1C-6

テキストノードへのパスに基づくWebページからの価格情報の抽出

福井　雅之†　　増田　英孝†　　中川　裕志‡

東京電機大学工学部†　　東京大学情報基盤センター図書館電子化研究部門‡

1.はじめに

現在、Web 上には膨大な量の情報が存在しているが、これらを効率よく扱うことが近年重要視されてきている。大量の情報の中から重要なデータのみを抽出することはデータマイニングと呼ばれ、特に、Web ページからデータを抽出することは Web マイニングと呼ばれる。

先行研究で、精度の高いものに、Liu らの MDR(Mining Data Records)[1]がある。これは、機械学習を用いないもので、繰り返し構造に着目して抽出を行う。繰り返し部分をデータレコード、それらを持つ領域をデータ領域として扱い、データレコードが連続した Web ページに対応している。MDR のもつ欠点としてデータレコードの不連続なものに適応できないということがあげられる。

本研究ではテキストノードのパスの類似性によりテキストノードをラベリングすることで、データレコードが不連続の場合にも適用できる手法を提案し、価格情報を含むページから価格情報の抽出を行う。

2.データレコードを持つページのパターン

データ領域の分類として、データレコードが連続しているか不連続かに分類できる。データレコードが不連続とは、画像や HR タグによりデータ領域が分割されることである。

Web ページを分類する際に、1 つのデータレコードを構成するセルの個数 x , y と行方向のデータレコードの個数 ℓ を組み合わせて $(x, y.\ell)$ とし分類を行う。ここで、テーブルの行数、列数をそれぞれ n , m とする(図１)。

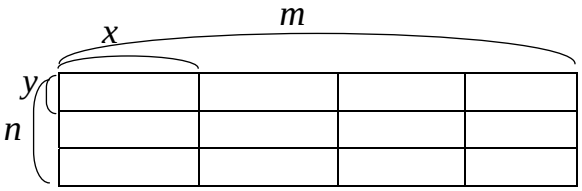

図 1:データレコードの表現方法

この方法を用いて以下の 4 つに分類する。なお、 j,k は $1<i<n$, $1<j<m$ となる任意の定数である。

(a)連続で 1 行に 1 個　　(1,*m*,1)
(b)連続で 1 行に複数個　　(1,1, *j*)
(c)不連続で 1 行に 1 個　　(*i*,1,1)
(d)不連続で 1 行に複数個　(*i*,1, *j*)

以下にその例を示す(図 2a,b,c,d)。破線で囲った部分がデータレコード 1 個分である。

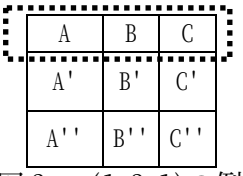

図 2a: (1,3,1)の例

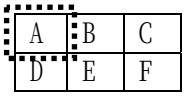

図 2b: (1,1,3)の例

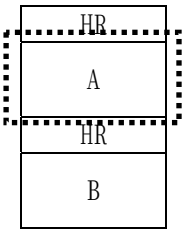

図 2c: (2,1,1)の例

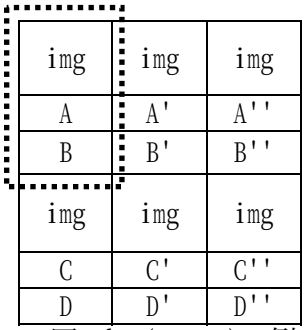

図 2d: (3,1,3)の例

3.Webページに関する考察

繰り返し構造を持つページの大部分は以上のタイプのいずれかに分類されるが、これらの多くがレイアウトのために Table タグを用いている。

そこで、本提案手法は次の 2 つの繰り返し構造を持つページに関する考察をもとにする。

(1)繰り返し構造となっているテキストノードへのパスは類似している。

Extracting price information from web based on text node path

† 東京電機大学工学部，東京都
School of Engineering, Tokyo Denki University, Kandanishikicho 2-2, Chiyoda, Tokyo, 101- 8457 Japan

‡ 東京大学情報基盤センター，東京都
Information Technology Center, University of Tokyo, Hongo 7-3-1, Bunkyo, Tokyo, 113- 0033 Japan

(2)繰り返し構造となっているテーブルは HTML ツリー上で同じ深さに存在している。

まず、１番目に関して、繰り返しとなっているレコードは同じテーブルに書かれていることが多い。そのため、HTML ツリーのルートノードからそのテーブルまでのパスが類似しているといるという特徴がある。また、これに関しては Table タグがネストしていても成立する。

次に、2 番目に関して、データベースから動的に出力される各商品データは基本的に同じルーチンを用いていると考えられる。そのため、個々のレコードは基本的には同じ深さ、同じ構造を持つといった特徴がある。

4.テキストノードへのパスを利用した手法

本手法の処理手順は次の 3 ステップに分けて行う。

(1)テキストノードのラベリング
(2)データ領域の抽出
(3)データレコードの認識

なお、前処理として HTML から HTML ツリーを構築する。

4.1 テキストノードのラベリング

まず、HTML ツリーからテキストノードのみを抽出し、ルートノードからのパスを作成する。そして、そのパスに出現したノードの集合 S を作る。このとき、各ノードの属性も考慮する。これは、メニューなどでは width 属性が意味を持つため、これらを識別するためである。以下に例を示す(図 3)。

```
HTML/BODY/TABLE(border=1,width=100%)/TR/TD/TEXT1
HTML/BODY/TABLE(border=0,width=20%)/TR/TD/TEXT2
                        ⇩
S = { HTML,BODY,TABLE(boder=1,width=100%),
            TABLE(boder=0,width=20%), TR, TD }
```

図 3:タグ集合 S の作成

この集合を元にテキストノード 1 つに対しベクトル V(text_num,html_tag) を作成する。text_num はテキストノードの番号であり、html_tag は含むタグの要素である。この値は html_tag を含むか含まないかの 0,1 である。ラベリングのアルゴリズムを以下に示す(図 4)。ここで、sort_by_freq は SortTagCountList をキーにソートする関数である。

```
/* cout up tag  */
for each tag in S {
    push count(tag) to TagCountList
}
/* sort Tag Count List by descendant */
SortTagCountList = sort( TagCountList )

/* sort by tag count  */
LabelList(node) =
  sort_by_freq(V(node,tag),SortTagCountList )
```

図 4: テキストノードのラベリング

4.2 データ領域の抽出

同レベルのテーブルを抽出するために、ツリーの幅優先探索を行う。そして、Table タグを発見後、含まれる要素の数を行・列ごとにカウントする。さらに、深さ３以上などのルールによってフィルタリングを行う。

4.3 データレコードの認識

データレコード不連続の場合には、同じテーブル内で、１つのデータレコードが複数行にまたがっている場合がある。それらをテキストノードのラベルを行ごとに編集距離[2]を用いて比較してグループにまとめ、個々のデータレコードを認識する。

5.評価実験

対象とするページはGoogle[3]のオンラインショップのカテゴリの中からランダムに100 件選び使用した。データ領域認識の精度は93%、データレコード認識の精度は81%であった。データレコード誤認識の多くは $(i,1,j)$ タイプであり、今後、データレコードの認識精度向上のためには、テキストノード内の文字列を活用する。

6.おわりに

本研究ではデータレコードが不連続の Web サイトに対応できる手法として、テキストノードへのパスからラベリングする手法を提案した。

参考文献

[1] B. Liu, R. L. Grossman, Yanhong Zhai. Mining data recordsin Web pages. In KDD, pages 601-606, 2003.

[2]Baeza-Yates, R. .Algorithms for string matching: A survey..*ACM SIGIR Forum*, 23(3-4):34-58, 1989

[3]Google:http://www.google.co.jp/