F-039

帰納論理プログラミングを用いたWebラッパー自動生成

Automatic Web Wrapper Generation and Induction using Inductive Logic Programming

河野 碧† 西山 裕之† 大和田 勇人†
Midori Kouno Hiroyuki Nishiyama Hayato Ohwada

1 はじめに

Web 上には大量のテキストデータが存在し，その数は日々増加している．それらの多くはブラウザで閲覧され，人間がそのデータを認識，理解するという利用方法が前提となっている．しかし，その量は膨大であり，さらに日々増加しているため，これを機械的に処理したいというニーズがある．このように Web を巨大な知識ベースと捉え，Web から有用な知識を取り出す情報抽出技術の研究が行われている [1]．この技術は Web 内容マイニングとも呼ばれている [8]．

しかし，Web から取得した情報をそのままデータとして利用しようとすると問題が生じることが多い．これは Web 上に存在する文書の多くが HTML などの半構造化文書のためである．HTML は文書の構造を表す言語であるが，HTML 文書の多くがブラウザ閲覧を前提にしているため，文書の構造を表す情報とブラウザで表示のためのレイアウトを指定する情報が混在した半構造化文書となっている．そこで，HTML 文書からデータを抽出し，構造化する **Web** ラッパーが注目されている．Web ラッパーとは，HTML などで書かれた Web ページから特定の情報を自動的に抽出するための，サイトレイアウトを利用した抽出ルールおよび抽出プログラムのことである [7]．Web ラッパーはこれら半構造化文書から，レイアウト情報を利用して特定の情報の位置を把握，抽出を行うことで，構造を再構成する．Web 上のデータの膨大さを考えると，Web ラッパーを手動で作ることは現実的ではなく，自動的に生成することが課題となる．

Web ラッパー自動生成の課題は情報の抽出ルールの導出である．抽出ルールは HTML ページ内に含まれるレイアウト情報を利用するため，サイトごと，ページの種類ごとに異なる．そこで，ページレイアウトを解析し，ページ内における抽出したい情報の位置の把握，すなわち抽出ルールの導出を自動で行う必要がある．本研究ではページレイアウトの情報およびユーザが抽出したい情報を学習器に学習させ，抽出ルールを導出し，Web ラッパーを生成する手法を提案する．また，この提案手法の有効性を実証すべく，提案手法を用いたシステムを実装し，評価実験を行う．

2 関連研究

抽出ルールを機械的に導出するための手法は多く研究されている [3][2]．その多くは，ユーザの労力を減らすべく，ユーザの抽出したい項目の推論を行う [4]．

2.1 Kushmerick のラッパー帰納

トレーニングサンプルを与えて，抽出すべきコンテンツの前後に現われる文字列を学習するラッパー帰納問題を示す．Kushmerick[5] が提案したラッパーのうち最も単純な LR ラッパーを説明する．例えば，表 1 のような

表 1: データを含むテーブルの例

CPU	2.8GHz
メモリ	512MB

テーブルを仮定すると，これは以下のようなコードで表されている．

```
<tr><td>CPU</td><td>2.8GHz</td></tr>
<tr><td>メモリ</td><td>512MB</td></tr>
```

テーブルに含まれるデータのみを抽出するルールを，データを囲む左右の文字列で示すことにすると，この表 1 に含まれるデータのみを抽出するルールは

```
a1=(<tr><td>,</td>)
b1=(<td>,</td></tr>)
```

となる．この抽出ルール群がラッパーであり，いったんラッパーを構築することができれば，任意の HTML ページから自動的に必要な情報を抽出可能である．

2.2 HTML 文法知識を利用した学習

Embley[3] らは，入力ファイルを HTML に固定し，いくつかのヒューリスティクスを組み合わせて，HTML 文法の知識を利用することで情報抽出規則を生成する高精度なアルゴリズムを提案している．ヒューリスティクスとは，例えば，

(1) 繰り返し現われるタグは区切り目である，

(2) <hr>,<td>,<tr>,<a>,<p>,
など特定のタグの直後に区切り目がある

などである．この手法ではトレーニングサンプルが不要であるという大きなメリットがあるが，HTML 以外のマークアップ言語には適用しにくく，特に，XML のように自由にタグを定義できる場合，特定のタグの意味を考えなければならない点が問題となる．

2.3 IEPAD:文字列の繰り返しの発見

IEPAD[2] は，入力された HTML ファイルの規則性を探そうとするものである．IEPAD では，訓練例に特別な加工は必要なく，手に入れた HTML をそのままアルゴリズムの入力として利用できる．IEPAD は繰り返し登場する HTML の文字列のパターンのうち，文字列数が最長のものを探し出してそこを切り出すことにより情報

† 東京理科大学大学院理工学研究科，Tokyo University of Science

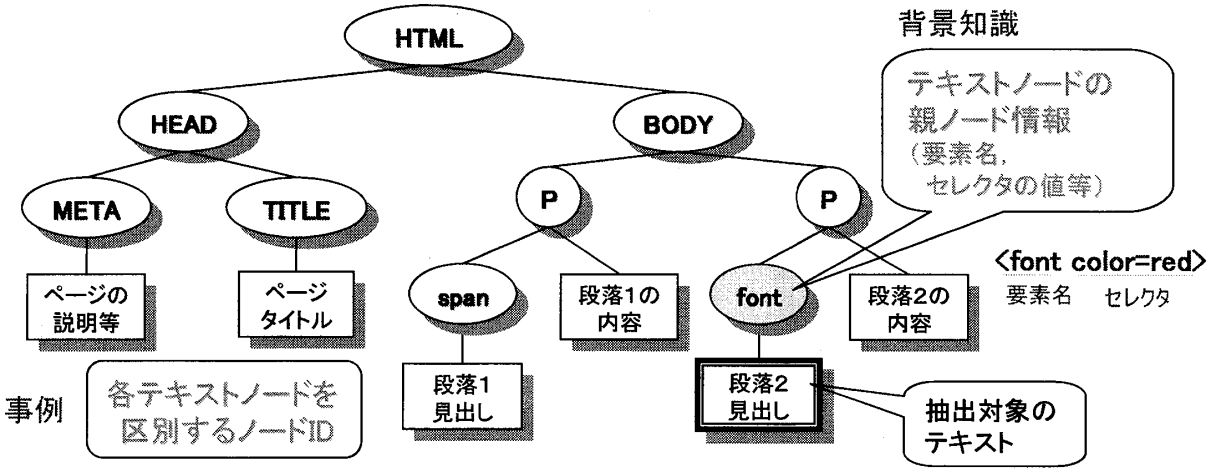

図 1: 木構造で表現した HTML の例

を抽出しようという考え方である．これにより，単一レコードの繰り返しを含むページであれば，自動的に Web ラッパーを生成可能である．この手法にはユーザの手間は最小限になるというメリットがあるが，対象ページを単一レコードを含むページに限定しているため，抽出不能なページに対しては何ら適用できないという問題点が存在する．

2.4 ユーザの労力を最小限に抑えたラッパー

ユーザが抽出したい項目を自動予測し，その項目の抽出ルールを導出するアプローチでは，ユーザの労力が少ない反面，精度が低いという問題がある．Irmak ら [4] は，入力ファイルに含まれる文字列群の中で，ラッパーで抽出可能な項目群を生成し，ユーザのニーズに沿う確率が高いと予想される順にランクづけしてユーザに提示し，ユーザに選択させる手法を提案している．この手法は，ユーザの労力を最小限に抑えたまま，ユーザの望むデータだけを抽出しようとするアプローチである．

3 提案手法

3.1 提案手法概要

本章では提案手法における Web ラッパーにおける抽出ルールの導出方法について述べる．既存研究では，対象ページを単一レコードのみに限定することで抽出ルール導出手法を単純化し，Web ラッパーの自動生成を可能にしていた．この手法ではユーザの労力をできる限り抑えることができるが，これでは抽出不能なページに対しては何ら適用できない．そこで本研究では，ユーザの労力を出来る限り抑え，任意のテキストデータを抽出可能にする Web ラッパーを生成する手法を提案する．幾つかのサンプルページを用意し，その中からユーザが希望するテキストデータを選択することで，各々のテキストデータに関して抽出ルールを導出する．これにより，導出した抽出ルールを保持した Web ラッパーを生成することで，任意のデータを抽出可能にする．以下に抽出ルールの導出方法について述べる．

3.2 抽出ルールの導出方法

抽出ルールは，抽出したい情報が HTML 文章のどこに含まれるかを示すルールである．例として，HTML 文章中のある 1 つのテキストノードを示す方法として，ルートからの Path を示す方法がある．しかしこの表現を用いた抽出ルールは，文書のタグ構造がわずかでも変化すると利用できなくなる．従って，抽出ルールは出来る限り一般的な記述であることが望ましい．

このような一般化されたルールを抽出するために，本研究では学習器を用いる．ここで対象となる問題を考える．タグの構造を利用して，抽出したいテキストの HTML 文書内の位置を説明し，他のテキストを説明しない概念を学習する必要がある．このために，抽出ルールの導出には帰納論理プログラミングによる学習器の 1 つである Progol を用いる．Progol で学習を行うためには Progol に学習データを入力する必要がある．これには，ユーザが抽出したい項目および同一 HTML ページに含まれるページレイアウトに関する情報を学習器である Progol に入力するために，Prolog 形式で問題を記述する必要がある．以下に，Progol に与える学習データの生成法について述べる．

3.3 学習データ生成

3.3.1 タグ構造解析

Progol に与える学習データの生成の前処理として，サンプルページのタグ構造解析を行う．これにより，事例や背景知識に必要となるテキストノードやタグに関する情報などを取得しやすくする．HTML 文章はタグによって文書構造を表す．このタグ構造は，HTML タグを根ノードとし，タグを関係ノードとする木構造 (DOM 木，ドキュメントオブジェクトモデル木) となる．簡単な HTML を木構造で表現した例を図 1 に示す．サンプルページに対する DOM 木を生成すると，その葉ノードのうちテキストノードは事例になり，テキストノードの先祖ノードは背景知識となる．なお，DOM 木の生成においては Xerces[10] と nekoHTML[11] を用いる．これにより，対象となるサンプルページの URL を入力することで DOM 木オブジェクトを生成することができる．

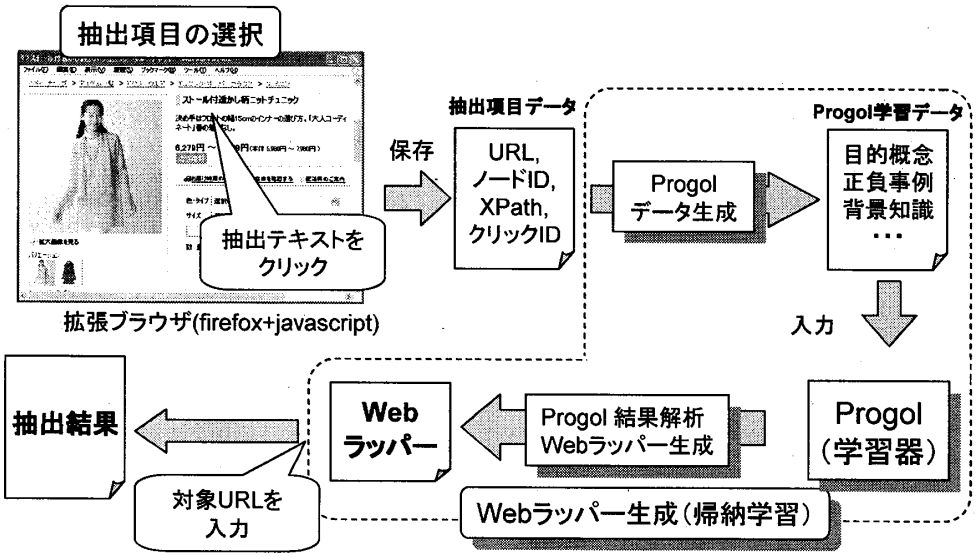

図 2: Web ラッパー生成の流れ

3.3.2 Progol への入力情報

Progol に学習させるために与える必要がある入力データは，モード宣言，正負事例，背景知識，パラメータなどである．モード宣言は問題概念の定義であるため，全抽出ルール導出問題において同一の定義となる．正負事例および背景知識の値は，3.3.1 で構造を解析したサンプルページの HTML ソースから抽出している．パラメータは Progol 4.4 におけるデフォルトの値を用いている．

モード宣言　モード宣言は，目的概念とそれを説明する概念を宣言するものである．本研究での目的概念は正事例を説明するルールの抽出を行うこと，目的概念を説明する概念には親タグの要素名，属性，属性値のペアの背景知識を用いることを宣言している．実際に抽出ルールを導出した際に用いたモード宣言を図 3 に示す．

【モード宣言】

```
:- modeh(*,node(+pageID,+nodeID))?
:- modeb(*,parentAttribute(+pageID,
    +nodeID,#attribute,#value,#depth))?
:- modeb(*,parentName(+pageID,+nodeID,
    #parent_name,#depth))?
```

【変数の説明】

- pageID:各サンプルページを区別する ID
- nodeID:各ノードを区別する ID
- attribute:タグに含まれる属性名
- value:attribute の属性値
- depth:テキストノードからの階層

図 3: モード宣言

正負事例　事例は，サンプルページに含まれるテキストノードである．実際には，サンプルページに含まれる各テキストノードにノード ID を付与し，その値を用いている．正事例は抽出項目のテキストノードであり，負事例は正事例以外の全てのテキストノードである．ただし，単一ページレイアウトから複数項目を抽出する場合は，抽出したいテキストノード全てを負事例から除く．学習には複数ページの正事例が必要であり，これをトレーニングサンプルと呼んでいる．ユーザは 2，3 ページ程度のサンプルページに対し，トレーニングサンプルを選択する．トレーニングサンプル選択時には，後述の拡張ブラウザ 4.1 が利用可能であり，これによってユーザの労力をなるべく減らしている．

背景知識　背景知識としては，事例を囲むタグに関する情報を与える．このタグは HTML ソースを木構造で表現した際に，抽出すべきテキストコンテンツの親ノードにあたる．ここでは事例を囲むタグを便宜上「親タグ」，親タグを囲む親タグを「先祖タグ」と呼ぶ．具体的には，親タグの要素名，属性，属性値のペア，階層の深さの情報を与えている．事例の親タグに関する情報だけではルールが生成されない場合，先祖タグに関して同様の情報を与えて再学習 (3.3.3) を行っている．

3.3.3 先祖タグの情報を含めた再学習

3.3.2 では，全正負事例について，親タグのみの背景知識を記述している．親タグの情報のみで学習できる場合，最も一般化された抽出ルールを得ることができるが，これでは抽出ルールが導出されないことがある．そこで，抽出ルールが導出されない場合，親タグだけでなく，一階層上の先祖タグの情報も追加して再学習を行わせる．これを抽出ルールが導出されるまで繰り返し，DOM 木の葉ノードからボトムアップで情報を与え，複数回の学習を行うことで，なるべく一般化されている抽出ルールを導出する．ページレイアウトによってはタグの親子関係やその属性だけではノードの位置を特定できない場

合も考えられる．このような場合，本アルゴリズムは根ノードまでの全てのノードの情報を与え，抽出ルールの導出に失敗し，終了する．このアルゴリズムは，HTML に含まれる全ノードの情報を利用するような複雑な抽出ルールの抽出の場合には何度も学習が必要となるため冗長となるが，各学習試行においては必要最小限の知識のみを与えているため，一般化された抽出ルールが単純な場合の導出に要する時間を短縮するとともに，必要となるメモリ量もなるべく少なく抑えている．

4 Web ラッパー生成システム

本研究では，3 で述べた提案手法の有効性を実証すべく，提案手法を用いたシステムを実装した．図 2 に Web ラッパー生成システムについて示す．本章では，図 2 の Web ラッパー生成の流れに沿って，トレーニングサンプル入力のためのユーザインタフェースであるブラウザの拡張について述べた後，3.3 で述べた Progol 学習データを実際に生成する手法について説明する．また，学習結果を解析し，Web ラッパーを自動生成する手法について述べる．

4.1 トレーニングサンプル入力インタフェース

トレーニングサンプル入力用ユーザインタフェースを拡張ブラウザを用いて実現した．トレーニングサンプル入力用ユーザインタフェースの利用例を図 4 に示す．

このユーザインタフェースにおいて実装した機能は，抽出可能テキストのハイライトおよび抽出項目の保存である．このユーザインタフェース用のブラウザには拡張性の高い Firefox を用い，独自拡張機能である抽出可能テキストのハイライトおよび抽出項目の保存は Firefox アドオンである Greasemonkey[12] を用いて，JavaScript で記述したユーザスクリプトを実行することにより実現した．Greasemonkey は，Firefox でページ閲覧を行う際，ソースのロード時に JavaScript で記述されたユーザスクリプトを反映させるアドオンである．以下に実際に実装した機能について述べる．

抽出可能テキストをユーザに示すため，各テキストノードをハイライトすることで，抽出可能テキスト単位を表示する．また，抽出登録済みテキストと未登録テキストを区別するため，登録済みテキストのアウトライン色を変更している．

1 項目の学習に対し複数の正事例を用いた学習を行うため，各学習項目を区別するための学習項目 ID の情報を付与する．通常 1 項目の学習には 1 テキストを登録するため，1 クリックごとに新しい学習項目とみなせばよいが，複数の項目を登録する際には，複数クリックを 1 項目の正事例とみなす必要がある．この問題を解決するべく，Ctrl 押下中のクリック時は複数の正事例を含む学習として，前のクリックと同じ学習項目 ID を付与するものである．

Ctrl キーと Shift キーの同時押下時に，抽出項目に関する情報をローカルに保存する．通常の JavaScript にはセキュリティの関係上，任意のデータをローカルに保存する機能はないが，Firefox では data スキームが application/octet-stream であるデータの場合，このデータの保存を行うため，保存データを URI にエンコードし，Firefox に通常のバイナリデータと認識させることで保存している．この結果，ファイル名がデータに依存した 8 文字の文字列，拡張子が part であるファイルが保存される．

4.2 事例および背景知識の抽出

3.3 で述べた，Progol への入力情報の生成を行う．Progol への入力のうち，正負事例および背景知識はサンプルページの HTML ソースから抽出する必要がある．そのため，各情報が HTML ソースのどの位置に含まれるか把握する必要がある．この際，4.1 で述べた GUI で保存した part ファイルの情報を用いる．まず，part ファイルからサンプルページの URL を取得し，この HTML ソースを取得する．この HTML ソースから DOM 木を生成し，この DOM 木に含まれるテキストノードの中で，ユーザが抽出したいと選択，入力した抽出したいテキストがどれかという情報をノード ID により把握する．また，この際，学習項目 ID を参照することで，同一の学習項目 ID を持つノードが複数登録されていた場合は，複数の正事例を登録した入力情報の生成を行う．

4.3 Progol 学習データの生成

本研究で実装したシステムでは，まず Progol に学習させる Prolog 形式のデータを Java に出力させ，pl ファイルとしてローカルに保存する．次に Java から外部コマンドとして Progol を実行し，保存した pl ファイルを学習データとして渡すことでシステムが自動的に学習することを可能にしている．そのための pl ファイルの出力方法について述べる．

Progol が学習に必要とする入力ファイルには，まず，表 3 に示したモード宣言を記述する．次に，Xerces+nekoHTML から得られた DOM 木の XML 型オブジェクトからテキストノードのみの配列を生成する．この配列から正事例を探し出し，全てのサンプルページに固有なページ ID を，全てのノードに固有なノード ID をつける．固有なページ ID およびノード ID はタイプ宣言として登録し，入力用の変数として利用する．探し出した正事例を Progol 入力ファイルに記述した後，残った配列要素を同様に負事例として記述する．次に，全正負事例について，ページ ID, ノード ID, 親タグの要素名，属性，属性値のペア，階層の深さの情報を引数に持つ背景知識として記述する．

4.3.1 Progol による学習および学習結果の解析

抽出ルール導出のため，4.3 で生成した学習データを Progol に学習させる．Java の外部コマンドによって Progol を呼び出し，4.3 で生成した pl ファイルを引数に与え，学習させる．結果表示のストリームを Java で受け取り，抽出ルールが導出できているか判定する．抽出ルールが導出されていなければ，3.3.3 で述べた背景知識を追加した再学習を行う．導出されていれば Prolog 形式で書かれている抽出ルールを 4.2 で述べた DOM 木に対する条件へと変換し，Web ラッパーを生成する．抽出ルールの変換および Web ラッパー生成については 4.4 に述べる．

4.4 学習結果の解析および Web ラッパー自動生成

本研究では，情報抽出したいページの URL を引数とし，実行すると情報抽出を行う Web ラッパーを生成した．Web ラッパーの雛型を用意し，抽出条件を埋め込むことで生成を行った．導出された抽出ルールは述語と変数にパースし，4.2 で述べた DOM 木に対する条件へと変換，雛型に埋め込んでいる．なお，DOM 木の生成に Xerces[10] と nekoHTML[11] を用いているため，生成した Web ラッパーのコンパイルおよび実行時にはそれらが必要となる．

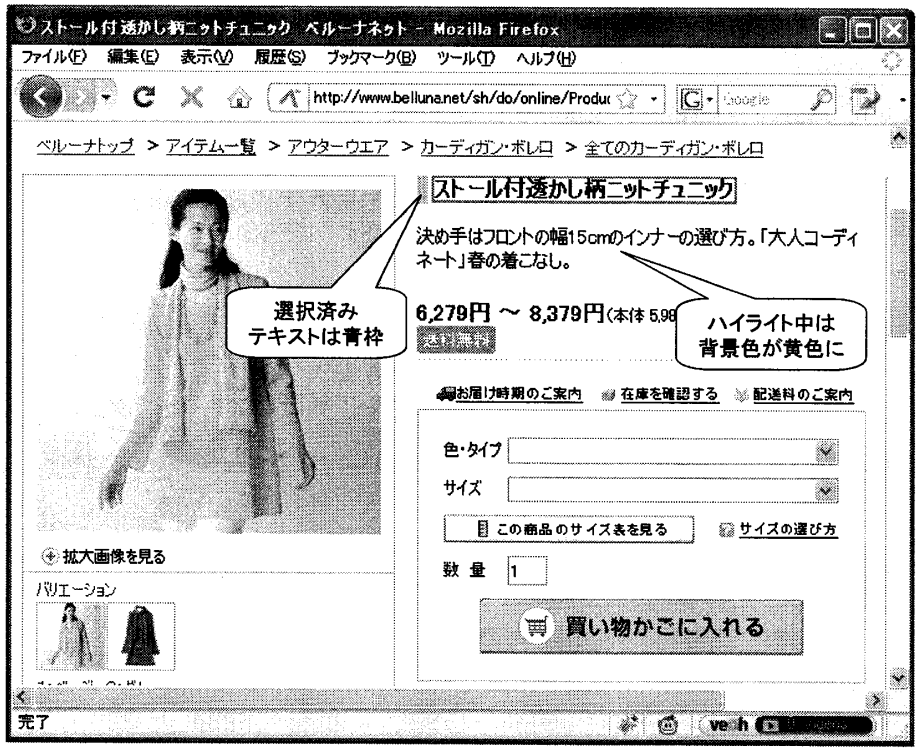

図 4: トレーニングサンプル入力用ユーザインタフェース

5 評価実験および考察

本章では，提案手法を用いて実装したシステムの評価実験および考察について述べる．評価実験は，抽出ルール導出実験，Web ラッパー自動生成実験および Web ラッパー応用実験を行った．各実験におけるハードウェアおよびソフトウェアの実験環境を表 2 に示す．

表 2: 実験環境

OPU	Intel(R) Core(TM)2 Duo CPU
メモリ	2GB
OS	Microsoft Windows XP Home Edtion Version 2002 SP3
Progol	CProgol 4.4 (+ cygwin1.dll)
Java	JDK:1.6.0_11, JRE: 1.6.0_07
ブラウザ	Firefox ver. 3.0.5

5.1 抽出ルール導出実験

女性向けアパレル販売 EC サイト，「ベルーナ」(http://www.belluna.net/) および「IMAGE」(http://www.st-image.com/) のサイトを利用して抽出ルール導出実験を行った．各サイトの商品詳細ページに対し，10 の抽出項目に対する抽出ルールを導出し，Web ラッパーを手動で生成した．10 の抽出項目とは，商品名，税込価格，税抜価格，商品のキャッチ文，カラーバリエーション，サイズバリエーション，商品の情報，商品の説明，商品カテゴリ，商品コードである．この実験の結果，ベルーナ，イマージュとも全ての抽出項目に対して抽出ルールを導出できた．また，手動で生成した Web ラッパーを各サイト 2,000 ページ以上に適用したところ，全てのページに対して情報抽出し，保存することができた．抽出ルール導出に要する時間はサイトの応答時間に依存するが，両者のサイトに関しては Progol 学習ファイルの出力および学習に要する時間は 800 ミリ秒から 2300 ミリ秒であり，実用に耐えうる時間であると考えられる．抽出されたルールは，ベルーナのサイトでは親タグの属性として class セレクタの値を用いたルールが多く，IMAGE のサイトでは id セレクタの値を用いたルールが多かった．また，本実験で導出された抽出ルールは全て親タグに関する条件のみで構成されており，再学習は行われなかった．

5.2 Web ラッパー自動生成実験

5 つのサイトに対して Web ラッパー自動生成実験を行った．amazon.co.jp (http://www.amazon.co.jp/)，ベルメゾンネット (http://www.bellemaison.jp/)，ファミマドットコム (http://www.famima.com/top/CSfTop.jsp)，Nissen On-line(http://www.nissen.co.jp/)，ヨドバシカメラドットコム (http://www.yodobashi.com/) の 5 サイトの各サイトの商品詳細ページに対し，商品名，価格，商品の説明の 3 項目の抽出ルールを導出し，Web ラッパーの自動生成を行った．この実験の結果，ベルメゾンネット，ファミマ・ドットコム，ヨドバシカメラドットコムの 3 サイトの抽出ルールを自

動抽出し，Web ラッパーを自動で生成することができた．amazon.co.jp，Nissen On-line の 2 サイトはサンプルページに含まれる DOM ノードの数が多すぎるため，Progol で割り当て可能なメモリの上限を超えてしまった．これら 2 サイトは，抽出ルールの自動導出は出来なかったが，手動で抽出ルールを導出し，Web ラッパーを生成したところ，抽出可能であった．このため，これら 2 サイトに関しては，Progol に対する割り当て可能なメモリ量の問題が解決できれば，抽出ルールの導出可能性があると考えられる．また，Web ラッパーの自動生成に 730 ミリ秒から 3400 ミリ秒を要し，Web ラッパーを実行し，抽出結果取得に 280 ミリ秒から 2480 ミリ秒を要した．なお，抽出結果取得時に連続してページ取得を続けるとページ提供サイトからページからアクセスを拒否されることがあるため，ページアクセスは一定の時間間隔を設けた方が良いことがわかっている．導出された抽出ルールは前述の実験同様，親タグに関する条件のみで構成されていたが，class セレクタや id セレクタ以外の条件を含むルールが導出されていた．

5.3 Web ラッパー応用実験

EC サイト以外のサイトに対する Web ラッパー生成実験を行った．対象サイトは Yahoo!天気予報（`http://weather.yahoo.co.jp/weather/`）および Yahoo!路線情報（`http://transit.yahoo.co.jp/`）である．本研究で実装したシステムが自動生成する Web ラッパーは，抽出した情報を標準出力に出力する単純なプログラムであるが，この出力を応用し，前者に対しては全国の週間天気予報を抽出するプログラムを生成することができた．この Web ラッパーを実行すると，全国 50 地点における実行日から向日一週間の天気，最高最低気温，降水確率を取得する．後者に対しては出発地点，到着地点，出発時刻もしくは到着時刻を入力として与えると，その経路と所要時間を含む XML を出力するプログラムを生成することができた．このように，提案手法は EC サイト以外の情報提供サイトにも適用可能であり，また，生成される Web ラッパーの出力を応用することで二次利用が可能であることを示した．また，Yahoo!路線情報における抽出ルール導出時に，再学習を行うことで 3 階層上の先祖タグに関する条件を含むルールを導出した．これにより，再学習アルゴリズムを用いることで，なるべく一般化されたルールを導出可能であることが示せた．

5.4 考察

抽出ルール導出実験により，提案手法による Web ラッパー抽出ルール導出が可能であることを示すことができた．導出された抽出ルールが class, id セレクタを用いていたのは，現状の HTML のタグだけでは文書の論理構造を明確化するのが困難であるため，サイト製作者が class や id を付与することでこの問題の解決を試みており，Web ラッパーがページの中から抽出情報の位置を把握する際に，この文書の論理構造を知る手掛かりとしてこれらの値が利用可能であったためと考えられる．また，Progol による抽出ルール導出を利用した Web ラッパーの自動生成実験も成功した．これにより，本提案手法を用いた Web ラッパー自動生成が可能であることを示した．さらに，抽出結果の応用実験により，二次利用の可能性も示すことができた．この実験では，再学習アルゴリズムによって 3 階層上の先祖タグに関する条件を含む抽出ルールが導出されたことから，再学習アルゴリズムの有効性が示せた．このアルゴリズムにより，抽出精度が 100% となる抽出ルールの導出を可能にしている．なお，抽出ルールの導出および Web ラッパーの生成に要する時間はサイトごとに異なるものの，3400 ミリ秒以下と短時間である．

6 おわりに

本研究ではユーザ入力の抽出項目に対する抽出ルールを帰納論理プログラミングによる学習器である Progol を用いて導出し，そのルールに基づく Web ラッパーの自動生成手法を提案した．また，提案手法による Web ラッパー自動生成システムを実装し，評価実験を行うことでその有効性を示した．

参考文献

[1] C.-H. Chang, M. Kayed, R. Girgis and K. F. Shaalan: A Survey of Web Information Extraction Systems, *IEEE Transactions on Knowledge and Data Engineering*, vol.18, pp.1411-1428, 2006.

[2] C.-H. Chang and S.-C. Lui: IEPAD: Information Extraction Based on Pattern Discovery, *the Tenth International Conference of World Wide Web (WWW2001)*, pp.4-15, 2001.

[3] D. W. Embley, Y. Jiang and Y. -K. Ng: Record-boundary discovery in Web documents. Proc. *ACM SIGMOD International Conference on Management of Data*, 1999.

[4] U. Irmak and T. Suel. Proc. of *WWW 2006*, pp.553-563, 2006.

[5] N. Kushmerick: Wrapper Induction: Efficiency and Expressiveness, *Artificial Intelligence*, Vol.118, pp.15-68, 2000.

[6] S. Muggleton: Inverse Entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4), pp.245-286, 1995.

[7] 池田 大輔，山田 泰寛，廣川 左千男： Web 上の多言語テキストデータからのラッパー自動生成, 九州大学情報基盤センター年報, Vol.3, pp. 7-14, 2003.

[8] 奥村学： WWW 上の情報の知的アクセスのためのテキスト処理，人工知能学会誌, Vol.19, No.3, pp.295-323, 2004.

[9] 古川 康一, 植野 研, 尾崎 知伸： 帰納論理プログラミング, 共立出版，2001.

[10] Apache Software Foundation: Xerces: XML parsers in Java and C++. `http://xml.apache.org/#xerces`

[11] CyberNeko HTML Parser: `http://nekohtml.sourceforge.net/`

[12] Greasemonkey: `http://addons.mozilla.org/ja/firefox/addon/748`

[13] Google デベロッパー: `http://code.google.com/intl/ja/`

[14] Yahoo デベロッパーネットワーク: `http://developer.yahoo.co.jp/`