知能と複雑系　132－16
(2003. 3. 14)

HTMLレイアウトからの知識発掘

櫻井 成一朗†

†東京工業大学　大学院情報理工学研究科　〒152-8552 東京都目黒区大岡山 2-12-1
E-mail: †sakurai@cs.titech.ac.jp

あらまし 統一的なWebページを保守していくために,HTML文書の内容と表示を分離しておくことが望ましい.HTML文書の内容と表示を分離しておくことで,HTML文書作成者は内容の保守に専念することができるからである.本論文では,HTML文書の表示をHTMLテンプレートとして分離し,内容を外部データとして取り扱う.HTML文書の表示のための知識をHTMLテンプレートとして分離することでHTML作成者のページデザインにおける知識を発掘する.

キーワード HTMLレイアウト，HTMLテンプレート，知識獲得，知識発掘

Knowledge Mining from HTML Layout Information

Seiichiro SAKURAI†

†Graduate School of Information Science and Technology, Tokyo Institute of Technology 2-12-1 O-Okayama, Meguro-ku, Tokyo, 152-8552 Japan
E-mail: †sakurai@cs.titech.ac.jp

Abstract In order to maintain a Web site, the separation of content and presentation of HTML documents is now a fundamental issue. The concept of HTML template is a method for the separation of content and presentation. By means of the template, the intention of the page designer can be represented as a HTML template. In this paper, knowledge mining method for Web page designer by using HTML templates.

Keyword HTML layout information，HTML template, knowledge acquisition，knowledge mining

1. はじめに

今や企業や行政機関においてWWWは重要な広報媒体の一つであり,その重要性が年々益々高まっている.WWWの普及に伴って,村田[8]や廣川[9]らの研究が示すようにWWW間のリンク構造に着目して有用な情報を抽出することができることが知られている.これはWebコミュニティ[7]がWeb作成者が関連深いと考えたという知識を抽出することに成功していると見ることができる.Web間のリンク情報から更に知識を得るべく実験した[10]ところ，リンク集のような構文的に制約のあるHTMLページから知識を発掘するためには，ページ作成者の意図を何らかの形で抽出する必要があることがわかった.本論文では，ページデザイナーとしてのページ作成者の意図を知識発掘の対象として捉え，ページ作成者の意図を抽出する方法について考察する.

WWWは顧客や利用者が接する企業や行政機関の顔であるので，利用者の便宜を最大限に考慮する必要がある.その結果，サイト内の閲覧を容易とするためには，統一的なページ構成が望まれる事になる.単に統一的なページ構成を採用するだけであれば，サイト内の全ファイルを一括して作成することで対応できるが，ページの更新は困難になる.積極的な情報公開が望まれている今日では，統一的なページ構成のために更新が滞ってしまう事態を避けなければならない.ページの更新を容易にしつつ，サイト内のページに統一性を保持するための様々な技術が開発されている.そのような技術の一つとして，HTML文書を直接作成するのではなく，HTML文書を動的に展開する方法が商用Webサイトでは積極的に導入されている.JSP(Java Server Page)やPHPに代表されるように,これらのWebサイトではHTML文書の中にスクリプトを埋め込み，スクリプトの実行結果で置換されたHTML文書がユーザに提供されるのである.また,DTML[3],Xtemplate[5]やamrita[4]等，直接HTMLを記述するのではなく，予めHTMLテンプレートを用意しておき，別途用意したデータによってHTMLテンプレート中の変項を置換するHTML文書生成方法もある.どちらの方法にしても，HTML文書の内容とHTML文書の表示を分離することができる.HTML文書の内容と表示を分離することは，

ページ作成者の内容に関する考えと表示に関する考えを分離することにもなる．すなわち，表示の部分を分離することで，ページデザイナーとしてどのようにページデザインを行っているのかという知識を分離することになる．本論文では，HTML 文書の内容と表示を分離することで，ページデザイナーとしての知識発掘を試みる．そのために表示知識を HTML テンプレートによって表現し，HTML テンプレートを具体的なデータで埋める事で，HTML 文書の表示と内容を分離する．

2. HTML 文書における内容と表示の分離

2.1. HTML 文書における内容と表示

HTML 文書では，限定的ながら表示を制御することができる．しかしながら，ブラウザ依存性が強く，作成者の意図通りに表示できないことが少なくなかった．このため W3C によって HTML 文書には内容だけを記述するように HTML4.01[2]が規定され，表示のためにカスケードスタイルシート[1]が規定された．カスケードスタイルシートによってブラウザによる制限は依然として残されているものの，HTML 文書の内容と表示を分離できるようになった．

2.2. サーバサイドプログラミングによる動的 HTML 文書生成

動的に HTML 文書を提供する方法として，SSI（サーバサイドインクルード）や JSP 等のサーバサイドプログラミングがある．サーバサイドプログラミングとは，WWW サーバ側でプログラムを実行して，その実行結果に従って HTML 文書を生成する技術である．例えば，SSI を利用することで，サイト内の HTML 文書に対して同一のヘッダ・フッタを用いる事ができ，統一的なレイアウトを採用することができる．サーバサイドプログラミングはデータベース等の外部サービスと WWW を連携することで動的にページを生成するのが主目的であるが，表示と内容を分離するという観点からは，統一的な書式で HTML 文書を生成している事に着目する必要がある．

2.3. HTML テンプレート方式による HTML 文書生成

サーバサイドプログラミングは動的に HTML 文書を生成するが，静的に HTML 文書を生成するのであれば，予め HTML テンプレートを用意しておき，HTML テンプレートに具体的データを埋め込む方法[4,5]が考えられる．HTML テンプレートによって表示を定めるので，統一的な書式で HTML 文書を作成でき，整合的な Web サイトも構築できる．HTML テンプレート自身は HTML や XML データとして与えるが，ここでの HTML テンプレートは表示のためのデータであって，内容と表示を完全に分離する事ができる．内容の部分については外部のデータベースに蓄積することができるので，データベースと連携して Web サイトを構築することもできる．

3. 表示知識発掘

3.1. 表示知識の例

オブジェクト指向スクリプト言語 Ruby で記述された HTML テンプレートシステム amrita[4]では，HTML テンプレートを使うことで，HTML 文書の内容と表示を分離することができる． HTML 文書の内容は Ruby のデータとして与えればよく，与えられたデータを使って HTML テンプレートの展開を行い，ユーザに提示すべき HTML 文書を生成する．

HTML テンプレートシステムを用いない場合でも，Web 作成者の頭の中では内容と表示を分離しながら，HTML 文書を作成することは少なくないであろう．例えば，東京地区と大阪地区の映画の観客動員数を HTML 文書化する際には，まず観客動員数を地区毎に並べ替えた後に，図 1 に示すような HTML 文書を作成することになるであろう．

```
<table>
 <tr>
  <td>東京地区</td><td>大阪地区</td>
 </tr>
 <tr>
  <td> 映画 A</td><td>映画 B</td>
 </tr>
 ...
</table>
```

図 1 ：HTML 文書の例

図 1 の HTML 文書を amrita を使って生成する場合には，まず図 2 に示す HTML テンプレートを与えておき，図 3 に示すデータを amrita に与えれば良い．HTML テンプレートに付加されたメソッドを呼び出す事で HTML 文書の生成が行われるのである．

```
<table>
 <tr>
  <td>東京地区</td><td>大阪地区</td>
 </tr>
 <tr id=t1>
  <td id=tt> </td><td id=to></td>
 </tr>
</table>
```

図2：amritaによるHTMLテンプレートの例

図2のHTMLテンプレートは構文的にはHTML構文そのものであり，データの埋め込み対象のタグであることを示すためにid属性を付加している．id属性を持つタグであれば，その要素は外部のデータによって埋め込みが行われる事を示す．図2のid属性値であるt1, tt, toはデータを展開すべき位置を指示しており，それぞれ図3の対応する値が埋め込まれる事になる．図3中の映画A，映画B，映画C，映画Dはそれぞれ具体的な映画名を表す文字列であり，[ ]は配列を表し，{ }はハッシュを表す．データの埋め込みが完了した後では，id属性は不要なので，当該タグの属性からは削除される．

```
Data = {
 :t1 => [
  {:tt=> 映画 A, :to => 映画 B},
  {:tt=> 映画 C, :to=> 映画 D},
  ...
 ]
}
```

図3：提供すべきデータの例

図3に示したようにHTMLテンプレートを展開するデータは言語 Ruby のハッシュ（名前で値を参照可能な配列）または配列として与える．任意個のデータを与えることで，データの長さに応じて図1に示したようなHTML文書が生成されるので，内容を分離した効果を最大限に利用することができる．

本論文では，HTML 文書の表示知識を amrita の HTMLテンプレートによって表現し，HTML文書の内容知識については Ruby のデータとして表現することについて検討する．

3.2. 表示知識としての頻出順序木の検討

ブラウザ上での最終的な見栄えを決めるのは，Javascript やカスケードスタイルシートに依存する場合もあるが，基本的な表示を決めるためのHTMLテンプレートを表示知識として考える．したがって，HTML文書のレイアウトを行う構造をHTML文書から抽出することになる．HTML文書から抽出するのであれば，半構造化された文書中に繰り返し出現する同型な構造を効率的に抽出する方法[6,7]が知られている．まず頻出する同型な構造が表示知識として利用できるかどうかについて検討する．

素朴な考えに従えば，一定回数以上繰り返し出現する構造を表示知識として見なすこともできるであろう．半構造化された文書から頻出する木構造やグラフ構造を抽出する方法としては，浅井らの FREQT[6]やGhazizadeh らの SEuS[7]が知られている．FREQT を用いれば，HTML文書中の頻出順序木を高速に抽出することができる．その場合，HTMLタグの属性を無視して，FREQTアルゴリズムを適用するだけで，HTML文書中の頻出順序木を得ることができる．しかしながら，タグの意味を無視した頻出順序木をいつでも表示知識と見なす事はできない．この場合，HTML文書としての文脈を無視するので，頻出する木構造やグラフ構造が単に抽出されてしまうからである．例えば，表形式を用いている HTML 文書であれば，“<td>項目名</td>”という頻出順序木が抽出されることになるが，この項目がどの表に出現していたのかという情報は伴わない．もちろん，元のHTML文書に戻れば，文脈を調べることはできるが，レイアウト情報を知るという立場からは頻出順序木をHTMLの表示知識としてそのまま利用することは適切とは言えない．

また表示知識として頻出順序木を採用する場合には，その表現能力が十分とは言えない．頻出順序木は同型な順序木を探索するのであるから，ノード数が異なる順序木は別の順序木としてみなされてしまう．現実には長さの異なるような表示であっても同一の表示として扱いたい事が少なくない．実際，図2に示したHTMLテンプレートでは行数は任意個で構わないので，表現が単純化されていると考える事ができる．頻出順序木を用いた場合には，直接的に要素数を表現しておく必要がある．もしコンパクトな表現を求めるのであれば，頻出順序木に実在しない仮想ノードを導入するなどして，木構造の違いを吸収しておく必要がある．

3.3. HTMLテンプレートの利用

文脈情報を表現するためにHTMLテンプレートの利用を考える．頻出順序木自体でもある程度の情報を獲得できるが，表示知識としては必ずしも十分な情報を獲得できるとは限らない．頻出順序木では，文脈構造の表現と中間ノードの抽象化ができないのである．

HTML文書中に頻出する繰り返し構造がHTML文書

の表示知識として意味を持つためには，その文脈が重要となる場合が少なくない．異なる文脈，すなわち異なる表やリスト環境の下では，異なる意味内容を与えることが少なくないからである．HTMLテンプレートのタグはHTMLタグの意味を継承するものであり，画面上でのHTMLレイアウト（表示）を表現していると考えられる.したがって,HTMLテンプレート上では，ひつようなHTMLタグを記述しておくことで，自然な形で文脈を表現することができる．

HTMLテンプレートでは要素数はRubyのイテレータによって実現されるので，ユーザが要素数を考慮する必要はない．またHTMLテンプレートの一部が変項として機能するため，順序木構造と比較して，中間ノードの抽象化も自然に行われる事になる．

AmritaのHTMLテンプレートにおける繰り返し表現の構造としては，HTML構文の<ol>タグや<ul>タグによるリスト形式と図2に示した<table>タグによる表形式が代表的である．これらの繰り返し構造については相互に入れ子となって出現することもあるので，局所的な繰り返し構造であると言える．これに対して，HTML文書の中で大域的な繰り返し構造を用いることがある．その例を図4に示す．

```
<h2>タイトル 1</h2>
<p>
タイトル 1 とは、…
</p>
<h2>タイトル 2</h2>
<p>
タイトル 2 とは、…
</tr>
```

図4：大域的な繰り返し構造

図4に示したような繰り返し構造については，図5に示すようなHTMLテンプレートによって実現することができる．具体的なHTMLテンプレートとでは，タグの名前を予め予想しておかなければならないが，amritaではそのためにクラスを導入して実テンプレート数を抑制することもできる．

```
<span id=sec>
  <h2 id=title></h2>
  <p id=para></p>
</span>
```

図5：大域的なHTMLテンプレート

図5のタグ自身はHTMLの表示上は何も意味を持たないタグであるが，HTMLテンプレートを展開するための連続した区画を表現するために用いる.尚，amritaでは属性を持たないタグ自体はデータの展開後削除される．したがって,最終的に生成されるHTML文書に無意味なタグが保持され続けるということはない．また図5のHTMLテンプレートの<p>タグであるが，現実のHTML文書では必ずしも<p>タグが用いられているとは限らないので，そのような場合には，タグを用いることで対応できる．

HTMLテンプレートに適用するデータは，Rubyのデータであればよいので，HTMLテンプレートを再帰的に適用することも可能である．すなわち，データとしての文字列の部分に対して再帰的にHTMLテンプレートを適用する事もできるし，ユーザの判断で適用を打ち切ってもよいし，適用可能である限り適用し続けることもできる．

4. 実験

HTMLテンプレートの抽出は以下のように深さ優先探索によって行う．

1. HTMLテンプレートが適用できるかを確認し，適用可能であれば，表示と内容に分離する．分離したデータ部分に対して再帰的にHTMLテンプレートの適用を行う．
2. 適用可能なHTMLテンプレートがなければ，終了する．

局所的HTMLテンプレートの適用は,<ol>タグ,<ul>タグ，<dl>タグ，<table>タグに限られるので，これらのタグの場合には高速に適用することができる．また再帰的にHTMLテンプレートを適用する際には，ユーザと対話をして，HTMLテンプレートの適用を段階的に抑制する事もできる．

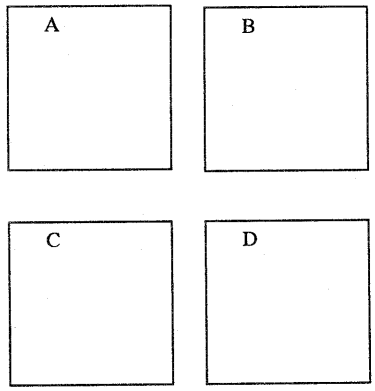

図6：HTML画面上のレイアウト

図6に,ある Web ページのレイアウト例を示す．このレイアウトでは表の要素として内部に表を持っている．このページでは A，B，C，D の各ブロックで表示しているのはアンカータグだけであるので，画面を節約するために表を使って配置したものと考えられる．

ブロック A，B，C については，ブロックの内容を説明する項目の背景色がオレンジ色で，ブロック D については項目の背景色が水色で内容に差がある事を示している．内容に差がある事はデータ構造の違いからも明らかであるので,ブロック D だけは別の HTML テンプレートであることがわかる．他の3つのブロックについては共通のテンプレートを利用できる．

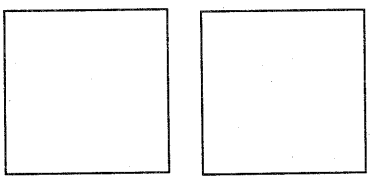

図7：表による2段組の実現

図7にしめすように，表を使って2段組を実現している Web ページもある．このようなページでは，2段組の各列の内容については更に繰り返し構造を見出すこともできる．調査した実際の Web ページの列の内容は国別の自動車メーカーのリンク集であり，国毎のリンク数は大きく異なっている．HTML テンプレートでは，表の行数や列数に依存せずに記述できるので，HTML テンプレートを再利用することができる.但し，注意しなければならないのは，通常の紙の文書と同じように左列から右列に連続するようにレイアウトされている場合には，左列の行数と右列の行数をバランスさせる必要がある．このような場合には，行数をバランスさせるための Ruby のプログラムコードをデータの一部として埋め込むことも可能である．データを加工する方針を採用すれば，HTML テンプレート自身は図2に示したテンプレートで十分である．また amrita ではテンプレートの一部に Ruby プログラムを埋め込むこともできるので，表示すべきデータの違いによって選択的に HTML コードを生成するような HTML テンプレートを用いることもできる．

5. 考察

本論文で提案した方法と頻出構造を抽出する FREQT や SEuS との違いは，FREQT や SEuS がタグを単なるラベルとして扱っているのに対して，本方法では HTML タグの意味を考慮していることにある．HTML タグの意味を考慮するので，例えば，<table>タグや<ol>タグ等の繰り返し構造を表現するタグに対しては，無条件に表示知識と見なしてしまう．これはページデザイナーがページデザインとして適切であると判断したために，これらの構造を導入したと考えている事に他ならない．これに対して，FREQT のようにタグの意味を考慮しない方法では，HTML 文書内の文脈も無視する事になってしまう．HTML テンプレートの場合には，特定のリスト形式や表形式中のデータ出現であるという文脈情報を，当該 HTML テンプレートに直接埋め込むことで容易に扱うことができるので，意味的な制約を課すこともできるようになる．また繰り返しを表現するための局所的テンプレートについては，タグ名によって探索を制御できるので，該当するタグが存在しない場合には探索を打ち切ることもできる．

本方法では，HTML 作成者が HTML 文書の内容と表示の分離を意識して作成しているという仮定の下で知識表現を考えている．このため，内容と表示の分離を意識せずに HTML 文書を作成している場合には，本来の用途とは異なるタグの使い方をする事がある．例えば，フォントサイズを変更するためだけに<h1>タグを用いたり，リスト形式でないにも関わらず<li>タグを用いることがある．このような本来の用途と異なる使い方をしている場合には，ブラウザ依存性が強くなるばかりか，分離された表示知識が正しくなくなる場合がある．

6. あとがき

本論文では,HTML 文書が内容と表示を分離して作成されることが少なくないことから，構文的繰り返し構造に基づき，単独の HTML 文書から表示知識を HTML テンプレートとして抽出する方法を提案した．Web 作

成者の分類知識を抽出することを Web ページに対して試み，その有効性を確認した．

HTMLテンプレートを抽出した結果，HTML文書で表現されていた内容についてはデータとして分離するので，分離したデータに対して更に知識獲得を試みることも考えられる．

HTMLテンプレートの柔軟性からHTMLテンプレートを十分なだけ用意しておけば，かなりのWebページに対して本方法を適用できる．しかしながら，ブラウザのバグに依存した方法を採用するなど，予期せぬ方法でHTML文書レイアウトを行っているページも少なくない．そのような場合にはHTMLテンプレートを自動作成する事が必要であるが，HTMLテンプレートの自動作成については今後の課題である．

文　献

[1] W3C, Cascading Style Sheets Level 2, CSS2 Specification, http://www.w3.org/TR/REC-CSS2/, 1998 .

[2] W3C, HTML 4.01 Specification W3C Recommendation 24 December 1999, http://www.w3.org/TR/html401/.

[3] Michel Pelletier and Amos Latteier, The Zope Book, ISBN 073511372, New Riders.

[4] Amrita, http://www.brain-tokyo.jp/research/amrita/index.html .

[5] Xtemplate, http://xtemplate.sourceforge.net/.

[6] T. Asai et.al., Efficient Substructure Discovery from Large Semi-structured Data, Proc. SDM02, pp.158-174,SIAM,2002.

[7] S. Ghazizadeh and S. Chawathe, SEuS: Structure Extraction using Summaries, In Proc. of The 5th International Conference on Discovery Science, Germany, November 2002.

[8] 村田剛，参照の共起性に基づくWebコミュニティの発見，人工知能学会誌， Vol. 16, No.3, pp. 316-323, 2001.

[9] 廣川佐千男，池田大輔，Web グラフの構造解析，人工知能学会誌，Vol. 16, No. 4, pp.525-529, 2001.

[10] 若月謙太郎，リンク解析によるWWWからの知識獲得に関する研究，東京工業大学　大学院情報理工学研究科　修士論文(2002).