Automatic Tree and String Based Wrapper Generation for Semi-structured Documents

山田, 泰寛
九州大学大学院システム情報科学府

池田, 大輔
九州大学情報基盤センター

廣川, 佐千男
九州大学情報基盤センター

出版情報：情報処理学会研究報告 ： 自然言語処理. 2003 (98), pp.115-122, 2003-09
バージョン：

半構造化文書に対する木構造と文字列を組合せたラッパーの自動生成法

山田泰寛[†] 池田大輔[††] 廣川佐千男[††]

同種の項目を多数含む半構造化文書群から，各項目を抽出するラッパーの自動生成法を提案する．本手法では，まず部分文字列の長さと出現頻度に基づき，半構造化文書を構造記述部分とコンテンツ記述部分に分離する．これにより，機械学習等によるラッパー生成で必要となる人手による訓練例の作成が不要となる．次に，対象文書を木構造として捉え，コンテンツ部分を含むノードに対するパスとして抽出部分を特定する Tree ラッパーを構成する．最後に，同じパスで特定されるノードに現れる文字列群に対し，共通部分を特定することにより，パターンとコンテンツの境界を厳密に分離する LR ラッパーを構成する．従来の Tree ラッパーで排除できなかった不要な文字列が削除できることを実験的に確認できた．

Automatic Tree and String Based Wrapper Generation for Semi-structured Documents

Yasuhiro Yamada [†] ,Daisuke Ikeda [††] and Sachio Hirokawa[††]

We propose an automatic wrapper generation algorithm to extract contents from semi-structured documents with a lot of same items. Our wrapper is expressed by paths of the tree structure and two common delimiters surrounding the item in the path. It can extract partial strings even if a node contains useless strings. The algorithm separates a page into templates and contents using optimal cut point $(n, a\%)$, where n is the length of a substring and a is the frequency of the substring. The top $a\%$ frequent substrings with length n appear on template parts. Experiments show that the algorithm discards useless substrings which tree wrappers extract as contents.

1. はじめに

WWW 上に存在する膨大な量の情報は頻繁に追加・更新・削除されている．その中には有用な情報が多く含まれているが，それらは様々なサイト上にあり，様々な形式で記述されているため，異なるサイト上の同種の情報を比較することは容易ではない．例えば，複数の自動車メーカーとディーラーのサイトから「各メーカーのハッチバックの車で，値段は○○以下のもの」という情報を探す場合，検索エンジンを利用したとしても，多くのリンクを辿り，多くのページを閲覧し比較を行なわなければならない．また，必要な情報を記憶・記録しておかなくてはならず，手間と時間のかかる作業である．異なるサイト間の情報を容易に閲覧・比較する為には，これらの情報を効率良く抽出し統合する必要がある．

† 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University, yshiro@matu.cc.kyushu-u.ac.jp
†† 九州大学情報基盤センター
Computing and Communications Center, Kyushu University, {daisuke, hirokawa}@cc.kyushu-u.ac.jp

異なるサイト間の情報を統合するサービスがいくつかある．例えば，GoogleNews☆は，ニュースサイト群を対象とし，新聞記事から見出しや本文の一部を抽出し，それらを統合した一覧を生成している．DealTime☆☆では，オンラインショッピングサイトを対象として，商品の値段やその詳細を統合している．Flipdog☆☆☆では，求人情報を統合している．

WWW 上に存在するこのような情報は，膨大にあるだけでなく，頻繁に追加・更新される．したがって，手動での統合には限界があり，自動化が必要となる．より詳細に分析すると，必要となるプロセスは，情報の (1) 発見・収集，(2) 抽出・整列，(3) 統合の 3 つから成る．

本論文では (2) の抽出について新たな手法を提案する．(2) 抽出・整列の為の技術として，同一サイトの表やリストの形式をとり，同種の項目が繰り返し現れている半構造化文書を対象とし，そこから各項目を抽出するプログラムの生成法について研究が行なわれている [1)~6),9)~15)]．このようなプログラムは**ラッパー**と呼ばれる．一度ラッ

☆ http://news.google.com/
☆☆ http://www.dealtime.com/
☆☆☆ http://www.flipdog.com/

パーを生成すると，次回からはラッパーを生成するプロセスなしに自動的に半構造化文書から同種の情報を抽出することができる．

しかし，このようなページは決まった構造を持っておらず，内容も様々である．これは，それらの多くは閲覧者が直接見たり読んだりし易いように記述されており，計算機が扱い易いように記述されていない為である．また，サイトが違えば，同種の項目を持つページであっても，その構造やフォーマットが異なっている．従ってサイト毎，同種の項目を持つページ群毎にラッパーを生成しなければならない．手動でラッパーを生成することはコストの大きい仕事である．また，使用されているマークアップ言語を熟知していなければラッパーの生成は難しい．よって，生成法自体も自動的であることが望ましい．また，WWW 上の情報の多様性を考えると，多言語に対応できる生成法が求められる．

これまで機械学習を用い，訓練例を手動で作成し，それを入力として与える半自動的なラッパーの生成法が多く提案されている [9),12),13)]．Baumgartner ら [2)] や Minton ら [11)] は，GUI を実装することにより，ラッパーの生成，訓練例の生成を支援している．いずれにせよ人手に対するコストの問題点がある．

自動的なラッパー生成で重要となる部分は，抽出箇所の特定もしくはテンプレート部分の特定である．Ashish ら [1)] は，<H1> やボールド体などの強調文字に着目し，見出しを抽出する為のラッパーを自動生成している．Embley ら [5)] は，境界はいくつかの特別なタグ <hr>，<td>，<tr>，<a>，<p>，
 であるという仮定等，レコードの境界についてのいくつかのヒューリスティックの組み合わせを用いることでレコードの境界を特定している．Crescenzi ら [4)] や Lerman ら [10)] は，入力をタグや単語のトークン列に変換し，複数のファイルに共通するトークン列をテンプレート部分として特定している．Chang ら [3)] では，文書中に繰り返し現れるタグやテキストの列を抽出する部分として特定している．

本論文で提案する手法では，まず共通部分特定アルゴリズム [7)] を用いて，同種の項目を多数含む半構造化文書の集合から共通部分を特定する．共通部分特定により，訓練例の作成は不要となる．共通部分特定アルゴリズムは，**交代数**という計数を用いて，部分文字列の長さ n と頻度の割合 $a\%$ を自動的に決定する．この時，長さ n の部分文字列のうち，頻度の上位 $a\%$ に含まれるものは，共通部分に出現する．交代数とは，文字列と部分文字列の集合が与えられたとき，文字列上でその部分文字列の出現する部分とそうでない部分の境界の総数を表す．

本手法では，次に非共通部分を抽出対象として抽出ルール生成を行なう．半構造化文書からの情報抽出としては，対象を文字列としてとらえる方法と木構造としてとらえる方法が提案されている．Kushmerick らによって提案された LR ラッパー [9)] は，抽出したい項目を囲む左区切文字列と右区切文字列の組からなる集合を抽出するものである．Kushmerick らの方法は機械学習によるもので，前提として訓練例が必要な半自動生成法であった．著者らは交代数を用いることにより，LR ラッパー自動生成の研究を行なって来た [14),15)]．しかし，LR ラッパーでは，抽出したい項目を囲む文字列がそれぞれ異なっている時には抽出できないという問題があった．

一方，半構造化文書のタグ構造に注目した Tree ラッパーの半自動的な生成の研究がある [12)]．半構造化文書はタグにより階層的な構造を持っている為，入力を木構造に展開し，ルートからノードまでのパスをラッパーの表現形式として用いるものである．しかし，ノードの中には不必要な文字列や，複数の項目が含まれている場合がある．このような不必要な文字列は，同種の項目間の対応づけを扱う Name Matching 問題における大きな障害となる．例えば，Ikeda[8)] は，抽出された文字列の文字コードに着目し項目間の類似性を測っている．この時，不必要な文字列が付いたものを用いると結果に影響し，対応が取れない．したがって，異なるサイト間の情報の統合の為にはこのような不必要な文字列を削除したり，ノードの中から細かく抽出する必要がある．

本論文では，木構造と文字列を段階的に組み合わせることによりこれらの問題を解決する **PLR ラッパー (Path-Left-Right ラッパー)** を提案する．

提案するラッパーは，まず木構造のパスによりコンテンツ部分を大局的に特定し，次にコンテンツを含むノードの部分について，左・右句切文字列を用いてより詳細に特定する．パスの示すノードにおいて，左・右句切文字列で囲まれるものを抽出することにより不必要な文字列を削除する．このような詳細部分の特定は，GUI を用いたり，典型的な学習例と抽出例を与えることでも実現可能だが，本手法はこれを自動で行なう．

本論文で提案するラッパー生成アルゴリズムは入力を単なる文字列として扱い，自然言語やマークアップ言語に依存する前処理や，サイトごとの特別な知識を用いない．空白文字についても，タグと同様に構造の一部を表していると考え，そのまま扱う．テキストの一部もパタンを特定する句切文字列になるので本手法は多言語に適応できる．実験では，Tree ラッパーでは抽出結果に残る不要な文字列を削除することに成功した．

本論文の構成は以下の通りである．2 節では，本論文で提案する PLR ラッパーのアイデアを述べる．3 節では，部分文字列の長さと出現頻度に基づくコンテンツ部分特定方法について述べる．特に，その基本概念である交代数と，それを用いた入力から共通部分を特定するアルゴリズムを述べる．4 節では，共通部分とテンプレート部

分がほぼ一致することを利用したラッパー生成アルゴリズムについて述べる．5 節では，実験とその評価を述べ，6 章でまとめと今後の課題について述べる．

2. PLR ラッパー

著者らは，これまで LR ラッパー [9] の自動生成について研究を行って来た [14),15)]．LR ラッパーとは，抽出したい項目を囲む左区切文字列 と右区切文字列の組からなる集合によって表現される．今，図 1 のような入力を考える．半構造化文書において，最も基本となる情報単位を**要素**と呼ぶ．body，font，a，BR，HR がそれにあたる．この時，要素が始まったことを示すタグは**開始タグ** と呼ぶ．その要素が終わったことを示すタグは**終了タグ**と呼ぶ．この 2 つで挟まれた部分を**テキスト**と呼ぶ．下位の要素をもたないタグを**空要素タグ**という．図 1 では，<body>，，<a> が開始タグ，</body>，， が終了タグ，
，<HR> が空要素タグ，“山田泰寛”，“Address: yshiro@matu.cc.kyushu-u.ac.jp” はテキストである．開始タグもしくは空要素タグに何らかの付属情報を与えたものを**属性**と呼ぶ．また，その属性の持つ値のことを**属性値**と呼ぶ．図 1 では，<font> は属性 size を持ちその属性値は 5 である．この例において名前を抽出するルールは，左区切文字列が 5”>，右区切文字列が < /f である．この 2 つの文字列によって，文書中の名前の位置を一意に特定できる．

```
<body>
<font size=‘‘5’’>廣川佐千男</font>
<BR>
<a href=‘‘mailto:hirokawa@xxx.jp’’>
Address: hirokawa@cc.kyushu-u.ac.jp
</a>
<HR>
<font size=‘‘5’’>山田泰寛</font>
<BR>
<a href=‘‘mailto:yshiro@yyy.com’’>
Address: yshiro@matu.cc.kyushu-u.ac.jp
</a>
</body>
```

図 1 LR ラッパーではルールの抽出が不可能な例

しかし，LR ラッパーでは，ある項目を囲んでいる適切な左区切文字列と右区切文字列が抽出できない場合がある．例えば，図 1 において，メールアドレスを囲んでいるアンカータグは属性値が人によってそれぞれ異なっている為，左区切文字列が抽出できない．左区切文字列が”> では，名前も一緒に抽出してしまう．

一方, [12)] などの Tree ラッパーでは，入力を木構造に展開し，ルートからノードまでのパスを用いて抽出箇所を指定する．図 2 は図 1 を木構造に展開したものである．図で丸で表したものを**ノード**と呼び，要素とテキストが

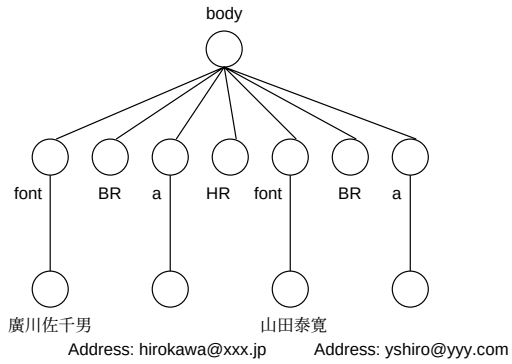


図 2 図 1 を木構造に展開したもの

対応している．Tree ラッパーを用いれば，図 1 からメールアドレスを抽出することは可能である．メールアドレスは body の下位の a の下位のテキストである為，ルールは “body-a-TEXT” である．このパスによって，メールアドレスの位置を一意に特定できる．

しかし，図 1 のメールアドレスには，“Address: ” という文字列が付いている．これはメールアドレスではない為不要な部分であるが，Tree ラッパーのルールではこの文字列もメールアドレスと一緒に抽出してしまう．このように同じパス中に，必要のない文字列が含まれる場合が存在する．Tree ラッパーでは，このように必要な部分を細かく指定することができない．

情報の統合を考えた時に，サイト A ではメールアドレスが抜きだされているが，サイト B では “Address: ” という文字列が付いたメールアドレスが抜きだされると，統合したときに問題となる．また，異なるサイトから抽出した項目群についてそれぞれ対応をつけ統合する際にも，ノイズとして影響を与える．よって，情報の統合を行う為に，より細かくコンテンツを抽出する必要がある．

我々は，このような問題を解決するために，ノードに対応する文字列からより詳細に抽出する **PLR ラッパー (Path-Left-Right ラッパー)** を提案する．

定義. 1(PLR ラッパー): PLR ラッパーは，入力として与えられた半構造化文書から各項目を抜きだす為の**ルール**の集合によって表現される．ルールとは各項目の出現する木構造の**パス**と，そのパスで特定されるノードに対応する文字列中の項目を囲んでいる**左区切文字列**と**右区切文字列**と呼ばれる文字列の組から成り立つ．

PLR ラッパーは，Tree ラッパーと LR ラッパーを組み合わせたものである．まず，項目部分を含むノードに対するパスを特定する Tree ラッパーを構成した後，共通パターンと項目を分離する LR ラッパーを構成する．

3. 交代数を用いた共通部分の特定

本節では，ラッパー生成アルゴリズムにおいて，入力として与えられた半構造化文書の集合から共通部分を求

める為に使われるアルゴリズム [7] について述べる．

共通部分特定アルゴリズムは，入力として同種の項目を多数含んでいる半構造化文書の集合を受け取り，それらを**高頻度部分**とそうでない部分 (以下，低頻度部分と記述) に分ける．この時，高頻度部分が共通部分つまりテンプレート部分と対応し，低頻度部分が非共通部分つまりコンテンツ部分と対応していると仮定し，テンプレート部分を特定する．

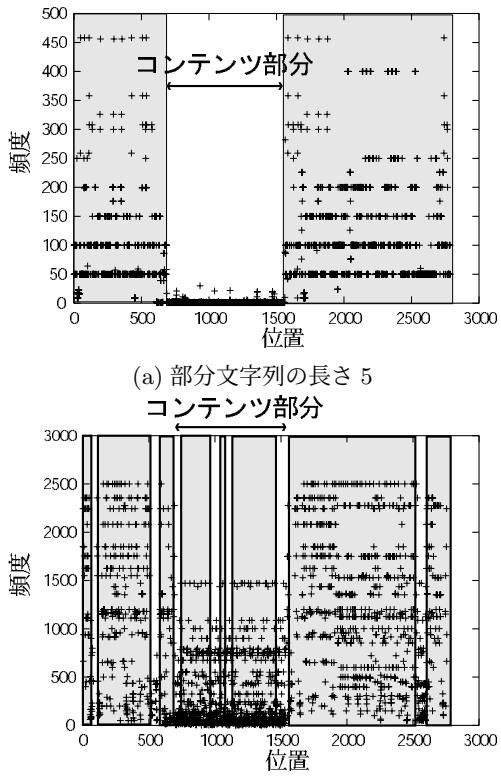
共通部分特定アルゴリズムは，**カットポイント**と呼ばれる 2 つの整数の組 (n, a) を出力する．n は部分文字列の長さ，a は割合 (パーセント) で $1 \leq a \leq 100$ の整数である．カットポイントを用いて，高頻度部分を以下のように定義する．D を文字列の集合とする．この時，D における全ての長さ n の部分文字列の内，頻度の上位 a%の部分文字列が D の各文字列上で現れる領域を高頻度部分と呼ぶ．

同種の項目を多数含んでいる半構造化文書は，テンプレート部分とコンテンツ部分から成り，異なるページであっても同一サイトであれば共通のテンプレートで記述されている．コンテンツ部分とテンプレート部分は，それぞれがある程度の長さを持っており，交互に複数回現れると考えられる．共通部分特定アルゴリズムは高頻度部分とテンプレート部分が対応するようなカットポイントを出力する．

図 3 は部分文字列の長さを長く設定した時 (図 3 (a)) 及び短く設定したとき (図 3 (b)) の文書中のある位置から始まる長さ n の部分文字列の頻度のグラフである．縦軸は，その位置で始まる部分文字列の出現頻度を表している．また，灰色の部分は高頻度部分を表している．入力として，ある新聞社のサイトから新聞記事 50 ファイルを収集し，頻度を調べた．そのうちの 1 文書では，約 700 文字目から約 1500 文字目の間がコンテンツ部分であった．

部分文字列の長さ n を大きく設定した時，コンテンツ部分の部分文字列の頻度が下がり，テンプレート部分と比べ小さくなっている．この時，テンプレート部分と高頻度部分は対応している．しかし，n が小さい時は，コンテンツ部分に現れる部分文字列のうちで，頻度が高くなっているものが数多く存在することが分かる．よって，テンプレート部分のみではなく，コンテンツ部分にも高頻度部分が多く現れる為，共通部分の特定に失敗している．

今，高頻度部分と低頻度部分の境界の数に注目する．この境界の総数を**交代数**と呼ぶ．交代数とは，文字列 x とそれに対する部分文字列の集合が与えられたときに，与えられた全ての部分文字列が x 上で出現する領域とそうでない領域とが変化する回数である．ただし，部分文字列が x 上で繋がっている場合は，繋がった領域を 1 つの領域と考える．例えば，$x = ac\underline{cba}a\underline{cb}c$ ，$W = \{cb, ba\}$ とした場合，下線部が x 上で W が出現する部分であり，この時の交代数は 4 である．



(a) 部分文字列の長さ 5

(b) 部分文字列の長さ 2

図 3 位置による部分文字列の出現頻度

n を大きく設定し，テンプレート部分が高頻度部分と対応している時は交代数は小さくなっている．一方，n を小さく設定し，共通部分の特定に失敗している時は交代数は大きくなっている．

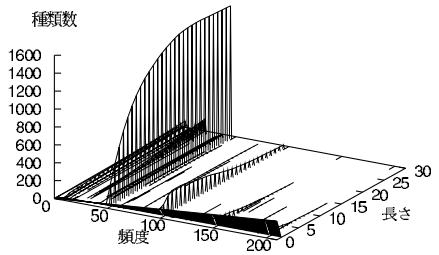
図 4 は入力をテンプレート部分 (a) とコンテンツ部分 (b) に分けて，部分文字列の頻度分布を調べたものである．横軸が部分文字列の頻度，縦軸が長さ，垂直軸がその頻度を持つ部分文字列の種類数を表している．図 4 (a) より，テンプレート部分は n が小さい時，大きい時，いずれも頻度の大きい部分文字列が存在する．また，図 4 (b) より，コンテンツ部分は n が小さい時は，頻度の大きい部分文字列が存在するが，n を大きくした時，出現する部分文字列の頻度は小さくなっている．

このことから，部分文字列の長さ n が大きい時は，高頻度な部分文字列はテンプレート部分のみに現れる．この時，より多くの部分文字列を与える，つまり割合 a を大きくすれば，複数の部分文字列の現れる領域が重なることにより，テンプレート部分が高頻度部分として覆われ，交代数が小さくなる．以上より，交代数が十分に小さくなった時，部分文字列の長さ n と割合 a は十分大きいと判断する．

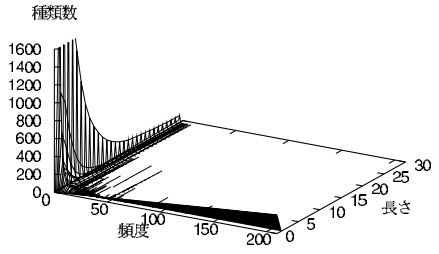
```
<!————★★ここから入れ替えてね ‥ ————->
<font color="#8b0000"> ■ </font><b> 中国　韓国国会議員にビザ拒否</b></font><hr>
<b> 朝鮮族への恩典に反発<BR></b>
<p>
<BLOCKQUOTE>
　【ソウル９日＝黒田勝弘】中国居住の朝鮮族に関する調査のため中国を訪問しようとした韓国の国会議員四人が中国当局から入国ビザを拒否され問題になっている。背景には韓国側で「在外同胞法」を改正して在中国の朝鮮族に恩典国を与えようという動きが出ていることに対する中国側の反発がある。(略) 違憲論議にまで発展してい た。<p>
　中国の反発については「中国自身が外国籍の在外華僑に対し優遇措置を取っていながら、韓国が血筋を同じくする同胞を優遇しようというのに対して非難するのはおかしい」との指摘もあ る。<p>
</BLOCKQUOTE>
<!————★★記事はここまでよ ‥ ————->

</td></tr></table></center>
<!———フッタ情報開始———>
<CENTER><a href=" internat.htm"><img src="../../../cut/left.gif" border=0 vspace=15 width="31" height="38"></a>
<br>
```

図 5　高頻度部分と低頻度部分への分割の例



(a) テンプレート部分



(b) コンテンツ部分

図 4　部分文字列の頻度と長さと種類数のグラフ

部分文字列の長さ n を更に大きくすると，今度はテンプレート部分の部分文字列の頻度が下がってしまい，頻度が高かった部分文字列の現れていた部分が低頻度部分になる．この時，長さ n が小さい時と同様に交代数が大きくなる．図 4 (a) において，頻度 100 や 150 を持つ部分文字列の種類数は，部分文字列の長さ n を更に大きくすると，少なくなることが分かる．

以上より，入力を共通部分と非共通部分に分ける為には，長さ n と割合 a を十分大きくし，交代数が小さくなるときの，n と a を決定する必要がある．

共通部分特定アルゴリズムは，カットポイント (2, 1) を初期状態とし，現在のカットポイント (n, a) における交代数と $(n+1, a)$，$(n, a+1)$ における交代数を比較し，交代数が少ないカットポイントへ遷移していく．そして，$(n+1, a)$，$(n, a+1)$ における交代数が現在のカットポイント (n, a) における交代数より大きくなったとき停止し，このカットポイントを出力する．

共通部分特定アルゴリズムは，入力を記述している自然言語やマークアップ言語に関する知識を用いない．大文字と小文字の区別，全角と半角の区別などについてなど，背景知識は用いずに，与えられたまま処理を行なう．

図 5 は，入力ファイルを高頻度部分と低頻度部分に分けた例であり，下線部が低頻度部分を表す．高頻度部分はテンプレート部分に対応しており，入力における共通部分であった．また，2 行目の “■” は，入力ファイル全てにおいて出現する文字であった．このようにタグ以外の文字でも入力において共通して現れる文字は，高頻度部分となる．また，本文中の “た。”，“る。” も高頻度部分になっている．このような文字は，日本語の文末によく現れる文字である為，高頻度部分となった．

4. ラッパー生成アルゴリズム

ラッパー生成の主要部分は，文書中から各項目の場所を特定することと，それを抽出する為のルールを生成することである．本論文で提案するラッパー生成アルゴリズムは，同種の項目を複数含む半構造化文書で同一サイト上にあるものの集合を入力として受け取る．

ラッパー生成アルゴリズムは入力から各項目を抜きだす為の**ルール**の集合を出力する．ルールは各項目の出現する木構造の**パス**と，パスにより特定されるノードで項目を囲んでいる**左区切文字列**と**右区切文字列**と呼ばれる文字列の組から成り立つ．

本論文で提案するラッパー生成アルゴリズムは共通部分の特定，ルールの抽出，不要なルールの削除の 3 つのステップから成り立つ．

4.1 共通部分の特定

ラッパー生成アルゴリズムは共通部分特定アルゴリズムから出力されたカットポイントを用いて，入力として与えられた半構造化文書を高頻度部分と低頻度部分に分ける．高頻度部分はテンプレート部分，低頻度部分がコンテンツ部分と大まかに重なるということを利用する．ただし，図 5 のように低頻度部分が完全にはコンテンツ部分とは一致していない．よって，次節以降のルールを生成するステップが必要となる．

4.2 ルールの抽出

始めに，入力として与えられた半構造化文書を木構造に展開する．木構造の各ノードは開始タグ，空要素タグ，テキストに対応する．開始タグ，空要素タグの場合はノー

ドにタグ名と属性を付与しテキストの場合は “TEXT” を付与する．例えば，図 1 は図 6 のような木構造に展開される．

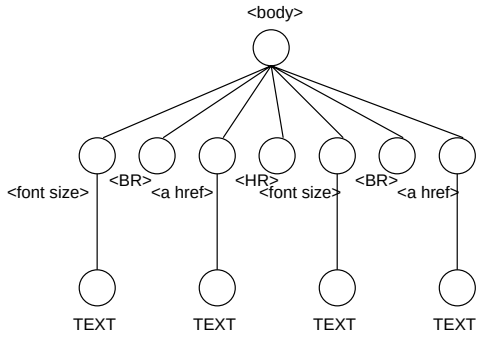


図 6 ラッパー生成アルゴリズムにおいて，図 1 を木構造に展開したもの

まず，文書中で低頻度部分を含むテキストもしくはタグに対応するノードのパスを特定する．この時，対象とするのは，テキストの他に属性を持つ開始タグと空要素タグとする．終了タグや属性の持たない開始タグ，空要素タグは抽出の対象としない．

次に特定したパスの内，同じパスを持つ複数のノードに対応する文字列から，低頻度部分を囲む高頻度部分の共通部分を見つける．低頻度部分の直前に現れる高頻度部分のうち，同じパスを持つ複数のノードに対応する全ての文字列に共通し，長さが一番長いものを左区切文字列とする．また，低頻度部分の直後に現れる高頻度部分のうち，全ての文字列に共通し，長さが一番長いものを右区切文字列とする．ただし，共通部分が見付からないときは，左区切文字列と右区切文字列は “NULL” とし，この時はノードに対応する文字列を全て抽出する．そして，このパスと左・右区切文字列を組み合わせたものをルールとする．

4.3 不要なルールの削除

機械学習による手法は，自動的な手法でないかわりに有用な項目をあらかじめ手動で指定できる．よって，抽出された項目が有用かどうかの判断は不要である．一方，ラッパー生成アルゴリズムによって出力されたルールによって抽出される項目が有用かどうか判断することは難しい．共通部分特定アルゴリズムによって，構造記述部分とコンテンツ部分の分離を行なっているが，そのコンテンツが有用かどうかの判断はしていない．

また，いくつかのコンテンツには，ある項目が含まれない場合もあり得る．例えば，名簿データの場合，何人かはメールアドレスの欄が空欄かもしれない．そこで，一部の入力文書に対して何も抜きださないルールも認めることにした．一方で，入力文書のほんの一部からしか文字列を抜きださないようなルールは不要であると考えた．

そこでルールを用いて入力文書から抽出される文字列の数に着目し，有用である割合を半数と決めた．生成されたルールの集合の内，入力の半数未満の文書から文字列を抽出できないルールは削除し，残ったルールの集合を出力する．

5. 実験と評価

前節で記述したラッパー生成アルゴリズムを実装し実験を行なった．表 1 は，産経新聞[☆]の新聞記事 50 ファイル (日本語) を入力として与えたとき，生成されたラッパーである．“見出し”，“日付”，“本文”，“ジャンル” の 4 つの項目を抽出する為のルールが生成された．

産経新聞における，ジャンルを抜きだす為のルールの左区切文字列は “Sankei-” だった．この項目は，“Sankei-international” や “Sankei-business” の様に “Sankei-” の後にその新聞記事のジャンルが記述されていた．4.1 節の共通部分の特定において，“Sankei-” が高頻度部分に含まれた為，この文字列が左区切文字列として抜きだされた．

washingtonpost.com[☆☆]の新聞記事 74 ファイル (英語) を入力として与えたときの実験では，見出しを抽出するルールの右区切文字列が “ (washingtonpost.com)” であった．washingtonpost.com の見出しは，“Bush Cabinet Meets On California Crisis (washingtonpost.com)” の様に見出しの後に “ (washingtonpost.com)” がついていた．

また，AltaVista[☆☆☆]は検索エンジンであるが，検索結果 50 ファイル (英語) を入力として与えたときの実験では，検索結果の件数を抽出するルールは，左区切文字列が “We found ”，右区切文字列が “ results” であった．検索結果の件数はファイル中で “We found 187,302 results” の様な形式で記述されていた．

このように木構造のパスで指定されるノードから不要な文字列を削除することに成功した．これは，異なるサイト間の統合に必要な処理であり，特に Name Matching 問題において不要な文字列が結果に影響を与えないことが期待できる．

一方，抽出すべき部分が句切文字列に含まれた為，項目全体が抜きだされない場合があった．産経新聞の日付を抽出するルールの左区切文字列は，“2002.01.1” だった．入力として与えたファイルは全て 2002 年 1 月 12 日もしくは 13 日の記事だった．この為，“2002.01.1” までが高頻度部分としてみなされた為に，ラッパー生成アルゴリズムは日にちの 1 の位を抽出するものを生成した．他の入力データにおいて，URL を抽出するルールを生

☆ http://www.sankei.co.jp/main.htm
☆☆ http://www.washingtonpost.com/
☆☆☆ http://www.altavista.com/

表 1 産経新聞のラッパー

項目名	パス	
	左区切文字列	右区切文字列
本文	<html> <body bgcolor> <table width> <tr> <td> <blockquote> TEXT	
	NULL	NULL
日付	<html> <body bgcolor> <i> <b> TEXT	
	2002.01.1	NULL
見出し	<html> <body bgcolor> <table width> <tr> <td> <b> TEXT	
	NULL	NULL
ジャンル	<html> <head> <title> TEXT	
	Sankei−	NULL

成する際にも，“http://www” のような文字列は多くの URL において共通するため抽出に失敗した．このように項目全体を抽出したい場合でも，項目の前後が共通部分に含まれる場合は，ルールの生成に失敗した．

句切文字列の特定に失敗した他の例として，ノードに複数の項目が含まれる場合がある．Citeseer☆の論文のリストのページでは，“Developing a Knowledge Network of URLs - Ikeda, Taguchi, Hirokawa (1999)” のように，同じノード内に論文名とその論文を発表した年の 2 つの項目が含まれていた．Vine Linux☆☆のセキュリティ情報のページでは，“[2003,07,26] LPRng にセキュリティホール” のように，日付とセキュリティ情報の種類の 2 つの項目が含まれていた．これらは，項目間の不要な文字列が低頻度部分に含まれたため，項目を挟む共通部分を見つけることができず，2 つの項目を 1 つの項目として抽出するルールが生成されたものである．

産経新聞の本文部分を抽出するルールのパスは，“<html> <body bgcolor> <table width> <tr> <td> <blockquote> TEXT” であった．図 7 は入力の本文部分のソースであるが，本文の段落間に “<p>” が挟まれている．このため本文部分が段落ごとに抜きだされた． このように，同じ項目の中にタグが挟まれる項

```
<blockquote>
段落 1<p>
段落 2<p>
段落 3<p>
</blockquote>
```

図 7 産経新聞の本文部分

目全体を抜き出すことができない場合があった．これは，パスを用いて抽出する Tree ラッパーについての一般的な問題である．この例のように，本文全体を抜き出したいのか，段落毎に抜き出したいのかはその情報を使うユーザ次第である為，どちらかに統一することが難しい．

☆ http://citeseer.nj.nec.com/cs/
☆☆ http://vinelinux.org/

6. おわりに

本論文では，同種の項目を多数含む半構造化文書群から，各項目を抽出する PLR ラッパーの自動生成法を提案した．PLR ラッパーは，まず対象文書を木構造として捉え，コンテンツ部分を含むノードに対するパスとして抽出部分を特定する Tree ラッパーを構成する．次に，そのパスによって特定されるノードで前後に共通する左・右句切文字列の組として LR ラッパーを生成し，抽出ルールを表現する．

この詳細な表現により，ノードに不要な文字列が含まれる場合でもそれらを分離して抽出することができる．これは，異なるサイトのコンテンツ統合のための Name Matching におけるノイズの除去に有効と考えられる．

機械学習のための訓練例では，不要な文字列は人手により予め削除されている．あるいは，その支援を行なうための GUI が提案されている．本論文において提案したラッパー生成アルゴリズムは，部分文字列の長さと出現頻度に基づく構造記述部分とコンテンツ記述部分に分離，パターンとコンテンツの境界の特定の 2 つの処理により不要な文字列の削除を自動で行なう．

境界特定の制度を向上すること，並びに単一ノード中に複数の項目が含まれる場合の処理が今後の課題である．

参 考 文 献

1) N. Ashish and C. Knoblock, Wrapper Generation for Semi-structured Internet Sources, Proc. of *Workshop on Management of Semistructured Data*, 1997.
2) R.Baumgartner, S.Flesca and G.Gottlob, Visual Web Information Extraction with Lixto, Proc. of *the 27th International Conference on Very Large Data Bases*, The VLDB Journal 2001, pp.119–128, 2001.
3) C.-H. Chang and S.-C. Lui, IEPAD: Information Extraction Based on Pattern Discovery, Proc. of *the 10th International Conference of World Wide Web*, pp. 4–15, 2001.
4) V. Crescenzi, G. Mecca and P. Merialdo, Road Runner:Towards Automatic Data Extraction from Large Web Sites, Proc. of *the 27th International*

Conference on Very Large Data Bases, 2001.
5) D. W. Embley, Y. Jiang and Y. -K. Ng, Record-Boundary Discovery in Web Documents, Proc. of *ACM SIGMOD Conference*, pp. 467–478, 1999.
6) S. Hirokawa, E. Itoh and T. Miyahara, Semi-Automatic Construction of Metadata from A Series of Web Documents, Proc. *16th Australian Joint Conference on Artificial Intelligence*, 2003. (to appear)
7) D. Ikeda, Y. Yamada and S. Hirokawa, Eliminating Useless Parts in Semi-structured Documents using Alternation Counts, Proc. of *the 4th International Conference on Discovery Science*, Lecture Notes in Artificial Intelligence, Springer-Verlag, Vol. 2226, pp. 113–127, 2001.
8) D. Ikeda, Instance Based Table Integration Algorithm for Multilingual Tables on the Web, Department of Informatics Technical Reports, 2003.
9) N. Kushmerick, D. S. Weld and R. B. Doorenbos, Wrapper Induction for Information Extraction, Proc. of *the 15th International Joint Conference on Artificial Intelligence*, pp. 729–737, 1997.
10) K. Lerman, C. A. Knoblock and S Minton, Automatic Data Extraction from Lists and Tables in Web Sources, Proc. of *Workshop on Adaptive Text Extraction and Mining*, 2001.
11) S. N. Minton, S. I. Ticrea and J. Beach, Trainability: Developing a responsive learning system, Proc. of *the 18th International Conference on Artificial Intelligence 2003 Workshop on Information Integration on the Web*, pp. 27–32, 2003.
12) 村上義継, 谷口力昭, 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, Vol. 42, No. SIG14-006, pp. 39–49, 2001.
13) 梅原雅之, 岩沼宏治, 永井宏和, 事例に基づく HTML 文書から XML 文書への半自動変換, 人工知能学会論文誌, Vol. 16, No. 5, pp. 408–416, 2001.
14) Y. Yamada, D. Ikeda and S. Hirokawa, SCOOP: A Record Extractor without Knowledge on Input, Proc. of *the 4th International Conference on Discovery Science*, Lecture Notes in Artificial Intelligence, Springer-Verlag, Vol. 2226, pp. 482–487, 2001.
15) Y. Yamada, D. Ikeda and S. Hirokawa, Automatic Wrapper Generation for Multilingual Web Resources, Proc. of *the 5th International Conference on Discovery Science*, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2534, pp. 332–339, 2002.