Abstract

Tables in HTML Web pages have become precious knowledge sources. Therefore it is reasonable and necessary to develop an algorithm to extract knowledge from them. For this, we need a system to identify the boundary between attributes and values of a table in HTML and transform tables into more understandable attribute-value pairs. In this paper, we propose an algorithm for this purpose. The outline of the algorithm is that if we find a row (or column) having low similarity with other rows (or columns), it is probably an attribute name row (or column), otherwise value data rows (or columns). The algorithm based on this idea results in 82% accuracy of recognition of lengthways and 78% accuracy of recognition of sideways for 300 tables in HTML of Web pages downloaded from the Web.

1 Introduction

Ever increasing number of HTML Web pages, tables are regarded as precious knowledge sources. It is necessary for extracting knowledge from HTML tables in order to utilize them. Then we propose an automatic table structure recognition system which can identify the boundary between attribute name rows (or columns) and its corresponding value data rows (or columns) in the table. In general attribute name cell is expected not to be similar to its value data cell in its form and its linguistics features. On the contrary, value data cells are expected to be very similar each other. Based on this idea, in our algorithm, we extract many linguistic features from each cell data of the table and use them to identify table structures at first. Then if the similarity between the feature vector of cell data of $n$-th row (or column) and that of other rows (or columns) is low, we regard the $n$-th row (or column) as attribute name, and if it is high, it is regarded as value data row (or column).

Section 2 classifies existing HTML tables on the Web, then we describe definition and calculation of the similarity of cells based on vector-space model used in our table recognition system, and we evaluate the algorithm experimentally in Section 3. Section 4 is our conclusion.

2 Classification of existing tables on the Web

In this section, we classify HTML tables into three classes according to objective of use of $<$TABLE$>$ tags as follows.

**Tables for Web page layout:** This type of table is used to tune up a Web page layout.

**Genuine tables:** This type of table has a structure of enumerating attribute data corresponding to attribute name. Genuine tables have the following characteristics in general. $<$TABLE$>$ tag’s BORDER attribute value can be more than one and the table consists of more than two cells.
Tables for emphasis: This type of table is used for emphasizing a picture or a simple list of, for instance, Web page links as a frame (only one cell with thick border). It does not pay any attention the logical and informational contents of tables such as data consistency.

In this research, we only focus on the second type: genuine tables because they are regarded as a sort of database and include valuable knowledge or data compared to the first or third types. Needless to say that identifying genuine tables among unseen HTML tables is essential. Wang, et al. have already proposed a good algorithm to solve this problem (WANG and HU, 2002). Their algorithm distinguishes type of tables whether genuine table or use of page layout in HTML documents in 95% accuracy. In addition, we expect that this recognition can be enhanced by applying our algorithm with some modification. That is our future work. Among genuine tables, we specifically focus on N by M table where either N or M or both are larger than 2 because 1) if N or M is one or both N and M are two, the table is regarded just as a list and no recognition is needed, and 2) in N by 2 or 2 by M table which N or M is larger than 2, our algorithm that we will describe the next section can be applied to the direction of N or M, so if the algorithm could not detect any attribute name from N or M side then the table is to be regarded as a list of attribute-value pairs.

Looking more closely at genuine tables, they have the structure which consists of attribute names and its value data (YOSHIDA, 2002). We further classify this type into the following three types based on where attribute names are located within the table.

Type of lengthways list: This type of table consists of two parts: Each data of the first row (or at most a few rows from the top of table) are used for attribute names of the table and the rest of rows are for values of each attribute.

Type of sideways list: This type of table is a transposed table of “type of lengthways list.”

Type of timetable: The type of timetable has both attribute names of row and column respectively.

We propose an algorithm which recognizes those three types among genuine tables in the following section.

3 Table Recognition Algorithm

Before going into the detail of our algorithm, we describe the previously proposed algorithms.

3.1 Previous works

Hurst, et al. (HURST and DUGLAS, 1997) have proposed a recognition system for table schema from preformatted tables in plain texts. Their targets, however, are tables normalized in advance, and they do not treat tables having multiple rows or columns of attributes. Ito, et al. (ITOH et al., 1999) are extracting information from tables. They did not consider multiple rows or columns of attributes either.

3.2 Outline of algorithm

Our targets are genuine tables as we said in the previous section 2. The outline of our algorithm is as follows.

At first, we normalize a table as shown in Section 3.3. Then we extract features from each cell of the table in order to make a vector representation of each cell. As features, we use linguistic or numerical feature that represent characteristics of each cell data in the table which we will later describe in 3.4.

Then, to recognize the boundary between attribute and value data, we calculate similarities between a cell and other cells on the same column or row. As already said, this similarity is low when a cell is for attribute name and high when a cell is for value data. Based on this similarity, we try to find the boundary of attribute name row (column) and value data row (column).

The remaining problem is how to chose the threshold which decides whether a given pairs of rows (columns) is boundary or not. We take the threshold value which maximize 10 fold cross validation. Finally, we evaluate precision of the algorithm experimentally in 3.6. In the remaining part of this section, we will describe the details of the proposed algorithm.
3.3 Table normalization

Before recognition process, we normalize a table. Figure 3.3 shows a table which has both rowspan and colspan options that combine more than two cells. In such cases, we normalize the table as shown in Figure 2. By this normalization, we become able to treat every cell in the same manner (Chen et al., 2000).

<table>
<thead>
<tr>
<th>菜名</th>
<th>食物名</th>
<th>ランゴ</th>
<th>バナナ</th>
<th>ミカン</th>
</tr>
</thead>
<tbody>
<tr>
<td>春のナツメ</td>
<td>カルシウム (mg)</td>
<td>10.1</td>
<td>2.1</td>
<td>3.5</td>
</tr>
<tr>
<td>夏のナツメ</td>
<td>ビタミンC (mg)</td>
<td>1000</td>
<td>2764.4</td>
<td>349</td>
</tr>
<tr>
<td>秋のナツメ</td>
<td>亜鉛 (pg)</td>
<td>376.2</td>
<td>3776.3</td>
<td>763.0</td>
</tr>
</tbody>
</table>

Figure 1: A table with rowspans and colspans.

<table>
<thead>
<tr>
<th>菜名</th>
<th>食物名</th>
<th>ランゴ</th>
<th>バナナ</th>
<th>ミカン</th>
</tr>
</thead>
<tbody>
<tr>
<td>春のナツメ</td>
<td>カルシウム (mg)</td>
<td>10.1</td>
<td>2.1</td>
<td>3.5</td>
</tr>
<tr>
<td>夏のナツメ</td>
<td>ビタミンC (mg)</td>
<td>1000</td>
<td>2764.4</td>
<td>349</td>
</tr>
<tr>
<td>秋のナツメ</td>
<td>亜鉛 (pg)</td>
<td>376.2</td>
<td>3776.3</td>
<td>763.0</td>
</tr>
</tbody>
</table>

Figure 2: Normalization of the table.

3.4 Features

Henceforth, we denote a cell of \(i\)-th row and \(j\)-th column in the table as \(Cell_{ij}\). Each cell is represented as a vector which consists of a number of features. A value of each element of vector is 1, if it has the corresponding feature, otherwise 0.

Formally we define the vector which consists of \(w_k, k = 1, \ldots, N\) as Formula (1) where \(N\) is the number of features we use.

\[
\overrightarrow{Cell_{ij}} = (w_1, w_2, \cdots, w_N)
\]

As for features, we use 107 features, that means \(N = 107\) in our experiment and they are described in the following.

Sequential numeric numbers (1 feature)

The feature \(\mathbf{L}\) value is 1 if a row (or column) has sequential numeric numbers increasing from initial value with the fixed interval value.

Punctuation marks (6 elements)

Since attribute names tend to be a brief string with no punctuation marks, we use the following six punctuation marks: “;”, “,”, “。”, “、“. Namely each of these punctuation mark corresponds to one feature. For instance, if a string in cell includes “;”, the feature corresponding to “;” is 1, otherwise 0.

In the explanation about the remaining features, if there are both one byte and two byte code for the same character, we give them different feature according to each code respectively.

Length of cell data (3 elements)

The length of attribute names tends to be short in comparison with the length of attribute values. Then we use the length of a string in a cell as features. However, assigning a feature for each length, namely, 1, 2, \cdots\, respectively consumes too many features and it may even loose the discrimination power. Thus we digitize them into three features as follows: (1) the length of string is zero (NULL), (2) the length of string is less than or equal to 10 bytes, and (3) the length of string is more than 11 bytes.

Prefix strings (16 elements)

In Japanese, attribute names often have a special prefix of the string like domestic year expression, day, time, place and organization expression. In English, these are usually expressed not as prefix but as suffix like “co.”, “-th” and so on. We select strings corresponding to Japanese prefix such as: “昭和 (era name □ )”, “特 (□ limited □)”

Suffix strings (45 elements)

By almost the same reason as prefix strings, attribute names often have a special suffix of the string. At this moment, we use 45 strings as suffix strings like: “位 (□ order □ )”, “□ 日 (□ date □ )”, “点 (□ point □ )”, “月 (□ month □ )”

Many of them indicate day, time, date, month, year, place like airport, station, organization, status like “sold”, and so on. Each of them has one feature.

Unit (16 elements)

The strings which indicate a certain unit are also good candidates of sign of attribute names. We use 16 strings as features like: “feet”, “inch.”
These include monetary unit, head count unit, time unit, length, width, volume and weight unit, and so on.

**Special characters** (13 elements)

Attribute names tend to be used with "〜" as time period, "(”, “)" as remarks and so on. Then we use 13 special characters.

**Kinds of character** (5 elements)

In Japanese we use the following five kinds of character sets: “hiragana”, “katakana”, “kanji”, “number” and “Latin alphabet.” Attribute name strings may consist of one or more character sets. On the contrary, strings expressing values usually consist of only one character set like numbers. Then, we use five features corresponding to the above five character sets respectively.

**Attribute of &lt;table&gt; tags** (2 elements)

Cell with colspan or rowspan option, next row to the cell with colspan or next column to the cell with rowspan tend to be attribute names. To take into account of this tendency, we use the following two cases as features:

1. Cell with colspan option or next row to the cell with colspan.
2. Cell with rowspan option or next column to the cell with rowspan.

We can distinguish cells into two classes: 1) related to spans and 2) not related to spans.

Of course, there is further possibility to find useful features for recognizing the boundary between attribute names and value data. Presumably it is one of our important future problem to enhance these features.

### 3.5 Recognition algorithm

We use a feature vector of each cell to calculate cosine of two vectors which represents the similarity of the corresponding two cells. Formally, we use representation, $\text{Cell}_{ij}$, as the cell of $i$-th row and $j$-th column where $1 \leq i \leq m$ and $1 \leq j \leq n$. Now we define the average of similarity: $\text{Sim}_{row}(i, j)$ between $\text{Cell}_{ij}$ and all other cells on the same column defined by the following equation where $\overline{\text{cell}_{ij}}$ means the feature vector of $\text{Cell}_{ij}$.

$$\text{Sim}_{row}(i, j) = \frac{1}{m-1} \sum_{k=1}^{m} \frac{\overline{\text{cell}_{ij} \cdot \text{cell}_{kj}}}{|\overline{\text{cell}_{ij}}||\overline{\text{cell}_{kj}}|} \quad (2)$$

In this equation, the range of summation $\sum$ takes $k = 1, \ldots, m$ except $k = i$. Also in this equation, $\overline{\text{cell}_{ij} \cdot \text{cell}_{kj}}$ means inner product between $\overline{\text{cell}_{ij}}$ and $\overline{\text{cell}_{kj}}$, $\frac{|\overline{\text{cell}_{ij}}|}{|\overline{\text{cell}_{kj}}|}$ stands for absolute value of $\overline{\text{cell}_{ij}}$ or $\overline{\text{cell}_{kj}}$ respectively. Therefore, the inside of the formula $\sum$ becomes a value of cosine between $\overline{\text{cell}_{ij}}$ and $\overline{\text{cell}_{kj}}$.

![Figure 3: Calculation of $\text{Sim}_{row}(m, 1)$](image)

We illustrate the methods of calculation about the first row, the second row and $m$-th row on the first column, as examples. In Figure 3, $\text{Cell}(1, 1)$ is the reference cell. We calculate values of cosine between the reference $\text{Cell}(1, 1)$ and each cell in the first column except $\text{Cell}(1, 1)$, then assign the average values of all of these cosines to similarity $\text{Sim}_{row}(1, 1)$. The reference cell will be changed to $\text{Cell}(2, 1)$. We make the average value of all of cosines between the reference $\text{Cell}(2, 1)$ and the other cells. The reference cell is moving to the adjacent lower cell repeatedly and calculation of cosine is done by the same manner until $\text{Cell}(m, 1)$ is used as the reference cell. Then we move to the next column, say $\text{Cell}(j, 2)$, $j = 1, 2, \ldots, m$ and do the same calculation of similarity, and so on till $n$-th column, say $\text{Cell}(j, n)$, $j = 1, 2, \ldots, m$.

$$\text{Sim}_{row}(i) = \frac{1}{n} \sum_{k=1}^{n} \text{Sim}_{row}(i, k) \quad (3)$$

Next, we make averages of similarities of each row using Formula (3) in order to determine the boundary between row of attribute names and
rows expressing values. Apparently, rows for value data probably have the same linguistic features and high similarity because they all express the value data of the specific attribute. On the contrary, rows for attribute names and rows for values are expected to share less linguistic features and have low similarities. In addition, the row(s) of attribute name will be at the top of the table in general. The rows of attribute names have low similarities, but the rows of attribute values remain high and almost the same similarity. To capture this distinction, we introduce \( R(i) \), which represents the ratio of similarities between \( \text{Sim}_{\text{row}}(i) \) and the average of \( \text{Sim}_{\text{row}}(i+1), \ldots, \text{Sim}_{\text{row}}(m) \), defined as Formula (4).

\[
R(i) = \frac{\text{Sim}_{\text{row}}(i)}{\frac{1}{m-i} \sum_{k=i+1}^{m} \text{Sim}_{\text{row}}(k)} \tag{4}
\]

Owing to this definition of \( R(i) \), if \( R(i) \) is low then the \( i \)-th row is regarded as attribute name, otherwise it is regarded as value. Based on this tendency of \( R(i) \), by Formula (4), the boundary between rows of attribute names and rows of its value data, which we call \( T \)-th row, is determined by using the following algorithm where \( \theta \) is the threshold value to decide whether there is the boundary between \( T \)-th row and \( T + 1 \)-th row.

Boundary searching algorithm

\[
\begin{align*}
T &= 0; \\
\text{for } (i=1; i<=m; i++) & \{ \\
& \quad \text{if } (R(i) < \theta) \{ T = i; \} \\
& \quad \text{else } \{ \text{break}; \} \\
& \} \\
\text{if } (T = 0) & \{ \text{There is no boundary between any rows.} \} \\
\text{else } & \{ \text{There are attribute names until } T \text{-th row.} \}
\end{align*}
\]

While this boundary searching algorithm is for identifying the boundary of rows between attribute names and its value data, if we exchange row and column in this algorithm and calculate \( \text{Sim}_{\text{col}}(j) \), the algorithm can search the boundary between attribute name column and its value data columns. Note that we have two threshold of \( \theta \) of this algorithm for boundary on rows and columns respectively. We decide these thresholds experimentally as described in the next section.

We decide which of three types the given table is by the following three criteria where \( T \) stands for the boundary identified by the algorithm described above.

1. If both row’s \( T \) and column’s \( T \) are more than 1 respectively, then the table type is a timetable.
2. If row’s \( T \) is more than 1 and column’s \( T \) is equal to 0, then the table type is a lengthways list.
3. If column’s \( T \) is more than 1 and row’s \( T \) is equal to 0, then the table type is a sideways list.
4. Otherwise, the table type is unknown.

3.6 Evaluation of table recognition algorithm

Now, we have to optimize the threshold \( \theta \) to detect the boundary between an attribute name and its value data based on the similarity ratio \( R(i) \) we mentioned 3.5. For this, we use 10 fold cross validation to fix the optimized threshold \( \theta \) and evaluate our algorithm.

Table 1: The result of lengthways.

<table>
<thead>
<tr>
<th>Kind of set</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>83.23%</td>
</tr>
<tr>
<td>Test set</td>
<td>82.11%</td>
</tr>
</tbody>
</table>

Table 2: The result of sideways.

<table>
<thead>
<tr>
<th>Kind of set</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>79.11%</td>
</tr>
<tr>
<td>Test set</td>
<td>78.11%</td>
</tr>
</tbody>
</table>

Table 3: The right boundaries of test data in 300 tables.

<table>
<thead>
<tr>
<th>Boundary</th>
<th>Row</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>70</td>
<td>183</td>
</tr>
<tr>
<td>1</td>
<td>202</td>
<td>115</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>300</td>
<td>300</td>
</tr>
</tbody>
</table>

We collected 2193 web pages with \(<\text{TABLE}>\) tag using Web search robot and selected genuine tables by hand. This time, the condition of genuine table is that it has ruled lines (BORDER attribute is more than 1) and the size is bigger than two times two. Then we extracted 300 genuine tables from them randomly. We identify the boundaries between attribute name rows (or columns)
and their value data rows (or columns) of 300 tables by hand. These tables are used as the training set and test set of 10 fold cross validation. We seek the optimized threshold $\theta$ by averaging through this cross validation. As the result, we found $0.90$ for rows $\theta$ and $0.70$ for columns.

We achieved $82\%$ accuracy of recognition of lengthways and $78\%$ accuracy of recognition of sideways by applying our algorithm. Our future problems are enhancing and optimize linguistic feature vector and applying machine learning methods.

Figure 4: Distribution of $R(i)$ of sideways.

Figure 5: Distribution of $R(i)$ of lengthways.

Table 1 shows the results of evaluation of lengthways. The result of sideways is shown in Table 2. The average of size of tables using the evaluation is 9.2 columns and 6.3 rows. Table 3 shows the number of each type and the breakdown of tables. Among of the result of Table 3, 66 tables become type of timetables. Looking more closely, 43 tables have the boundaries with the first row and the first column, 22 tables have the second row and the first column and only one table has the second row and the second column.

The distribution of the average of lengthways $R(i)$ for our evaluation is shown in Figure 4, and Figure 5 shows the average of sideways $R(i)$. In both Figure 4 and 5, $R(i)$ ($i = 1, 2, \cdots$) are the almost same when the tables have no boundary. On the other hand, when there is the boundary between the $K$-th row (or column) and the $K + 1$-th row (or column) $R(1), \cdots, R(K)$ are low and $R(K), \cdots$ are relatively high and almost the same. By our method with the optimized threshold $\theta$, our system could recognize the attribute names of tables with $82\%$ accuracy of lengthways and $78\%$ of sideways for Table 1 and 2.

Next, we evaluated which linguistic features contribute how much to improve accuracy. For this we exclude one of the following five sets of linguistic features and evaluate accuracy of each case: 1.kinds of character, 2.prefix and suffix strings, 3.punctuation marks and unit, 4.special characters, 5.length of cell data. Figure 6 shows the result of how much an accuracy is degraded when excluding each of the above mentioned sets features. We found that the most influential category is the length of cell data from the result. The second most influential category is prefix and suffix strings, the third one is punctuation marks and unit, and the fourth one is special characters.

Figure 6: The effects to accuracy of each category in the vector.

4 Conclusion

In this paper, we described the system which can recognize table attribute names and their value data in order to extract knowledge from HTML tables. We achieved $82\%$ accuracy of recognition of lengthways and $78\%$ accuracy of recognition of sideways by applying our algorithm.

Our future problems are enhancing and optimize linguistic feature vector and applying machine learning methods.
References


