ヘッダ抽出によるWeb文書構造化

吉田 稔　　中川 裕志
東京大学 情報基盤センター　　東京大学 情報基盤センター
mino@r.dl.itc.u-tokyo.ac.jp　　nakagawa@dl.itc.u-tokyo.ac.jp

1　はじめに

本研究は、**Web 文書の文書構造**を抽出することを目的とする。一般に文書構造とは、「見出し」と「本文」により構成される階層構造のことを指すが、本稿では Web 文書の見出しとして**ヘッダ**という概念を提案する。ヘッダは、「文書内の他の部分を修飾する文字列」として定義され、「題名」「セクション名」といった単純なタイトルだけでなく、「属性」や「日付」[1]等も含むものとする。例えば図 1 の Web 文書では、“自己紹介” や “名前”, “年齢” がヘッダであり、それぞれページ全体、名前 “山田　太郎”, 年齢 “25” を修飾している。この階層構造はまた木構造としても表現することができ、これを**ヘッダ木**と呼ぶ（図 2）。ヘッダ木を抽出することにより、構造を反映したより高度な検索システムや、音声対話によるブラウザ等の応用が可能となる。

Web 文書の文書構造の解析は、Wrapper Induction 等の分野で盛んに研究が行われている。これらの研究の多くでは、教師あり学習を用いて特定のサイトから特定のデータを抽出することを目的としているが、HTML タグの繰り返し構造を利用した手法もいくつか提案されており、Mukherjee[2] らは、このアプローチによって意味的な階層構造の抽出を試みている。この手法は、企業の Web サイトのように整った文書を対象とした場合は有用であるが、「HTML タグの繰り返し構造」という特定の手掛かりに依存しているため、記号や言語表現による階層構造表現を解析することができない。他に、Web ページを階層構造化する研究としては、松本らの研究 [4] があるが、対象を自治体サイトの文書のみに絞った教師あり学習を行っており、一般の Web 文書への適用可能性は未知数である。

また、近年、タイトルやヘッドラインを抽出するシステムもいくつか報告されている。Hu ら [1] は、HTML、Word ファイルをはじめとする様々なタイプのファイルからタイトルを抽出する手法を提案している。しかし、手掛かりとして、「フォントの大きさ」や、「文書中の位置（文書の最初にあるか、等）」の素性を用いている等、タイトル以外の見出し要素の抽出については考慮されていない。Tatsumi ら [3] は、タイトルに限らない見出し要素を、人手によるルールを用いて抽出する手法を提案している。しかし、解析対象を一つの企業の Web サイトに絞っており、ルールの Web 文書一般への適用性については議論されていない。

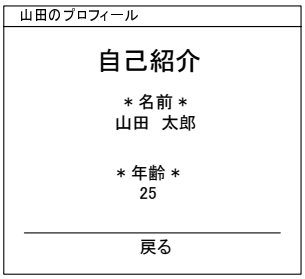

図 1: Web 文書の例

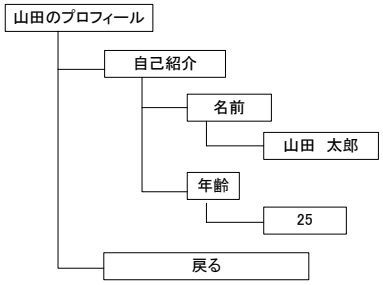

図 2: 図 1 の Web 文書から抽出されるヘッダ木

これに対し本研究では、言語表現･記号表現と HTML タグによる視覚効果を統一的に扱える生成モデルを用い、EM アルゴリズムによる教師なし学習を行うことで、適用範囲の広い階層構造抽出アルゴリズムを提案する。これにより、特に、従来研究では扱われてこなかった、属性・属性値表現に多く見られる「記号による階層構造の表現」についても解析が可能となる。

[1] 日記等における見出しとして使われる場合。

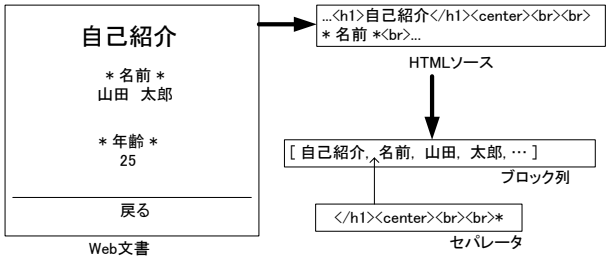

図 3: ブロック列生成の例

2　ヘッダ抽出アルゴリズム

システムは、入力された Web 文書を、**ブロック列**と呼ばれるデータに変換する。これは、文書の HTML ソースを HTML タグや記号を切れ目として分割し、結果をリストに格納したものである（図 3）。この切れ目のことを、**セパレータ**と呼ぶ。この結果、各 Web 文書は、ブロック列 $\mathbf{b} =< b_0, b_1, b_2, \cdots, b_n >$ と、セパレータ列 $\mathbf{s} =< s_1, s_2, \cdots, s_n >$ により表現されることになる[2]。

ヘッダ抽出アルゴリズムは、**セパレータ分類** (STEP-1)、**ブロッククラスタリング** (STEP-2) の 2 ステップに分けられる。STEP-1 で隣接するブロック同士の関係を推定し、STEP-2 で残りの関係をページ全体のレイアウトを元に決定する。以下、各ステップの詳細を説明する。

2.1　STEP-1: セパレータ分類

STEP-1 では、各セパレータを **NONBOUNDARY**, **REL**, **UNREL** の 3 つのクラスに分類することにより、セパレータを挟むブロック間の関係を推定する。NONBOUNDARY は同一ブロックとなるべきものを誤って切ってしまったセパレータ、REL はヘッダとその修飾先を挟むセパレータ、UNREL はそれ以外のセパレータである。例えば、図 3 のブロック列に対するセパレータ分類の正解は、図 4 のようになる。

分類の際は、各セパレータとその周辺のブロックを確率モデルで表現し、最尤推定を行う。ブロックの出現確率は、直前のブロックとセパレータのみに依存すると仮定し、文書全体の出現確率を以下で定義する。

$$P(\mathbf{s}, \mathbf{b}) = \prod_{i=1}^{n} P(s_i, b_i | b_{i-1})$$

ここで、各セパレータ s_i に対し、クラスを表す隠れ変数 c_i を導入する。さらに、(s_i, b_i) が隠れ変数 c_i

[2] s_i は、b_i の直前のセパレータを表す。

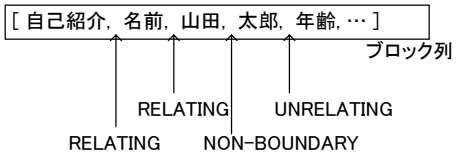

図 4: セパレータ分類の例

のみに依存することを仮定し、各 $P(s_i, b_i | b_{i-1})$ を以下のように分解する。

$$P(s_i, b_i | b_{i-1}) \quad = \quad P(c_i | b_{i-1}) P(s_i | c_i) P(b_i | c_i)$$

このモデルに基づき、各セパレータのクラスは以下の式で決定される。

$$\begin{aligned} \hat{c_i} &= \arg\max_c P(s_i, b_i | b_{i-1}) \\ &= \arg\max_c P(c | b_{i-1}) P(s_i | c) P(b_i | c) \\ &= \arg\max_c \log P(c | b_{i-1}) P(s_i | c) P(b_i | c). \end{aligned}$$

但し、データスパースネス問題を回避するため、各ブロック b_i はその suffix に、各セパレータ s_i はその prefix に事前に置き換えられる[3]。さらに、s_i 内に記号と HTML タグが混在する場合は、両者の特徴をモデルに生かすため、記号と HTML タグを分割し、それぞれに出現確率を設定することを行う。（このさい、$P(s_i|c)$ も独立性仮定によって分割する。）

各パラメータ $P(c|b_{i-1})$, $P(s_i|c)$, $P(b_i|c)$ は、EM アルゴリズムにより推定される。このさい、各クラスを特徴付けするために、各クラス毎にそれぞれ**典型セパレータ**を定義する。セパレータが典型セパレータであった場合は、無条件で対応するクラスに分類される。各典型セパレータは以下のように定義される。

NONBOUNDARY: 前処理で NONBOUNDARY に分類されたもの全て。前処理では、文章どうしの接続と文章内の記号を、主に句読点の存在を基準としたルールにより NONBOUNDARY に分類する。

RELATING: HTML タグを含まず、閾値以上の空白文字を含むセパレータ。

UNRELATING: 閾値以上の数の HTML タグを含むセパレータ。

2.2　STEP-2: ブロッククラスタリング

STEP-1 においては、隣接するブロック同士の関係を推定したが、STEP-2 では、それ以外のブロック

[3] suffix,prefix の長さはヒューリスティクスによるルールで決定される。

間関係を決定する。STEP-2 では、まず各ブロックに**depth** を与え、その depth に基づきヘッダ木を構成する。depth とは、ヘッダ木におけるノードの深さを表し、一般的に、depth が低いほど、タイトル等の重要な情報を表していると考えられる。各ブロックに depth が与えられれば、「隣接するブロック間で depth が増加していれば（右のブロックの depth が左のブロックの depth より大きければ)、そのブロック間は親子関係にあると見做す」「同一の depth にあるブロック同士は、姉妹関係にあると見做す[4]」というルールによりヘッダ木が構成できる。

以下、STEP-2 のメインである、「各ブロックに depth を与えるアルゴリズム」について説明する。

2.2.1 depth 付与アルゴリズム

各ブロックの depth は、他のブロックの depth に対する相対位置として格納される。あるブロック同士の間で相対位置が定義されているとき、そのブロック同士は**接続されている**と言う。アルゴリズムに入力として与えられるのは、部分的に接続されたブロック列であり、アルゴリズムは、すべてのブロックを接続し、接続された結果から、先頭のブロックの depth を 1 とすることで、すべてのブロックの depth を出力する。

接続を表現するために、各ブロック b_i に対し、相対位置を表すペア (p_i, q_i) を定義する。ここで p_i は、b_i が参照する先のブロックの番号であり、q_i は、参照先に対する b_i の相対位置を表す。

入力として与えられる部分的な接続は、STEP-1（セパレータ分類）の結果から生成される。例えば、AGE と 25 の関係が REL であった場合、25 の depth は AGE の depth より一つ大きくなる。より一般的には、s_i の値が REL であれば $(p_i, q_i) = (i-1, 1)$、NONBOUNDARY であれば $(p_i, q_i) = (i-1, 0)$ となる。それ以外の (p_i, q_i) は未定義とする。（このとき、$p_i = -1$ と設定する。）

ここで、depth 推定にあたり、以下のような仮定を置く。

1. リストの先頭に近いブロック程、重要であり、depth が低い。
2. 類似したレイアウトを持つブロックは、同一の depth を持つ。

ブロックのレイアウトは、隣接するセパレータによって特徴付けられる。例えば、図 1 の HTML ソースにおいて “名前” と “年齢” はそれぞれ

[4]但し、両者の間により低い depth のブロックがない場合に限る。

Given:
a list of blocks $\{b_1, b_2, \cdots, b_n\}$
a list of relative relations $\{(p_1, q_1), (p_2, q_2), \cdots, (p_n, q_n)\}$

for $i := n$ **to** 1 **step** -1 **do**
  if $p_i = -1$ **then**
    if $\max_k sim(b_i, b_k) > threshold$
    then $j := \arg\max_j sim(b_i, b_k)$; $p_i := p_j$; $q_i := q_j$
    else $j := i - 1$; $p_i := p_j$; $q_i := q_j + 1$
end

図 5: depth 付与アルゴリズム

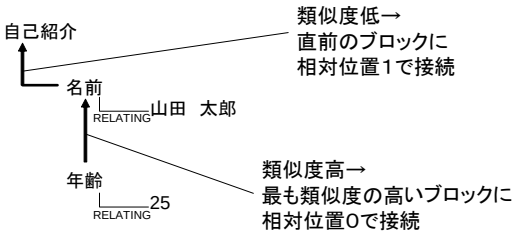

図 6: 接続の例

```
... <br><br>* 名前 *<br> ...
... <br><br>* 年齢 *<br> ...
```

と表現され、両者とも「左側に

がある」「右側に
がある」として特徴づけできる。各ブロック b_i に対し、s_i 内のタグ・記号の頻度をベクトルで表現したものを**左コンテキスト**、同様に s_{i+1} のベクトル表現を**右コンテキスト**と呼ぶ。ブロック間で、左コンテクスト同士と右コンテキスト同士のコサイン類似度を計算し、大きいほうをそのブロック間の類似度と定義する。

図 5 にアルゴリズムを示す。ここで、$sim(b_i, b_k)$ は b_i と b_k の類似度を表す。アルゴリズムは、ブロック列を後方から処理し、各ブロックの接続先と相対位置を決定していく。接続先は、最大の類似度を持つブロックが選択されるが（この場合、相対位置は 0 となる)、もしもその類似度が閾値（現在は 0.8）を下回った場合は、類似するブロックは存在しないと判断し、仮定 1 に基づいて、直前のブロックに相対位置 1 で接続する。図 6 に、接続の例を示す。

3 実験

WWW から集めた 3,000 個の Web 文書を対象に、実験を行った。ただし、集める文書には、「1,000 バイト以上 10,000 バイト以下」「“src” “script” タグを含まない」といった制限を加えてある。

[山田のプロフィール, [自己紹介, [名前, [山田　太郎]], [年齢, [25]]], [戻る]]

図 7: 図 2 のヘッダ木に対するリスト表現

Algorithm	Recall	Precision	F-measure
micro-averaged			
EM	0.548	0.439	0.487
BASELINE	0.530	0.413	0.465
macro-averaged			
EM	0.492	0.392	0.424
BASELINE	0.481	0.402	0.416

表 1: 実験結果。“EM” が提案手法を表す。

評価は、ヘッダ木の**リスト表現**を用いて行う（図 7)。システムによるカッコ付けと、人手によるカッコ付けを比較し、recall, precision, f-value を算出する。各カッコ付けは、開きカッコとそれに対応する閉じカッコの位置のペアとして表現され、このペアがシステムと人手で一致したときに正解と見做す。上記 3000 文書のうち 50 文書を対象に人手によるカッコ付けを行い、テストデータとして用いる[5]。

3.1　ベースライン

比較のため、STEP-1 において EM 学習を行わないヒューリスティクスによるアルゴリズムをベースラインとして実装した。このアルゴリズムは、STEP-1 において、以下のルールにより各セパレータ s のクラス分類を行う。

1. もし s が典型セパレータであれば、それに応じたクラスに分類する。
2. それ以外の場合、もし s が HTML タグを含まなければ、NONBOUNDARY に分類する。
3. それ以外の場合、UNREL に分類する。

3.2　結果

結果の評価は、**bracketed recall** と **bracketed precision** を用いて行う。bracketed recall は、「正解のカッコ付け数 / 人手によるカッコ付け数」として、bracketed precision は、「正解のカッコ付け数 / システムによるカッコ付け数」として定義される。また、F-value は、recall と precision の調和平均である。これらの値を各文書で評価し、その平均 (macro-average) を計算した結果と、全テストデータを一つの文書と見做し、全体での bracketed recall, bracketed precision を計算した結果 (micro-average) を、図 1 に示す。

EM アルゴリズムの使用により、ある程度の精度の改善が認められた。これらの改善は、主に REL に分類される記号（“:” 等）や、NONBOUNDARY に分類される HTML タグ（文章が
により切られている場合）を正しく分類できたことによるものであった。しかし、改善幅はそれほど大きなものではなく、これは、主に STEP-1 で間違った分類をしてしまった場合に、その影響が STEP-2 により文書全体に波及してしまうことによる精度低下が原因であった。従って、精度向上のためには、信頼度の低い分類をクラスタリングの際に無視する等、STEP-1 と STEP-2 との結果を統合するための処理が必要となると思われる。

4　おわりに

本稿では、Web 文書の文書構造をあらわすデータ構造である**ヘッダ木**を提案し、EM 学習とクラスタリングを用いたヘッダ木抽出アルゴリズムを示した。実験では、EM 学習により精度が僅かながら改善することを確認した。しかしながら、実用的アプリケーションのためには、より高い精度が求められる。学習データの増加や、EM 学習とクラスタリングの結果をブースティングする等の工夫により精度の改善を行うことが今後の課題である。

参考文献

[1] Y. Hu, G. Xin, R. Song, G. Hu, S. Shi, Y. Cao, and H. Li. Title extraction from bodies of HTML documents and its application to web page retrieval. In *SIGIR'2005*.

[2] S. Mukherjee, G. Yang, W. Tan, and I.V. Ramakrishnan. Automatic discovery of semantic structures in HTML documents. In *ICDAR 2003*.

[3] Y. Tatsumi and T. Asahi. Analyzing web page headings considering various presentation. In *WWW2005 Poster Session*.

[4] 松本吉司, 乾健太郎, 松本裕治. Web ページのテキストセグメント階層構造の抽出. 言語処理学会第 11 回年次大会論文集, 2005.

[5] カッコの総数は、1,388 であった。