WebDB の Query Form におけるメタデータ自動抽出

Automatic Metadata Extraction from Query Form of WebDBs

中藤 哲也[†] 大森 敬介[‡] 廣川 佐千男[§]
Tetsuya NAKATOH Keisuke OHMORI
Sachio HIROKAWA

各フィールドの属性値を指定することによりレコード検索を行うことができる Web 上のデータベース（WebDB）が増えている．WebDB が提供するレコードの各フィールドの属性名のリストは，その WebDB のメタデータと考えることができ，Web インタフェースの背後にあるデータベースを理解するために必須である．多数の WebDB に対し，分類や選択を行ったり，同種の（homogeneous）WebDB あるいは異種（heterogeneous）の WebDB のメタサーチを構築するためにも，このメタデータは必須である．本論文では WebDB の検索画面を構成する HTML のフォームからメタデータを自動的に抽出する方式を提案する．また，国内の 2,800 件の WebDB から無作為に選んだ 134 件のサイトについて，メタデータ抽出実験を行い，適合率，再現率，F 値の 3 つの観点から抽出性能を評価する．

There are increasing number of WebDBs (Web Databases) which return a list of records with respect to a complex query that specifies each field of records to be searched for. The attribute set of a WebDB can be thought as a metadata and is important to understand the database behind the Web interface. Moreover, it is indispensable to classify, to chose and to create homogeneous and/or heterogeneous metasearch engines from a large pool of WebDBs. This paper proposes a method which extracts the metadata from the HTML form of the WebDB. An empirical evaluation is conducted by assessing precision, recall an F-measure of extracted attributes for 134 sites randomly chosen from 2,800 WebDBs.

1. はじめに

WWW 上にはキーワードによる検索が可能な Web ページが数多く存在する．そのようなサイトは特定のテーマに限定した質の高い情報やサービスを提供している事が多く，またそれらの持つ情報量は直接参照可能な Web ページの持つ情報量よりも多いと言われている．しかしながら，その検索結果の Web ページの多くは背後のデータベースから動的に生成されるため直接参照することができず，統合などの再利用が容易ではない．それゆえ，それらの Web ページは Invisible Web, Deep Web, Hidden Web などと呼ばれ，その利用手法が重要な研究テーマになっている．更に近年の動向として，これまでのキーワード検索とは異なる新しい方向性もみられる．検索クエリーとして複数の異なる項目にキーワードを指定することで，URL の単純な一覧ではなく，複数の項目から構成されたレコードの一覧を返す検索サイトの増加である．例えば，Amazon.com は本のリストを返し，kakaku.com は PC のリストとともにそれらの価格を返す．Travelocity は指定されたエリアのホテルのリストを返す．単純なキーワード検索を用いたサイトと区別し，Web のインタフェースを持つデータベースという意味でこれらのサイトを特に **WebDB**（Web Database）と呼ぶ．

我々は現在，これらの WebDB の情報を連携・結合することを目的に研究を進めている．WebDB の結合により，利用者は自分が定常的に使うサイトを組み合わせたものを単一のサービスとして使えるようになり，使い勝手が向上する．提供側としては，利用者数と利用頻度の向上が期待できる．例えば，PC パーツに関する複数の WebDB を統合することで，最も安い PC パーツを扱う店を常に手早く探すことができるようになる．普段自分が用いるホテル予約と航空機予約を組み合わせれば，自分の好みに沿った出張の準備を手早く行うことが可能となる．

一方，WWW 上の情報サービスの新しい形として近年 Web サービスが注目されており，Web サービスの連携に関する研究がなされている [5]．しかしながら，Web サービスはイントラネットでの運用が主であり，公開され利用可能な Web サービスはごく限定的である．今後公開される Web サービスの増加には期待ができるものの，それを上回る多数のサイトで人間に対するユーザインタフェースを用いた情報サービス，すなわち WebDB が提供され続けると考えられる．したがって，WebDB の連携・結合に関する研究は，Web サービスの発展に反するものではなく，むしろそれらと連携することで，より機能的に充実したサービスの構成を可能とする重要な研究テーマである．

これらの WebDB が持つ機能・サービスの動的な連携・結合を行うためには，主に（1）WebDB への検索実行の自動化，（2）WebDB の検索結果の出力ページからのレコードの抽出*，の 2 つの機能が必要である．本稿ではこれらのうち，WebDB への検索実行の自動化に必要な機能として，入力フォームの自動的な分析，及び入力フィールドのメタデータ抽出を扱う．従来のデータベース，あるいは Web サービスであれば，データの受け渡し方法（HOW），及びデータスキーマ（WHAT）が明示的に与えられている．しかし，WebDB ではブラウザ経由での人間による利用しか想定されていないため，それらの情報は一般に明示されない．したがって，データの受け渡し方法，及びメタデータは，HTML を用いて記述された入力インタフェースから自動的に抽出する必要がある．

我々は，属性の異なる複数のクエリー入力を持った入力フォームの現状についての調査を行ってきたが，その過程で得られた知見から，そのような入力フォームの多くが `TABLE` タグを用いた表で記述されていることに着目した．従来の研究 [4, 1, 7] においては，入力フィールドのメタデータとして，各入力フィールドに近接する文字列を想定している．しかしながら，複数の入力フィールドを持つサイトでは，`TABLE` タグを用いて表示画面の構造を記述している事が多い．具体的な調査 [8] でも，調査した 2,800 サイト中 1,359 サイト，すなわち 48.5% のサイトにおいて，入力フィールドを内部に持つ `TABLE` タグが存在している．また一般的な HTML の表からの情報抽出に関する研究 [2, 6] などからも，同様の事が示唆される．WebDB の入力フォームの多くが表で記述されている事実に基づいて，新たに考案した抽出方

† 正会員 九州大学情報基盤センター
nakatoh@cc.kyushu-u.ac.jp
‡（現）三菱電機情報ネットワーク株式会社
ohmori.keisuke@mind.co.jp
§ 九州大学情報基盤センター
hirokawa@cc.kyushu-u.ac.jp

* この点に関しての研究も別途行っている [3]．

法を第 2 章で示す．更に，第 3 章で国内の 2,800 件の WebDB から無作為に選んだ 134 件のサイトに対し人手により抽出したメタデータと提案手法で抽出したメタデータとの比較，及びナイーブな手法と提案手法との比較を行い，提案手法の抽出性能を評価する．

2. メタデータの自動抽出

2.1 コンポーネントリストの生成

WebDB での自動的な検索を可能とするには，WebDB の入力画面における各入力フィールドの属性名（入力データのメタデータ）が必要である．しかし，それらは入力画面の HTML 上では明確な関連付けがされておらず，自明ではない．したがって，それらメタデータは入力フィールドの HTML が持つ幾つかの手掛りから推定し抽出する必要がある．

入力フィールドのメタデータを抽出するため，まず各入力フォーム中における入力フィールド，及びメタデータ候補をリストアップする．しかし HTML におけるタグは表示をコントロールする側面が強く，タグのままでは解析に必要な論理的な単位として取り扱うことができない．また，本稿では TABLE タグが構成する表構造に注目してメタデータの抽出を行うが，その処理の為に各候補は表中における位置を保持する必要もある．

我々は，それらを統一的に取り扱うために，図 1 に示すデータ構造を持ったコンポーネントを定義した．テキスト入力要素はそれ自体独立したコンポーネントとする（TextInput）．タグの終了を表す “>” とタグの開始を表す “<” の間に現れる文字列，すなわち “>[^<]*<” 中のパターン “[^<]*” で表現される文字列はメタデータと成り得るので，候補としてコンポーネント化する（String）．“radio” や “checkbox” といった属性を持つ INPUT 要素は，name 属性が等しい複数の要素から一つ以上の情報が選択的に入力されるので，それら要素をまとめて 1 つのコンポーネントとする（Radio と Checkbox）．同様に SELECT 要素は，その内部に持った複数の OPTION 要素の内容を選択肢としてプルダウンメニューを構成するので，まとめて 1 つのコンポーネントとする（Select）．各コンポーネントは，位置の情報として表の番号（Table_No），表中の行（Row）と列（Col）を共通して持つ．また，各要素の NAME 属性の値をコンポーネントの Name の値とするが，String コンポーネントについてはその文字列自身を Name の値とする．Radio，Checkbox，Select の各コンポーネントはその選択肢のリストを OptionList として持つ．

```
Component ::= TextInput | String | Select |
              Radio | Checkbox
TextInput ::= (Name, Table_No, Row, Col)
String ::= (Name, Table_No, Row, Col)
Radio ::= (Name, OptionList, Table_No, Row, Col)
Checkbox ::= (Name, OptionList, Table_No, Row, Col)
Select ::= (Name, OptionList, Table_No, Row, Col)
```

図 1 コンポーネントの定義

Fig. 1 Definition of Component

入力画面の HTML からコンポーネントへの変換を行う前に，その取り扱いを容易にするために前処理を行う．まず，文字の修飾など表示のみをコントロールするタグ † を取り除き，そのコンテンツ（文字列）部分のみ残す．次に，表構造の整形を行う．内容を持たないセルだけからなる行，及び列を削除する．加えて，colspan 属性や rowspan 属性を用いて結合されたセルを単純なマトリクスになるよう分割し，内容もコピーする．

更に，図 1 の定義に従って，前処理を終えた HTML を一連の番号を持ったコンポーネントに変換する．この一連のアルゴリズムを図 2 に示す．最初に呼ばれる Pre_Process では，前処理を行った上で要素リストを生成し，ExtractComponents へ引き渡す．同手続きは要素毎に ExtractComp*を呼び出す．ExtractComp*は，得た要素が TABLE であれば入れ子の表としての再帰処理を行い，そうでなければ MakeComp を呼び出す．MakeComp では，まず name 属性が等しい Radio や Checkbox 要素があればそれらをまとめる．その後，実際にコンポーネントを生成（MakeComponent）し，それをコンポーネントリストに追加する．

```
Pre_Process() {
  ListOfForms = FindForms(HTML);
  Global_Table_No = 0;
  ComponentList = ();
  foreach F in ListOfForm {
    Delete TAGs for presentation;
    Rebuild TABLE to matrix;
    Translate TABLE to Element;
    Translate INPUT and SELECT to Element;
    Translate String to Element;
    ElementList = FindElements(F);
    ExtractComponents(ElementList,
                      Global_Table_No, 0, 0);
  }
}

ExtractComponents(ElementList, Table_No,i, j) {
  while (E = shift(ElementList))
      ExtractComp*(E, Table_No, i, j);
}

ExtractComp*(E, Table_No, Row, Col) {
  if (type(E)= "TABLE") {
    Global_Table_No++;
    for (i = 1; i ≤ RowSize(E); i++) {
      for (j = 1; j ≤ ColumnSize(E); j++) {
        ExtractComponents(E(i, j),
                          Global_Table_No, i, j);
      }
    }
  } else {
    ComponentParts = ();
    push(ComponentParts, E);
    MakeComp(ComponentParts, Table_No, Row, Col);
  }
}

MakeComp(CP, Table_No, Row, Col) {
  if ((type(CP[1]) = type(ElementList[1])) and
      (name(CP[1]) = name(ElementList[1]))) {
    push(CP, shift(ElementList));
    MakeComp(CP, Table_No, Row, Col);
  } else {
    push(ComponentList,
         MakeComponent(CP, Table_No, Row,
         Col));
  }
}
```

図 2 前処理アルゴリズム

Fig. 2 Pre-processing Algorithm

† 現在は，<SCRIPT>, <!>, <FONT>,<LABEL>, <B>, <I>, <U>, <S>, <A>, <TT>, <SUP>, <SUB>, <NOBR>, <CENTER>, <ADDRESS>の各タグ（終了タグを含む）を予め取り除いている．

2.2 メタデータの抽出

前節で生成したコンポーネントリストに対して，コンポーネント同士の位置関係を解析し，各入力要素コンポーネントに対するメタデータ候補のコンポーネントを抽出する．メタデータの位置としては，もっともナイーブな手法で用いられる直前と，本稿で提案する表の構造に基づく上端，左端の 2 ヶ所，計 3 ヶ所を対象とする．すなわち，位置 i にあるコンポーネントを C_i については，表の構造に関わらず左隣の要素 C_{i-1} がメタデータ候補となる．メタデータ候補の C_{i-1} が `TextInput` であった場合は，特例としてその C_{i-1} のメタデータ候補をそのまま受け継ぐ [‡]．表中の位置 (j,k) のコンポーネント $C_{j,k}$ については，$C_{j,0}$ と $C_{0,k}$ をメタデータ候補とする．この抽出アルゴリズム全体の概略を図 3 に示す．

```
Main(){
    Pre_Process() ;
    for (i = 1, i ≤ SizeOf(ComponentList), i++) {
        ExtractMetadata(i, Table_No, Row, Col);
    }
}

ExtractMetadata(i, Table_No, Row, Col) {
    if (type(C_{i-1}) = "TextInput") {
        ExtractMetadata(i − 1, Table_No, Row, Col);
    } else output i, C_{i-1} ;
    output i, GetComponent(Table_No, 0, Col);
    output i, GetComponent(Table_No, Row, 0);
}
```

図 3 メタデータ抽出アルゴリズム

Fig. 3 Metadata Extraction Algorithm

3. 抽出アルゴリズムの評価

前章で述べたメタデータ抽出アルゴリズムの評価を行うため，本アルゴリズムを実装し実験を行った．

評価実験の対象として，収集済みの WebDB 2,800 件 [8] から，複数のテキスト入力フィールドを持つ Web データベース 150 件を無作為に選び，その検索ページの HTML ファイルを取得した．150 件のうち，有効な HTML を取得できた検索ページ 134 件を今回の評価実験の対象とした．

また，テストベッド等が存在しないため，メタデータ抽出に対する正解を生成する必要がある．多数の WebDB サイトの管理者から個別に情報を入手するのは困難である．加えて，入力フィールドのメタデータは，背後のデータベースのメタデータや出力のメタデータとは異なっている可能性があり，入力画面のみから情報を得る必要がある．したがって，本稿では人手によってメタデータを抽出し，評価実験における正解とした．

3.1 人手によるメタデータの抽出

抽出手順を示す．(1) 134 件の WebDB それぞれにテキスト入力フィールドを識別する ID，プルダウンメニューを識別する ID を付加する．(2) 作業員 5 名は，134 件の WebDB をそれぞれ閲覧，各テキスト入力フィールドのメタデータとして適切な文字列あるいはプルダウンメニュー[§]を判断し，その文字列自身かプルダウンメニューのいずれかの ID を記録する．(3) 各テキスト入力フィールドのメタデータとして記録された 5 人分のデータのうち，3 人以上で一致したデータを正解として採用する．

評価対象とした 134 サイトには，714 のテキスト入力フィールドがあるが，人手によるメタデータの抽出結果が，5 人全員で一致したのは 498 フィールド，4 人以上で一致したのが 564 フィールド，3 人以上で一致したのは 638 フィールドであった．過半数，すなわち 3 人以上で一致したデータは 124 サイトの 638 テキスト入力フィールドであり，それを正解として，実際に抽出プログラムの評価対象とした．

3.2 提案手法の再現率，適合率，F 値による評価

人手により作成したメタデータを正解とし，提案手法によるメタデータ抽出結果の再現率（Recall）R，適合率（Precision）P，および F 値（F-measure）F の 3 つの値を求めた．各値の計算式は，一般的な情報検索における定義にならい，それぞれ以下のように定義した．

ある WebDB において n 個のテキスト入力フィールドがあるとする．各 $i = 1, 2, ..., n$ について，人手で準備した正解例のメタデータの集合を H_i，本アルゴリズムにより抽出されたメタデータ候補の集合を A_i とする．このとき，その WebDB における R，P，F は次の式 $R = 1/n \cdot \sum_{i=1}^{n} |H_i \cap A_i|/|H_i|$，$P = 1/n \cdot \sum_{i=1}^{n} |H_i \cap A_i|/|A_i|$，$F = RP/2(R+P)$ で表される．

実験の結果得られた F 値を図 4 にグラフ化した．横軸は F 値の降順にサイトをソートし，縦軸を F 値とした．

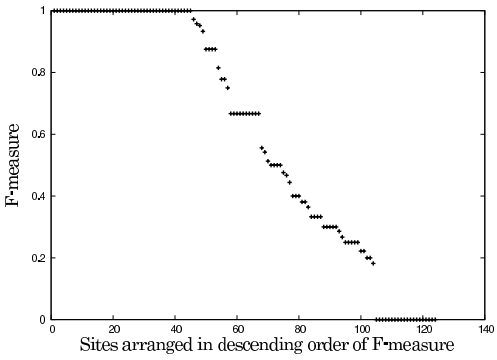

図 4 F 値による評価

Fig. 4 Evaluation by F-measure

この結果より次のことが分かる．(1) メタデータの抽出が完全に成功した F 値が 1 のサイトは 124 件中 45 件（全体の 36%）であった．(2) F 値が 0 のサイトは 124 件中 20 件（全体の 16%）であった．(3) それ以外のサイトは 124 件中 59 件（全体の 48%）であった．

この (2) の 20 件はメタデータが抽出できなかったか間違えていたことになる．これらについて詳細に調査した結果，抽出に失敗した理由は以下の 3 点である事が明らかになった．(A) 本アルゴリズムによる抽出が文や文節も取得しているのに対して，人手による抽出はその一部である単語が多い．内容はほぼ同じであるが，字面の違いで不正解となった．(B) 本アルゴリズムで想定していない位置にテキスト入力フィールドのメタデータが存在した．具体的には，テキスト入力フィールドの後方にあったもの，表の中の上端及び左端では無い位置にあったもの，が存在する．(C) `TABLE` タグでなく，`<dt>`，`<li>`，`<dd>`などのリストを生成するタグを用いて表と同等な表示結果を構成していた．(D) メタデータの判断が難しいページ構成であった．人手による正解例も完全には一致していなかった．

また (3) は，本アルゴリズムを用いて正解例の一部を取得で

[‡] これまでの調査における観察に基づく．一つのメタデータに対して，複数のテキスト入力フィールドが並んでいる場合が想定される．

[§] これまでに行った研究 [8] における観察に基づく．人手によって抽出したメタデータ（3.1 節）では，実際に 714 あった入力フィールドに対して 118 のメタデータがプルダウンメニューに存在することが確認できている．

きたものである．この 59 サイトについて，正解例以外が取得された原因を確認するため，再現率と適合率の関係を調べた．図 5 は再現率と適合率の相関関係を表したグラフ（Recall, Precision 値の小数点第 2 位を四捨五入した値で grid 上に配置したグラフ）であり，x 軸は適合率，y 軸は再現率，z 軸はサイト数である．

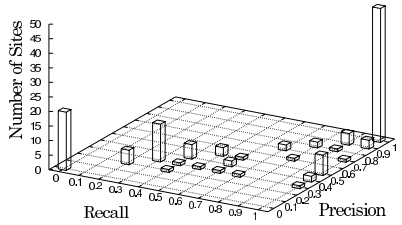

図 5　再現率と適合率による抽出結果の評価

Fig. 5　Evaluation of Result by Recall and Precision

このグラフから，再現率が高いが適合率が低いサイトが比較的多く存在することが見て取れる．本アルゴリズムでは上端，左端，直前の最大 3 つのメタデータ候補が取得されるが，正解は基本的に 1 つであるため，そのようなサイトにおいて適合率が下がることが理由であると考えられる．

結果全体を通して本アルゴリズムを再検討した結果，メタデータ抽出の適合率及び再現率を改善するためには，次の対応が考えられる．(1) 複数取得されたメタデータ候補から正しいメタデータを選ぶための，選択方法やその重み付けの手法．(2) <dt>, <li>, <dd>などのリストを構成するタグを考慮したアルゴリズムの改良．(3) 表におけるメタデータ候補の位置の検討と追加．入力フィールドの直後，及び表の左上隅の実例が見受けられた．ただし，(1) の手法を含めた検討が必要である．

3.3　表構造の情報を用いない手法との比較

本稿では，表構造の情報を利用したメタデータ抽出を提案しており，本来なら既存研究のメタデータ抽出手法との比較が必要である．しかしながら，この分野における一般的なテストベッドが存在しないため，各研究論文における評価実験は個別の対象について行われている．また，それらの手法を論文中の情報だけから構築することも困難であり，直接の比較が不可能である．したがって本節では，入力フィールドの直前の要素のみをメタデータとする手法をナイーブな手法として選択し，提案手法との直接比較を行った．この比較評価により，表構造を直接解析する手法のメタデータ抽出への寄与が明らかになると考える．

両手法による F 値を，2 次元上にマッピングしたグラフが図 6 である．円の大きさがその位置での頻度を示しており，例えば最左の列の円はナイーブな手法ではメタデータが取得できないが，提案手法では取得可能だった WebDB が数多く存在することを示している．全体として，多くの WebDB において提案手法の F 値が改善している事が読み取れる一方，グラフの右下領域には F 値が悪化した 9 の WebDB がある事が示されている．この理由は，複数候補の取得により適合率が下がった（3.2 節）為であり，今後の対応が必要である．

4. まとめ

WebDB を自動的に連携・結合するためには，WebDB のメタデータの自動抽出が必要である．属性の異なる複数の入力フィールドを持つ入力フォームは表で記述されることが多いことに着目し，新たな抽出方式を提案した．国内の 2,800 件の WebDB から無作為に選んだ 134 件のサイトに対し，人手により抽出したメタデータ集合と提案手法で抽出したメタデータ集合の評価を行い，表を用いない方法と比べて F 値が向上することも確認した．今後，提案手法の性能向上を試みるとともに，ラッパー生成技術との協調により複数の WebDB の連携を行うシステムを開発予定である．

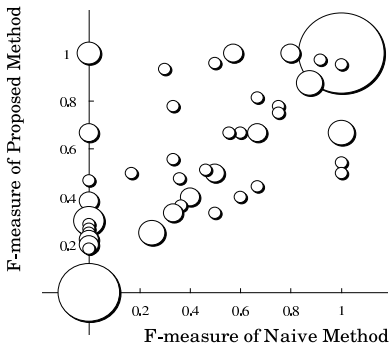

図 6　表構造の情報を用いない手法との比較

Fig. 6　Comparison of Proposed Method and Naive Method

[文献]

[1] He, H., Meng, W., Yu, C. and Wu, Z.: “Automatic Integration of Web Search Interfaces with WISE-Integrator”, VLDB Journal, Vol.13, No.3, pp.256-273, 2004.

[2] Lerman, K., Knoblock, C. and Minton, S.: “Automatic Data Extraction from Lists and Tables in Web Sources”, Proc. of ATEM-10, IJCAI, 2001.

[3] Nakatoh, T., Yamada, Y. and Hirokawa, S.: “Automatic Generation of Deep Web Wrappers based on Discovery of Repetition”, Proc. of AIRS2004, pp.269-272, 2004.

[4] Raghavan, S. and Garcia-Molina, H.: “Crawling the HiddenWeb”, Proc. of the 29th International Conference on VLDB, pp.129-138, 2001.

[5] Thakkar, S., Knoblock, C. A., Ambite, J. and Shahabi, C.: “Dynamically Composing Web Services from On-line Sources”, Proc. of the AAAI-2002, 2002.

[6] Yoshida, M., Torisawa, K. and Tsujii, J.: “Integrating Tables on the World Wide Web”, Trans. of the JSAI. Vol.19, No.6, pp.548-560, 2004.

[7] Zhang, Z., He, B. and Chang, K. C.: “Understanding Web Query Interfaces: BestEffort Parsing with Hidden Syntax”, SIGMOD2004.

[8] 大森敬介, 中藤哲也, 山田泰寛, 原由加里, 廣川佐千男: “複雑な検索機能を持つ検索サイトの動向調査”, DEWS2004, I-1-05, 2004.

中藤 哲也　Tetsuya NAKATOH

九州大学情報基盤センター助手．1992 九州大学総合理工学研究科修士課程修了．検索エンジン，Web マイニング，文字列処理の研究に従事．日本データベース学会正会員．情報処理学会正会員．

大森 敬介　Keisuke OHMOMI

三菱電機情報ネットワーク（株）．2002 九州大学大学院システム情報科学研究科修士課程修了．Web マイニングの研究に従事．

廣川 佐千男　Sachio HIROKAWA

九州大学情報基盤センター教授．1979 九州大学大学院理学研究科修士課程修了，理学博士．検索エンジン，Web マイニング，計算論理学の研究に従事．情報処理学会正会員．電子情報通信学会正会員．ACM 正会員．