Detecting Common Parts and Wrapper Generation for Multilingual Web Documents Using Alternation Counts(Data Mining/Data Warehousing)

YAMADA YASUHIRO
Department of Informatics, Kyushu University

IKEDA DAISUKE
Kyushu University Library

HIROKAWA SACHIO
Computing and Communications Center, Kyushu University

出版情報：IPSJ Journal. 45 (9), pp.2138-2145, 2004-09-15. Information Processing Society of Japan (IPSJ)
バージョン：

交代数を用いた多言語Webテキストからの共通部分特定とラッパーの生成法

山田泰寛[†] 池田大輔[††] 廣川佐千男[†††]

本稿では，同種の項目を多数含むWebページから各項目を抽出するラッパーを自動生成するアルゴリズムを提案する．提案手法では，まず部分文字列の出現頻度に着目し，交代数という指標を用いてテンプレート部分とコンテンツ部分を識別する．部分文字列の長さ n と，出現頻度の割合 a%に対する交代数とは，長さ n の部分文字列で頻度が上位 a%以内に含まれるものが連続して出現する領域とそうでない領域の境界の総数である．提案手法では交代数が極小となる (n, a) を求め，高頻度な部分文字列の出現する領域をテンプレート部分とする．次に，テンプレートの先頭あるいは末尾の文字が “>”，“<”，改行，タブ，空白のような特徴的な文字となっていることを用い，各項目を囲む文字列の組を特定する．この文字列の組からラッパーを生成する．提案手法は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．実験では，4種類の自然言語，2種類のマークアップ言語によるページ群について評価を行い，高い再現率を示すことを確認した．

Detecting Common Parts and Wrapper Generation for Multilingual Web Documents Using Alternation Counts

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

We propose an algorithm to generate a *wrapper* which extracts contents of Web pages written with the same template. First, the algorithm separates each page into template and contents parts using an *alternation count*. An alternation count with respect to $(n, a\%)$ is the sum of boundaries between frequent parts and non-frequent parts, where n is the length of a substring and a is the frequency of the substring. The algorithm searches for a local minimal (n, a) of the alternation count then specifies the template parts as ones on which frequent substrings appear. Next, the algorithm determines the strings which enclose the contents assuming that the last character and the first character is one of the symbols in “>”, “<”, new line, tab and space. The algorithm does not use any preprocessing depending on mark-up and natural languages and knowledge for each site. Experiments show this algorithm works well for inputs written in four natural languages and markuped with HTML and XML.

1. はじめに

WWW上には膨大な量のテキストデータが存在しており，かつ日々増加し続けている．これまで，膨大な量のHTMLやXMLなどの半構造化文書を対象とし，そこから有用なルールや知識を抽出するための様々なWebマイニングの手法が提案されている．

Webマイニングの研究の1つとして，Webページのコンテンツを再処理し，新たな情報としてユーザに提供する研究が行われている．情報の再構成のためには，Webページからある特定の箇所を自動的に抽出するための情報抽出技術が必要となり，そのためのプログラムは**ラッパー**と呼ばれる．情報抽出技術は，Webページの要約作成やWeb上のデータを用いた百科事典の生成などに応用できる．また，異なるサイト間の情報統合を行い，同種の情報の比較を容易に行うことを可能にする．

従来提案されているラッパー生成法[1),3)~7)]は検索エンジンの検索結果や新聞記事など，同一のテンプレートを用いて記述され，同種の項目が繰り返し現れているページを対象としている．情報抽出における抽出箇所の粒度は提供するサービスによって異なり，単語，文，段落など自然言語単位のものもあれば，同じ種類の項目（フィールド）やそれらを1つにまとめた単位

† 九州大学大学院システム情報科学府
Department of Informatics, Kyushu University
†† 九州大学附属図書館
Kyushu University Library
††† 九州大学情報基盤センター
Computing and Communications Center, Kyushu University

（レコード）の場合もある．たとえば，典型的な名簿が記述された Web ページにおいてはフィールドとして名前や電話番号，メールアドレスなどがあり，これらを 1 人分まとめたものがレコードになる．従来提案されているラッパーは，同じ種類の項目を抽出するものが多い．

WWW 上には膨大な量の半構造化文書が存在しており，サイトが違えば同種の項目を持つページであっても，テンプレートが異なっている．よって，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り，間違う可能性も高い．このため，自動的な生成法が望ましい．また，WWW 上のテキストは様々な自然言語で記述されており，現在は HTML による文書化が主流であるが，今後普及するであろう XML を考えると，多言語に対応できる生成法が求められる．

Kushmerick ら[3] は機械学習を用い，訓練例を入力として与え，ラッパー生成を行った．ラッパーの表現形式として，**LR（Left-Right）ラッパー**と呼ばれるラッパーを提案し，それらを生成するためのアルゴリズムを与えた．LR ラッパーは，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を用いて抽出を行う．機械学習によるラッパー生成には，入力として切り出すコンテンツの位置を示す訓練例が必要となる．このとき，訓練例を手動で作成しなくてはならないため，自動的な生成法とはいえない．

本稿で提案するラッパー生成アルゴリズムは，手動の操作や機械学習における訓練例を必要としない自動的な手法である．また，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．

人手のかからない自動的なラッパー生成のために，本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[2] を用いて，同種の項目を多数含む半構造化文書群から共通部分を特定する．共通部分特定は，機械学習における訓練例の作成に相当し，非共通部分がほぼ抽出したい項目の部分である．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．このとき，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．

ラッパー生成アルゴリズムの有効性を評価するために，HTML と XML の 2 種類のマークアップ言語に対して実験を行った．その結果，新聞記事の見出し，日付，本文など人が見て 1 項目としてまとまって認識されるものをほぼ確実に抽出できることが実証された．

2. 交代数を用いた共通部分の特定

本章では，半構造化文書の集合から共通部分を求めるためのアルゴリズム `FindOptimal` について述べる．

2.1 レンジ文字列と交代数

記号の有限集合をアルファベットといい，Σ と記述する．Σ の要素の列 $w = a_1a_2\cdots a_n$ $(a_i \in \Sigma)$ を Σ 上の文字列と呼び，Σ 上の文字列の集合を Σ^* で表す．文字列 $w = a_1a_2\cdots a_n \in \Sigma^*$ $(a_i \in \Sigma)$ に対し，n をこの文字列の長さといい $|w|$ で表し，i $(1 \le i \le |w|)$ 番目の文字を $w[i]$ と表す．また，文字列 w に対し，文字列 $a_ia_{i+1}\cdots a_j$ $(1 \le i \le j \le n)$ を，w の部分文字列と呼ぶ．文字列 $w_1 = a_1a_2\cdots a_n$ と $w_2 = b_1b_2\cdots b_m$ に対し，$a_i = b_i$ $(1 \le i \le n = m)$ が成立するときにこれらの文字列は等しいといい $w_1 = w_2$ と書く．x を文字列 $w = a_1a_2\cdots a_n$ に対するある部分文字列とする．このとき，x の w における出現とは，$a_ia_{i+1}\cdots a_{i+m} = x$ $(1 \le i \le m \le n)$ となるような正の整数 i の集合である．

x を文字列とし，$W = \{v_1, \ldots, v_n\}$ を x の部分文字列の集合とする．x 上の W による**レンジ文字列**（$r_x(W)$ と表記する）とは，以下の条件を満たす長さ $|x|$ の $\Sigma = \{0, 1\}$ の上の文字列である：x における部分文字列 v_k $(1 \le k \le n)$ のある出現 i に対し，$i \le j \le i + |v_k| - 1$ を満たす j 番目の位置は 0 $(r_x(W)[j] = 0)$，それ以外の j に対しては $r_x(W)[j] = 1$ である．たとえば，$x = ac\underline{cba}a\underline{cb}c$，$W = \{cb, ba\}$ とした場合，下線部が cb，ba の現れる部分であるためレンジ文字列は $r_x(W) = 110001001$ である．

次に，文字列に対する統計量である**交代数**をレンジ文字列を用いて以下のように定義する．

定義.1（交代数）： x を文字列とし，$W = \{v_1, \ldots, v_n\}$ を x の部分文字列の集合とする．このとき，交代数とはレンジ文字列 $r_x(W)$ において，$r_x(W)[i] \neq r_x(W)[i+1]$ $(1 \le i \le |x| - 1)$ となる位置 i の総数である．

上記の例において，交代数は 4 である．

2.2 共通部分特定アルゴリズム `FindOptimal`

`FindOptimal` 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**と低頻度部分に分ける．このとき，高頻度部分が共通部分つまりテンプレート部分に対応し，低頻度部分が非共通部分つまりコンテンツ部分に対応してい

ると仮定し，共通部分を特定する．

FindOptimal 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．$X = \{x_1, \ldots, x_n\}$ を文字列の集合とする．X の各文字列における長さ n の部分文字列のうち，頻度の上位 a%に含まれる部分文字列が $x_1, \ldots, x_n$ 上で現れる領域を**高頻度部分**と呼ぶ．高頻度部分をレンジ文字列を用いて以下のように定義する．

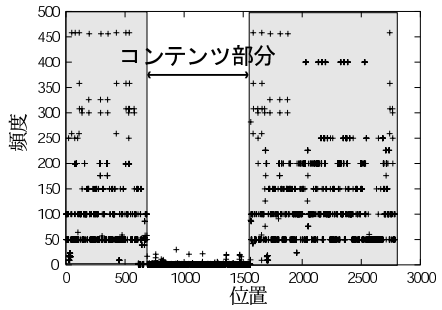
定義.2（高頻度部分）： $X = \{x_1, \ldots, x_n\}$ を文字列の集合とする．X の各要素における長さ n のすべての部分文字列のうち，頻度の上位 a%に含まれる部分文字列の集合を W とする．このとき，$x_1, \ldots, x_n$ 上の W によるレンジ文字列 $r_{x_1}(W), r_{x_2}(W), \ldots, r_{x_n}(W)$ において，$r_{x_i}(W)[j] = 0$ $(1 \leq i \leq n, 1 \leq j \leq |x_i|)$ となる位置 j の集合が文字列 x_i の高頻度部分である．

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．FindOptimal 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．

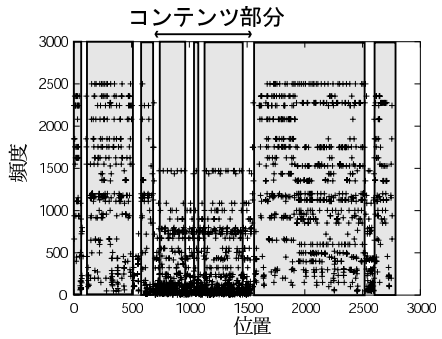
図 1 は部分文字列の長さを長く設定したとき (a)，および短く設定したとき (b) の文書中のある位置から始まる長さ n の部分文字列の頻度である．縦軸はその位置で始まる部分文字列の出現頻度を表している．また，灰色の部分は高頻度部分を表している．入力として，ある新聞社のサイトから新聞記事を 50 ファイル収集し，頻度を調べた．そのうちの 1 文書では，約 700 文字目から約 1,500 文字目の間がコンテンツ部分であった．

n を大きく設定したとき，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．このとき，テンプレート部分と高頻度部分は対応し，交代数が小さくなる．しかし，n が小さいときは，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ため共通部分の特定に失敗する．このとき，交代数は大きくなる．

図 2 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 2 (a) より，テンプレート部分は長さが小さいとき，大きいとき，いずれも頻度の大きい部分文字列が存在する．また，図 2 (b) より，コンテンツ部分は

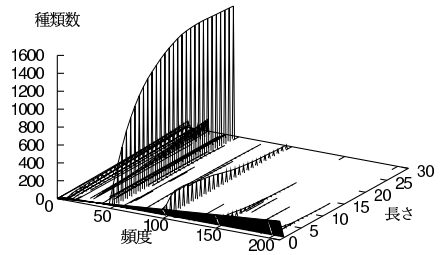


(a) 部分文字列の長さ 5

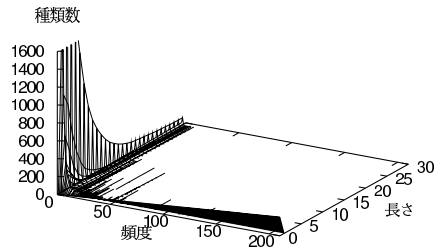


(b) 部分文字列の長さ 2

図 1 位置による部分文字列の出現頻度
Fig. 1 Frequencies of substrings on each position.



(a) テンプレート部分



(b) コンテンツ部分

図 2 部分文字列の頻度と長さと種類数のグラフ
Fig. 2 The graph for frequencies, lengths and varieties of substrings.

部分文字列の長さが小さいときは，頻度の大きい部分文字列が存在するが，部分文字列の長さを大きくしたとき，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 が大きいときは，高頻度な部分文字列はテンプレート部分のみに現れる．このとき，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とき，部分文字列の長さ n と割合 a は十分大きいと判断する．

部分文字列の長さ n をさらに大きくすると，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．このとき，長さ n が小さいときと同様に交代数が大きくなる．図 2 (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 をさらに大きくすると，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ためには，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．

`FindOptimal` は，カットポイントが (2, 1) を初期状態とし，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 (n,a) における交代数より大きくなったとき停止し，この (n,a) を出力する．

`FindOptimal` は $O(nN\log N + n^2N)$ 時間で動作する．ただし，n は出力されたカットポイントにおける部分文字列の長さであり，N は入力として与えられた文字列の合計の長さである．実験では，n は N に比べ十分小さいため，$O(N\log N)$ で近似できる．

`FindOptimal` 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．初期の `FindOptimal` [2] では，入力として与えられた文字列に対して，あらかじめ，(1) 改行，タブはスペースとして扱う，(2) 連続するスペースは，1 つスペースとして扱うという前処理を行っていた．しかし，本稿においては，そのような前処理を行わず，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う．

図 3 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．高頻度部分は構造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 “■” は入力として与えられた各文書のテンプレート部分に含まれる文字であった．本アルゴリズムは，タグ以外の文字であっても，入力文書のテンプレート上に出現するものを，高頻度部分として特定できるという特徴がある．

```
<!————★★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>
```

図 3 高頻度部分と低頻度部分への分割の例
Fig. 3 Division into frequent and non-frequent parts.

また，文末に頻出する “た。” や “る。” は，コンテンツ部分に含まれるため，低頻度部分に含まれるのが理想である．しかし，日本語の文では，このような部分文字列は頻度が大きいため，高頻度部分に含まれてしまうという問題点がある．

3. ラッパー生成アルゴリズム

本稿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．このような文書は同一のテンプレートをもとに記述されていると考えられる．

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ための**ルール**の集合を出力する．ルールとは各項目を囲んでいる**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．このようなラッパーは **LR ラッパー**[3] と呼ばれる．

ラッパー生成アルゴリズムはルールを生成する際に，`FindOptimal` によって特定された共通部分と非共通部分の境界に注目する．半構造化文書はタグによりコン

テンツを構造化している．また，空白文字も文書の構造の一部として見ることができる．よって，項目はタグもしくは空白文字に囲まれていると仮定し，直前を $E_l = \{>$, 改行，タブ，スペース $\}$，直後を $E_r = \{<$, 改行，タブ，スペース $\}$ という特別な文字に囲まれているとする．E_l と E_r を**境界文字**と呼ぶ．

本稿で提案するラッパー生成アルゴリズムはコンテンツ部分特定，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．

3.1 コンテンツ部分特定

FindOptimal を用いて，入力として与えられた半構造化文書を共通部分と非共通部分に分ける．

3.2 ルールの抽出

前節によって，抽出する項目にあたる部分がすべて非共通部分となれば，その部分を抽出することにより，ラッパー生成を行うことができる．しかし，図 3 において，文末に頻出する “た。” や “る。” のような部分文字列は高頻度部分となるため，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．このような問題点を解決するために，ルールの作成が必要となる．

まずラッパー生成アルゴリズムは，各非共通部分を囲んでいる直前の E_l の要素で終わり，タグもしくは空白文字を 1 つ含む文字列を左区切文字列の最初の候補（l と表記）とする．その非共通部分の直後の E_r の要素で始まり，タグもしくは空白文字を 1 つ含む文字列を右区切文字列の最初の候補（r と表記）とする．

もし，2 つの区切文字列の候補 l，r がその文書中で異なる頻度を持つならば，頻度の大きい区切文字列の長さを伸ばす．つまり，もし l (r) が r (l) より頻度が大きいならば，l (r) の長さを直前（直後）のタグもしくは空白文字が含まれるまで伸ばす．これを l と r が同頻度になるまで繰り返し，そのときの l と r をルールとして出力する．

ただし，l と r が同じ文字列もしくはタグとして対応している場合は（たとえば，l =<tagA><tagB>, r =< /tagB>< /tagA>），その文字列中で 2 つが同頻度であっても，両方の長さを増やす．このような文字列は文書中で頻度が同じになる確率が高く，句切文字列としては適切ではないと考えたためである．

3.3 不要なルールの削除

抽出する項目は提供するサービスによって異なる．このため，どの項目を抽出するかを自動で選択することは難しい問題である．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，前節で出力されたルールに関する知識は，左区切文字列 と右区切文字列の頻度が同じということだけであり，その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．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うる．そこで，一部の入力文書に対して何も抜き出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出さないようなルールは不要であると考えた．

ある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，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た．

4. 実　　験

入力として扱う半構造化文書は 15 サイトから 1410 ファイルを収集した（**表 1**）．これらの文書は 4 種類の自然言語（日本語，英語，中国語，ドイツ語）で記述され，HTML もしくは XML の 2 種類のマークアップ言語で構造化されている．

4.1 評 価 手 法

情報抽出における抽出箇所の粒度は提供するサービスによって異なる．実験では，従来提案されているラッパーと同様に，たとえば新聞記事における，見出しや本文のような同種の項目を抽出することを目的とした．そこで，いくつかの入力ファイルをブラウザ上で実際に見ることで，抽出すべき項目を決めた．**表 2** の項目名の列がこれにあたる．このとき，項目名は手動で付けた．

FindOptimal を評価するために，再現率，精度，適合率を計算した．入力文字列に対して正解のレンジ文字列 c を手動で作成し〔コンテンツ上の文字を 1，テンプレート（共通部分）を 0〕，FindOptimal によって出力されたカットポイントを用いて作成されたレンジ文字列 r と比較を行った．再現率は，c における共通部分がどのくらい r によって共通部分と見なされたかを意味する．精度は，r における共通部分が実際にどのくらい共通部分だったかを意味する．適合率は，c におけるコンテンツ部分が r におけるコンテンツ部分と，また c における共通部分が r における共通部分とどのくらい一致したかを意味する．

次に，ラッパー生成アルゴリズムを評価するために，手動でラッパーを生成した．そして，各項目に対し各入力ファイルから手動ラッパーによって抽出されたものと，自動生成されたラッパーによって抽出されたものを比較し，一致した割合を求め，それをその項目に対する再現率とした．そして，再現率が高いとき，

表 1 実験で用いたサイト
Table 1 All sites used in experiments.

ID	URL	言語	#	タイプ
altavista	http://www.altavista.com/	英語	17	検索エンジン
freebsd	http://docs.freebsd.org/mail/	英語	49	メールアーカイブ
ftd	http://www.ftd.de/	ドイツ語	101	ニュース
kyushu-u(XML)	非公開	日本語	50	データベース
lycos	http://www.lycos.com/	英語	50	検索エンジン
mail	http://www.mail-archive.com/	英語	50	メールアーカイブ
mainichi(XML)	http://www.mainichi.co.jp/digital/newsml/	日本語	470	ニュース
peopledaily	http://www.peopledaily.co.jp/	中国語	127	ニュース
redhat	http://www.redhat.com/mailing-lists/	英語	50	メールアーカイブ
reuters	http://www.reuters.de/	ドイツ語	50	ニュース
sankei	http://www.sankei.co.jp/main.htm	日本語	108	ニュース
sigmod(XML)	http://www.acm.org/sigmod/record/xml/	英語	50	データベース
toho-jp	http://www.toho-jp.com/	中国語	144	ニュース
yahoo	http://www.yahoo.com/	英語	45	検索エンジン
yomiuri(E)	http://www.yomiuri.co.jp/index-e.htm	英語	49	ニュース

表 2 `FindOptimal` とラッパー生成アルゴリズムの実験結果
Table 2 Results for `FindOptimal` and wrapper generation algorithm.

ID	(n, a)	項目名（ラッパーの再現率）	再現率	精度	適合率
altavista	(21, 15)	タイトル (1.00), 説明 (1.00), URL (1.00)	0.603	0.911	0.532
freebsd	(22, 4)	日付 (1.00), From (1.00), To (0.939), タイトル (1.00), ID (1.00), 本文 (1.00)	0.974	0.737	0.587
ftd	(36, 19)	タイトル (0.00), 見出し (1.00), 記事の要約 (1.00), 本文 (1.00)	0.910	0.748	0.253
kyushu-u(XML)	(20, 1)	ID (1.00), 苗字 (1.00), 名前 (1.00), 所属 (0.00), 学科部門名 (1.00), 職名 (0.00), メールアドレス (1.00), URL (0.00), 電話番号 (1.00), FAX 番号 (1.00), 取得学位 (1.00), 専門分野 (1.00), 研究教育社会活動 (1.00), 研究のキーワード (1.00), 研究業績 (1.00), 所属学会協会 (1.00), 授業担当科目 (1.00), 社会連携活動 (1.00)	0.994	0.915	0.907
lycos	(30, 10)	タイトル (1.00), 説明 (0.955), URL (1.00)	0.603	0.917	0.814
mail	(15, 16)	subject (0.00), From (0.600), タイトル (0.00), 日付 (0.00), 本文 (0.00)	0.962	0.791	0.470
mainichi(XML)	(11, 4)	見出し (1.00), 日付 (1.00), キーワード (1.00), 本文 (0.00), 関連語 (1.00), 他の新聞記事の見出し (1.00)	0.997	0.949	0.730
peopledaily	(4, 2)	タイトル (1.00), 日付 (1.00), 見出し (0.984), 本文 (0.992)	0.870	0.934	0.635
redhat	(18, 4)	タイトル (1.00), 本文 (0.980)	0.773	0.930	0.900
reuters	(15, 12)	見出し (1.00), 日付 (1.00), 本文 (1.00)	0.765	0.978	0.836
sankei	(11, 2)	見出し (1.00), 2 番目の見出し (1.00), 本文 (1.00)	1.00	0.984	0.949
sigmod(XML)	(4, 1)	出版巻 (0.00), 出版番号 (1.00), 出版月 (1.00), 出版年 (1.00), タイトル (1.00), 開始ページ (1.00), 終了ページ (1.00), 著者 (1.00)	0.557	0.662	0.423
toho-jp	(23, 2)	見出し (1.00), 本文 (1.00)	0.896	0.949	0.965
yahoo	(78, 9)	タイトル (1.00), 説明 (1.00), URL (1.00)	0.307	0.731	0.934
yomiuri(E)	(19, 1)	見出し (1.00), 本文 (1.00)	0.997	0.993	0.993

ラッパー生成が成功していると判断した．ただし，同じルールで異なる項目を抽出する場合はルールの生成が失敗していると考え，再現率は 0%とする．

4.2 評　　価

4.2.1 `FindOptimal` の評価

表 2 の 2 番目の列は `FindOptimal` の出力したカットポイントを表し，“再現率”，“精度”，“適合率” の列が `FindOptimal` の実験結果である．それぞれの平均は，0.832，0.891，0.750 であった．実験結果から，`FindOptimal` が入力を記述している言語に依存せず有効であることが分かる．

ただし，実験の評価において以下のような問題があった．本実験では，入力として与えられたページを実際にブラウザ上で見ることで，抽出すべき項目を手動で決めた．しかし，これらの項目以外で `FindOptimal` が低頻度部分つまりコンテンツ部分と見なした項目があった．たとえば，新聞記事において，手動では，ライターの名前は不要と判断したが，`FindOptimal` はコンテンツ部分として見なした．また，ページのソースを見てみると，ブラウザには表示されない，meta タグやコメントタグの中に日時，キーワード，記事の要約など，有用な情報が記述されていた．`FindOptimal` は，このようなタグもコンテンツ部分と見なし，このことが一部の実験結果に影響した．

4.2.2 ラッパー生成アルゴリズムの評価

表 2 の “項目名（ラッパーの再現率）” の列がラッパー生成アルゴリズムの評価である．それぞれの実験において完璧ではないものの高い再現率で各項目の抽出に成功している．

提案するラッパー生成アルゴリズムでは，タグだけでなく，テキスト部分も区切文字列の一部として抽出することがある．これは，テキスト部分であっても，入力として与えられたファイルすべてにおいて，出現する文字ならば，それらをテンプレート部分として扱うためである．

“sankei” の各ファイルにおいて，サブ見出しの行頭

表 3 “sankei” から生成されたラッパー
Table 3 A wrapper generated from “sankei”.

項目名	ルール (左区切文字列/右区切文字列)
見出し	■
サブ見出し	\n
本文	<BLOCKQUOTE>\n\n <p>\n\n\n

の文字として “■” が出現する．このため，“■” はテンプレート部分に含まれる．**表 3** より，“■” が区切文字列の一部として抽出されていることが分かる．“■” はマルチバイト文字であり，日本語の文章において，何らかのリストの先頭の文字として使われることがある．このように，入力を記述している言語の文字が区切文字列の一部として抽出される場合がある．このことより，テキスト部分が何語で書かれていても問題なくルールの生成を行うことができる．

“sankei” の実験において，“サブ見出し” の抽出に成功している．入力として 108 ファイルを与えたが，各ファイルにおいて，この項目の出現数に違いがあった．しかし，このような場合においても，このアルゴリズムで生成したラッパーは，それらをすべて抜き出すことに成功した．

ラッパー生成アルゴリズムは，meta タグや comment タグの中に含まれた隠れた情報も抽出した．このような情報は他のラッパー生成アルゴリズムでは無視されてきた．“ftd” においては，記事の要約が記述された meta タグを抽出するためのルールを生成した．

“kyushu-u(XML)” は九州大学の教官データベースの XML ファイルである．このデータにおける XML ファイルには，タグの名前に日本語が含まれているが，これらのタグも句切文字列の一部として抽出した．

しかし，いくつかの項目に関して，ルールの生成に失敗する場合があった．たとえば，ある項目の多くは “<abc” という文字列が後に続くが，いくつかは，“<ABC” となっている場合があった．本稿で提案するアルゴリズムは，入力に対して知識をほとんど用いていない．よって，大文字と小文字も異なる文字として扱うため，後者のインスタンスを抽出するためのルールの生成は行えなかった．

HTML におけるタグは，同じタグであればレイアウト上は大文字，小文字いずれも同じレイアウトがなされる．ただし，Web ページ作成者が意図して大文字と小文字を区別して使用し，構造を作っている場合がある．このとき，大文字と小文字の区別をなくしたために，異なる項目が同じ項目として抽出されるという問題が起こる．たとえば，ある項目 A の左区切文字列が “
” で項目 B の左区切文字列が “
” であるとき，これらは異なる項目であるのに，同じ項目として認識して抽出されてしまう．本稿では，Web ページ作成者が意図して大文字と小文字を区別していると仮定し，抽出することが自然だと考えたため，大文字と小文字を区別している．

他の問題点として，区切文字列となるべきタグの属性値が異なるため，項目が抽出できない場合があった．区切文字列を生成する際に属性値を無視するという方法が考えられるが，作成者が属性値も含めて構造を記述しているとも考えられるため，本稿では属性値も含めてルールを抽出している．

抽出精度の向上のために，LR ラッパーの代わりに Tree ラッパーを使用することにより解決できると考える．文献 4) では，LR ラッパーで抽出できなかったものが，Tree ラッパーにできることが報告されている．提案手法は，あらかじめ入力からテンプレート部分とコンテンツ部分を自動的に特定しているため，生成するラッパーの種類の変更は容易に行えると考える．

Tree ラッパーは，木構造のパスを用いることにより，そのパスに対応する文字列を抽出する．しかし，パスに対応する文字列には不必要な文字列がついている場合がある．たとえば，毎日新聞の記事では，あるパスに対応する文字列が “［毎日新聞 3 月 1 日］(2002-03-01-11:06)” のように，日付・時間の周りに不要な文字列がついている．

この問題を解決するために，文献 7) において，Tree ラッパーと LR ラッパーを組み合わせた **PLR ラッパー** を提案している．PLR ラッパーにおけるルールは，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．実験では，パスに対応する文字列から不必要な文字列を削除できることを確認した．

5. まとめと今後の課題

本稿では，交代数を利用した WWW 上の多言語テキストに対する共通部分の特定手法と，これを応用した LR ラッパーの生成法について提案した．これらは，訓練例の作成など手動の操作を必要としない自動的な手法である．また，入力を単なる文字列として扱い，言語に依存した処理を行わないため，多言語テキストに対応できる．

Web ページから情報抽出を行う際に，抽出する項目は提供するサービスによって異なる．たとえば，検索

エンジンの検索結果においては，あるサービスでは，URL やタイトルが必要かもしれない．あるサービスでは，検索結果の件数の総数が必要かもしれない．このため，サービスの要求する項目を自動で選択することは難しい問題である．

本稿では，選択する項目の判断として，項目の出現する回数に着目し，入力文書のほんの一部にしか出現しない項目は不要であると考えた．この判断基準を用いたとしても，余分な項目が抽出されるなど，抽出した項目と要求する項目が完全に一致しない場合が存在する．そこで，半自動的な手法ではあるが，抽出された項目から，GUI を用い項目を選択する方が，機械学習のように例を生成する手法に比べ，労力が小さいと考える．GUI の実装は今後の課題である．

また，自動的に生成されたラッパーによって抽出された項目が，抽出したい項目と一致している場合は，手動の操作なしで生成できる．いずれにしても，手動による生成や機械学習による生成よりも労力は小さくなると考えるため，本稿で提案するラッパー生成法は意味があると考える．

現在，情報抽出の研究では，統一した評価手法やテストベッドが存在していないという問題がある．他のラッパー生成法との比較のために妥当な評価手法の提案やテストベッドの作成も今後の課題である．

参 考 文 献

1) Embley, D.W., Jiang, Y.S. and Ng, Y.-K.: Record-Boundary Discovery in Web Documents, *Proc. ACM SIGMOD Conference*, pp.467–478 (1999).
2) Ikeda, D., Yamada, Y. and Hirokawa, S.: Eliminating Useless Parts in Semi-structured Documents using Alternation Counts, *Proc. 4th International Conference on Discovery Science*, Lecture Notes in Artificial Intelligence, Vol.2226, pp.113–127, Springer-Verlag (2001).
3) Kushmerick, N., Weld, D.S. and Doorenbos, R.B.: Wrapper Induction for Information Extraction, *Intl. Joint Conference on Artificial Intelligence*, pp.729–737 (1997).
4) 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).
5) 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).
6) Yamada, Y., Ikeda, D. and Hirokawa, S.: Automatic Wrapper Generation for Multilingual Web Resources, *Proc. 5th International Conference on Discovery Science*, Lecture Notes in Computer Science, Vol.2534, pp.332–339, Springer-Verlag (2002).
7) 山田泰寛，池田大輔，廣川佐千男：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，第 72 回情報学基礎研究会，情報処理学会研究報告，Vol.2003, No.98, pp.115–122 (2003).

(平成 15 年 6 月 23 日受付)
(平成 16 年 7 月 1 日採録)

山田 泰寛（学生会員）

昭和 53 年生．平成 15 年九州大学大学院システム情報科学府情報工学専攻修士課程修了．現在，九州大学大学院システム情報科学府情報理学専攻博士後期課程在学中．ウェブマイニングの研究に従事．日本学術振興会特別研究員，修士（工学）．

池田 大輔（正会員）

昭和 46 年生．平成 8 年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．平成 9 年九州大学大学院システム情報科学研究科情報理学専攻博士後期課程中退．九州大学大型計算機センター助手，九州大学情報基盤センター講師を経て，現在，九州大学附属図書館助教授．ウェブマイニングの研究に従事．ACM 会員．博士（理学）．

廣川佐千男（正会員）

昭和 29 年生．昭和 54 年九州大学大学院理学研究科修士課程了．静岡大学工学部情報工学科助手，九州大学教養部助教授，九州大学理学部助教授，九州大学大学院システム情報科学研究科教授を経て，現在，九州大学情報基盤センター教授．リンク情報による WWW 空間の解析，半構造化データからのデータマイニング，および型理論の研究に従事．電子情報通信学会，人工知能学会，ソフトウェア科学会，数学会，ASL，EATCS 各会員．博士（理学）．