リンク情報と構造情報を用いたHTML群からのテキスト自動切り出しアルゴリズムの実装

筒井　淳平†　　伊藤　公人††　　有村　博紀†††

† 北海道大学工学部　〒 060–0814 札幌市北区北 14 条西 9 丁目
†† 北海道大学人獣共通感染症リサーチセンター　〒 060-0818 札幌市北区北 18 条西 9 丁目
††† 北海道大学大学院情報科学研究科　〒 060–0814 札幌市北区北 14 条西 9 丁目
E-mail: †supli-mail@m2.dion.ne.jp, ††itok@czc.hokudai.ac.jp, †††arim@ist.hokudai.ac.jp

あらまし　ウェブ上で収集した HTML テキストから，目的とするデータを自動的に取り出すことをテキストの自動切り出し，または情報抽出と呼ぶ．また，これを行うプログラムを HTML Wrapper と呼ぶ．本研究では文献（村上，坂本，有村，有川，情報処理学会 TOM，2001）で提案された情報抽出アルゴリズムを元に，与えられた HTML 群のリンクによる巡回の例示から，所望の HTML テキストデータを切り出すための HTML Wrapper を自動生成するアルゴリズムを構築する．さらにこのアルゴリズムを実装し，その有効性を検証する．

キーワード　ウェブからの情報抽出，ウェブ巡航，帰納的ウェブラッパー構築，例示による学習機械学習

Web Information Extraction Based on Generalized User Navigation Path

Junpei TSUTSUI†, Kimihito ITO††, and Hiroki ARIMURA†††

† Graduate School of Information Science and Technology, Hokkaido University　N14 W9, Sapporo 060–0814, Japan
†† Research Center for Zoonosis Control, Hokkaido University　N18 W9, Sapporo 060–0818, Japan
††† Graduate School of Information Science and Technology, Hokkaido University　N14 W9, Sapporo 060–0814, Japan
E-mail: †supli-mail@m2.dion.ne.jp, ††itok@czc.hokudai.ac.jp, †††arim@ist.hokudai.ac.jp

Abstract　This paper studies information extraction from the Web. Our method combines the advantages of an approach based on HTML page navigation and an approach based on HTML wrapper induction. The keys of our method are construction of extraction tree from navigation records and the use of machine learning techniques for automatically constructing wrappers to achieve flexible information extraction.

Key words　information extraction from the Web, learning from Web navigation, machine learning

1. 序　　論

ウェブは相互にリンクによって結合された多数の HTML ページの集まりである．これらのウェブ上のデータからの情報抽出は，ウェブ検索やウェブマイニングにおける基本的な問題であり，ウェブ上の情報の探索，情報の自動抽出，ウェブ知識データベース，文書の自動分類，キーワード発見といった様々なタスクの基礎となる [1], [3]〜[6], [8]．本稿では，ウェブ上の情報を探索し，情報を抽出する作業を支援する半自動ツールを提案する．この鍵として，HTML ページ群の巡回の例示から抽出スクリプトを学習するアルゴリズムを与える．

多くのウェブサイトでは，情報を求めるユーザーが探索しやすいように，そのリンク構造がローカルなディレクトリ構造に似た階層構造になっていることが多い（図 2）．このときに，ある階層に存在する HTML ページや，ある条件を満たすリンク構造で指し示される HTML ページ，またはそのような HTML ページの部分 HTML テキストのみをまとめて抽出したいことも多い．

単一の HTML ページからの情報抽出として，文献 [7] では，HTML ページをタグまたはテキストをノードとした木構造に展開し，与えられたいくつかの切り出し対象の訓練例から抽出規則を学習する手法を提案している．一方，本稿ではこれを拡

張し，ウェブの巡回の例示から部分 HTML テキストの抽出スクリプトを学習する．学習は，与えられた例を一般化することで行い，例示には数個の切り出し例で十分である．

1.1 提案手法の概要

本節では，提案手法の概要を説明する．抽出の対象となるウェブサイトとして，図 2 のようなディレクトリ構造に似た階層構造をリンク構造としてもつ HTML ページ群を考える．ここでは，HTML ページ群にはただ一つのトップページが存在し，そこからいくつかの分岐点となる中間の HTML ページを経由して，端点となる目標 HTML ページ群へ到達すると仮定する．ただし，これらの HTML ページは，同じサイトにある必要はないし，HTML リンクでつながっている他には特別な構造は仮定しない．

部分 HTML テキストの抽出規則として，次のような形の多段階の繰り返しをもつ抽出スクリプトを扱う．

---

x_0 := `GoPage`(URL_0);
for x_1 in `Follow`(x_0, π_1/A/`@href()`) **do**
  for x_2 in `Follow`(x_1, π_2/A/`@href()`) **do**
    ... ...
      for x_n in `Follow`(x_{n-1}, π_n/A/`@href()`) **do**
        `Select`(x_n, π_{n+1});

図 1　多段階の繰り返しをもつ部分 HTML テキスト抽出規則

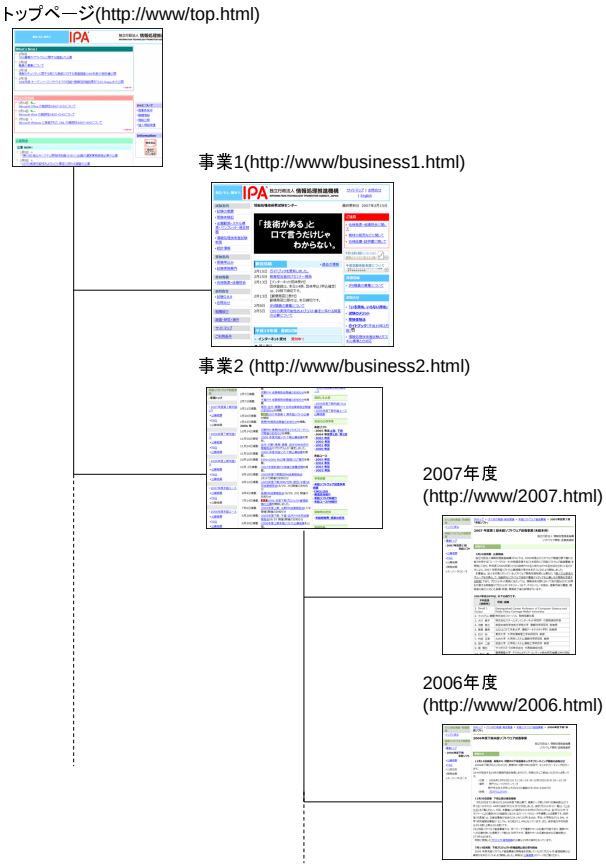

図 2　ディレクトリ構造に似たリンク構造

この抽出規則は，次のように抽出作業を行う．

- まず，アルゴリズムは，指定された開始場所 URL_0 をもつ HTML ページ x_0 から出発する．
- 次に，ステージ $i := 1, \ldots, n$ で次を繰り返す：現在の HTML ページ x_{i-1} 上で，パス π_i で到達可能なアンカーノードを訪問し，その HTML リンクが指す HTML ページ x_i へ移動する．
- 最後に到達した HTML ページ x_n 中で，パス π_{n+1} で指される部分 HTML テキストを抽出する．

しかし，人間の利用者が，上記のような複雑な URL やパスを含む抽出規則を直接手で作成することは難しい．そこで利用者による例示から抽出スクリプトを学習することを考える．

そのために，利用者は図 3 のような操作履歴を記録可能なウェブブラウザを用いて，HTML ページ群の巡回記録を作成する (3 節)．この巡回記録から例を生成し，これらを一般化する．これにより，最終的な部分 HTML テキストの抽出スクリプトを学習する (4 節)．これらのステップの詳細については，以降の節で説明する．

1.2 応　　用

この様な巡回記録からの例示による学習を用いると，ウェブからの情報収集に関して，次の様な応用が可能になる．

- **部分テキストの収集：** トップページから，離れた HTML ページ群にある部分 HTML テキストを収集することができる．
- **表の集約：** 上記と同様に，離れた HTML ページ群にある多数の表を一つの大きな表に集約できる．
- **ページの閲覧：** あらかじめ数個のサンプル HTML ページを訪問しておき，後はボタン一つで類似したパスをもつ HTML ページ群を順に表示することができる．

1.3 本稿の構成

本稿の構成は以下の通りである．2 節では準備として以降の節に必要な基本的な定義と結果をまとめる．3 節では学習の対象となるナビゲーションについて説明し，4 節では学習の方法について説明する．5 節では実行例を報告する．6 節で本稿をまとめ，今後の課題を与える．

2. 準　　備

本章では，本稿で説明するアルゴリズムの準備として，村上ら [7] によって提案された WebWrapper の概要について説明する．村上らの手法は，HTML ページとそれに対する抽出対象部分の組を訓練例としていくつか与え，新たに与えられた HTML ページから対応するテキストを抽出しようとするものである．以下では HTML ページを単にページと呼ぶ．

2.1 HTML 木

$\mathbb{N}$ で非負整数全体の集合を表す．要素の集合 S に対して，S^* で S の要素の有限列全体の集合を表す．互いに素な可算集合を，$\mathcal{N}, \mathcal{V}', \mathcal{A}$ とおく．$\mathcal{N}, \mathcal{V}', \mathcal{A}$ の元は，それぞれ，**タグ名**，**値**，**属性名**と呼ばれる．ここで，任意を表す特別な記号 $* \notin \mathcal{N} \cup \mathcal{V}' \cup \mathcal{A}$ を仮定し，$\mathcal{V} = \mathcal{V}' \cup \{*\}$ とする．**HTML 属性集合**は，定義域 $\mathrm{dom}(A) \subseteq \mathcal{A}$ をもつ関数 $AS : \mathcal{A} \to \mathcal{V}$ であり，HTML 属性名

と HTML 属性値の対 $\langle a, v \rangle$ の集合として表現される．$\mathcal{AS}$ で HTML 属性集合の全体を表し，$\mathcal{URL}$ で URL の全体集合を表す．要素 $name \in \mathcal{N} \cup \mathcal{A} \cup \mathcal{URL}$ を**名前**という．また，ページの全体集合を $\mathcal{P}$ と書く．

ページ $P \in \mathcal{P}$ は以下で定義される HTML 木 $T = (\mathcal{DOM}, child, sibling, root)$ で表される．集合 $\mathcal{DOM}$ はノードの有限集合であり，$child$ と $sibling$ は親子関係と兄弟関係，$root$ は根である．T の各ノードはノードラベル $NL(n) = \langle N(n), V(n), AS_n, Pos(n) \rangle$ をもつ．ここで $N(n) \in \mathcal{N}$ はノード名，$V(n) \in \mathcal{V}$ はノード値，$AS_n : \mathcal{A} \to \mathcal{V} \in \mathcal{AS}$ は HTML 属性集合，$Pos(n) \in \mathbb{N}$ は n の兄と自分自身のうち，n と同じノード名であるノードの数を表す．

ページのすべての空要素は T のタグ名をノード名とした子供を持たないある要素ノードに対応する．タグに関する文字を含まない文字列 w は，ノード名として特別な名前 ♯TEXT を持ち，ノード値として w をもつ葉ノードに対応する．開始タグとそれに対応する終了タグの組は，その内部の部分 HTML テキストによって構成される HTML 木の並びを子の並びとして持ち，タグ名をノード名として持つ要素ノードに対応する．

ページと HTML 木は自然な対応をもつ．

2.2 パ　　ス

HTML 木において，アクセスや抽出の対象はパスで指定する．パスは，子ノード軸と，タグと属性に関する条件だけを持つ XPath [9] と本質的に同等である．

［定義 1］ **ステップ**（step）は三つ組 $step = \langle N, Pos, AS \rangle \in \mathcal{N} \times \mathbb{N} \cup \{*\} \times \mathcal{AS}$ であり，$step = N[Pos][a_1 = AS(a_1), a_2 = AS(a_2), \ldots]$ と書く．また，**パス**(path) はステップの列 $path = step_1 \cdots step_l$ であり，$path = /step_1/ \cdots /step_l/$ と書く．ステップとパスについて，そのどの部分にも $*$ を含まないものを特に**基礎ステップ**および**基礎パス**といい，そうでないものを**一般化ステップ**および**一般化パス**という．

$\mathcal{STEP}$ と $\mathcal{PATH}$ で，それぞれステップとパスの全体集合を表す．

2.3 パスの半順序

HTML 木に関するいろいろな対象の上の一般化順序 $\leqq$ を導入する．

［定義 2］ ステップとパスを含む対象の上に半順序 $\leqq$ を次のように導入する．添え字を $i = 1, 2$ とする．

（1） 名前 $name_i \in \mathcal{N} \cup \mathcal{A} \cup \mathcal{URL}$ に対して，$name_1 \leqq name_2$ なのは $name_1 = name_2$ のとき．

（2） 値 $value_i \in \mathcal{V}$ に対して，$value_1 \leqq value_2$ なのは $value_1 = value_2$ または $value_1 = *$ のとき．

（3） 属性集合 $AS_i : \mathrm{dom}(A_i) \to \mathcal{V} \in \mathcal{AS}$ に対して，$AS_1 \leqq AS_2$ なのは，$\mathrm{dom}(A_1) \subseteqq \mathrm{dom}(A_2)$ かつ $\forall a \in \mathrm{dom}(A_1) : AS_1(a) \leqq AS_2(a)$ のとき．

（4） 位置 $Pos_i \in \mathbb{N} \cup \{*\}$ に対して，$Pos_1 \leqq Pos_2$ なのは，$Pos_1 = Pos_2$ または $Pos_1 = *$ のとき．

（5） ステップ $step_i = \langle N_i, Pos_i, AS_i \rangle \in \mathcal{STEP}$ に対して，$step_1 \leqq step_2$ なのは，$N_1 \leqq N_2$ かつ $Pos_1 \leqq Pos_2$，$AS_1 \leqq AS_2$ のとき．

（6） パス $path_i = /step^i_{l_i}/ \cdots /step^i_1/ \in \mathcal{PATH}$ に対して，$path_1 \leqq path_2$ なのは，$step^1_k \leqq step^2_k$ が任意の $k = 1, \ldots, l_1$ に対して成立するとき．

2.4 パスの一般化

［定義 3］ n 個の対象 $o_1, \ldots, o_n$ から，$\forall i \in \{1 \ldots n\} : o \leqq o_i$ をみたす対象 o を見つけることを，対象を**一般化**するという．

［定義 4］ 二つの対象に対する一般化演算子 $\oplus$ を以下のように帰納的に定義する．添え字を $i = 1, 2$ とする．

（1） 名前 $name_i \in \mathcal{N} \cup \mathcal{A} \cup \mathcal{URL}$ に対して，$name_1 = name_2$ のときは $name_1 \oplus name_2 = name_1$ とし，それ以外は無定義とする．

（2） 値 $value_i \in \mathcal{V}$ に対して，$value_1 = value_2$ のときは $value_1 \oplus value_2 = value_1$ とし，それ以外は $*$ とする．

（3） 属性集合 $AS_i : \mathrm{dom}(A_i) \to \mathcal{V} \in \mathcal{AS}$ に対して，$AS_1 \oplus AS_2 = AS_*$ とする．ただし AS_* は，$\mathrm{dom}(A_*) = \mathrm{dom}(A_1) \cap \mathrm{dom}(A_2)$ かつ，$\forall a \in \mathrm{dom}(A_*), AS_*(a) = AS_1(a) \oplus AS_2(a)$ を満たす属性集合である．

（4） 位置 $Pos_i \in \mathbb{N} \cup \{*\}$ に対して，$Pos_1 = Pos_2$ のとき $Pos_1 \oplus Pos_2 = Pos_1$ とし，それ以外は $*$ とする．

（5） ステップ $step_i = \langle N_i, Pos_i, AS_i \rangle \in \mathcal{STEP}$ に対して，$N_1 \oplus N_2$ が定義されるとき，$step_1 \oplus step_2 = \langle N_1 \oplus N_2, Pos_1 \oplus Pos_2, AS_1 \oplus AS_2 \rangle$ とし，それ以外は無定義とする．

（6） パス $path_i = /step^i_{l_i}/ \cdots /step^i_1/ \in \mathcal{PATH}$ に対して，$path_1 \oplus path_2 = /(step^1_k \oplus step^2_k)/ \cdots /(step^1_1 \oplus step^2_1)$ とする．ただし k は任意の j $(1 \leqq j \leqq k)$ について $step^1_j \oplus step^2_j$ が定義される最大の非負整数である．

［例 1］ 表 1 に，パスの一般化の例を示す．下線部が path1 と path2 の異なる部分であり，`TR` の位置，`TD` の属性 `bgcolor` の属性値がそれぞれ一般化によって変化する．また，タグ名が一致しない前半のいくつかのステップは一般化によって取り除かれる．

2.5 HTML 木とパスのマッチング

［定義 5］ HTML 木のノード n とステップ $step = \langle N, Pos, AS \rangle$ について，$N \leqq N(n), AS \leqq AS_n, Pos \leqq Pos(n)$ のとき，n は $step$ に**マッチする**という．また，HTML 木のノード n への根からの経路 $n_k \cdots n_1$ $(n_1 = n, n_k$は根) とパス $path = step_l \cdots step_1$ について，$l \leqq k$ かつ，任意の i $(1 \leqq i \leqq l)$ に対し n_i が $step_i$ にマッチするとき，n は $path$ にマッチするという．

2.6 WebWrapper の学習とテキストの抽出

WebWrapper の学習は，訓練例によって与えられたパスの一般化によって達成される．テキスト抽出対象のページの HTML 木 T と学習によって得られたパス $path$ に対し，WebWrapper は $path$ にマッチする T のすべてのノードから，対応するタグ以下のテキストを抽出する．

3. 巡回記録と抽出スクリプト

本節では，ページ群の巡回を定式化する．

3.1 記録再生ブラウザ

図 3 の**記録再生ブラウザ**は，通常のウェブブラウザの機能に

表 1　パスの一般化の例

path1	/HTML[1]/BODY[1]/TABLE[1]/TBODY[1]/TR[1]/TD[3]/ TABLE[2]/TBODY[1]/TR[2]/TD[2][bgcolor="blue", width="100"]/
path2	/HTML[1]/BODY[1]/ TABLE[2]/TBODY[1]/TR[4]/TD[2][bgcolor="red", width="100"]/
path1⊕path2	/TABLE[2]/TBODY[1]/TR[*]/TD[2][bgcolor=*, width="100"]/

図 3　記録再生ブラウザ

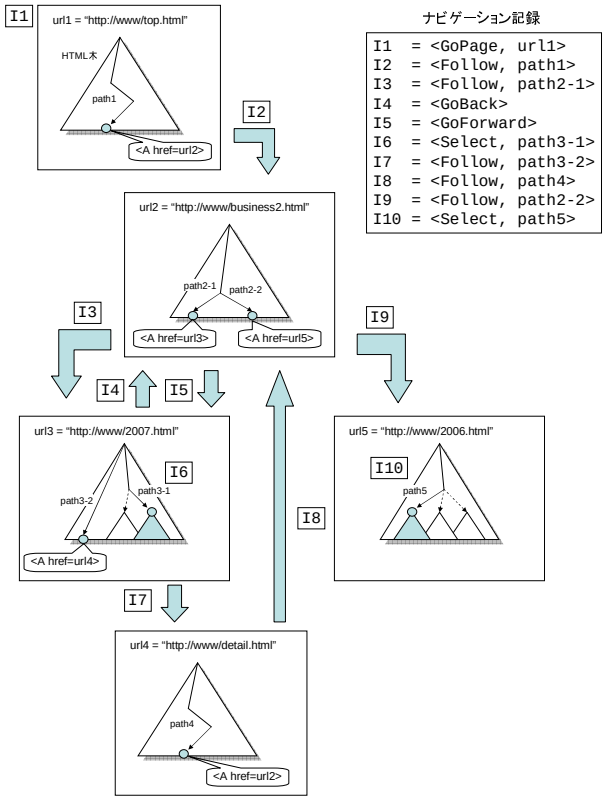

図 4　ウェブの巡回と巡回記録

加えて，利用者の操作履歴を巡回命令の列，すなわち巡回記録として，記録する機能を持つプログラムである．上部の表示タブを選択することで，本稿で以後に説明する巡回木や抽出スクリプトなどを作成，および確認することができる．さらに，後述する抽出スクリプトを自動的に実行することができる．

記録再生ブラウザにおける利用者の各操作は，次の巡回命令に対応する．ただし履歴行列とはブラウザが過去に表示したページを順に保持したリストである．

`GoPage`(url): URL url を受け取ってそれが指すページ $P(url)$ へ移動する．

`GoBack`(): 履歴行列中で一つ前のサイトへ移動する．

`GoForward`(): 履歴行列中で一つ先のサイトへ移動する．

`Follow`($path$): 選択部分の基礎パス $path$ にマッチするノードが指すページへ移動する．

`Select`($path$): 選択部分の基礎パス $path$ にマッチする部分 HTML テキストを抽出対象にする．

3.2　巡回記録

命令の集合 $\mathcal{INST}$ = {`GoPage`, `GoBack`, `GoForward`, `Follow`, `Select`} とおき，$\mathcal{INST}$ の各要素を**命令名**と呼ぶ．さらに，$\Theta = \mathcal{URL} \cup \mathcal{PATH}$ として命令名 $inst \in \mathcal{INST}$ とその引数 $\theta \in \Theta$ の組 $\langle inst, \theta \rangle$ を**巡回命令**と呼び，以下で $inst(\theta)$ と書くことがある．巡回命令の全体集合を $\mathcal{CMD}$ とする．直感的には，巡回命令はウェブブラウザ（ウェブ閲覧プログラム）における利用者の操作を抽象化したものである．

［定義 6］　巡回命令の列 $cmd_1 \cdots cmd_n \in \mathcal{CMD}^*$ を**巡回記録**と呼ぶ（図 4）．

3.3　抽出スクリプト

図 1 のような多段階の繰り返しをもつ抽出スクリプトを抽象化して，次の様に定義する．ページ群の巡回と抽出の手順は，次の抽出スクリプトとして表現される．

［定義 7］　**抽出スクリプト**は，scr = $(\langle$`GoPage`$, url_0 \rangle, \langle$`Follow`$, \pi_1 \rangle, \ldots, \langle$`Follow`$, \pi_{n-1} \rangle, \langle$`Select`$, \pi_n \rangle)$ の形をした巡回命令列である．ここに，url_0 はあるページの URL であり，π_i はパスである．

抽出スクリプトのうち，そのどの部分にも $*$ を含まないものを特に**基礎抽出スクリプト**，そうでないものを**一般化抽出スクリプト**という．また，抽出スクリプトの全体集合を $\mathcal{SCR}$ で表す．抽出スクリプトは，記録再生ブラウザによって，自動的に実行される．

手続き 1 巡回記録から巡回木を作るアルゴリズム

Input: 巡回記録 $R = cmd_1 \cdots \in \mathcal{CMD}^*$
Output: 巡回木 $\mathcal{T}_R$
  $current\_URL$ /* 現在地の URL */
  $next\_URL$ /* 移動先の URL */
  $node\_stack$ /* ノードへのポインタのスタック */
  $select\_node\_list$ /* 命令名 `Select` をラベルに持つすべてのノードへのポインタを持つリスト */
  $i = 1$;
  while cmd_i != $NULL$ **do**
    $inst := cmd_i$ の命令名;
    $arg := cmd_i$ の引数;
    switch ($inst$)
      case `GoPage` **then**
        ラベル $\langle cmd_i, current\_URL \rangle$ を持つノードを作り，そのノードへ移動し，そのノードを根とする;
        break;
      case `GoBack` **then**
        現在のノードを指すポインタを $node\_stack$ へ入れ，親ノードへ移動する;
        break;
      case `GoForward` **then**
        $node\_stack$ から取り出したポインタの指すノードへ移動する;
        break;
      case `Follow` **then**
        $next\_URL := current\_URL$ の HTML 木上で arg_i によりページを移動した先の URL;
        if $next\_URL$ をラベル url に持つノードが存在する **then**
          そのノードへ移動する;
        else
          現在のノードの子にラベル $\langle cmd_i, next\_URL \rangle$ を持つノードを追加し，そのノードへ移動する;
        end if
        break;
      case `Select` **then**
        現在のノードの子にラベル $\langle cmd_i, NULL \rangle$ を持つノードを追加し，そのノードを $select\_node\_list$ に追加する;
        break;
    end switch
    $i := i + 1$;
  end while

4. 巡回例からの学習

本節では，抽出の例示から一般的な抽出規則を学習する方法を与える．

4.1 巡回記録からの抽出スクリプトの発見

［定義 8］ $R \in \mathcal{CMD}^*$ を巡回記録とする．R の**巡回木**は，手続き 1 で計算される根付き木 $\mathcal{T}_R = (V, E, root)$ である．ここに V は**命令ノード**と呼ばれるノードの集合であり，E は有向辺の集合，$root \in V$ は根である．巡回木の各ノード $v \in V$ はラベルとして $\langle cmd(v), url(v) \rangle \in \mathcal{CMD} \times \mathcal{URL}$ を持つ．

［例 2］ 図 5 に，図 4 の巡回記録から構成される巡回木を示す．

［定義 9］ $R \in \mathcal{CMD}^*$ を巡回記録とする．R の巡回木の根からあるノードまでの経路 $v_1 \cdots v_n$ 上のラベル列 $\pi = cmd(v_1) \cdots cmd(v_n) \in \mathcal{CMD}^*$ が抽出スクリプトであるとき，

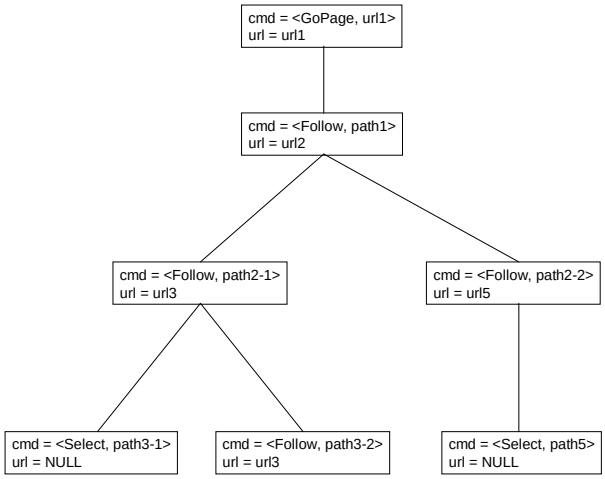

図 5　巡回木の例

手続き 2 巡回木から基礎抽出スクリプトを取得するアルゴリズム

Input: 巡回木 $\mathcal{T}_R$,
  命令名 `Select` をラベルに持つすべてのノードへのポインタを持つリスト $select\_node\_list$
Output: 基礎抽出スクリプトのリスト $basescript\_list$
  Parent(n); /* ノード n の親ノードを返す関数 */
  $current$; /* 現在のノード */
  $script$; /* 巡回命令列 */
  for all $select\_node$ in $select\_node\_list$ **do**
    $script$ を空にする;
    $current := select\_node$;
    while $current$!= NULL **do**
      $script$:= cmd($current$) の末尾に $script$ を連結した命令列;
      $current$:= Parent($current$);
    end while
    $script$ を $basescript\_list$ に格納する;
  end for

この経路を**抽出枝**と呼ぶ．任意の抽出枝は，一つの基礎抽出スクリプトに対応する．

手続き 2 は入力として与えられた巡回木が持つすべての基礎抽出スクリプトを取得し，そのリストを返すアルゴリズムである．巡回木の根から命令名 `Select` を持つ巡回木のすべてのノードへの経路を，ノードを親へと順にさかのぼることで取得している．

［例 3］ 図 6 に，図 5 の巡回木から構成される基礎抽出スクリプトを示す．

4.2 抽出スクリプトの一般化

［定義 10］ 抽出スクリプトについて，半順序 $\leqq$ を拡張する．添え字を $i = 1, 2$ とする．

（1） 命令名 $inst_i \in \mathcal{INST}$ に対して，$inst_1 \leqq inst_2$ なのは，$inst_1 = inst_2$ のとき．

（2） 巡回命令 $cmd_i = \langle inst_i, \theta_i \rangle \in \mathcal{CMD}$ に対して，$cmd_1 \leqq cmd_2$ なのは，$inst_1 \leqq inst_2$ かつ $\theta_1 \leqq \theta_2$ のとき．

（3） 抽出スクリプト $scr_i = cmd_1^i \cdots cmd_{l_i}^i \in \mathcal{SCR}$ に対して，$scr_i \leqq scr_2$ なのは，$l_1 = l_2$ かつ $\forall j \in \{1, \ldots, l_1\} : cmd_j^1 \leqq cmd_j^2$ のとき．

基礎抽出スクリプト1

<GoPage, url1>
<Follow, path1>
<Follow, path2-1>
<Select, path3-1>

基礎抽出スクリプト2

<GoPage, url1>
<Follow, path1>
<Follow, path2-2>
<Select, path5>

url1 = “http://www/top.html”
path1 = /HTML[1]/BODY[1]/TABLE[1]/TBODY[1]/TR[2]/TD[2]/A[1][href=“./business2.html”]/
path2-1 = /HTML[1]/BODY[1]/.../TBODY[1]/TR[2]/TD[2]/A[1][href=“./2007.html”]/
path2-2 = /HTML[1]/BODY[1]/.../TBODY[1]/TR[4]/TD[2]/A[1][href=“./2006.html”]/
path3-1 = /HTML[1]/BODY[1]/.../TR[1]/TD[1]/TABLE[4]/TBODY[1]/TR[2]/TD[1]/
path5 = /HTML[1]/BODY[1]/.../TR[1]/TD[1]/TABLE[4]/TBODY[1]/TR[4]/TD[2]/

図 6 基礎抽出スクリプトの例

一般化抽出スクリプト

<GoPage, url1>
<Follow, path1>
<Follow, path2-1⊕path2-2>
<Select, path3-1 ⊕path5>

url1 = “http://www/top.html”
path1 = /HTML[1]/BODY[1]/TABLE[1]/TBODY[1]/TR[2]/TD[2]/A[1][href=“./business2.html”]/
path2-1⊕path2-2
= /HTML[1]/BODY[1]/.../TBODY[1]/TR[*]/TD[2]/A[1][href=*]/
path3-1⊕path5
= /HTML[1]/BODY[1]/.../TR[1]/TD[1]/TABLE[4]/TBODY[1]/TR[*]/TD[*]/

図 7 一般化抽出スクリプトの例

［定義 11］ 抽出スクリプトについて，一般化演算子 $\oplus$ を拡張する．添え字を $i = 1, 2$ とする．

（1） 巡回命令 $cmd_i = \langle inst_i, \theta_i \rangle \in \mathcal{CMD}$ に対して，$inst_1 = inst_2$ のときは $cmd_1 \oplus cmd_2 = \langle inst_1 \oplus inst_2, \theta_1 \oplus \theta_2 \rangle$ とし，それ以外は無定義とする．

（2） 抽出スクリプト $scr_i = cmd_1^i \cdots cmd_{l_i}^i \in \mathcal{SCR}$ に対して，$l_1 = l_2$ かつ任意の j $(1 \leq j \leq l_1)$ について $cmd_j^1 \oplus cmd_j^2$ が定義されるとき，$scr_1 \oplus scr_2 = (cmd_1^1 \oplus cmd_1^2) \cdots (cmd_{l_1}^1 \oplus cmd_{l_1}^2)$ とし，それ以外は無定義とする．

［例 4］ 図 7 に，図 6 の 2 つの基礎抽出スクリプトから作られる一般化抽出スクリプトを示す．基礎抽出スクリプトの巡回命令はそれぞれ一般化されるが，上から 1 番目と 2 番目の巡回命令は両方の基礎抽出スクリプトで違いが無いので一般化された結果変化しない．3 番目と 4 番目の巡回命令の引数であるパスはそれぞれ一般化される．3 番目では `TR` タグの位置を自由に選択でき，`A` タグのリンク先も制限されないため，巡回記録では到達されていない URL http://www/2005.html, http://www/2004.html, ... へアクセスできる．4 番目では `Select` 命令のパスが一般化されており，`TABLE` 中の複数のエントリを抽出可能である．

5. 実 行 例

本稿で提案した巡回記録を作るための記録ブラウザを C#言語で実装し，これに例の生成アルゴリズムと抽出スクリプトの学習アルゴリズム，スクリプトの実行アルゴリズムを組み込んだ．これを用いて，ウェブ上の実際のページに対し，PC(Pentium4 3.60GHz, 2GB memory, WindowsXP, .NET) 上で実験を行った．

村上ら [7] の手法では，まず抽出対象とするすべてのページを

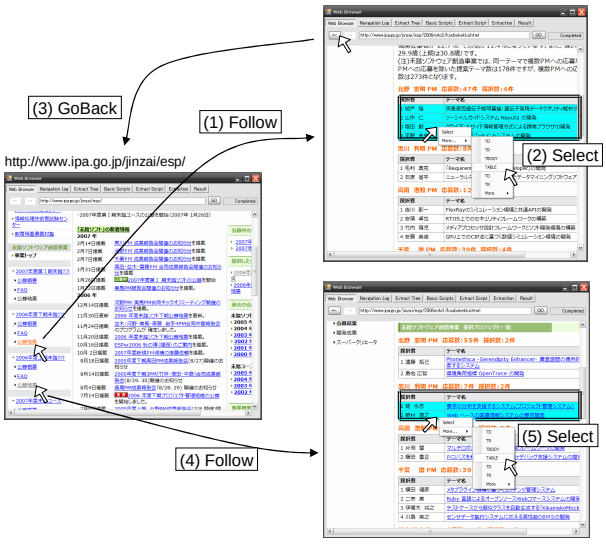

図 8 実行例 1 のナビゲーションの様子

あらかじめ取得しておき，抽出アルゴリズムにそれらを順に与えて抽出作業を行っている．本実験では，記録ブラウザによって作られた巡回記録から抽出スクリプトを学習することで，例示の巡回では訪問していないサイトも抽出対象として選び出し，それらから部分 HTML テキストを抽出することができる．

5.1 実 行 例 1

IPA（独立行政法人 情報処理推進機構）の「未踏ソフトウェア創造事業」についてのページ（http://www.ipa.go.jp/jinzai/esp/）から巡回を開始し，採択者とテーマ名を掲載している表のうち，二つを抽出対象として選択した．この巡回記録から，過去の採択者とテーマ名を掲載するすべての表を抽出し，一つのページとして表示させる実験を行った．

図 8 は，上記目的のためのページ群の巡回の様子を図示したものである．利用者は番号の順に操作を行っている．(2) と (4) の `Select` 命令では，コンテキストメニューに表示されたタグ名を利用者が選択することで選択範囲の背景色が変わり，そのタグを選択することがどの範囲を抽出することになるかを知らせている．図では二重線で囲まれた範囲が，それぞれの選択範囲である．

この実行例で得られた巡回記録と，それから得られる基礎抽出スクリプト，さらにそれらから得られる一般化抽出スクリプトはそれぞれ表 2 および，表 3，表 4 の通りである．なお，表で HTML 属性は省略した．図 9 に，実験で抽出された部分 HTML テキストを連結したものを示す．

5.2 実 行 例 2

セブンイレブン・ジャパンのホームページ（http://www.sej.co.jp/index.html）から，北海道にある店舗の店舗名をいくつか抽出対象として巡回し，北海道にあるすべての店舗の店舗名を抽出し，一つのページとして表示させる実験を行った．

このウェブサイトの構成は，利用者の視点からみると次のとおりであった：(i) トップページは都道府県名の表を掲載したページである．(ii) ここから都道府県名を選択すると，都道府県の市区町村名が頭文字によって五十音の行ごとに分けられた

表 2　実行例 1 の巡回記録

巡回記録
⟨GoPage, "http://www.ipa.go.jp/jinzai/esp/"⟩
⟨Follow, /HTML[1]/BODY[1] /TABLE[3]/TBODY[1]/TR[1]/TD[1]/TABLE[1]/TBODY[1]/TR[18]/TD[2]/P[4]/A[1]/⟩
⟨Select, /HTML[1]/BODY[1]/TABLE[3]/TBODY[1]/TR[1]/TD[4]/TABLE[4]/⟩
⟨GoBack⟩
⟨Follow, /HTML[1]/BODY[1]/TABLE[3]/TBODY[1]/TR[1]/TD[1]/TABLE[1]/TBODY[1]/TR[20]/TD[2]/P[4]/A[1]/⟩
⟨Select, /HTML[1]/BODY[1]/TABLE[3]/TBODY[1]/TR[1]/TD[4]/TABLE[5]/⟩

表 3　実行例 1 の巡回記録から得られた二つの基礎抽出スクリプト

基礎抽出スクリプト 1
⟨GoPage, "http://www.ipa.go.jp/jinzai/esp/"⟩
⟨Follow, /HTML[1]/BODY[1]/TABLE[3]/TBODY[1]/TR[1]/TD[1]/TABLE[1]/TBODY[1]/TR[18]/TD[2]/P[4]/A[1]/⟩
⟨Select, /HTML[1]/BODY[1]/TABLE[3]/TBODY[1]/TR[1]/TD[4]/TABLE[4]/⟩

基礎抽出スクリプト 2
⟨GoPage, "http://www.ipa.go.jp/jinzai/esp/"⟩
⟨Follow, /HTML[1]/BODY[1]/TABLE[3]/TBODY[1]/TR[1]/TD[1]/TABLE[1]/TBODY[1]/TR[20]/TD[2]/P[4]/A[1]/⟩
⟨Select, /HTML[1]/BODY[1]/TABLE[3]/TBODY[1]/TR[1]/TD[4]/TABLE[5]/⟩

表 4　実行例 1 の二つの基礎抽出スクリプトから学習された一般化抽出スクリプト

一般化抽出スクリプト
⟨GoPage, "http://www.ipa.go.jp/jinzai/esp/"⟩
⟨Follow, /HTML[1]/BODY[1]/TABLE[3]/TBODY[1]/TR[1]/TD[1]/TABLE[1]/TBODY[1]/TR[*]/TD[2]/P[4]/A[1]/⟩
⟨Select, /HTML[1]/BODY[1]/TABLE[3]/TBODY[1]/TR[1]/TD[4]/TABLE[*]/⟩

表を掲載したページが表示される．(iii) さらにここから，市区町村名を選択すると，その市区町村の店舗名がまとめられたページへと辿り着く．

このとき，トップページから巡回し，異なるページにある二つの店舗名を選択した．これを元に学習と抽出を行った結果，図 10 のようなページが得られた．

5.3　考　　察

実行例 1 では意図通りに表の項目をすべて抽出することができ，満足する結果が得られた．一方実行例 2 では，結果として，目的であった北海道のすべての店舗名を得ることはできなかった．この原因として，利用者の視点から判断した上記のウェブサイトの構成に誤りがあったことがあげられる．つまり，実際のウェブサイトの構成は，上記のものに加え，ある市区町村に店舗が一つしか存在しない場合，その市区町村名を選択することで直接店舗の詳細を表示するページに移動するものになっている．このため，上記の項目 (iii) に対応するページにおいてパスがマッチせず，店舗名を抽出することができなかった．

また，実行例 1 と 2 に共通する問題として，選択する数が少ないときには，その場所によって，抽出対象の集合が大きく異なる点があげられる．この防止には，多数個選択するか，後から選択箇所を追加すると良い．

以上のことから，ウェブサイトの構成がある程度単純な場合には意図通りの結果を得ることが可能であり，ウェブサイトの構成が状況によって異なる場合には適切に得られない場合があることが確かめられた．

図 9　実行例 1 の抽出結果

6.　結　　論

本稿では，単一 HTML ページを対象とした情報抽出手法を拡張することで，ウェブの巡回例の例示からの学習を用いた情報抽出手法を提案した．

提案手法の問題点として，トップページから抽出対象 HTML ページへのリンクの長さが同一のものしか扱えないという制約をもつ．今後の課題として，繰り返しや可変長ギャップを許した正規表現を用いた抽出スクリプトの拡張があげられる [1], [2].

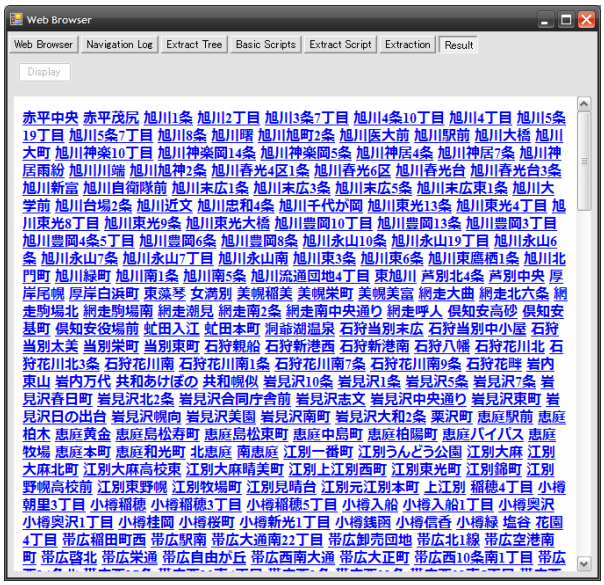

図 10　実行例 2 の抽出結果

また，本手法に，HTML ページの内容や構造に関する制約を付加して，より利用者の意図に合った部分 HTML テキストを抽出できるようにすることも今後の課題である．もう一つの課題として，ウェブに対する柔軟なアクセス手法の実現のための抽出済みデータの再構成や検索があげられる．また，このための構造データの半自動的な統合やパターン発見等も興味深い問題である．

文　　献

[1] H. Arimura, H. Sakamoto, S. Arikawa, Efficient Learning of Semi-structured Data from Queries, Proc. the 12th ALT, LNAI 2225, 315-331, 2001.

[2] A. Brazma, Efficient Algorithm for Learning Simple Regular Expressions from Noisy Examples, Proc. AII/ALT'94, LNAI, 260-271, 1994.

[3] J. Fujima, A. Lunzer, K. Hornbak, Y. Tanaka, Clip, connect, clone: combining application elements to build custom interfaces for information access, Proc. UIST 2004, 175-184, 2004.

[4] K. Ito, Y. Tanaka, Web Application Wrapping by Demonstration, Proc. EJC'03, 266-281, 2003.

[5] N. Kushmerick, Wrapper induction: Efficiency and expressiveness, Artif. Intell., 118(1-2): 15-68, 2000.

[6] A. Morishima, H. Kitagawa, A. Matsumoto, A Machine Learning Approach to Rapid Development of XML Mapping Queries, Proc. IEEE ICDE'04, 276-287, 2004.

[7] 村上義継，坂本比呂志，有村博紀，有川節夫, HTML からのテキストの自動切り出しアルゴリズムと実装, 情報処理学会論文誌：数理モデル化と応用，**42**(SIG 14) (TOM 5), 39-49,2001.

[8] H. Sakamoto, Y. Murakami, H. Arimura, S. Arikawa, Extracting partial structures from HTML documents, Proc. 14the Int'l FLAIRS Conference, 264-268, AAAI Press, 2001.

[9] W3C, XML Path Language (XPath) Version 1.0, W3C Recommendation, 1999.