(a)　　(b)

図 3　提案手法による結果

たノードをルートノードとする部分木と考えることが自然である．しかし，文書指向要素も混在しているために，そのような部分木を返すと余分な情報を与えすぎてしまうことになり，利用者の負担も大きくなってしまう．図 2 に示す XML 文書に対して，Yi かつ XML かつ snippet の三つのキーワードを用い，SLCA ノードを検索結果として取得すると，例えば図 2 の article(0.0) が得られる．しかし，article ノードをルートノードとする部分木全体を検索結果とすると，利用者の負担があまりにも大きすぎるものとなっている．

そのため，図 3(a)のように必要な情報のみを組合せ，検索結果を構築することは利用者の負担を軽減することに繋がる．さらには図 3(a)だけでなく，例えばその他にも図 3(b)のようにその組合せ方にも様々なものが存在するため，これらを提示することによって利用者の理解支援にも繋がるのではないかと考えられる．

LCA 型検索において，このように検索結果の構築に関する研究[7]はなされているが，それらは要素の重複を許さない組合せ方法の提案を行っている．これは，重複した要素を利用者に提示することは同じ情報を利用者に与えることになってしまい，冗長であると考えられるためであるが，データ指向要素と文書指向要素が混在している場合には必ずしもそうではないと我々は考えた．混在している場合，データ指向要素はそれぞれの文書指向要素と関連があり，複数の文書指向要素に一つのデータ指向要素が対応するような場合が多々ある．そのため，データ指向要素に関して重複した結果を返した方が効果的である場合があると考えられる．

データ指向要素と文書指向要素が混在するような XML として，ここでは XML で構造化された論文を考える．論文には本文となる文章の他にも著者名やタイトルなどといった情報があり，そのような書誌情報に相当する部分はデータ指向要素であると考えられる．データ指向要素である著者名などは本文のどの部分にも関係のある情報だといえる．そのような著者名などが利用者の問合せに含まれていた場合，重複なしに必要となる要素を組合せただけでは利用者が満足する結果を得ることはできない．図 2 のように著者名やタイトルが問合せの語として含まれていた場合，適切な結果を返すためには，利用者が問合せの語として用いている”Yi”に関する情報を含んだものを返すべきである．この例では”Yi”というキーワードはフィルタのような役割を果たしていると考えることができ，重複を許して検索結果を返すことが結果の妥当性にも繋がると考えられ，例えば図 3(a)，(b)のようにデータ指向要素である name と title に出てくる”Yi”，”XML”を重複して返すことが，適切な検索結果を返すことに繋がると考えられる．このような検索結果の返し方は既存研究で

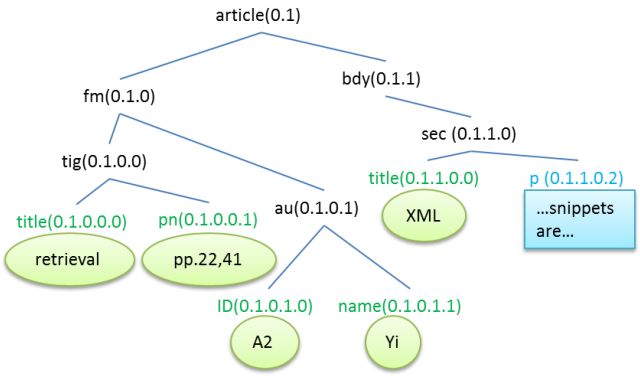


図 4　IR 型検索を適用させた結果

もある XSeek[10]などではできず，我々は検索結果の構築にあたってデータ指向の部分における重複を許す要素にも着目する．

3.2. IR 型検索の適用

次に，データ指向要素と文書指向要素が混在した XML に対して，従来の IR 型検索を適用することを考える．IR 型検索では取得の粒度を問題とすることが多いため，全てのキーワードを含むことを重視しない場合がある．さらには，我々はデータ指向要素について粒度問題はないものとして扱うべきだと考えているが，IR 型検索を適用すると粒度問題があるものとして扱われ，結果的には図 2 において，”Yi”を含むノードとして name ノードであったり fm ノードであったりといった様々な結果が検索結果となり得る．

データ指向要素についても粒度があるものとした場合，図 2 の”Yi”を含むノードとして，例えば，図 4 のように fm ノードが返される場合がある．今，”Yi”はフィルタのような働きをしていると考えられるが，このように fm ノードが検索結果の単位となった場合，余分な情報も利用者に提示してしまう．つまり，検索結果としては，粒度問題を考慮せずに，tig ノードなどを含まない name ノードが妥当なノードだと考えられる．

このように，既存研究をそのまま適用するだけでは適切な検索結果を返すことができない．また，必要となる部分だけを組合せて検索結果を構築することによって利用者の負担を減らすことにも繋がるため，データ指向の部分と文書指向の部分を考慮することによってよりよい結果が構築できると考えられる．

4. 問合せ意図の推定

検索結果を構築するに当たり，大きく二つの問題点がある．まず一つ目の問題として，利用者の問合せ意図をどのように推定するかがある．問合せに用いられているキーワードがどの部分に一致することを期待しているか，あるいは検索結果としてどのような単位で利用者が閲覧したいと望んでいるかといったことが分かっていれば，より適切な検索結果を構築できると思われる．もう一つの問題としてはデータ指向要素と文書指向要素が混在しているような XML に対して，どのように検索結果を構築していき，取得された結果に対してランキングを行うのかということである．

まず，本節において前者の問合せ意図の推定について議論し，5 節において後者の検索結果の構築およびランキングについて議論する．図 2 のように，問合せとして”Yi XML snippet”が利用者から与えられたとする．ここで，”Yi XML snippet”は”Yi”を含みかつ”XML”を含みかつ”snippet”を含むことを意味する．今”Yi”は，著者を示す要素 au(0.0.0.1)，au(0.1.0.1)に現れており，また，段落を示す要素 p(0.0.1.0.3.1)，p(0.0.1.1.1)にも現れている．”XML”や”snippet”についても色々な要素に出現している．このように，問合せ一つを見るだけでも様々なマッチの仕方があることがわかる．

問合せの推定については，Motomura[13]らの手法を応用することによって問合せに対する複数の解釈を取得し，その中から選択することで行うことができると思われる．Motomura らの手法は主にデータ指向 XML を対象としており，特に要素名に着目してなされており，どの要素名に問合せがマッチしているかに応じてクラスタ化することによって複数の解釈を取得しているが，我々が今検討している対象は，データ指向部分と文書指向部分が混在している XML であり，文書指向要素の要素名には意味のあるものが少ないため，そのまま手法を適用することは望ましくない．データ指向要素では例えば author などといったものは著者名を指し，それは非常に意味のあるものとなっているが，文書指向要素では例えば paragraph や sec などに関してどの要素名の要素を取得するかは問題とならない．そのため，要素名に着目した手法をそのまま適用させると，文書指向要素では問題が生じると考えられる．

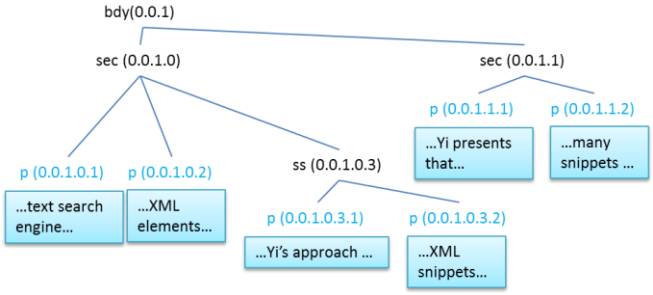
ここで問合せの解釈について，文書指向要素全般に対して”*”を割り当て，Motomura らの表現を拡張して用いることとする．例えば，”Yi XML snippet”に対して(//article, ((Yi, .//au), (XML, .//title), (snippet, .//*)))といった解釈が得られることが考えられる．ここで，(//article, ((Yi, .//au), (XML, .//title), (snippet, .//*)))は部分木のルートノードが article 要素で，”Yi”，”XML”それぞれが article 要素以下の au 要素，title 要素にマッチし，”snippet”が article 要素以下の文書指向要素にマッチすることを示している．文書指向要素にマッチするといったように特定の要素名を指定していないのは，先に述べたように文書指向部分の要素名は意味のあるものが少なく，なおかつ粒度問題を考慮するにあたっ

Algorithm 1 データ指向要素と文書指向要素の混在するXMLに対する検索

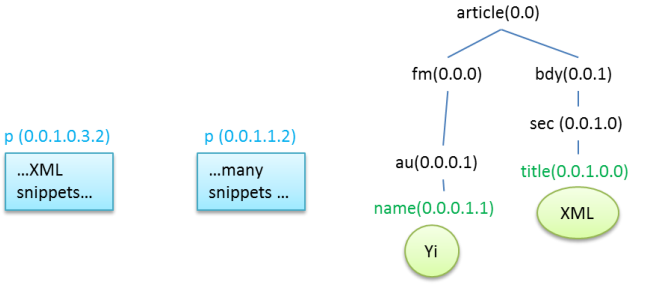
入力: 解釈付きの問合せ Q = (p_r, ((k_1, p_1), ...)), 対象となる XML の木 T, ストラテジー S
出力: 検索結果となる枝抜き部分木の集合 PT
1: Q からデータ指向要素にマッチすると判定されているキーワードとパスのペアの集合(Q_{dat})を取り出す
2: Q から文書指向要素にマッチすると判定されているキーワードの集合(Q_{doc})を取り出す
3: T から文書指向部分を抽出 T^{doc}
4: T^{doc} において Q_{doc} を問合せとして IR 型検索を行う(T_{doc})
5: T において p_r のパスを持つノード集合を取得する(T_r)
6: for each t_i in T_r
7: t_i の中からデータ指向要素の部分を抽出する t_i^{dat}
8: t_i^{dat} において, Q_{dat} にマッチする部分を抽出し, 枝抜き部分木 pt_{dat} を作成する
9: T_{doc} から t_i をルートノードとする部分木に含まれる要素集合(E_{doc})を抽出する
10: PT = ∅
11: if(S = Thorough)
12: for each e_i in E_{doc}
13: pt_{dat} と e_i を結合したものを PT に追加する
14: end
15: end
16: if(S = Focused)
17: E_{doc} から non-overlapping elements を取得する(E_{doc}')
18: for each e_i' in E_{doc}'
19: pt_{dat} と e_i' を結合したものを PT に追加する
20: end
21: end
22: if(S = Fetch and Browse)
23: E_{doc} から non-overlapping elements を取得する(E_{doc}')
24: pt = pt_{dat}
25: for each e_i' in E_{doc}'
26: pt に e_i' を追加する
27: end
28: ptをPTに追加する
29: end
30: end
31: return PT

てマッチすると考えられるものが paragraph 要素であったり sec 要素であったりと変化するためである.

問合せ意図の推定に関しては, 利用者の負担を考慮して自動で推定を行えることが望ましいが, 候補を利用者に提示することによって負担を減らすことができるため, 自動での推定は本論文の対象外とし, 候補を利用者に提示し, 選択してもらうものとする.



(a)



(b) (c) (d)

図 5 枝抜き部分木と要素集合

5. 検索結果の構築

検索結果の構築に際しては, まずデータ指向要素と文書指向要素を分けた後に, 解釈に基づいて既存技術を適用させる.

検索結果の構築アルゴリズムについては, アルゴリズム 1 に示す. 例えば, ((//article, ((Yi, .//au), (XML, .//title), (snippet, .//*)))という解釈が与えられたとする. "Yi", "XML"はそれぞれデータ指向要素にマッチしているために, データ指向要素のみを対象に"Yi", "XML"にマッチする部分を抽出する (7 行目). 一方, 文書指向要素にマッチすると判定されている"snippet"については, XML 情報検索技術を適用させる(4 行目). この際に, 文書指向要素と判定された要素のみを対象として行う. なぜならば, 対象となる XML の木にはデータ指向要素も混在しており, その部分も一緒に XML 情報検索技術を適用させると, データ指向要素についても粒度問題があるものとして検索されてしまうためである. また, XML 情報検索技術を適用させる際に, まず XML の木の文書指向要素全体を対象として XML 情報検索技術を適用させ, その後に検索結果の単位毎に要素集合を取得している(3,4,9 行目).

次に得られた結果から検索結果を構築していくが, その検索結果の構築にあたって我々は Thorough(9-15 行目), Focused(16-21 行目), Fetch and Browse(22-29 行目)という三つのストラテジーが基本的なものになる

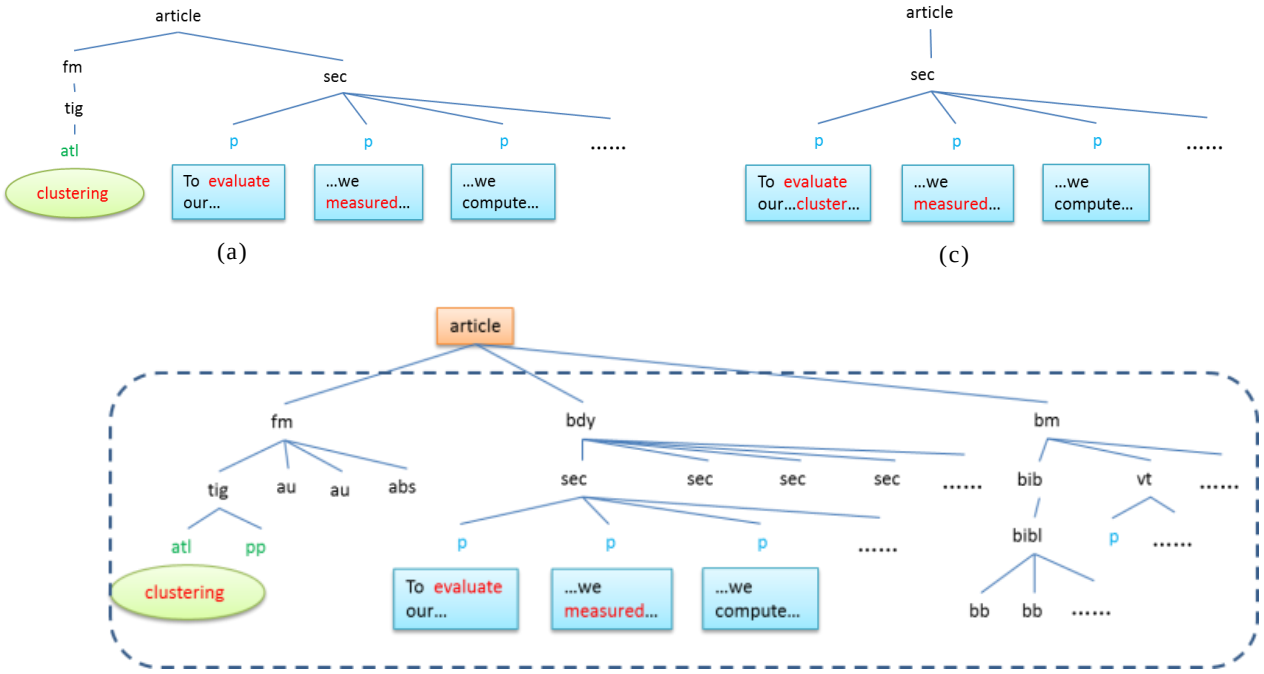


図 6　適用事例の結果

と考えた．

ここで，アルゴリズム 1 を適用することによって図 5 のような文書指向要素の集合とデータ指向要素による枝抜き部分木が得られたとし，文書指向要素の集合におけるランキングが順に図 5(b)，図 5(a)，図 5(c)となったとする．

Thorough では，ランキング順にデータ指向要素に関する部分と文書指向要素に関する部分を結合していき，まず図 5(d)と図 5(b)が結合されたものが検索結果の一つとなる．次に図 5(d)と 5(a)が結合されたものが結果となり，最後に図 5(d)と図 5(c)が結合されたものが結果となる．

Focused では，まず重複のない文書指向要素集合をランキング順に取得する．最初に図 5(b)を取得するが，次の図 5(a)には図 5(b)という既に取得した部分を含むために検索結果とせず，次に図 5(b)と重複しない図 5(c)を取得する．つまり，文書指向要素の集合として用いるものは，図 5(b)と図 5(c)の二つとなる．これら二つと図 5(d)を用いて，あとは Thorough の時と同様にして検索結果を構築する．つまり，まず図 5(d)と図 5(b)が結合されたものが結果となり，次に図 5(d)と 5(c)が結合されたものが結果となる．このように，重複したものを取り除いている点が Thorough とは異なっている．

Fetch and Browse では，Focused の場合と同じく，まず文書指向要素の集合として図 5(b)と図 5(c)を取得する．次に，図 5(d)と図 5(b)を結合し，さらにその結果に図 5(c)を結合したものが検索結果となる．つまり，例えば利用者が検索結果の単位として article 要素を単位に検索結果を閲覧したいといったような場合では，Fetch and Browse を用いると，一つの article 要素に対しては多くとも一つの検索結果が得られるだけであり，複数の検索結果が一つの article 要素以下で得られるわけではない点が，他の二つのストラテジーと異なっている．

このように三つのストラテジーのどれかを利用することによって検索結果を構築していく．

検索結果の構築に際して，我々はデータ指向要素については重複を許している．先にも述べたように，データ指向要素と文書指向要素が混在している場合，データ指向要素一つに対して複数の文書指向要素が関連するといったような混在していない場合には起こらなかった状況が発生する．このような場合，データ指向要素は複数の文書指向要素に関連しているため，我々はデータ指向要素については重複して返すことが効果的だと考えた．

6. 適用事例

検索結果の構築に関して，INEX の 2004 年のデータから Rangarajan らによる” Adaptive Neural Network Clustering of Web Users”というタイトルの論文を対象に，実際にどのような結果が得られるのか調査を行った．解釈としては INEX の実験データとして与えられた問合せを基に，(//article, ((cluster, .//atl),

(evaluate, .//*), (measure, .//*)))という解釈が得られたものとして扱った．我々の手法を適用させると，この解釈における検索結果として，ランキング 1 位のものとして図 6(a)が返ってきた．今，ランキングのスコアの算出には，TF(Term Frequency)を用い，TF は問合せ語の出現回数を語数で割ったものを利用した. 例えば，p という要素に問合せ語の evaluate が二回出現し，p 内には語が百語あったとすると, evaluate の TF は 2/100 で求める．

図 6(b)は，図 6(a)と同じ位置に問合せがマッチする場合に文書指向要素とデータ指向要素を分けずに LCA 型検索を適用させ得られた結果であり，ノードの特定を行うと検索結果として article ノードが返ってくる．今，その article ノードの内容は破線内のようになっており，文書指向要素も混在していることから，利用者が閲覧するには非常に量が多く，わかりにくいものとなっている．

一方，図 6(c)は文書指向要素とデータ指向要素を分けずに XML 情報検索技術を適用させ得られた結果であり，検索結果としては図 6(a)からデータ指向要素に該当する部分を取り除いたものが検索結果となっている．今，(//article, ((cluster, .//atl), (evaluate, .//*), (measure, .//*))) という解釈は，”cluster” に対する”evaluate”と”measure”が知りたいということを表している．検索結果としては，この例においてはほとんど図 6(a)と同じものが返ってきているが，図 6(a)ではタイトルとしての”cluster”という情報を提示することによって，図 6(c)よりも何についてのものかが一目瞭然であり，効果的な結果を返せているのではないかと考えられる．また，IR 型検索について，全ての問合せ語が含まれていることを条件に適用させているが，その条件がない場合，タイトルの”cluster”に関する部分について粒度があるものの結果として, tig ノードであったり fm ノードであったりと様々なノードも検索結果となる．

ランキングに関しては，先に述べたように文書指向要素のみを対象に TF を用いてスコアの算出を行った．これはデータ指向要素に関してはフィルタのような働きをするため，スコア算出においてはデータ指向要素に関する部分はあるかないかといった情報だけが必要なためである．

しかし，スコア算出においては文書指向要素の集合とデータ指向要素による枝抜き部分木を結合する際にも何かしらのスコアが加わると考えられるため，ランキングのスコア算出に当たり，そのことについても考慮する必要がある．

7. まとめ

本稿では, XML キーワード検索において要素の特性に着目し，その特性を考慮して検索結果を構築する手法を提案し，議論した．データ指向要素と文書指向要素を分けることによってそれぞれに適した扱いをし，検索結果を構築する．

検索結果の構築の際，分けられたデータ指向要素と文書指向要素を組み合わせることが必要となるが，その際に XML の構造にも着目することによってランキングを行い，検索結果を返すことを考えている．構造に着目したランキング手法を具体化し，実際に実験を行うことや，利用者による問合せの推定を行うことが課題となっている．

参 考 文 献

[1] Sujeet Pradhan, “An algebraic query model for effective and efficient retrieval of XML fragments”, VLDB, pp.295-306, 2006.

[2] Michael Ley, “DBLP – Some Lessons Learned.”, PVLDB, pp.1493-1500, 2009.

[3] Ludovic Denoyer and Patrick Gallinari, “The Wikipedia XML Corpus”, SIGIR Forum, pp.64-69, 2006.

[4] Yu Xu and Yannis Papakonstantinou, “Efficient LCA based Keyword Search in XML Data”, EDBT, pp.535-546, 2008.

[5] Yu Xu and Yannis Papakonstantinou, “Efficient Keyword Search for Smallest LCAs in XML Databases.”, SIGMOD Conference, pp.527-538, 2005.

[6] Lin Guo, Feng Shao, Chavdar Botev and Jayavel Shanmugasundaram, “XRANK: Ranked Keyword Search over XML Documents”, SIGMOD Conference, pp.16-27, 2003.

[7] Paavo Arvola and Johannna Vainio, “When is in-Context Retrieval Beneficial?”, INEX, pp.41-49, 2010.

[8] Rui Zhou, Chengfei Liu and Jianxin Li, “Fast ELCA Computation for Keyword Queries on XML Data”, EDBT, pp.549-560, 2010.

[9] Ziyang Liu, Jeffrey Walker and Yi Chen, “XSeek: A Semantic XML Search Engine Using Keywords”, VLDB, pp.1330-1333, 2007.

[10] Ziyang Liu and Yi Chen, “Identifying Meaningful Return Information for XML Keyword Search”, SIGMOD, pp.329-340, 2007.

[11] Zhifeng Bao, Tok Wang Ling, Bo chen and Jiaheng Lu, “Effective XML Keyword Search with Relevance Oriented Ranking”, ICDE, pp.517-528, 2009.

[12] Martin Theobald, Ralf Schenkel and Gerhard Weikum, “An Efficient and Versatile Query Engine for TopX Search”, VLDB, pp.625-636, 2005.

[13] Tetsutaro Motomura, Toshiyuki Shimizu, and Masatoshi Yoshikawa, “Alternative Query Generation for XML Keyword Search and Its Optimization”, DEXA, pp.410-424, 2011.

[14] Umaporn Supasitthimethee, Toshiyuki Shimizu, Masatoshi Yoshikawa and Kriengkrai Porkaew, “XSemantic: An Extension of LCA Based XML

Semantic Search", IEICE TRANSACTIONS on Information and Systems, pp.1079-1092, 2009.

[15] Kenji Hatano, Hiroko Kinutani, Masatoshi Yoshikawa and Shunsuke Uemura, "Information Retrieval System for XML Documents", DEXA, pp.758-767, 2002.

[16] Kei Fujimoto, Toshiyuki Shimizu, Norimasa Terada, Kenji Hatano, Yu Suzuki, Toshiyuki Amagasa, Hiroko Kinutani and Masatoshi Yoshikawa, "An Implementation of High-Speed and High-precision XML Information Retrieval System on Relational Databases", Advances in XML Information Retrieval and Evaluation, Vol.3977 of Lecture Notes in Computer Science (LNCS), pp.254-267, 2006.

[17] 徳田 隆志, 田島 敬史, "XML タグの文書構造上の役割に関する自動分類", 日本データベース学会論文誌, Vol.8, No.1, pp.53-58, 2009.