自然言語処理 154−26
(2003. 3. 7)

繰り返し構造を用いたWebページの構造化に関する研究

南野 朋之† 齋藤 豪‡ 奥村 学‡

概 要

World Wide Webは，サイトの数においても有用な情報の量においても，急速に成長している巨大な情報源である．しかしながらWeb上の情報は，レイアウト記述言語で記述された，人が目で見て理解するための情報であるため，計算機で直接扱うには困難な点がある．そこで本研究では，このようなWeb上の情報を人間が理解する構造に近い形で計算機が扱うことが出来るようにするために，HTML文書中に含まれる要素の繰り返し構造に注目し，自動的な情報のセグメンテーション，構造化を行うことを目的とする．

Structuring Web pages based on repetition of elements

Tomoyuki NANNO† Suguru SAITO‡ Manabu OKUMURA‡

Abstract

The World Wide Web is a vast source of information accessible to computers, but understandable only to humans, because Web pages are described in layout description languages, such as HTML. In this paper, we propose the technique of automatically segmenting and structuring Web pages based on repetition of elements.

1 はじめに

今日の世界において，インターネットは急速に成長しているメディアの一つである．しかも，その成長はなお加速しており，このことはインターネットが未だ拡大の絶頂期に達していないということを示している．その中でも特に，World Wide Web(WWW)の拡大は，誰もが容易に世界中の情報にアクセスすることを可能にした．また同時に，情報発信を行うことが容易になったため，インターネット上には膨大な量の情報が蓄積されている．

しかしながら，そのような膨大なWebページを選択的に閲覧することは容易ではない．そこで，Zero-Click[1]では，WWWを利用して情報を検索する時に不可欠であるリンク先の情報の閲覧作業に注目し，ユーザがリンク先情報を容易に取得できるシステムの構築を行った．このシステムを構築する過程で，ユーザへの情報提示を行う際，よりユーザに適応した形で情報を提示するなど，より知的な情報提示を行う必要性があるとの認識に至った．

このような知的情報提示を行うためには，システムがHTML文書を理解し，自動的にセグメンテーションや構造化を行う必要がある．つまり，一つのWebページ中に含まれる情報のうち，どの部分が情報のひとかたまりで，またそれらがどのような構造で配置されているかなどの情報を計算機で扱えなければならない．

しかしながら，計算機によって，これらのことを理解することは非常に困難である．なぜなら，HTMLによって記述されている情報は，人間がレンダリングされた結果を見て理解するために記述された情報がほとんどであるからである．

そこで，本研究では，「人間はWebページを見るときページのレイアウトから情報の「まとまり」を理解することで，そのページの構造を理解することができる」という点に注目し，HTML文書中の繰り返し構造を発見することで情報のセグメンテーション，構造化を実現する．

計算機によってWebページの自動的なセグメンテーション，構造化が可能になれば，Web上の情報を扱う様々なアプリケーションでの利用が可能となる．例えば，構造の情報を利用した読み上げ順序で情報を提示する音声ブラウザや，表示領域の限られた携帯端末向けにWebページを分割するシステム，また，Webページから情報を抽出するwrapperシステムなどでの利用が考えられる．本稿では，それらの応用のための基礎となる研究について報告する．

2 関連研究

Webページの構造化に関する研究は，Webからの情報抽出やWebマイニングの分野でなされている．

Vijjappuら[2]は，HTMLタグによって与えられる潜在的な情報の利用によって，Webからの情報抽出を行っている．この研究では，例えばニュースサイトの記事では，著者名の前には「by」がつくことが多いという性質や，記事の日付は「mm-dd-yyyy」や「month-dd-year」の形式で現れることが多いといったようなテキストの性質や，記事のタイトルに相対リンクが含まれる，また個々の記事が<p>タグや タグによって

†東京工業大学大学院 総合理工学研究科
Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology
nanno@lr.pi.titech.ac.jp
‡東京工業大学 精密工学研究所
Precision and Intelligence Laboratory, Tokyo Institute of Technology
{suguru,oku}@pi.titech.ac.jp

囲まれるなどといったHTMLタグによって与えられる性質を利用して，Webページ中の複数の記事の抽出を行っている．そして，それらニュースサイトに特徴的なルールを作成することで，他のニュースサイトに対しても同様に情報を抽出できるとしている．

しかしながら，そのようなルールは人手によって記述しなければならず，また，Webの多様性を考慮すると，様々なWebページに共通するルールを作成することは，非常に難しいと考えられる．また，そのような手法が適用できる分野も，ニュースサイトや検索エンジンの出力など，ある程度決まったパターンのあるページに限定されると考えられる．

Cohenら[3]は，Webページから構造化されたデータを抽出するwrapperを学習によって得るための研究を行った．学習に使用している素性は，HTML文書のトークンレベルの性質，DOMレベルの性質，また，テーブルに関しては，レンダリングされた二次元の幾何学的な性質を利用している．素性の抽出には，ビルダーと呼ばれるモジュール化された抽出器を使用するため，新しい素性を追加することも容易なシステムになっている．

しかしながら，学習アルゴリズムには，多くの正解データが必要になると共に，そのようにして得られたモデルは学習データに依存するため，ある種のページに特化したシステムとしては有効ではあるが，汎用的なシステムにするのは困難である．

3 提案手法の概要

本節では，本研究の提案手法について概要を述べる．

3.1 繰り返し構造

本研究では，「人間はWebページを見るときページのレンダリングされた結果から情報の“まとまり”を理解することで，そのページの構造を理解することができる」という観察に基づき，HTML文書中の情報の構造化を行う．

例として，図1に示すWebページを考える．

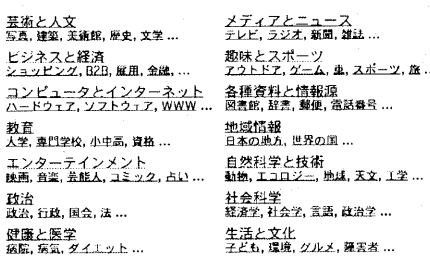


図1: Yahoo!Japanの一部

人間は，このWebページを見ると，以下のような構造を理解すると思われる．

- 「写真」は「建築」「美術館」などと同じレベルの構造
- 「芸術と人文」と「写真」は別のレベルの構造
- 「芸術と人文」は「ビジネスと経済」などと同じレベルの構造

なぜこのような構造を理解できるのかであるが，Webページ作成者は，ページのセグメントを人間が得やすいようにWebページを記述するためであると考えられる．例えば，Webページの作成者は，同じタイプの項目が複数ある情報セグメントを記述する際には，各項目を同じ文字サイズ，同じフォントカラーで表現し，また，それらの情報を並べて配置するであろう．先ほどの例では，「写真」と「建築」は同じフォント属性で記述され，それに対し「芸術と人文」は別のフォント属性で記述されているという特徴が，見る人に対してWebページの構造を理解する際の手がかりになっていると考えられる．

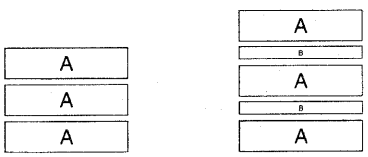
本研究では，このような表現上の特徴を，HTML文書中に含まれる繰り返し構造を検出することによって得られると考える．例えば「写真」「建築」などは，<a>タグや<small>タグによって特徴づけられるテキストの繰り返し構造になっている．

このようなHTML文書中の繰り返し構造を検出すると，情報のセグメントを検出する事もできる．例えば，以下の例を考える．

今日の天気	晴れのち雨
	降水確率 80%
明日の天気	晴れ時々くもり
	降水確率 50%

これを見ると，人間は二行が一つの情報の単位であることを理解することが出来る．しかし，HTML文書中には，このような情報のセグメントに関する記述はされていない．「降水確率」が含まれるセルの前のセルが空白であるという情報を用い，二行単位で繰り返しているという情報を得ることができれば，このような人間の理解するセグメントと同じ構造を得ることが出来る．このように，繰り返し構造を発見し，その繰り返し単位を利用することで，情報のセグメンテーションは可能になると考える．

なお，本研究で考慮する繰り返し構造のタイプは図2に示す二つのタイプである．セパレータありの繰り返し構造は図3のような構造を得る際に使用される



(a) セパレータなし繰り返し構造　(b) セパレータあり繰り返し構造

図2: 繰り返し構造のタイプ

メール - 住所録 - カレンダー - 挨拶状

図3: セパレータありの繰り返し構造の例

3.2 構造化

前節で述べたように，繰り返し構造を検出することで，同じレベルの情報のセグメンテーションが可能に

なる．しかしながら，図1の例では，「芸術と人文」と「地域情報」は同じ繰り返し構造にはならない．なぜなら，「芸術と人文」はサブカテゴリを5つ持っているのに対し，「地域情報」は2つしかサブカテゴリを持っていないからである．

そこで，本研究ではまず，図4に示すようなもっともプリミティブな繰り返し構造を検出する．その後，その繰り返し部分をトークンに置き換える．その際，繰り返しの回数が異なるが，繰り返しの基本単位が同じ繰り返し構造は，同じトークンで置き換える．すなわち，図4中で網掛けした部分は，繰り返しの基本単位が同じため，含まれる要素数が異なっても同一視される．

そのような状態で，再び繰り返し構造を検出することで，図5で網掛けしたようなさらに大域的な構造を検出することが出来るようになる．

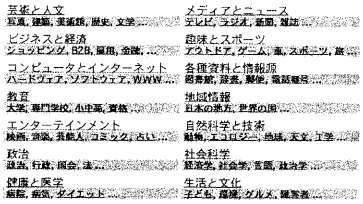


図 4: 繰り返し構造

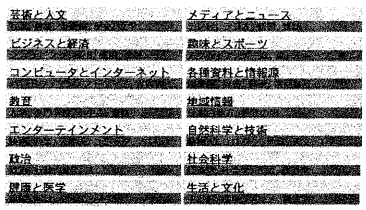


図 5: 構造化

本研究では，このようにもっともプリミティブな構造を検出し，見つかった繰り返し構造をつぎつぎとトークンに置き換え，再帰的に繰り返し構造を発見していくことで，Webページ上の情報をボトムアップに構造化する手法を提案する．

4 システム構成

本節では，前節で述べた提案手法を実現するためのシステムに関して詳細を述べる．

4.1 システム概要

システム全体のフローチャートを図6に示す．

本研究で構築するシステムは，大きく分けて「前処理」，「セグメンテーションと構造化」，「後処理」の三つの部分に分かれる．以下では，それぞれの部分について詳説する．

4.2 前処理

本節では，本研究で構築するシステムの前処理部分について詳説する．

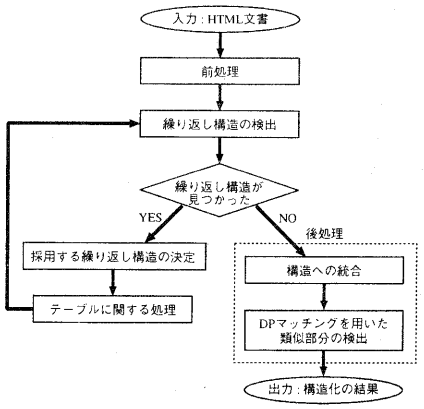


図 6: システム全体のフローチャート

Tidy による HTML 文書の整形

本システムでは，後の処理でHTML文書中のある部分において，「タグのバランスがとれているかどうか」ということを制約として用いる．「タグのバランスがとれている」とは，以下のような条件を満たすことである．

- 終了タグの必要なタグにおいては，開始タグと終了タグの対応が正しくなければならない．
- 開始タグに対し，終了タグの出現順序が正しくなければならない．

そこで，このような条件を満たすために，HTML文書中に含まれる上記のような誤りを自動的に訂正するユーティリティである Tidy[5] を使用する．また Tidy は asxml オプションを使用することで，HTML 文書を well-formed な XML 文書に変換することが出来る．その際，HTML において終了タグの必要ないタグは，空要素として記述される．

フォントの修飾に関するタグ

Tidy による整形を行った HTML 文書に対して，HTML に含まれるタグのうち，フォントの修飾に関するタグは，テキスト属性として付加する．これは，`<b><i>text</i></b>` と `<i><b>text</b></i>` のレンダリング結果がどちらも同じになるためである．これらの要素を同等として扱うことを可能にするために，本研究では `texti=1,b=1` のように置き換える．これは，このテキストがイタリック属性を持ち，かつボールド属性を持っているということを示している．

使用しないタグの除去

HTML に含まれるタグのうち，文書の構造に関係ないタイプのタグおよび部分を除去する．除去する部分を以下に示す．

- `<!-- -->` で囲まれるコメント部分
- `<script> </script>` で囲まれるスクリプト部分
- `<map> </map>` で囲まれるクリッカブルマップに関する部分

また，繰り返し構造を検出する上で問題となる以下のタグを除去する．

- 改行タグ関連

, <nobr>
- フォント修飾タグ関連
  <b>, <i>, <s>, <tt>, <u>, <blink>, <font>

本来構造を検出するためにどのタグを使用するかはシステムを適用するページに応じて決定するべきものではあるが，動的に決定するのは難しいため固定のタグセットでの汎用的なシステムを目指す．

4.3 セグメンテーションと構造化

本節では，本システムの処理本体である，セグメンテーションと構造化を行うモジュールについて詳説する．

4.3.1 繰り返し構造の発見

前節で行った前処理の結果の中から，図2に示した二つのタイプの繰り返し構造を発見する．繰り返し構造の定義は以下のような制約を満たすものである．

- セパレータなし繰り返し構造の場合
  - すべての「A」に含まれる要素が，完全に一致する
  - 「A」の範囲に対して，タグのバランスがとれている
- セパレータあり繰り返し構造の場合
  - すべての「A」に含まれる要素が，完全に一致する
  - すべての「B」に含まれる要素は，タグ以外のテキストをトークンに置き換える以前の要素がそれぞれ等しい
  - 「A」「B」共に，タグのバランスがとれている
  - 「B」の大きさは，「A」の大きさ以下である

以上の制約を満たす繰り返し構造を前処理を適用したWebページ全体から検出する．また本研究では，もっともプリミティブな要素からボトムアップに構造化するという手法を用いているため，「A」「B」の内部に繰り返し構造を持つ構造は，繰り返し構造として検出しない．

4.3.2 採用する繰り返し構造の決定

前節で提案した手法は，HTML文書中のあらゆる繰り返し構造を取得する．しかし図7に示すように，その中にはHTML文書中の出現位置が重複するものが含まれるため，全てを採用することは出来ない．

本節では，見つかった繰り返し構造のうち最も適している組み合わせを発見し，構造化する手法について詳説する．

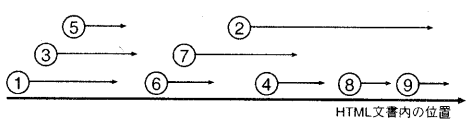


図7: HTML文書中での出現位置

最適解の発見

本研究では，見つかった繰り返し構造の中から，最もプリミティブなものを発見しボトムアップに構造化を行う．よって，発見すべき最適な組み合わせは，「繰り返し回数が最大となる組み合わせ」かつ「どの二つの要素もHTML文書中の出現位置が重ならない組み合わせ」であると考えられる．

このような最適解を発見するために，見つかった繰り返し構造をそれぞれ点とし，出現位置の重なる対を辺で結んだグラフを考える．図8に例を示す．

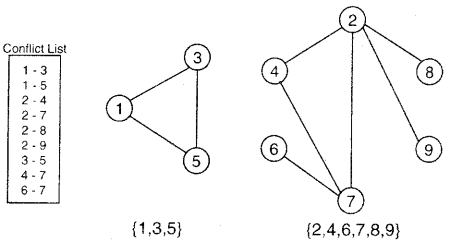


図8: 出現位置が重なるものを辺で結んだグラフ

この例では，グラフが非連結グラフとなっている．このことは，全体の最適解は，各連結成分中の最適解の和であることを示している．なぜなら，繰り返し構造1,3,5のどれもが2,4,6,7,8,9とは出現位置が重ならないからである．このように，グラフ中の連結成分を検出することで，問題を部分問題へ分割する．

次に，各連結成分に対し，出現位置が重ならない2点を辺で結んだグラフを作成する．すなわち，先ほどの各連結成分の補グラフを図9(a)のように考える．求める繰り返し構造の集合は，このグラフ中の極大クリークのうち，最大の繰り返し回数を持つものである．なぜなら，このグラフ中のクリークに含まれる点は，すべてお互いにHTML文書中での出現位置が重ならない繰り返し構造となっており，また，あるクリークの部分グラフとなるクリークは，必ず繰り返し回数の合計が元のクリークに含まれる繰り返し回数の合計以下になるからである．

そこで，図9(b)のように，すべての極大クリークを探索し，その中から，繰り返し回数の和が最大となる繰り返し構造の集合を発見する．

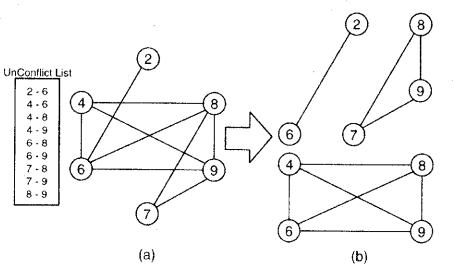


図9: 補グラフの極大クリーク

4.3.3 グループ化

以上のような処理により，見つかった繰り返し構造のうちお互いに出現位置が重ならず，繰り返し回数が

最大となる繰り返し構造の組み合わせが決まる．

次に，採用された繰り返し構造に該当する部分を，単一のトークンによって置き換えることでグループ化を行う．その際，繰り返しの単位が同じ構造の場合には，同一のトークンで置き換える．このようにすることで，次の繰り返し構造を検出する際には，同一の要素として扱われる．

4.3.4 テーブルに関する処理

HTML 文書中に含まれるテーブルには，さまざまな利用方法が存在する．例えば，通常の“表”を記述するために使用される以外に，二段組などを実現するためのレイアウトを目的とするテーブルなども多く存在する．また，縦に読むようなテーブルでは，関連するテキストがHTML 文書中の離れた場所に存在してしまうため，特別な処理が必要となる．

テーブルのタイプ

本研究では，テーブルを，後述する三つのタイプに分類して考える．ここでいうテーブルとは，列の数が同じ行が繰り返している部分を指す．また，一つの<table>タグ中に複数のテーブルが入れ子で存在する場合もある．その場合は，外側のテーブル，内側のテーブルを考え，それぞれ行数が同じ部分が連続する場所全てを考慮する．

以下にそれぞれのタイプと処理を説明する．

- type 1
  <tr>タグ中に含まれる<td>タグで囲まれる要素がそれぞれ繰り返している場合
  ⇒ このタイプのテーブルは，縦に読むテーブルであると考える．縦に読むテーブルの場合，HTML 文書中では，離れた位置にグループ化するべきテキストが配置されるという問題がある．そのため，このタイプのテーブルに対しては，HTML 文書中で，行列の転置を行うことで，横に読むテーブルに変形する．
- type 2
  すべての<td>タグで囲まれる部分が繰り返しており，かつ，各セルに含まれる要素が下位の構造もしくはリンクを含んでいる場合
  ⇒ このタイプのテーブルは，レイアウトのためのテーブルであると考える．よって，これらの行，列には特別意味がないと思われるため，一行にまとめる．
- type 0
  それ以外のテーブルすべて
  ⇒ 上記以外のテーブルについては，テキストの情報を使用しないという本研究の手法では，どのように構造化するべきか，判断ができないと考える．よって，このようなテーブルは，そのままにしておき，構造化を試みることにする．

4.3.5 組み上げ

以上の処理を図 6 で示したフローチャートに従って，新たな繰り返し構造が発見されなくなるまで行う．繰り返し構造の存在する部分が，トークンに置き換えられていくため，さらに大きな繰り返し構造が新たに検出される．

4.4 後処理

前節で述べた処理を繰り返し適用することで，ページ全体の構造化を行うが，繰り返し構造の定義が「完全一致」と厳しいため，繰り返し構造に含まれない要素が存在したり，構造が完全に一つに組みあがらないといった問題がある．そこで，本節では，繰り返し構造に含まれなかった要素に対して，要素間の類似度を計算し，構造に組み込む手法について述べる．次に，これまで厳密な定義であった繰り返し構造の定義に類似度を導入し，さらなる繰り返し構造を検出することで，これまで構造化出来なかった部分の構造の組み上げについて述べる．

このように本研究では，前節で述べた「完全一致」による構造化の後で，「DP マッチングを用いた類似度」による構造化を行う．このような二段階手法を用いる理由は，以下の点を考慮したためである．

- 最初から類似度を用いて構造化を行うと，本来繰り返し構造となるべきでない構造まで繰り返し構造になってしまうおそれがある
- 最初から類似度を用いて構造化を行うと，繰り返し構造となる候補が増大するため，計算量の面で問題が生じる
- 構造化されなかった部分に対して，後処理の段階で類似度を利用した構造化を行うことで，Web ページ全体を構造化するという目的が達成できる

4.4.1 類似度の定義

類似度の計算には，DP マッチング [4] を利用する．利用した漸化式を以下に示す．

$$g(i,j) = \min \begin{cases} g(i-1,j) + d(a_{i-1}, *), \\ g(i-1,j-1) + d(a_{i-1}, b_{j-1}), \\ g(i,j-1) + d(*, b_{j-1}) \end{cases} \quad (1)$$

$*$ は対応する要素がなかったことを示す．

また，上式中のコスト d は以下の式で与える．

$$d(\alpha,\beta) = \begin{cases} 0 & \text{if } \alpha = \beta \\ 1 & \text{if } \alpha \neq \beta, \alpha = * \ or \ \beta = * \\ 3 & \text{if } \alpha \neq \beta, \alpha \neq *, \beta \neq * \end{cases} \quad (2)$$

以下に DP マッチングの例を示す．

```
<tr>    td=2                    <tr>    td=2
<td>                            <td>
<img />                         <img />
</td>                           </td>
<td>                            <td>
(text)  a=1 center=1            *
*                               <img />
</td>                           </td>
</tr>                           </tr>
```

この DP マッチングの結果を利用して類似度 Sim の計算を行う．類似度の計算は，以下の式によって行われる．

$$Sim(a,b) = \frac{(\text{一致数})}{(\text{一致数}) + \max([a_i, *] \text{ の数}, [*, b_j] \text{ の数})} \quad (3)$$

この指標は，直感的には全体の何割が一致しているかを示す．本研究では，この類似度が 0.5 以上である場合に，類似しているとする．

4.4.2 グループへの統合

本節では，本来繰り返し構造に含まれるべき要素が，要素の欠落や付加により繰り返し構造とならなかった場合の処理について述べる．

図 10 の例を考える．

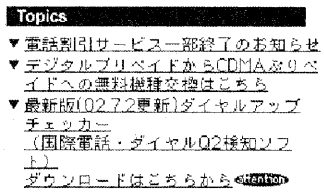


図 10: 繰り返し構造にならない例

この例では，前節までの構造化の処理により，図 10 中の三つの記事がグループ化されている．しかしながら，本来，三つ目の記事の直後の「ダウンロードはこちらから」という部分も，三つ目の記事に含まれる形で構造化されるべきであるが，直後に画像があるため繰り返し構造とならず，グループ化されない．

そこで，あるグループに対して，その前後にある構造を調べて，それらがグループに含まれるべきであるかを判断する．これらの要素が組み入れられる候補は，このグループタグもしくは，このグループタグが下位にもつグループタグであると考えられる．例として，図 11 を考える．

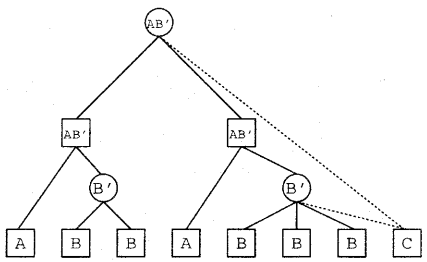


図 11: 組み込むべきグループの位置

この図中の C が繰り返し構造にならなかった要素である．この C が組み込まれる場所は，点線で結ばれた AB' もしくは B' である．つまり，この C が AB' と類似していれば，トップレベルのグループに，B' と類似していれば，その下位グループに統合される．

4.4.3 DP マッチングを利用した類似部分の検出

本節では，これまで厳密な定義であった繰り返し構造の定義に類似度を導入し，さらなる繰り返し構造を検出することで，これまで構造化出来なかった部分の構造の組み上げについて述べる．

基本的な考え方は，これまでの繰り返し構造の発見と同じである．ある部分と，それに連続する部分が共にタグのバランスがとれており，かつそれらが類似している場合に繰り返し構造と見なし，構造化を行う．このような処理を新たな繰り返し構造が見つからなくなるまで，繰り返し行うことで，これまで構造化出来なかった部分の構造化を行う．

5 提案手法の評価

本節では前節で実装したシステムについての評価と考察を行う．なお，システムの出力例の一部を図 12 に示す．

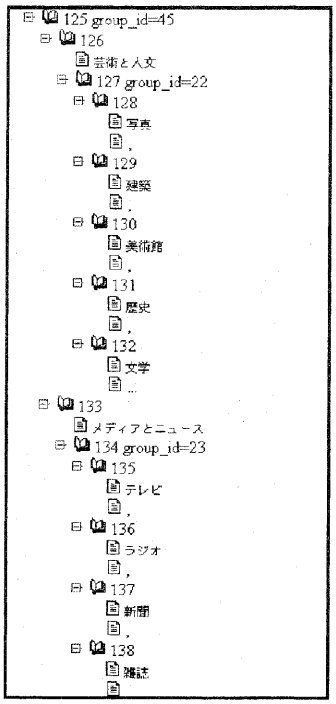


図 12: システムの出力例

5.1 システムの頑健性に関する評価実験

本節では，まずシステムの頑健性に関する実験について説明する．その後，結果を示し，結果について考察を行う．

5.1.1 テストセット

システムの頑健性に関する実験には，以下のカテゴリ毎に 10 ページずつ，合計 80 ページの HTML 文書を使用した．

- ポータルサイト
- 企業のトップページ
- サイトマップの存在するページ
- 機械生成されたページ
- CSS(Cascading Style Sheet) 利用ページ
- 外国語で記述されたページ
- 研究室内の Web ページ
- Yahoo!のランダムリンクから取得したページ

5.1.2 頑健性に関する実験の結果と考察

ランダムに収集した 80 ページに対して，本研究で構築したシステムを適用した．以下にシステムが受理で

きなかったケースについて述べる．

処理のできないページ

80 ページ中，15 の Web ページについてシステムが処理が出来なかった．これらの原因は，以下の二つのエラーによるものであった．

- Tidy のエラー
- 文字コードのコンバートのエラー

処理の終わらないページ

上記のエラーページ以外に，9 の Web ページではシステムによる処理が終わらなかった．「終わらない」とは，24 時間経過した時点で処理が終わらなかったことを指す．

処理の終わらないページは，繰り返し構造の取り方が多いケース，言い換えれば，非常に規則正しく出来ている Web ページに見られる．原因を調べたところ，このようなページでは「採用する繰り返し構造の決定」を行う部分で，非常に時間がかかっていることが判明した．また，同様の処理が，DP マッチングを使用した繰り返し構造発見の部分でも使用されているため，そこでも同様に時間がかかる．

これは，辺の多いグラフから，すべての極大クリークを発見することは，非常に計算時間がかかることに由来する．なぜなら，最大の物を一つ見つけるだけでも NP 問題だからである．

上記以外の処理時間については，10 秒以内に終わるものが全体の 45%，一分以内に終わるものは，全体の 71%であった．

5.2 構造化の結果に関する主観評価実験

前述したテストセットから，システムが受理できないものを差し替えたテストセットを再び用意し，構造化の結果を評価する．

5.2.1 テストセット

文字コードの自動判定に問題がある「外国語」カテゴリ以外のカテゴリについて，各 10 ページの Web ページを再び用意した．よって，使用した Web ページの総数は，7 カテゴリ，70 ページである．

5.2.2 評価方法

システムの評価は，被験者がシステムの出力を実際の Web ページと見比べることで行う．正解データを用意しなかった理由は，人間が人手で Web ページを構造化すると，うまく全体を構造化するのが難しいという理由と，人により正解とする構造に揺れが生じるという理由からである．よって評価は，システムが出力した結果が人間に受け入れられるかどうかに焦点を合わせて行った．

評価は，セグメンテーションと構造化に関して，それぞれ被験者の主観との一致度で評価で行う．各 Web ページ毎に 3 名の被験者から，セグメンテーション，構造化についてそれぞれ五段階評価をしてもらった．また，それぞれ不具合のある箇所についても被験者に指摘をしてもらった．

カテゴリ	Seg	Str
ポータルサイト	4.43	3.40
トップページ	4.63	4.33
サイトマップ	4.87	3.97
機械生成	4.30	3.53
css 利用ページ	4.33	3.83
奥村研究室	4.40	3.13
ランダム	4.20	3.50
全体	4.45	3.67

表 1: カテゴリ毎の評価結果

5.2.3 主観評価実験に関する結果

表 1 に，各被験者の五段階評価をカテゴリ毎に平均した結果を示す．なお，表中の “Seg” は，セグメンテーションに関する五段階評価，“Str” は，構造化に関する五段階評価を示す．数値が大きいほど，セグメンテーション，構造化の結果が被験者の主観と一致していることを示している．

5.2.4 考察

主観評価実験から，システムを構築する上で想定していたようなページに対しては，人間の理解する構造と同様の構造が得られていることがわかる．しかしながら，想定していなかったページに対しては，うまく構造化出来ないという問題点が明らかとなった．

以下では，本研究の提案手法で扱えなかった部分に対して，今後の解決すべき課題となる点について考察する．

繰り返し構造の検出に利用するタグについて

人間の直感と一致する構造が得られなかった原因のうち，最も大きな原因は繰り返し構造を検出する際に使用するタグの決定にあると思われる．

本研究では，繰り返し構造の検出に使用するタグを固定することで，汎用的なシステムを目指した．しかしながら，今回は削除した
タグや<font>タグなどは，場合によっては，セグメントテーションを行う際に不可欠な情報になりうる．このように本来このような利用するタグの決定は，ページあるいはページの部分に動的に決定するべき問題である．

また，タグ以外の部分にもセグメント境界が存在するということも実験により明らかになっている．これらを解決するためには，セグメントの境界となりやすい，括弧，空白などの文字の利用も今後検討するべきである．

ボトムアップに構造を構築する手法について

本研究の提案手法では，HTML 文書中の繰り返し構造を発見しグループ化していくことで，ボトムアップに構造を組み上げる．このような方針の最も危険な点は，下の階層で生じた，セグメンテーションの誤りが，上の構造に非常に大きな影響を与えてしまうということである．

別のアプローチとして，テーブルの部分や，リスト構造の部分などをあらかじめ DOM の構造を利用し，トップダウンに大まかなセグメンテーションを行っておき，ある程度の細かさになったら，その部分部分に対して，現在のようなボトムアップな構造化を行う方法も考え

られる．

このような手法についても今後検討する必要がある．

要素に応じた構造の構築手法の適用

現在の実装では，どのような要素に関しても，連続する繰り返し構造を発見することで構造化を行っている．テーブルに関しては，転置などの処理を行っているが，連続する繰り返し構造を発見するという構造化の手法は同じである．

しかしながら，リスト構造などは，DOMの構造がそのまま人間の理解する構造を示している．また，<pre>タグでレイアウトがなされている部分は，現在の枠組みでは構造化する事が出来ない．このようにテーブルはテーブルを処理する手法，リスト構造は，リスト構造を扱う手法と，要素毎に最適な手法を用意するべきである．

使用していない情報の利用

現在のシステムでは，フォントの色，サイズなどの情報や，リンクの種類，画像のサイズなどの情報を一切用いていない．しかしながら，特にフォントの情報などは，人間が理解する構造に影響を与えているはずである．このような情報を何らかの形で使用することができれば，構造化の精度もより向上するのではないかと考えられる．

また，言語の意味に関する情報なども考えられる．例えば，単語の属するカテゴリの近いもの同士が並んでいる場合は，優先的に繰り返し構造と見なしても良いかもしれない．また，レイアウトからは検出できない構造を新たに発見できるかもしれない．

このように，現在の実装では利用していない，さまざまな情報を利用することで，現在のシステムでは扱えないような問題が解決できると考えられる．

6 おわりに

6.1 本研究のまとめ

本研究の目的は，Web上にある情報を計算機が自動的に扱えるようにするために，情報の自動的なセグメンテーション，構造化をおこなうことであった．

Semantic WebやWeb APIに代表されるように，計算機がそのまま扱うことの出来る構造化された情報は非常に有用である．しかしながら，このような構造化がされた情報は，現状ではなかなか手に入れづらいという問題がある．これは，Web上にある情報のほとんどは，人が目で見て理解するための情報であるためである．よって，このようなレイアウト記述言語でマークアップされた情報をそのまま計算機で扱うことは非常に難しい．

本研究では，「人間はWebページを見るときページのレイアウトから情報の“まとまり”を理解することで，そのページの構造を理解することができる」という観察に基づき，HTML文書中から繰り返し構造を抽出することで，ページに含まれる情報を構造化する手法を提案した．前節で述べたように現状のシステムはまだ完全ではないが，いくつかのWebページでは，ほぼ人間の理解する構造と同じ構造が得られている．

このように，Web上の情報から構造化された情報を自動的に生成することが可能になると，Web上の情報を計算機で自動的に処理することが可能になり，様々な利用法が考えられる．例えば，構造化された情報を階層的にたどることの出来る音声ブラウザや，情報のセグメンテーションの結果を利用して，1ページに含まれる情報を携帯端末用に複数ページに分割することも可能になるだろう．このように，本研究で構築したシステムは，Web上の情報を処理する様々なアプリケーションの前処理として使用できるシステムである．

6.2 今後の課題

考察でも述べたが，現状のシステムには扱うことができないさまざまケースが存在する．今後の課題として，これらの問題を解決するためのシステムの改良がある．また，構造化の結果を利用したアプリケーションを試作しての主観評価についても考えていく必要があると考えている．

構造化が可能になった後のさらなる課題としては，構造化されたセグメントへのタイプ付けがある．例えば，サイト外部へのリンクが含まれるセグメントは，リンク集的な属性を持っているだろう．また，今回の実験を通じて特徴的であった，「あるグループの直前に位置するテキストや画像には，そのグループに対するタイトル的な意味合いを持っている」など，タイプ付けの手がかりになりそうな情報も構造化の結果から得られる可能性がある．このような構造化の結果を利用したWebページのセグメントへのタイプ付け，さらには，そのようなセグメントのタイプ付けの結果を利用したWebページ全体へのタイプ付けなどが可能になれば，Web上の情報を計算機で自動的に扱う上で，さらに有用であると考えられる．

参考文献

[1] Tomoyuki NANNO, Suguru SAITO, Manabu Okumura,
“Zero-Click : a system to support Web browsing.”,
In Proc. of The Eleventh International World Wide Web Conference,
2002.

[2] Lakshmi Vijjappu, Ah-Hwee Tan, Chew-Lim Tan,
“Web Structure Analysis for Information Mining”,
In Proc. of the First International Workshop on Web Document Analysis (WDA2001)
2001.

[3] William W. Cohen, Matthew Hurst, Lee S. Jensen,
“A Flexible Learning System for Wrapping Tables and Lists in HTML Documents”,
In Proc. of The Eleventh International World Wide Web Conference (WWW2002),
2002.

[4] 上坂 吉則, 尾関 和彦,
“パターン認識と学習のアルゴリズム”,
文一総合出版, 1990.

[5] Dave Raggett, ‘Clean up your Web pages with HTML TIDY’,
URL: http://www.w3.org/People/Raggett/tidy/.

[6] Unicode Home Page,
URL: http://www.unicode.org/.