A method to integrate tables of the World Wide Web

Minoru Yoshida[1] Kentaro Torisawa[2,3] Jun'ichi Tsujii[1,4]
mino@is.s.u-tokyo.ac.jp torisawa@jaist.ac.jp tsujii@is.s.u-tokyo.ac.jp

[1] Department of Computer Science, Graduate school of Information Science and Technology,
University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-0033, Japan
[2] School of Information Science, Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi, Ishikawa, 923-1292, Japan
[3] Information and Human Behavior, PRESTO, Japan Science and Technology Corporation
4-1-8 Kawaguchi Hon-cho, Kawaguchi-shi, Saitama, 332-0012, Japan
[4]CREST, JST(Japan Science and Technology Corporation)
4-1-8 Kawaguchi Hon-cho, Kawaguchi-shi, Saitama, 332-0012, Japan

Abstract

The World Wide Web (WWW) allows a person to access a great amount of data provided by a wide variety of entities. However, the content varies widely in expression. This makes it difficult to browse many pages effectively, even if the contents of the pages are quite similar. This study is the first step toward the reduction of such variety of WWW contents. The method proposed in this paper enables us to easily obtain information about similar objects scattered over the WWW. We focus on the tables contained in the WWW pages and propose a method to integrate them according to the category of objects presented in each table. The table integrated in a uniform format enables us to easily compare the objects of different locations and styles of expressions.

1 Introduction

The World Wide Web (WWW) allows a person to access a great amount of data provided by various entities. However, content varies widely in *expression* because the content provider uses their own particular style of communication and expression. This difference in styles makes it difficult to browse a large number of pages effectively, even if the contents of the pages are *similar*.

To reduce this vast variety inherent in the WWW, we focused on the tables contained in the WWW pages and developed a method to integrate them according to the *category* of *objects* represented in each table. Figure 1 shows an example of such an integration. In this case, a set of self-introduction tables and PC specification tables were clustered and merged independently. In spite of the wide variety of presentational forms of tables on the WWW, our method is able to align data of objects in the same category. Such an integration enables us to obtain a total perspective on the categories on the WWW and easily compare the objects of different locations and expressions.

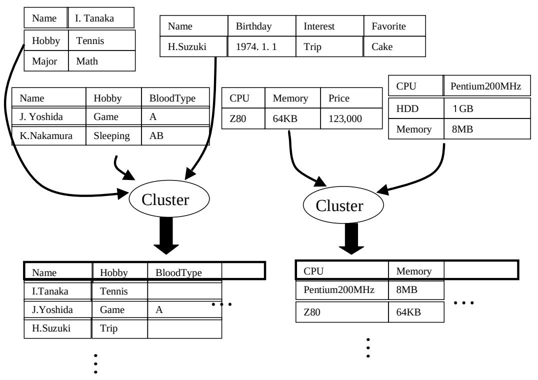


Figure 1. Sample Integration of Tables

Our method consists of the two processes: *Table Structure Recognition* and *Table Integration*.

We assume that a table represents one or more *objects*, each of which is described by a set of *attribute-value pairs*. For example, Table (b) in Figure 2 has the "Name" attribute and the "Hanako" value. We call the layout of attributes and values the *table structure*. In order to obtain a merged single table from tables having various layouts, we first recognize the table structures of the tables, since they are not given explicitly in web pages. We call this task *table structure recognition*. *Table integration* can be done according to the recognized table structures. In the following, we discuss some details of these subtasks.

Table Structure Recognition Table structure recognition detects which part of a given table is an attribute or a value. There has been some research on this task. Most stud-

Name	Tel.	Recommendation
Lake Restaurant	31-2456	special lunch
Cafe Bonne	12-3456	chicken curry
Metro Restaurant	56-7890	fried rice

(a)A table representing restaurants

Name	Hanako	BloodType	A
Gender	Female	Birthday	22 Feb.
Nationality	Japanese	Tel.	12-3456

(b)A table representing a person

John	Richard	Tom
Jude	Mary	Bill

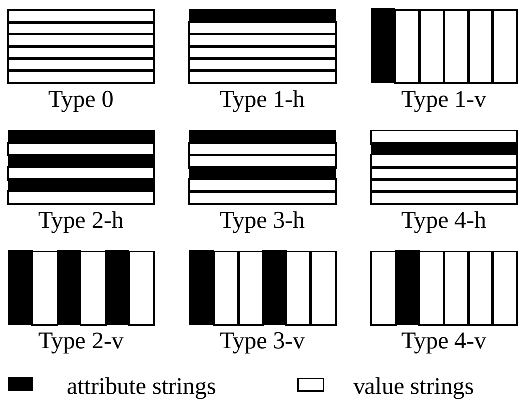
(c)A list of names

Figure 2. Sample Tables

ies have used surface features, such as the number of numeric words, the length of strings [4] [1], HTML tags [5], or visual cues such as the thickness of lines [3]. Although these methods achieved significant success, we view this task from a different perspective. Table interpretation by a human requires ontological knowledge. For instance, if a table describes people, it should have attributes such as "Name", "Birthday" and "Hobby". In addition, there should be strings which are likely to be values of these attributes. We think that these insights are based on generic ontological knowledge. Our approach is to recover a part of such ontological knowledge from tables about various objects in various formats, and to use it in table recognition. This is achieved by utilizing the Expectation Maximization algorithm [2]. The algorithm estimates the probabilities that a string appears as attributes (or values). Then, the algorithm determines a table structure according to the estimated probability distribution.

Table Integration After obtaining table structures for many tables, our procedure integrates tables in different formats into a single one. First, our algorithm determines which tables should be integrated. The tables to be integrated must be in the same category. Then, *table merging* is performed on each cluster. Table merging is a task to represent all tables by one large table. The main problem is that different attributes can be used with the same meaning, such as the attributes "Birthday" and "Day of birth" in self-introduction tables. To align such attributes we need a method for *attribute clustering*, which classifies attributes. Finally, we obtain the set of table clusters each of which is represented by a single large table describing objects in the same category, as shown in Figure 1.

Outline In Section 2 of this paper we describe our algorithm for table structure recognition. In Section 3 we provide our table-integration algorithm. In Section 4 we will describe the results of our experiments in which tables extracted from the WWW were used.



Suffix h: the attribute words are arranged horizontally.
Suffix v: the attribute words are arranged vertically.

Figure 3. Types

2 Table Structure Recognition

Definition We represent a table T as $(\langle s_1, s_2, ..., s_{xy}\rangle, x, y)$ where x is the number of columns in T and y is the number of rows in T. $\langle s_1, s_2, ..., s_{xy}\rangle$ is a sequence of strings appearing in the table.

We assume that each string in the table can be classified as an *attribute string* or a *value string*. The table structure denotes the positions of attributes and those of values. We represent this structure as a set of *labels*, which are assigned to each string of the table, where a label is a member of the *set of labels* $\{att, val\}$. The att label stands for $attributes$ and the val label stands for $values$.

More formally, the table structure is defined as the function whose argument is a table defined above and has a value of the corresponding sequence $\langle (s_1, l_1), (s_2, l_2), ..., (s_n, l_n)\rangle$ where l_i is the label that corresponds to the string s_i.

We also assume that table structures can be categorized into the nine *types* illustrated in Figure 3. We use types instead of table structures in the remaining of the paper for the sake of simplicity. When Table (a) in Figure 2 has type 1-h, att is assigned to the words "Name", "Tel." and "Recommendation", while val is assigned to "Lake Restaurant", "31-2456", etc. We can say that type 2-v is plausible for table (b) and type 0 for table (c).

Algorithm The algorithm chooses the most plausible sequence of types $\mathcal{M} = \langle m_1, ..., m_n\rangle$ for input sequence of tables $\mathcal{T} = \langle T_1, \cdots, T_n\rangle$, according to the estimated probabilities as follows.

$$\begin{aligned}\mathcal{M} &= \arg \max_{\mathcal{M}=\langle m_i\rangle_{i=1}^{n}} \prod_i P(m_i|T_i) \\ &= \arg \max_{\mathcal{M}=\langle m_i\rangle_{i=1}^{n}} \prod_i P(m_i, T_i)\end{aligned}$$

Then, we express the probability $P(m_i, T_i)$ with the following parameter set θ and denotes the probability by $P_\theta(m_i, T_i)$.

$$\begin{aligned}
\theta &= \{P_\theta(m|x,y)\} \cup \{P_\theta(s|l)\} \\
P_\theta(m,T) &= P_\theta(m, \{\langle s_i\rangle_{i=1}^n, x, y\}) \\
&= P(x,y)P_\theta(m|x,y)P_\theta(\langle s_i\rangle_{i=1}^n | m, x, y) \\
&\approx P(x,y)P_\theta(m|x,y)P_\theta(\langle s_i\rangle_{i=1}^n | m) \\
&\approx P(x,y)P_\theta(m|x,y) \prod_{(s,l)\in m(T)} P_\theta(s|l)
\end{aligned}$$

In the last transformation, we made an approximation such that $P_\theta(\langle s_i\rangle_{i=1}^n|m)$ is the product of $P(s|l)$ for all the pairs (s,l) in $m(T)$ where s is a string and l is a label. Intuitively, $P(s|l)$ denotes the probability of s appearing in the position labeled with l, and $P(m|x,y)$ denotes the probability that tables which is the size of (x,y) have the type m.

The EM algorithm improves the value of $\log \prod_i P(m_i, T_i)$ by repeatedly adjusting the parameter set θ according to the following formulae.

$$P_\theta(m|x,y) = \frac{1}{|\mathcal{T}_{xy}|} \sum_{T\in\mathcal{T}_{xy}} P_{\theta'}(m|T)$$

$$P_\theta(s|l) = \frac{1}{Z_{\theta'}(l)} \cdot \sum_{i,k} \sum_{k(m(T_i))=(s,l)} P_{\theta'}(m|T_i)$$

Here $k(m(T))$ means the kth element of the ordered set $m(T)$ and $\sum_{k(m(T_i))=(s,l)}$ means the summation over all possible values of m such that the kth element of $m(T)$ is (s,l). θ' is an old parameter set and $Z_{\theta'}(l)$ is the normalizing factor such that $\sum_s P_{\theta'}(s|l) = 1$. $\mathcal{T}_{xy}$ is a sequence of tables with the size of (x,y).

3 Table Integration

As stated in the introduction, table integration is divided into two processes, table clustering and table merging. This section discusses these processes in detail.

Table Clustering We define the concept of *unique attributes* for clustering tables. The unique attributes are attributes *peculiar* to certain classes of objects (or tables describing certain classes of objects.) For instance, the attribute “Hobby” is peculiar to a table about self-introduction and the attribute CPU is peculiar to catalogues of computers. We express the degree of peculiarity of an attribute a by the function $uniq(a)$ defined below. $U(a)$ is a set of tables in which a appears. $V(a)$ is the set of attributes appearing in $U(a)$. $Freq(b,a)$ is the frequency of attribute b in the tables in $U(a)$.

$$uniq(a) =_{def} cooc(a) \cdot excl(a)$$

where

$$cooc(a) =_{def} \frac{1}{|V(a)|} \sum_{b\in V(a)} Freq(b,a)$$

$$excl(a) =_{def} \frac{1}{|V(a)|} \sum_{b\in V(a)} \frac{Freq(b,a)}{|U(b)|}$$

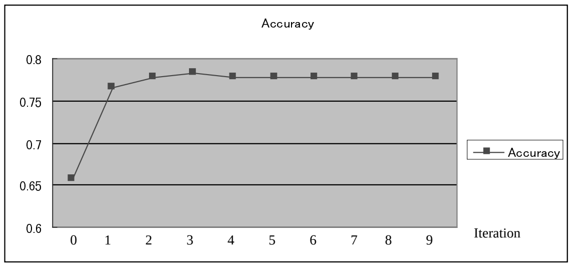


Figure 4. Improvement of Accuracy with Iteration

Intuitively, if $uniq(a)$ is large, $U(a)$ is likely to be a set of tables describing similar objects. $cooc(a)$ expresses how consistently the attributes in $V(a)$ co-occur with a. On the other hand, $excl(a)$ represents the degree of exclusiveness of the attributes in $V(a)$. The algorithm selects an attribute a with a large value of $uniq(a)$ and takes $U(a)$ as a cluster.

Table Merging Next, we explain our table merging algorithm. Table merging is realized by performing *attribute clustering*, which classifies similar attributes into the same cluster. In the resulting tables, attributes in the same cluster are aligned as the same attribute. The similarity between attributes a and b is calculated using a simple cosine measure where each attribute is represented by a vector whose element is the frequency of each value appearing with that attribute.

4 Experiments

Table Structure Recognition We applied our algorithm to S, a set of HTML tables gathered from the WWW. S contained 35232 tables. Most of these tables were in Japanese.

After the parameters were estimated from all over S, we estimated the accuracy with which types were assigned to tables by randomly selecting 175 tables from S and applying the algorithm to them. Note that, therefore, accuracy was evaluated in a closed test. The degree of accuracy was calculated as $n/175$ where n is the number of tables to which correct types were assigned. Of these 175 tables, the number of tables with type 0, 1-h, 1-v, and others, were 76, 61, 35, and 3, respectively.

Figure 4 shows the accuracy on each iteration. The accuracy increased from 0.66 to 0.78. Note that this accuracy is significant if we consider the distribution of types.

Next we compared the performance of our algorithm with that of [1]. They reported that their algorithm filtered out non-tables (i.e., Type 0 tables) with a precision of 92.92% and a recall of 80.07% . According to their criteria, the precision of our result was 79.44% and the recall was 85.86%.

Although a precise comparison is not possible, it would at first appear that our method does not perform as well as

Cluster	Categories	# of Tables
SHUMI (Hobby)	Person(10)	811
NAMAE (Person's Name)	Person(10)	668
JUUSHO (Address)	Building(10)	271
SHOZAITI (Address)	Building(7), Noted Place(3)	243
SIMEI (First/Last Name)	Person(8), Univ.-Course(1), Record(1)	425
JUN'I (Rank)	Ranking(10)	351
TAITORU (Title)	Game(5), Movie(3), Song(2)	623
NAIYOU (Contents)	TV-Program(7), Lecture(1), Rugby Game(1), Schedule(1)	292
NAME	Person(10)	159
NITIJI (Date/Time)	Schedule(8), Record(2)	253
GAPPI (Date)	Schedule(5), Record(5)	149
ZENCHO (Length)	Car(6), Machine(2), Pig(1), Fishing Rod(1)	41
CPU	PC(10)	95
NAMAE (Person's Name)	Person(10)	25
SIHONKIN (Capital Money)	Company(10)	40

Table 1. Result of Table Clustering

that of Chen et al. However, their experiments were performed on a set of tables selected from airline information pages. We therefore believe that these tables were more suitable for their method because these tables most likely contain numbers. We therefore expect that our method can outperform Chen's if the performances are evaluated on a set containing a greater variety of tables.

Table Integration Table clustering was evaluated on a set of 44691 tables, which is larger than S. We selected clusters with the top 15 values of $|V(a)|$, assuming that if $|V(a)|$ is large it means that tables in that cluster came from various kinds of locations and were therefore appropriate for evaluation. Thus, we can avoid choosing a cluster that contains many similar tables written by only one or just a few persons. We investigated the clustering result by checking 10 randomly selected objects in each cluster. Table 1 shows the resulting clusters denoted by their unique attributes. "Categories" column contains the categories of objects represented by the tables in the clusters. We observed that some attributes, such as "Hobby", "CPU", and "Capital Money", which are we think appropriate as unique attributes, were properly used for clustering. On the other hand, some attributes such as "Contents" and "Title" are ambiguous because it is hard to express the category of objects in the cluster by one word (such as "People" in the "Hobby" cluster.) Therefore, there is still room for improvement of clustering result by adjusting the definition of the $uniq$ function or relying on other clustering techniques.

Cluster	Precision	Recall	F-measure
SHUMI(Hobby)	0.98	0.79	0.87
CPU	0.90	0.82	0.86
SIHONKIN(Capital)	0.94	0.77	0.85

Table 2. Result of Table Merging

Next, we show the performance of table merging. The algorithm selects the top seven highest occurring attributes to present merged tables. We call them the *main attributes* of each cluster and use them for evaluation. For example, the main attributes in the "Hobby" cluster were Name, Birthday, Address, Bloodtype, Job, Hobby, and Favorite foods. We evaluated table merging for three clusters whose unique attributes were "Hobby", "CPU", and "Capital Money." For each cluster we selected 10 objects randomly and checked if their values in the original table appeared in the merging result (recall), and if their values in the merging result were correct (precision), for each main attribute in the cluster. The result is shown in Table 2. Although these experiments were done on a rather small set of tables, we can see that the merging was done properly to some extent.

5 Conclusion

We described an algorithm to integrate the tables describing similar objects. Our algorithm integrates tables by using the automatically recognized table structures on the basis of probabilistic models where parameters are estimated by the EM algorithm. Our algorithm achieved an accuracy of 78%. We also demonstrated that our algorithm automatically integrated tables of self-introduction or PC-catalogue.

References

[1] H. Chen, S. Tsai, and J. Tsai. Mining tables from large scale HTML texts. *18th International Conference on Computational Linguistics (COLING)*, pages 166–172, 2000.

[2] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Soc., B*, 39:1–38, 1977.

[3] E. Green and M. Krishnamoorthy. Model-based analysis of printed tables. *In Proceedings of International Conference on Document Analysis and Recognition (ICDAR)*, pages 214–217, 1995.

[4] M. Hurst and S. Douglas. Layout and language: Preliminary investigations in recognizing the structure of tables. *Fourth International Conference on Document Analysis and Recognition (ICDAR)*, pages 1043–1047, 1997.

[5] F. Itoh, N. Otani, T. Ueda, and Y. Ikeda. Two-way navigation system for the information space and the real space using extraction and integration of attribute ontologies. *Journal of the Japanese Society for Artificial Intelligence*, 14 (In Japanese):1001–1009, 1999.