DEIM Forum 2010 C7-3

# XMLデータに対するキーワード検索結果の分類とそれに基づく理解支援

(0)

本村徹太郎† 清水 敏之† 吉川 正俊†

† 京都大学 情報学研究科 〒606-8501 京都市左京区吉田本町

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

**あらまし** XML データに対するキーワード検索には，キーワードがテキスト値あるいは要素名にマッチするという曖昧性に起因した解釈の多様性が存在するため，検索意図と異なる検索結果が出力されうるという問題がある．また，流通している多くの XML データから横断的な検索を行う際，同一のエンティティを表現しているにもかかわらず構造が異なるデータが存在する場合があり，データ探索を行いにくくしている．本論文は，XML キーワード検索において，解釈の多様性に基づく検索結果の揺らぎを解消し，検索意図に合致した解の効率的な発見を支援するために，解釈ごとに検索結果をクラスタリングする方法を提案する．さらに，分類された結果に対して，探索的検索を支援するために，検索結果を再構築する方法を提案する．

**キーワード** XML，キーワード検索，クラスタリング

## 1. はじめに

近年，XML はデータ記述フォーマットとして急速に普及し，標準的なデータ表現の一つとして利用されている．XML 形式で流通しているデータは多種多様であり，かつデータの量は膨大である．特に，多種多様なデータの中には，データを出版している組織は異なるが，類似したエンティティを扱っている場合があるため，データを出版している組織を跨いだ，横断的な検索を行いたいという要求がある．一般的に検索対象となる XML データの量は膨大であり，その中からユーザが目的のデータを効率よく探し出すために，様々な検索方法が提案されている．

XML や HTML は半構造データと呼ばれ，テキスト値のみではなく構造も含めて記述される．キーワードを利用して XML 検索を行う場合，検索対象の XML は，データ指向 XML および文書指向 XML の二種類に大別することができる．両者は明確な定義はなされていないが，直感的には，DBLP における XML [1] はデータ指向 XML と考えることができ，INEX XML テストコレクション [2] は文書指向 XML と考えることができる．我々は，データ指向 XML におけるキーワード検索について焦点をあてている．

XML 文書の中から構造に関する情報を利用して検索を行う場合に，基本的な問合せ言語として，XPath を用いることが可能である．XPath 問合せは，パスパターンによって記述され，XML 木のノード集合からパターンに適合する部分集合を選択する．しかし，多くの一般ユーザは，XPath などの XML 特有の問合せ言語に関する知識，および検索対象の XML データの構造に関する知識を持たないため，検索意図を正確に反映したパス式を記述することが困難である．

一方で，Web 検索におけるキーワード検索のような，直観的な単語を用いて XML 検索が可能であれば，ユーザは XML に対する問合せ言語および対象とする XML 文書のスキーマ情報などの予備知識を必要とせず，キーワードのみを用いて探したい情報を探し出すことが可能となる．従来の Web における，テキストに対するキーワード検索とは異なり，XML は要素が名前を持つため，キーワードはテキスト値にも要素名にも含まれうる．その結果，XML キーワード検索では，入力として与えたキーワードがユーザが予期せぬノードとマッチし，検索意図とは異なる結果が得られることがある．例えば，図 1 のような講義情報データベースに対して，“course art”というキーワード集合による検索を行った場合，course というキーワードは，テキスト値および要素名にマッチすると考えられる．結果として，先ほどの入力は，講義名に art という単語を含む course を検索しているという解釈，course および art を含むテキストを検索しているという解釈などが可能である．また，XML キーワード検索においては，キーワードが出現する位置によって解釈が異なる場合がある．たとえば，あるユーザが先ほどと同様に，“course art” という語で講義情報データベースを検索したとき，(1) 講義名に art という単語を含む course を探しているという解釈や，(2) 講師名が art である course を探しているという解釈などが可能である．(1) に対しては，art というキーワードが title 要素の直下のテキスト値に出現した場合が適合し，(2) に対しては，art というキーワードが instructor 要素の直下のテキスト値に出現した場合が適合する．

横断的な検索を行う際には，異なる要素名であるが同じ意味を表す要素名が存在するという要素名の多様性，および，構造は異なるが，同じ意味を表している構造が存在するという，構造の多様性があると考えられる．しかし，前者はシソーラスを利用することにより解決可能であり，後者は，スキーママッチング [3] の研究を応用することで，解決することが可能であると考えられる．一方で，前述した例のように，XML キーワード検索では，キーワードが要素名にもテキスト値にも出現し得るような曖昧性，あるいはキーワードが出現する文脈によって意味が異なるという曖昧性が存在するため，入力キーワードに対し，

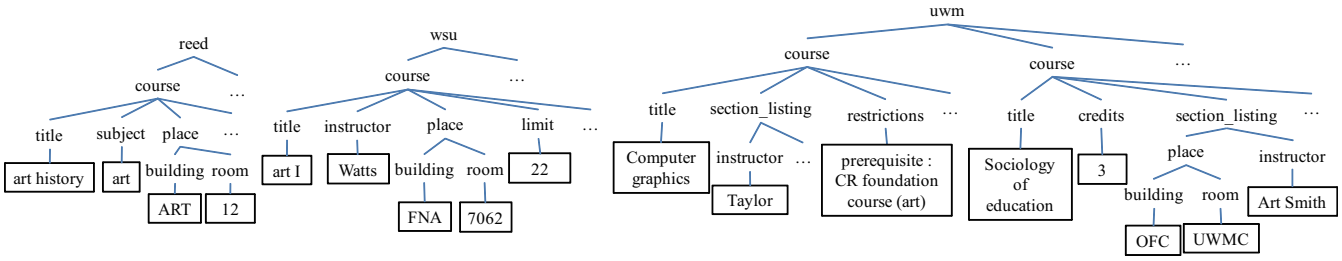


図 1　三大学の講義情報データベースの XML 木.

様々な解釈の元で検索が実行される．ユーザの検索意図を一意に特定できない限り，どの解釈が正しいかを，一概に決定することはできない．XML キーワード検索に関しては，XML 木から入力キーワードを全て含む部分木を求める研究 [4] [5] [6] [7]，さらに，入力キーワードをすべて含む部分木の中から有意な情報のみを選択した木構造を求める研究 [8] [9] [10] などがある．たとえば，SLCA (smallest lowest common ancestor) [6] のそれぞれの結果は，入力キーワードをすべて含むが，その部分木の子孫中にすべての入力キーワードを含む部分木は存在しないような，部分木である．既存研究では，複数の解釈に基づく検索結果が混在するため，ユーザは検索結果の中から自身の意図に合致する検索結果を見つけるために解を絞り込まなくてはならない．また，仮に自身の意図に合致した検索結果を見つけることができたとしても，検索結果は十分に絞り込まれているとは限らない．特に，キーワードによる検索は，検索対象のデータを予め知っていなければ，厳密な条件を記述できず，キーワードのみでは複雑な条件を指定することが困難がある．

例 1:　図 1 に示す，三大学の講義情報データベースの XML 木に対し，“course art”という入力キーワードで問い合わせたとする．仮に SLCA を採用した場合，図 2 に示す四つの結果 (a), (b), (c), (d) が得られる．ここで，(a)，(b) および (d) はルートノードが course 要素であり，(c) はルートノードが restrictions 要素である．この場合に，複数のエンティティが検索結果として得られるが，すべてのエンティティがユーザの検索意図に合致している場合は稀であるため，ユーザの検索意図に近い解のみを出力することが望ましい．しかしながら，システムが入力キーワード集合からユーザの検索意図を推定することは難しいため，ユーザの検索意図に合致する結果のみを出力することは困難である．他方，検索結果を course ごと，restrictions ごとに分類することで，ユーザは自身の検索意図に近い部分解集合を容易に選択できる．

例 1 のような，多種多様なデータを横断的に検索する際に，検索結果を解釈ごとにクラスタリングする，あるいは複数の解釈が存在することを提示することにより，ユーザは効率的に探している結果を発見することが可能になると考えられる．しかしながら，ユーザが探している検索結果のみを出力したとしてもなお，検索結果を閲覧する際には問題が生じうる．

例 2:　例 1 で用いた問合せにより得られた検索結果に対して，

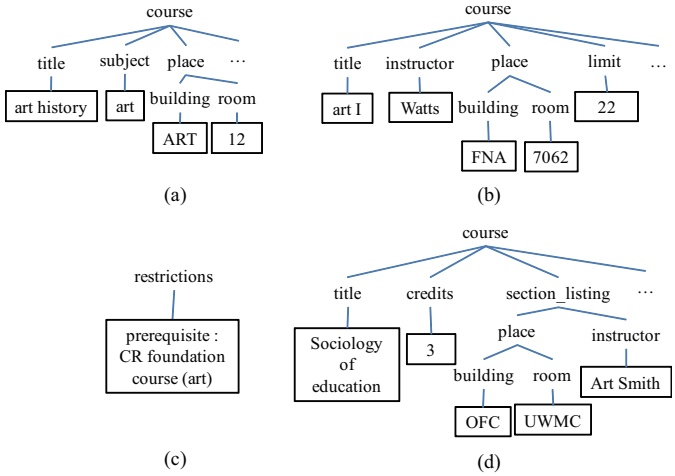


図 2　多様な XML インスタンス.

course に関するデータのみを選択したとする．この場合，図 2 の (a)，(b)，(d) などが検索結果に含まれる．しかしながら，(a)，(b)，(d) は異なる組織が出版しているデータであるため，三つすべてが course という情報に関して XML 形式で記述されているにもかかわらず，それらの構造には互いに異なっている部分がある．たとえば，(a) と (d) に注目すると，(a) には subject 要素が出現しているが，(d) には出現しておらず，逆に，(d) には credits 要素が出現しているが，(a) には出現していない．また，(a)，(d) ともに，place 要素をルートノードとする部分木を含んでいるが，place 要素が出現する位置が異なっており，同じ種類のデータを表現しているにもかかわらず，構造が異なっている．

例 2 のように，データごとに構造が異なっていては，ユーザが結果から詳細なデータを探しづらいと考えられる．この問題を解決する方法として，XML のスキーマを統一することが考えられる．しかしながら，様々な組織あるいは個人がデータを出版する中で，同じエンティティを表現するデータのスキーマを統一することは困難である．また，様々な要求に対応するため，柔軟にスキーマを変更することが可能な XML データに対して，スキーマを固定することは実利用上の観点からは非現実的な方法であると言える．

別の解決策として，構造の違いを吸収するために，さらに

キーワードを追加して解を絞り込むことが考えられる．しかし，条件を指定するためにはあらかじめデータの詳細を知っておく必要があり，また，キーワードのみでは複雑な条件を指定できない．このような場合は，キーワードを追加することによる絞り込み検索を行うことができない．

一方で，データ検索のための異なるアプローチとして，探索的検索であるファセット検索がある．ファセット検索は，絞り込みの条件としてデータの属性およびその属性が取る値をユーザが選択し，絞り込まれた部分検索結果に対してさらに条件を指定していく，という操作を繰り返して，徐々に結果を絞り込みながら解を探していく方法である．前述したように，キーワード検索において，キーワードのみでは解を絞り込むことは困難な場合があるため，検索結果に対してファセット検索の検索インタフェースを提供することは有用である．そのため，本論文は，データの効率的な発見を支援を行うために，検索結果を探索的検索が可能な形に再構築する．

以下は本論文の構成である．第 2 節では関連研究について述べる．第 3 節では解釈の違いに基づく検索結果のクラスタリング手法について，第 4 節では検索結果の部分木に対する動的な再構築手法について，第 5 節はまとめと今後の課題である．

## 2. 関 連 研 究

XML キーワード検索の基本的なアプローチとして，入力キーワードをすべて含む部分木 *Lowest Common Ancester* (LCA) を出力するという方法がある．既存研究では，部分木単位で表示するために，文献データベースにおける一本の論文や，顧客データベースにおける一人の顧客情報といった，意味としてまとまりのある部分木を求める方法が提案されてきた [5]．また，LCA の中に他の LCA を含まない最小の部分木 (SLCA) [6] を出力するという方法が提案された．これらの既存研究は解釈の違いが考慮されておらず，異なる解釈の結果が混在した検索結果が得られる．

検索結果の中に複数の検索意図が混在する問題に対して，XReal [7] は，キーワードが要素名にもテキスト値にも出現し得る曖昧性，およびキーワードが出現する文脈によって意味が異なるという曖昧性，の二つの曖昧性について考慮し，tf-idf 法を拡張した XML tf-idf 法を LCA に適用する方法を提案した．XReal では，曖昧性を解消するために，問合せ中に出現するキーワードの順序を利用して，ユーザの検索意図を推定し，意図に沿ったと判断された SLCA の集合を検索結果として返す．つまり，複数の解釈が存在した場合に，一つの解釈のみをユーザに提示していると考えることができる．それに対して，我々の手法は，複数の解釈が存在した場合に，それぞれをクラスタリングして表示する．

XML キーワード検索における検索結果の表示方法として，単純に部分木を表示するのではなく，不要な情報と考えられるノードを削除し，有意であると考えられる情報のみで構成される部分木を表示する方法が提案された [8] [9] [10]．これらの表示方法では，結果は検索対象の XML 木の構造に依存する．一方で，斉藤ら [11] は，Functional Dependencies (FD) を利用し，XML データの中から relation を取り出す方法を提案した. [11] は，ある XML 木を同値な XML 木へと変換可能であることを示唆している．

検索結果の理解支援として，Huang ら [12] は，検索対象の XML データの要約情報として，XML のスニペットを抽出する方法を提案した．また，Liu ら [13] は，複数のリポジトリから得られる検索結果に対し，Differentiation Feature Sets (DFS) と呼ばれる，検索結果を特徴づけるデータを関連情報として付加する方法を提案した．

## 3. 解釈の違いに基づくクラスタリング

XML キーワード検索は，キーワード集合のみが入力として与えられる．一般に，キーワード集合が問合せとして与えられたときに，問合せは多様な解釈がなされうる．たとえば，講義情報データベースに対して “course art” という問合せが与えられたとき，art に関する course，course および art という単語が含まれるテキスト，をそれぞれ検索しているといった，複数の検索意図が存在する．

XML キーワード検索に関する既存研究では，上述の例のような，複数の検索意図を区別せず，様々な解釈に基づくインスタンスが混在した集合をユーザに返す．ユーザは，膨大な検索結果の中から，自身の検索意図に合致した結果を探さなくてはならない．目的の結果を探すのはユーザにとって非常に負荷がかかる作業であると考えられるため，キーワードを追加して解を絞り込むことなどが考えられる．検索結果を出力する際に解釈ごとに分類して出力することができれば，ユーザは結果を理解しやすく，ユーザが求めている情報を絞り込む手間を軽減することが可能である．我々は，異なる複数の解釈が混在する結果を，解釈ごとに分類する手法を提案する．以降では，3.1 節では XML キーワード検索における解釈の違いについて，3.2 節では解釈の違いに基づく検索結果のクラスタリング方法について，それぞれ述べる．

### 3.1 解釈の違い

XML キーワード検索において，キーワードはテキスト値および要素名にマッチする．そのため，システムが入力キーワード集合からユーザの検索意図を解釈しようとした場合，複数の解釈が存在する．

例 3: 図 1 の講義情報データベースに対して，入力キーワードが “course art” であった場合について考える．SLCA によって検索結果を取得すると，図 2 に示すような多様な XML インスタンスが得られる．図 2 (a) は art という単語を title に含む course 要素（あるいは，art という単語を building に含む course 要素，art という単語を subject に含む course 要素）を示している．同様に，図 2 (b) は art という単語を title に含む course 要素を，図 2 (c) は art および course という単語を含む restrictions 要素を，図 2 (d) は art という単語を instructor 要素の子として含む course 要素を，それぞれ示している．

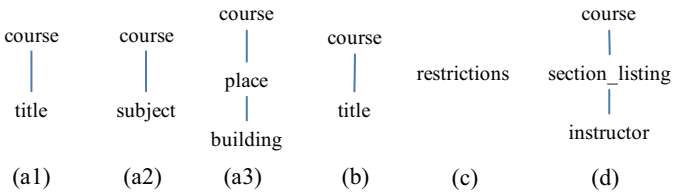


図 3　枝抜き部分木構造.

検索により四つのインスタンスが得られ，その中において，ルートノードの要素名が異なる二種類のインスタンスが得られた．これは，キーワード検索には曖昧性が存在するため複数の検索意図が存在し，解釈の違いによって異なる結果が得られたと考えることが可能である．一方，検索結果の中で，同一の解釈がなされうる結果には，共通の構造が存在する．特に，入力キーワードにマッチする要素およびテキスト値の構造内での位置は，解釈を特徴づける要因であると考えられる．たとえば，同じ解釈の元で実行されたと考えられる結果である図 2 (a) および 図 2 (b) は，ともに course を表現したデータであり，title 要素のテキスト値には “art” が含まれている．我々は，部分木構造から入力キーワードに関連した部分を抜き出した構造を，問合せの解釈に対応付けることが可能であると考えた．

検索結果の分類方法について議論する前に，本論文で用いるデータモデルについて定義する．

**定義 1** (XML 木)．ある *XML* 木 $t$ は $t = (N_t, E_t, r_t)$ として表現する．ここで，$N_t$ はラベル付けされたノード集合，$E_t$ はエッジ集合，$r_t \in N_t$ は $t$ のルートノードである．また，属性ノードは要素ノードとしてみなす．さらに，$\lambda(x)$ はノード $x \in N_t$ の要素名を，$\nu(x)$ は，$x$ の直下のテキスト値を，それぞれ表す．

**定義 2** (枝抜き部分木)．ある *XML* 木および入力キーワード集合 $K = \{k_1, k_2, ..., k_n\}$ が与えられたとき，*XML* 木に対して $K$ で問い合わせることにより，検索結果 $T$ が求まる．ある *XML* 部分木 $t = (N_t, E_t, r_t)$ $(t \in T)$ に対して，$t$ の $K$ に関する枝抜き部分木 *(pruned subtree)* $p$ を $p = (N_p, E_p, r_p)$ と定義する．ただし，$N_p$ は，$N_p \subseteqq N_t$ かつすべての $k_i \in K$ に対して，ある $n \in N_p$ が存在し，$k_i$ が $\lambda(n)$ あるいは $\nu(n)$ に含まれる，という条件を満たす．また，$E_p = (N_p \times N_p) \cap E_t$，$r_p = r_t$ である．

**定義 3** (枝抜き部分木構造)．枝抜き部分木 $p = (N_p, E_p, r_p)$ が与えられたとき，$p$ の枝抜き部分木構造は，$N_p$ からすべてのテキストノードを除いた部分木である．

例 4:　図 2 (a) に対して，入力キーワード “course art” に関する枝抜き部分木構造は，図 3 (a1)，(a2)，(a3) に示す構造である．また，図 2 (b)，(c)，(d) に対しては，それぞれ図 3 (b)，(c)，(d) に示す構造である．ここで，
図 3 (a1) は，course の title に関して検索しているという解釈，図 3 (c) は，restrictions を検索しているという解釈，図 3 (d) は，course の instructor に関して検索しているという解釈，とそれぞれ対応づけることが可能である．

例 4 に示すように，XML キーワード検索におけるある解釈は，一つの枝抜き部分木構造に対応していると考えることが可能である．我々は，枝抜き部分木構造を用いて検索結果を分類することにより，解釈の違いに基づいて検索結果を分類することが可能であると考えた．

### 3.2　枝抜き部分木構造による検索結果の分類

3.1 節では，枝抜き部分木構造が問合せの解釈に対応するとした．我々は，枝抜き部分木構造が XML インスタンスの特徴を表す指標として利用可能であると考え，この構造を用いて検索結果を分類する．一方で，枝抜き部分木構造を用いて検索結果を分類する際には，どこまで構造が一致した場合に同一のクラスタに分類するかが問題となる．

例 5:　図 3 に示した枝抜き部分木構造の中において，ルートノードの要素名が他と異なる (c) と，他の枝抜き部分木構造とでは，両者は別のエンティティを表現していると考えられるので，異なるクラスタに分類してもよいと考えられる．一方，(a)，(b)，(d) をクラスタリングする際は，例えば，(a1) および (b) は構造が全く同じであるため，(a) と (b) を同じクラスタに分類する，という方法がある．ところが，course 要素の直接の子要素に instructor 要素が出現する枝抜き部分木構造が存在した場合，上述した方法を用いると，この構造と図 3(d) は別のクラスタに分類されてしまう．

例 5 の問題を解決するために，我々は，二つのインスタンスを同一のクラスタに分類するか否かを決定する基準を複数用意する．クラスタリングを行う際は，それらの基準の中から一つ基準を選択し，その基準に基づき分類を行う．

最初に 3.2.1 節において，クラスタリングの基準を示す．その後，3.2.2 節において，クラスタリングアルゴリズムに関して述べる．

#### 3.2.1　クラスタリング基準

二つの XML 部分木インスタンス $I_1, I_2$ が与えられたとき，これらを同じクラスタに分類するか否かを判定する基準は複数考えられる．我々は，二つのインスタンスを同一とみなす基準として，以下の四つを考えた．

基準 1.　$I_1$ の枝抜き部分木および $I_2$ の枝抜き部分木の，ルートノードの要素名が同一である．

基準 2.　$I_1$ の枝抜き部分木構造が $I_2$ に包含されており，かつ $I_2$ の枝抜き部分木構造が $I_1$ に包含されている．

基準 3.　$I_1$ の枝抜き部分木構造と $I_2$ の枝抜き部分木構造が等価である．

基準 4.　$I_1$ の枝抜き部分木と $I_2$ の枝抜き部分木が等価である．

例 6:　図 1 とは異なる講義情報データベースにおいて，
“course art” というキーワードを用いて検索した結果，
図 4 (a)，(b) に示す二つのインスタンスが得られたとす

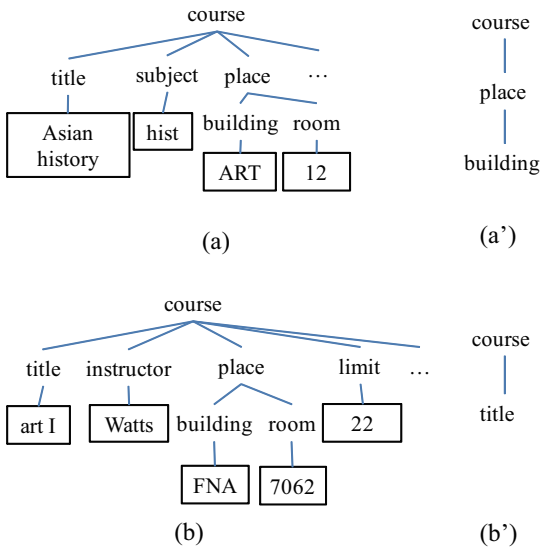


図 4　クラスタリング基準の判断例.

**Algorithm 1** 解釈に基づくクラスタリング.

**Require:** $S$ : SLCA node set.

```
1: C ← {}, P ← {}
2: for all s in S do
3:     flag ← false
4:     for all p_i ∈ P do
5:         if p_i matches s.pattern() then
6:             c_i.add(s)
7:             flag ← true
8:         end if
9:     end for
10:    if !flag then
11:        P.add(s.pattern())
12:        create new cluster c
13:        c.add(s)
14:        C.add(c)
15:    end if
16: end for
17: return C
```

る．(a) および (b) の “course art” に関する枝抜き部分木構造は，それぞれ図 4 (a') および (b') となる．(a) と (b) が同じクラスタに属するかを判断するときに，基準 1 あるいは基準 2 を採用するならば，(a) と (b) は同じクラスタであると判断され，基準 3 あるいは基準 4 を採用するならば，(a) と (b) は異なるクラスタに属すると判断される．特に，基準 2 を採用した場合，(a) は (b') を包含し，かつ (b) は (a') を包含しているため，(a) と (b) は同じクラスタに属すると判断される．一方，基準 3 を採用した場合は，(a') と (b') は等価ではないので，(a) と (b) は異なるクラスタに属すると判断される．

以降では，基準 1 を採用して議論を進めるが，他の基準を採用した場合でも同様の議論が成り立つ．

**3.2.2**　枝抜き部分木構造による分類

枝抜き部分木構造による分類方法をアルゴリズム 1 に示す．なお，ここでは検索結果として SLCA の集合が得られることを想定しているが， SLCA に限らず，ELCA [4] や他の LCA に基づく手法で得られた結果に対しても適用可能である．

今，図 1 に示す XML 木に対して，入力キーワード集合 $K = \{\text{“}course\text{”}, \text{“}art\text{”}\}$ により問い合わせたとする．このとき，SLCA のノード集合 $S$ は，図 2 の (a)，(b)，(c)，(d) である．(ここでは，簡単のため，$S = \{(a), (b), (c), (d)\}$ と表現する．) ここで，$S$ を枝抜き部分木構造の違いによりクラスタリングすることを考える．最初に，$s = (a)$ に対し，$s$ がどのクラスタに入るかを調べる（4 - 9 行目）．今回，$P = \{\}$ なので，$s$ はどの枝抜き部分木とも合致せず，新しい枝抜き部分木構造および新しいクラスタを生成する（11 - 14 行目）．クラスタを生成した結果，$P = \{\text{“}course\text{”}\}$，$C = \{\{(a)\}\}$ となる．次に，$s = (b)$ に対する処理を考える．現在 $P = \{\text{“}course\text{”}\}$ より，$p_1 = \text{“}course\text{”}$ である．$s$ の枝抜き部分木構造 $s.pattern()$ は，図 3(b) に示す枝抜き部分木構造であり，そのルートノードの要素名は course である．したがって，基準 1 に従い，$p_1$ と $s$ の枝抜き部分木構造は合致するため，$s$ をクラスタ $c_1$ に加える．(b) に対して処理が終了した段階で，$c_1 = \{(a), (b)\}$ となる．さらに，$s = (c)$ に対する処理に関して考える．今，$P = \{p_1\} = \{course\}$ である．$s$ の枝抜き部分木構造 $s.pattern()$ は，図 3 に示す構造であり，そのルートノードは restrictions である．ゆえに，$flag$ は $false$ のままであり，新しいクラスタを生成する（10 - 15 行目）．最初に，$s.pattern()$ を $P$ に追加し（11 行目），新しいクラスタ $c$ を生成する（12 行目）．その上で，$c$ に $s = (c)$ を追加し，クラスタ集合 $C$ に $c$ を加える．結果として，$P = \{\text{“}course\text{”}, \text{“}restrictions\text{”}\}$，$C = \{\{(a), (b)\}, \{(c)\}\}$ となる．以上の操作を繰り返すと，(d) に対する処理を終えた時点で，$C = \{\{(a), (b), (d)\}, \{(c)\}\}$ という結果が得られる．

## 4.　検索結果の部分木の動的な再構築

XML キーワード検索においては，検索結果内に多様な解釈の元で検索された結果が混在しうる．我々は，第 3 節で示したように，解釈の違いに基づきクラスタリングすることによって，解の効率的な発見を支援する．しかし，解釈ごとに分類してもなお，検索結果の中には多種多様なデータが含まれうる．特に，様々な XML データを横断的に検索する場合においては，各 XML データ内に記述されているデータの構造は特有である場合が多い．それゆえ，検索対象とする XML データ群には，同じエンティティを表現しているデータであるにも関わらず，異なる構造を持つデータが多数存在する．検索結果をユーザが閲覧する際に，データごとに構造が異なっていては，ユーザが注目する詳細なデータがどこに出現しているか分からず，データを探索しづらいと考えられる．

キーワードによる検索を行った際，ユーザが望む検索結果を

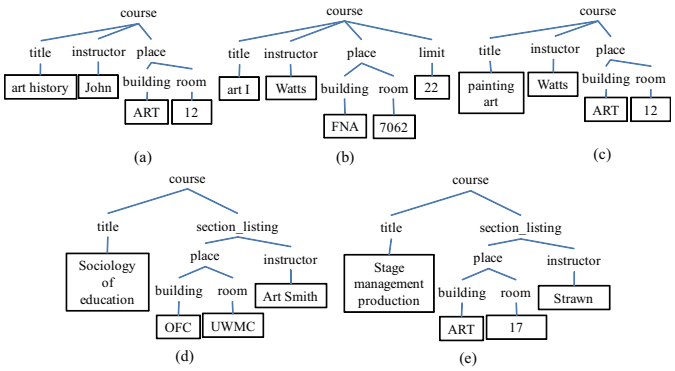


図 5　“course art”による検索結果の例.

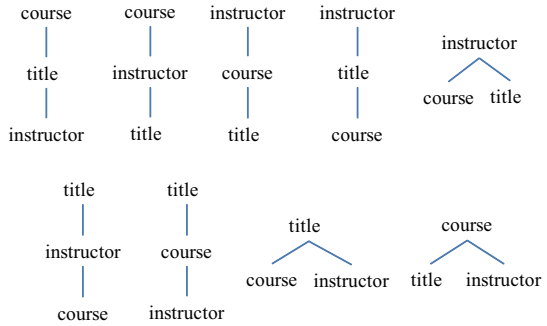


図 6　course, title, instructor に関する amoeba 構造 [11].

すぐに発見することができなかった場合は，膨大な検索結果の中から解を一つずつ閲覧していかなくてはならない．データを一つずつ確認していくことは非常に労力を伴う作業であるので，ユーザはさらにキーワードを追加するなどして，解を絞り込むと考えられる．しかしながら，キーワードを追加し，より厳密な条件を指定する方法は，事前に検索対象データの詳細を知っていなければ，適切なキーワードを指定することができない．この場合は，検索対象データに対する知識がなくとも検索可能な，探索的検索手法が有効である．

XML データに対して，探索的検索を適用する際，前述した，構造の異なるデータが存在する場合，統一した処理を適用することが困難となる．そのため，我々は，検索結果を要素名およびテキスト値の組に分解し，統計的指標を利用することにより，効率的な探索的検索が可能な探索木へと再構築する方法を提案する．

第 3 節では，入力キーワードに対し，複数の解釈が存在すると述べた．以降では，特に断らない限り，検索結果はクラスタリングにより解釈ごとに分類されており，一つの解釈に対応した結果が得られていることを前提とする．部分木の再構築は，最初に部分木集合から属性集合を抽出し，各属性の下で出現する値および値の出現頻度を計算する．その上で，属性集合の各要素に対して，効率的な探索が可能であると考えられる属性を求め，優先順位を付けていく．最後に，優先順位に従って部分木集合の各部分木を組み替え，組み替えられた結果を出力する．

優先順位を求める際は，最初に各属性における出現頻度の特徴から，効率的な探索が可能な属性の度合いを数値化し，比較する必要がある．我々は，そのため，値の出現頻度を出現確率とみなし，ある属性の探索を示す指標として，平均情報量を用いることにした．また，我々は，効率的な探索が可能な属性が備えている性質として，以下の二点があると考えた．

（1）　同一の属性の下で出現する，値の異なる個数が少ない．

（2）　同一の属性の下で出現する，値の出現頻度として，大きい値が多く出現する．

(1) に関しては，ユーザは検索対象データに対する事前知識がほとんどないと考えられるため，異なる値が出現する個数は少ない方が探索時に選択しやすいと考えられるためである．(2) に関しては，値の出現頻度が大きい値が多く出現する属性，つまり，同じ値が何度も出現する属性は，探索する際にその属性の値を選択することにより，その属性の下で出現する，値が異なる個数が少なくなると考えられる．上述した条件を満たす属性は，平均情報量の値が小さくなる．そのため，平均情報量の値が小さい属性の優先順位を高く設定する．

部分木の組み替えに関しては，組み替え後の部分木として非常に多くの候補が考えられるため，どの部分木に組み替えるかを選択する必要がある．一般に，$n$ 個の要素からなるノード集合に対し，ノード集合中のあるノードをルートノードとする部分木は，$n^{n-1}$ 通り存在する．ノード集合 $N = \{n_1, n_2, ..., n_k\}$ に対し，ノード $n_l$ が他のノードの共通の祖先であるとき，$N$ は *amoeba* [11] であると言う．year 要素, author 要素, title 要素からなるノード集合が与えられたとき，このノード集合の要素数は $n = 3$ であるから，図 6 に示す通り，$3^{3-1} = 9$ 通りの amoeba 構造が存在する．ここで，任意の amoeba 構造 $x, y$ に対し，$x$ のノード集合と $y$ のノード集合が等しいとき，$x \sim y$ と定義する．このとき，二項関係 $\sim$ は amoeba 集合の同値関係である．我々は，検索結果のインスタンスを amoeba 構造とみなし，この amobea 構造と同値である amoeba 構造の中で，優先順位に従った属性の配置である amoeba 構造に変換し，変換後の構造に基づき結果を出力することで，再構築を行う．

我々が提案する検索結果部分木の要素の優先順位決定方法を，アルゴリズム 2 に示す．アルゴリズム 2 では，最初に，部分木中に出現する要素名を属性集合 $E$ としてあらかじめ取得しておく．その上で，各要素名に対して直下のテキスト値を取得する．最後に，要素名ごとに平均情報量を求め，平均情報量の値が最も小さい要素名を順に $Output$ に加えていく．ここで，アルゴリズム 2 では，リストは [ ] によって，集合は {} によって，連想配列は $Map < key, value >$，あるいは $\{key => value\}$ によって表現する．

今，入力キーワード “course art” を用いて検索し，解釈に基づくクラスタリング手法により分類した結果の一部として，図 5 に示す五つの結果が得られたとする．最初に，図 5(a) の部分木に対し（4 行目），(a) のノード集合からノードを取り

**Algorithm 2** 検索結果部分木の要素の順序決定方法.

**Require:** $T$: List of result subtrees,
$E$: All element names occurring in result sets,
$\theta$: Threshold of non-null value ratio,
$\kappa$: Threshold of occurrence of non-null value.

1: $Output \leftarrow [\,]$
2: **while** $E$ is not empty **do**
3: $C = Map < String, Map < Set, Integer >>$
4: **for all** $t$ in $T$ **do**
5: **for all** $n$ in $N_t$ **do**
6: $prefix \leftarrow generatePrefix(Output, N_t)$
7: $prefix.add(\nu(n))$
8: $c \leftarrow C.get(\lambda(n))$
9: **if** $c.containsKey(prefix)$ **then**
10: $c.get(prefix) \leftarrow c.get(prefix) + 1$
11: **else**
12: $c.add(prefix, 1)$
13: $C.add(\lambda(n), c)$
14: **end if**
15: **end for**
16: **end for**
17: **for all** $e$ in $E$ **do**
18: **if** $p_{null} \leqq \theta$ or counts of non-null value $\geqq \kappa$ **then**
19: calculate entropy $H_e$
20: **end if**
21: **end for**
22: $Output.add$(E.$getSmallestEntropyNode$)
23: $E.remove$(E.$getSmallestEntropyNode$)
24: **end while**
25: **return** $Output$

**Algorithm 3** $generatePrefix(A, N_t)$.

**Require:** $A$: List of attribute names,
$N_t$: Node set of an XML tree t.

1: $prefix \leftarrow \{\}$
2: **for all** $a_i$ in $A$ **do**
3: **for all** $n$ in $N_t$ **do**
4: **if** $a_i = \lambda(n)$ **then**
5: $prefix.add(\nu(n))$
6: **end if**
7: **end for**
8: **end for**
9: **return** $prefix$

出す（5 行目）．ここでは，$n$ として要素名が course のノードが得られたとする．一つノードを得たら，$generatePrefix$ によりキーを生成し，キーを用いて単語の出現回数を調べる（6 - 7 行目）．$generatePrefix$ の挙動をアルゴリズム 3 に示す．$generatePrefix$ は属性名のリスト $A$，およびある XML 木 $t$ のノード集合 $N_t$ が与えられたときに，各属性 $a \in A$ の値を $N_t$ から求め，それらを集合として返す．今回，$Output = [\,]$ より，アルゴリズム 3 における $generatePrefix$ の $A$ も空リストとなる．そのため，$generatePrefix(Output, N_t)$ の結果は空集合となるので，$prefix = \{\}$ となる（6 行目）．また，$\nu(n) = null$ より，$key = prefix.add(\nu(n)) = \{null\}$ となる（7 行目）．$c = \{\}$ であるため，$c$ は $key$ の値を含まず（8 行目），$C = \{\{course \Rightarrow \{\{null\} \Rightarrow 1\}\}\}$ となる（12 - 13 行目）．(a) に含まれるすべての要素名に対して上述した処理を行うと，(a) に関する処理が終わった段階で，$C = \{\{course \Rightarrow \{\{\{null\} \Rightarrow 1\}\}\}, \{title \Rightarrow \{\{\{\text{“art history”}\} \Rightarrow 1\}\}\}, \{instructor \Rightarrow \{\{\{\text{“}John\text{”}\} \Rightarrow 1\}\}\}, \{place \Rightarrow \{\{\{null\} \Rightarrow 1\}\}\}, \{building \Rightarrow \{\{\{\text{“}ART\text{”}\} \Rightarrow 1\}\}\}, \{room \Rightarrow \{\{\{\text{“12”}\} \Rightarrow 1\}\}\}\}$ となる．同様にして，他の四つのインスタンスに対しても処理を適用していくと，最終的に，表 1 に示す結果が得られる．ここで，frequency は，各値が出現した回数を，その値が出現した要素が出現した回数で割った数値である．最後に，各要素名のうち，ノイズとなる要素を取り除くため，次の二つの条件：(1) null 値の割合が閾値 $\theta$ 以下である，(2) null ではない値が閾値 $\kappa$ 以上出現する，のいずれかを満たす属性名に対し（18 行目），平均情報量 $H$ を $H = -\Sigma_{i=1}^{n} p_i \cdot log_2(p_i)$ で算出する（19 行目）．今回は，$\theta = 0.5$，$\kappa = 2$ とする．ここで，$p_i$ は出現頻度である．たとえば，表 1 において，course 要素，place 要素，section-listing 要素は， null 値のみしか含んでいないため，平均情報量の計算は行わない．また，limit 要素は，$\kappa = 2$ 回以上出現していないため，同様に計算を行わない．平均情報量を計算すると，title に関しては，$H_{title} = -0.2 \cdot log_2(0.2) - 0.2 \cdot log_2(0.2) - 0.2 \cdot log_2(0.2) - 0.2 \cdot log_2(0.2) = 2.32$ となる．同様に，instructor，building，room に関して平均情報量を計算すると，$H_{instructor} = 1.92$，$H_{building} = 1.37$，$H_{room} = 1.92$ となる．$H$ の値が一番小さい要素を $Output$ に加え（22 行目），$E$ から取り除く（23 行目）．今回は，平均情報量の値が一番小さい要素は building 要素であるので，building を $Output$ に加え，$E$ から取り除く．ここで，$getSmallestEntropyNode$ は，属性集合の中から，一番平均情報量の値が小さい属性を取得する．

続いて，二番目に平均情報量が低い要素を求める．今，$E = \{author, title\}$，$Output = [building]$ である．$t$ として図 5 (a) のインスタンスからノード集合を取得し，ノード $n$ として，要素名が course であるノードを取り出す（5 行目）．次いで，$generatePrefix$ により $prefix$ を生成する（6 行目）．$generatePrefix$ では，$A = Output = [building]$ の各属性名 $a_i$ に対し，各 $n \in N_t$ に対し，$a_i$ と $n$ の要素名が同一であった場合，$n$ の値を $prefix$ に加える．$t$ は図 5(a) のインスタンスであるので，$generatePrefix$ は $prefix = \{\text{“}ART\text{”}\}$ を返す．また，$key = \{\text{“}ART\text{”}, null\}$ である（7 行目）．以降の処理は，上述した手順と同じである．(a) と同様に，他の四つのインスタンスに対しても処理を適用していくと，最終的に，表 2 に示す結果が得られる．ただし，表 2 では，前述した二つの条件：(1) null 値の割合が閾値 $\theta$ 以下である，(2) null ではない値が閾値 $\kappa$ 以上出現する，を満たしていない属性は，掲載していない．表 2 から，title，instructor および room に関して平均情報量を計算すると，$H_{title} = 2.32$，$H_{instructor} = 2.32$，

表 1　各要素に対する値の出現頻度.

| attribute name | value | count | frequency ($p_i$) |
|---|---|---|---|
| course | null | 5 | 1.0 |
| title | art history | 1 | 0.2 |
| | art I | 1 | 0.2 |
| | painting art | 1 | 0.2 |
| | Sociology ... | 1 | 0.2 |
| | Stage ... | 1 | 0.2 |
| instructor | John | 1 | 0.2 |
| | Watts | 2 | 0.4 |
| | Art Smith | 1 | 0.2 |
| | Strawn | 1 | 0.2 |
| place | null | 5 | 1.0 |
| building | ART | 3 | 0.6 |
| | FNA | 1 | 0.2 |
| | OFC | 1 | 0.2 |
| room | 12 | 2 | 0.4 |
| | 7062 | 1 | 0.2 |
| | UWMC | 1 | 0.2 |
| | 17 | 1 | 0.2 |
| section-listing | null | 2 | 1.0 |
| limit | 22 | 1 | 1.0 |

表 2　各要素に対する値の出現頻度（2 回目）.

| attribute name | prefix | value | count | frequency |
|---|---|---|---|---|
| title | ART | art history | 1 | 0.2 |
| | FNA | art I | 1 | 0.2 |
| | ART | painting art | 1 | 0.2 |
| | OFC | Sociology ... | 1 | 0.2 |
| | ART | Stage ... | 1 | 0.2 |
| instructor | ART | John | 1 | 0.2 |
| | FNA | Watts | 1 | 0.2 |
| | ART | Watts | 1 | 0.2 |
| | OFC | Art Smith | 1 | 0.2 |
| | ART | Strawn | 1 | 0.2 |
| room | ART | 12 | 2 | 0.4 |
| | FNA | 7062 | 1 | 0.2 |
| | OFC | UWMC | 1 | 0.2 |
| | ART | 17 | 1 | 0.2 |

$H_{room} = 1.92$ となり，room 要素が値が小さいため，room を $Output$ に加える．

最終的に，$E$ が空になるまで繰り返すと，$Output = [building, room, instructor, title]$ となる．それゆえ，検索結果を再構築すると，図 7 に示すように，ルートノードに近い方から順に，building，room，instructor，title となる．

## 5. おわりに

我々は，XML キーワード検索の解釈の多様性から生じる，解釈の異なる検索結果に対して，枝抜き部分木を用いて分類する方法を示した．分類された結果は，ユーザの検索意図に合致した解を効率的に発見することを支援することが可能であると考えられる．また，クラスタリングにより分類された各検索結果に対し，ユーザに対する検索結果の理解支援を行うために，動的に部分木を再構築する方法を示した．

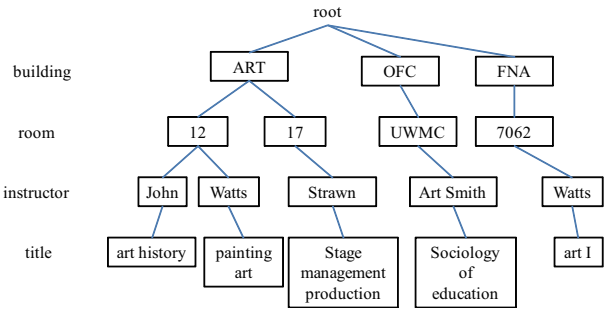

図 7　再構築結果の例.

今後は，検索結果のさらなる理解支援を行うために，検索結果の大まかな情報を把握することが可能な属性を抽出し，概観情報として付加することを考えている．概観情報は，検索結果を網羅的に把握している属性と考えられるので，ファセット検索における，ファセットの自動抽出などに適用可能であると考える．さらに，抽出された概観情報の検索結果の理解支援に対する有効性を検証することも課題である．

## 文　献


[1] M. Ley: “The DBLP computer science bibliography”. http://www.informatik.uni-trier.de/ ley/db/.

[2] INEX: “Initiative for the evoluation of XML retrieval”. http://inex.is.informatik.uni-duisburg.de/.

[3] J. Madhavan, P. A. Bernstein and E. Rahm: “Generic schema matching with cupid”, VLDB, pp. 49–58 (2001).

[4] L. Guo, F. Shao, C. Botev and J. Shanmugasundaram: “XRANK: ranked keyword search over XML documents”, SIGMOD Conference, pp. 16–27 (2003).

[5] Y. Li, C. Yu and H. V. Jagadish: “Schema-free XQuery”, VLDB, pp. 72–83 (2004).

[6] Y. Xu and Y. Papakonstantinou: “Efficient keyword search for smallest LCAs in XML databases”, SIGMOD Conference, pp. 527–538 (2005).

[7] Z. Bao, T. W. Ling, B. Chen and J. Lu: “Effective XML keyword search with relevance oriented ranking”, ICDE, pp. 517–528 (2009).

[8] V. Hristidis, N. Koudas, Y. Papakonstantinou and D. Srivastava: “Keyword proximity search in XML trees”, TKDE, **18**, 4, pp. 525–539 (2006).

[9] Z. Liu, J. Walker and Y. Chen: “XSeek: a semantic XML search engine using keywords”, VLDB, pp. 1330–1333 (2007).

[10] U. Supasitthimethee, T. Shimizu, M. Yoshikawa and K. Porkaew: “XSemantic: an extension of LCA based XML semantic search”, IEICE Transactions, **92-D**, 5, pp. 1079–1092 (2009).

[11] T. L. Saito and S. Morishita: “Relational-style XML query”, SIGMOD Conference, pp. 303–314 (2008).

[12] Y. Huang, Z. Liu and Y. Chen: “Query biased snippet generation in XML search”, SIGMOD Conference, pp. 315–326 (2008).

[13] Z. Liu, P. Sun and Y. Chen: “Structured search result differentiation”, PVLDB, **2**, 1, pp. 313–324 (2009).