中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-06-005

A Semantic Approach to Internet Tabular Information Extraction

Shih-Hung Wu, Huei-Long Wang, I-Chi Wang, Cheng-Lung Sung,
Wei-Kuan Shih and Wen-Lian Hsu*

May 23, 2006 || Technical Report No. TR-IIS-06-005

http://www.iis.sinica.edu.tw/LIB/TechReport/tr2006/tr06.html

A Semantic Approach to Internet Tabular Information Extraction

（A short version appeared in CIKM 2004）

Shih-Hung Wu [a], Huei-Long Wang[c], I-Chi Wang[b], Cheng-Lung Sung[c], Wei-Kuan Shih[c] and Wen-Lian Hsu[bc*]

[a] *Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung County, Taiwan, R.O.C*

[b] *Institute of Information Science, Academia Sinica. Taipei, Taiwan, R.O.C.*

[c] *Department of Computer Science, National Tsing Hua University Hsin-Chu, Taiwan, R.O.C.*

***Abstract*—**Extracting information from tables is essential for Internet information extraction. However, most web tables are designed in HTML format. To decipher their semantic meanings a system needs to deal with various layouts, which is quite cumbersome. Previous works have two major approaches: layout enumeration approach and wrapper approach. The first approach is to match the table with presorted layout. This approach normally does not perform appropriate information integration since it does not use table semantics. The second approach treats different tables in a case-by-case manner laboriously. We present a semantic approach to automatically recognize tables (in specified knowledge domains) with various layouts. Our system first tags table cells using domain knowledge, solve the multiple tagging ambiguities, and then apply layout-syntax rules to transform tables into database format. Experimental results show that the precision and recall are higher than 93% and 82% respectively in four different domains. Our approach has high precision and is suitable as a front end for wrapper construction.

Keywords: Table Understanding; Data Integration; Tabular information extraction, Domain Knowledge, Table layout syntax

---

[*] Corresponding Author
Tel: 886-2-27883799ext.1804
Fax: 886-2-27824814
E-mail: *hsu@iis.sinica.edu.tw* (W.L. Hsu), *shwu@cyut.edu.tw* (S.H.Wu)

1. INTRODUCTION

Tables are a concise way to represent information; massive useful information in table is available on the Web. Table layouts might vary widely, it is hard for machine to understand the logical meaning of a table [20, 27]. Furthermore, to integrate information from different sources, domain knowledge is necessary [3, 4, 6, 36]. A recent survey of table recognition from Zanibbi et al. describes that table recognition is an interaction of table models, observations, transformations and inferences [37]. However, the core problem is to recognize the table logic, i.e., the relationships of table cells.

Previous works for table recognition have two major approaches, one is document understanding with layout enumeration approach for tables in general documents [5, 11, 21-24, 33], another is a wrapper approach for Internet documents [1, 2, 7, 10, 12-15, 25, 26, 30]. The two approaches have different assumptions and achieve different level of understanding.

Layout enumeration approach focuses on finding the right layout of a table with least human intervention. Systems have predefined knowledge of all possible layouts but no domain knowledge of table content. Wang analyzes characteristic information on tabular layouts [33]. Douglas and Hurst present an approach to enumerate all possible layouts to recognize and interpret tables in plain text. They also presents the cell cohesion function and area templates to distinguish labels and entries [5, 21-24]. This approach can recognize most of the tables but fails on special cases, where tables contain unexpected information. Thus, human intervention is still necessary. Supervised machine learning technique can be applied once there are enough training set [29].

On the other hand, wrapper approach relies on human editors for each table and treat different tables in a case-by-case manner through manual tagging, even though the information content of the table is within the same domain. The advantage of wrapper construction is that it works for any given table and can have some tolerance of table layout with the help of machine learning technique [10, 28, 29]. However, each table must be processed first by a human editor. A method with less human intervention is desirable.

The previous works focus on extracting a table's logical structure and do not emphasize on the semantic relation of the table cells. Without the semantics, tables might not be recognized correctly and user might not be able to query them easily. Transferring the table into database is a practical way for further information integration. However, number of fields, names of fields, and types of fields are necessary to construct a database table. Therefore, a controlled vocabulary and relation between concepts are necessary. In our previous work [32], we proposed a way to recognize the table with the help of semantic relations of cells. We find that the concept-instance relation of cells could help to recognize the table in various layouts. Embley et al. also suggest a way that can recognize HTML tables to a given target database schema [8].

A practical approach of table recognition should combine the fully automatic approach with manual approach. For example, if one wants to collect timetable information for all hospitals in an area, the number of tables to wrap is enormous and manual tagging is almost impossible. However, with pre-designed hospital-timetable domain knowledge, our approach can automatically transform these tables into database correctly. Only a little effort is required to resolve the ambiguities in the remaining tables (less than 5% of total) by human editors. Thus, with our system can

serve as a front end tagging system for wrapper construction, we can get a practical system.

In this paper, we propose a method that can reuse the knowledge of human editor, such that the system can deal with tables in the same knowledge domain without respect to various layouts. The paper is organized as follows. In section 2, we describe the problem definition and analyze the problem. In section 3, we present the system that we implement base on the analysis. In section 4, we report our results of experiments. In the last section, we make conclusions and list some future works.

2. PROBLEM DEFINITION AND ANALYSIS

The table recognition problem can be defined as: given an HTML table *T* and the **domain knowledge** of the information to be extracted, the system outputs a database table *D*, containing the information extracted from *T*.

In the following subsections, we analyze the difficulties of this problem. To solve the problem, we first describe the table we want deal with. Next, we analyze the necessary knowledge. Then, we define the model of table syntax. Finally, we define the table transformation as a result of table recognition.

2.1. General tables and HTML tables

Due to presentation space, the size of a HTML table is often quite limited. Since tables are usually used to arrange or compare a lot of data, the length of the string in a cell is often short. We assume that strings in tables are the representation of concepts and their instances. Table headers often represent concepts, and table body entries represent instances of the concepts. However, since nested headers are very usual, header and the nested header are often concept-instance pair too. In Figure 1(a) Column Header located at the first row and the Nested Column Headers at the second row do have concept-instance relation. That is. Ass1, Ass2, and Ass3 are instances of

the concept “Assignment”, and the data are instances of the concepts of Ass1, Ass2, and Ass3. Figure 1 (a) shows a general table anatomy, while Figure 1 (b) shows the same table in HTML style.

Since there is no specific tags in HTML to indicate the stub part. To recognize which cells of an HTML table belonging to header or body require more semantic knowledge. Since a string in a cell is a representation of a concept or an instance of certain concept, a system can recognize the string through template matching, and then tag the string with the concept. However, due to the ambiguity of natural language, sometimes multiple tags for a string is quite common. Domain knowledge of the semantic relations is necessary to disambiguate the multiple tags. It should be noted that even if the table is given in XML format, extracting cell information could be still very difficult since the semantic relations among the XML tags are not necessarily specified.

2.2. Knowledge to understand a table

There is no any generic algorithm that can understand all kinds of tables. To decipher the meaning of a table, a system requires the knowledge of table layout and domain knowledge. Some tables cannot be understood without domain knowledge. For example, consider the train timetable in Figure 2. Human can use the concept *train number* and knowledge of train number to access the information. That is, read the second row and can get the *train code* (CHU KUANG), *departure station* (TAI-TUNG NEW), and *destination station* (KAO-HSIUNG). However, there is no explicit clue for a machine to do the same thing. The concepts of *train code, departure station* and *destination station* are hidden concepts of the table. Human with common sense can understand this table, but machines fail to recognize such hidden concepts. Domain knowledge must be brought into the system before

machines can read such table.

We investigate three kinds of knowledge that a system requires to understanding tables. 1. Basic table layout knowledge, 2 Knowledge to distinguish concept and instance, and 3 Knowledge to distinguish different layouts. Systems with different kinds of knowledge achieve different kinds of understanding.

Level-1: Basic table layout knowledge is supported. With the level-1 knowledge, a system can recognize the position of each text string, and than separate label and data. Traditional table recognizing systems understand a table in this way.

Level-2: Knowledge to distinguish concept and instance is supported. With level-2 knowledge, system can identify the semantics in each cell. Wrapper approach uses human editors to achieve this level of knowledge.

Level-3: Knowledge to distinguish different layouts is supported. With level-3 knowledge, we can determine which algorithm should be applied to different layouts. Our idea is to denote the possible layouts by **layout syntax grammar** in different table domains and use these denotations to do template matching. The matched templates will be used to determine the semantics of cells' content. After that, a **semantic preserving transition** will be applied to transform the layout. Finally, a database form is acquired. In this paper, we test tables in 4 domains with level-3 knowledge.

2.3. Table Classification

Previous works on table recognition focused on extracting a table's logical structure, however, this is insufficient to extract the semantic meaning of the table. We classify tables based on both semantics and layout knowledge.

In the layout aspect, we propose three table classes: 1-dimensional tables,

2-dimensional tables, and complex tables. The first two classes are used more often than the complex tables. Herein, table cells are divided into labels and entries according to semantic identification.

1-Dimensional Tables:

1-dimension tables have one row of labels over several rows of entries. In this paper we have a stricter definition. The relations between labels and entries should be concepts and instances in the same column.

The tables in a relational database are 1-dimensional. Entries in different columns represent instances of different concepts. The label of each column identifies the categories of concepts by which we access entries in the column. Figure 3(a) illustrates the style of a normal 1-dimensional table and Figure 3(b) provides an 1-dimensional table example. The notation of the abstract table will be described in section 2.4.

2-Dimensional Tables:

2-dimensional tables have a rectangular area of similar entries. Each entry in this rectangular area represents an instance of the same concept. One or several rows of labels are above the rectangular area. These labels are instances of other concepts. Instances in different rows belong to different concepts. The left-hand side of the rectangular area could have one or several columns of labels. These labels are also instances of different concepts. Instances in the same column belong to the same concept. Instances in different columns belong to different concepts. At times, table designers did not include a label concept either because it is intrinsically understood, or there is no common way to define it. Figures 4(a) & (b) show the styles of 2-dimensional tables, and Figures 4(c) & (d) provide examples.

Complex Tables:

Complex tables are variants of 1-dimension and 2-dimension tables and can have

various features. For each feature, a corresponding heuristics can be designed. There is no way to list all the features, since the table constructor might have different reason to design a special table. Here, according to our observation, we list seven common features.

a. Partition label: Special labels appear in the data entries area making several partitions of the data entries. Each partition shares the same labels at the top of the table. For example, Figure 5(a) illustrates this situation with an example in Figure 8. Semantic identification of cells in these tables is complicated, and concept-instance relation cannot be obtained in neighboring cells.

b. Over-expanded label: In Figure 5(b), each entry I_X spans two or three I_Y under the same label C_X. For example, in Figure 7, each "Fixed Deposit" spans six rows and "Regular Fixed Deposit" spans three rows under the same label "Items & Periods".

c. Combination: Some large tables are a combination of several similar smaller tables. In Figure 5(c), four small tables merge into a larger one. In Figure 6, two tables merge into a larger one.

d. Multiple items in a cell: Some cells may contain two or more items that can be recognized as instances of the same or different concepts. In Figure 5(d), each entry has two instances in the second column. In Figure 6, many cells have two different items: doctor name and doctor ID number[1]. The relations between items in the same cell are referred to as inner cell relations [21, 22].

e. Vertical writing: The sequence of text is presented vertically in a cell, which commonly occurs in Chinese web pages. For example, in Figure 5(e), the words "ABC" and "DEF" are written vertically in a cell. However, the original sequence in

[1] Multiple items can occasionally be represented as one item if the distribution in every cell is regular. For example, the doctor name and ID is an example of the concept: 'doctor name & id'.

the HTML source is “ADBECF”.

f. Forward reduction: Comparing Figure 5(f) and Figure 5(g) reveals two empty cells in Figure 5(f), which are omissions. It is an intentional omission that is named as forward reduction.

g. Footnotes: Some tables have one or more cells containing special information with descriptions in footnotes. These footnotes often produce many tags, which will create noises when selecting a correct tag for cells in the same columns and the same rows.

2.4. Layout Syntax

In order to represent the complex tables, we design a **layout syntax grammar** that can represent the semantic relations of all the HTML tables. Our layout syntax consists of two kinds of symbols. They are used as area notations and concatenation operators.

Semantic Cell Notation:

$\mathbf{C_X}$ denotes a string that represents concept X in a cell.

$\mathbf{I_X}$ denotes a string that represents an instance of concept X in a cell.

? is a don’t-care symbol, it can be any kinds of cell.

ə denotes an empty cell.

Empty cells can be also matched by the above denotations.

Area Notation:

^{*c} denotes a repetition in a column. $\mathbf{I^{*c}_X}$ represents Figure 9(a).

^{*r} denotes a repetition in a row. $\mathbf{I^{*r}_X}$ represents Figure 9(b).

*,* denotes a repetition in a rectangle area. $\mathbf{I^{*,*}_X}$ represents Figure 9(c). The first * is the row size and the second * is the column size.

All of the above * can be replaced by numbers, variables or expressions to denote

a deterministic size of repetition. For example, $\mathbf{I^{3r}_X}$ represents 3 concatenated cells of $\mathbf{I_X}$ in the same row.

Concatenation Operator:

| is an operator to concatenate two layouts from top to bottom. For example, $\mathbf{C_X|I^{*c}_X}$ represents Figure 9(d).

- is an operator to concatenate two layouts from left to right. $\mathbf{C_X\text{-}I^{*r}_X}$ represents Figure 9(f).

Π represents a repetition of left-to-right concatenation. In Figure 9(e), $\Pi^n_{i=1}(C_{Xi}|I^{*c}_{Xi}) = (C_{X1}|I^{*c}_{X1})\text{-}\ldots\text{-}(C_{Xn}|I^{*c}_{Xn})$ and $I^{nr}_X = \Pi^n_{i=1}I_{Xi.}$

Σ represents a repetition of top-to-bottom concatenation. In Figure. 9(g), $\Sigma^n_{i=1}(C_{Xi}\text{-}I^{*r}_{Xi}) = (C_{X1}\text{-}I^{*c}_{X1})|\ldots|(C_{Xn}\text{-}I^{*c}_{Xn})$ and $I^{nc}_X = \Sigma^n_{i=1}I_{Xi.}$

2.5. The Representation Lemma

The **layout syntax grammar** is induced from the complex tables, and is general enough to represent all the tables we observed. The following lemma gives a theoretical description on how general it is.

Definition: A **monotonic area** is an area of cells in the same concept. Monotonic areas are $\mathbf{C_X}$, $\mathbf{I_X}$, $\mathbf{C^{*r}_X}$, $\mathbf{I^{*r}_X}$, $\mathbf{C^{*c}_X}$, $\mathbf{I^{*c}_X}$, $\mathbf{C^{*,*}_X}$, and $\mathbf{I^{*,*}_X}$. See figure 9 for details.

Definition: A **binary-partition** is to divide a table into two partitions, vertically or horizontally, without breaking any monotonic area.

Representation Lemma: If a table can be recursively binary-partitioned until each partition is a monotonic area, then the table can be represented by layout syntax grammar.

Proof. The lemma can be proofed by mathematical induction. First, for any monotonic area, it can be represented by the layout syntax grammar. Then, for any

table partition **P**, a binary partition produces two partitions $\mathbf{P_X}$ and $\mathbf{P_Y}$. We assume that each of $\mathbf{P_X}$ and $\mathbf{P_Y}$ can be represented by the layout syntax grammar $\mathbf{G(P_X)}$ and $\mathbf{G(P_Y)}$ respectively. Since $\mathbf{P_X}$ and $\mathbf{P_Y}$ are produced by a binary partition, they are either concatenated from top to bottom or from left to right. Hence, $\mathbf{G(P)} = \mathbf{P_X}|\mathbf{P_Y}$ if they are concatenated from top to bottom, $\mathbf{G(P)} = \mathbf{P_X}\text{-}\mathbf{P_Y}$ if they are concatenated from left to right. Since $\mathbf{P_X}$ and $\mathbf{P_Y}$ can be further binary partition into smaller partition until the partition is a monotonic area, and the monotonic area can be represented by the layout syntax grammar. The whole table can be represented by layout syntax grammar.

2.6. Semantics Preserving Transition

The semantic preserving transition maps from layout to layout while preserving relations between cells. We find that the transition from 1-dimensional or 2-dimensional tables into a standard database can be automatic. However, the complex tables ins section 2.3 must be divided into 1-dimensional or 2-dimensional tables before the transition can take place.

We define an operator Γ to denote a set of layouts. For example, $\Gamma^{n}_{i=1}(I_{Xi}\text{-}I^{*r}_{Y})$ is a set of tables in the form of $I_{Xi}\text{-}I^{*r}_{Y}$. For 1-dimensional and 2-dimensional tables, the transformations are in Table 1.

Consider the table in Figure 4(c), with the semantics in domain knowledge it is recognize as a 2-dimensional table as the abstract table in Figure 10 (a). The abstract table can then be transformed into the database format in Figure 10 (b) by the transformation rules in Table 1. In the database format table, every row is a reading path of the table in Figure 4(c).

2.7. Semantic Support Database Queries

The database form of Figure 4(c) is shown in Figure 10(b). In a traditional database table, the first row is field names and the following rows are field values. In

order to support semantic queries, we define the semantic database table to be a restricted database table in which its field values must be instance values of the same concept and their field names must be the names of concepts. Therefore, the query constraints listed in the WHERE closure of SQL queries are composed by field names and field values. This would pave the way to transform a natural language queries into SQL queries, where field names and field values in query constraints are concepts and instances.

3. METHODOLOGY

According to the analysis in section 2, finding the relation between cells is an important step of table recognition. In this section, we describe how a system can tag the cells and find the relations among the cells based on manually edited domain knowledge. A brief review of our ontological knowledge representation framework, INFOMAP, is in section 3.2.

3.1. Table Recognition Process

A four-step process is designed for a system to recognize tables in a given domain. 1. Normalize the HTML tables into virtual tables, 2. Tag each cell with semantic tags, 3. Resolve the ambiguity, and 4. Restructure a tagged table into a semantic database table. Figure 11 shows the processes.

Normalization of the HTML tables

The first step is to normalize the HTML table because the source files might contain excess information. A virtual table is a two dimensional array with the same number of columns and rows as the original HTML table. A virtual cell contains the tuple (index of column, index of row, size of column span, size of row span, content). The content is the original text string in the HTML cell. The information in virtual cells is necessary for the following processing.

Our system ignore the function of <th> and <td> HTML tags since their original functions are often ignored by table creators. Our system expands the HTML spanning cells, where the attributes “rowspan” or “colspan” are used. That is, a spanning cell will be normalized into cells with the same content.

Tagging cells

The core of the tagging process is template matching. The template matching technique is to tag the string in each cell with the concepts or instances in domain knowledge.

A tag contains the tuple *(type, concept index, location)*. The value of the tag *type* can be concept or instance. The *concept index* is the semantic index on INFOMAP. The *location* is the location of the substring that matches the representation of the information described by *type* and *concept index*, both the starting and ending point of the substring is recorded.

Our system can use the concept index to compare the relation between tags. If one concept index is on a sub-tree of another, we say that the former concept is a descent concept of the latter.

Ambiguity resolving process

When there is more than one tag in a cell, our system applies an ambiguity resolution process and an omission refill process based on heuristics.

To disambiguate tags in a cell, our system applies two heuristics, **longer-first heuristic** and **left-first heuristic**, to eliminate ambiguity by comparing the string length and position. When there are two or more tags exactly overlapped, we need to use a **neighbor-reference heuristic**. Our system estimates the likelihood of these overlapped tags to the tags of the neighboring cells, and then selects the tag with the best correlation. This is based on the analysis that related data usually arranged in the

same row or column. A detailed algorithm is presented in section 3.3.

Restructuring of a tagged table into a database table

After the process of recognizing information in a string through template matching, the system can recognize the table layout and transfer the table into a semantic database. The system recognizes the layout according to the layout syntax grammar. In section 2.6, we present layout transition rules to convert generic table layout to semantic database table layout.

3.2. Domain Knowledge Editing

Ontological domain knowledge of a table domain contains the concepts and relationships of the concepts. Templates of various wording are also included. For each domain, we construct the domain knowledge by our ontological knowledge representation framework, INFOMAP [19], which is a concept-modeled ontology [7] that can use various semantic template to representation domain concepts. We develop it from the result of our previous works [16-18]. Our knowledge map has a tree structure where each information node represents a concept and the relationship between two concepts is labeled by a relation node, such as hyponym, synonym or attribute. Each relation node contains a name that defines the relation between its parent information node and its children information nodes. Figure 12(b) gives an example of domain knowledge in INFOMAP that are used to recognize the table in Figure 12(a). In this domain, there are four important concepts: Flight, Depart Info, Arrive Info, and Aircraft. The target content is the Flight, that is, the number of rows of the database is depends on the number of Flight cells.

The concept-instance relation can be defined on the tree structure. An information node is treated as a **concept**. Information nodes are treated as **instances** of another concept if they are within a sub-tree connected to it's the given concept and the relation nodes are "hyponym". Information nodes are treated as **synonyms** if they are within a sub-tree connected to its parent information node by a relation node called as "*synonym*".

The INFOMAP template syntax can represent various wordings of a concept. Table 2 shows some examples of templates. A template can be denoted as a sequence of **template unit groups** separated by colons.

A **template unit** can be a string of keywords quoted by two quotation marks. For example, in Figure 12, there are two instances of "City". Each "City" is a template unit.

A template unit can also be a **selection** of a set of template units. We use braces to enclose all the template units, double quotation to enclose a string, and comma to separate two template units. For example, {"Mr.", "Dr."} is a selection of two template units "Mr." and "Dr.". A template unit group can be repeated.

A template unit group is **repeated** when we append two numbers separated by two dots and enclosed by parentheses. The first number denotes the minimum number of repetitions and the second number denotes the maximum number of repetitions. For example, "0"(3..5) is a template unit group that repeats the template unit "0" at least 3 times and at most 5 times. It can be "000", "0000", or "00000".

A template unit can also be a substitution by any representation of instances of a concept. Such template units are denoted by the concept reference quoted by brackets. For example, '[Airline Timetable, Departure, City]'[2] is a substitution of the sub-tree defined for the instances of the concept *Departure-City* in the *Airline Timetable* domain. Such template units are called **reference template units**.

3.3. Tag Selection

The meaning of the string in a cell sometimes cannot be clearly identified without referring to neighboring cells. For example, consider the airline timetable in Figure 12(a), the meaning of "City" in this table is not clear, because our system does

[2] We call it a concept pathname because we can look up the real location through the tree structure one by one. In the map of "Airline Timetable" there is only one node named as "Departure". There is also only one node named as "City" in the sub-tree of "Departure".

not know whether it describes an arrival or departure city. The tagging system can create two tags for “City” in concepts of “Arrival City” and “Departure City”. In order to resolve the ambiguity, referring to other cells is necessary.

For the second row, “TPE” and “YVR” have two different meanings. The right and the left cells cannot help resolving the ambiguity. However, if one observes the cells above, the “Departing” has a greater correlation to “Departure City” than to “Arrival City”.

In our tag selection process, our system computes the correlation degree of a tag to other tags vertically and horizontally. Based on the assumption that table designer tend to put related information in the same row or column [31, 34].

The relation scoring function, *relation-score(T_1, T_2)*, computes the score of the relation between two tags T_1 and T_2. The score values of all relations are defined as follows:

(1) s_1: concept - instance relation.

(2) s_2: concept - descent concept relation.

(3) s_3: concept - descent instance relation.

(4) s_4: concept - same concept.

(5) s_5: instance - same instance.

(6) s_6: otherwise.

Given a tag T, its total vertical relation scoring function is defined as:

v-score (T) := $\sum_i$ *relation-score (T, T_i), $\forall T_i \in$ tags in cells in the same column with T.*

Its total horizontal relation scoring function is defined as:

v-score (T) := $\sum_j$ *relation-score (T, T_j), $\forall T_j \in$ tags in cells in the same column with T.*

Its total relation scoring function is defined as:

*score (T) := w_h * h-score (T) + w_v * v-score (T), where w_h and w_v are weights for the importance of a relationship in different directions.*

For example, the "City" in cell (1, 1)[3] is tagged as concept T_1: "Departure City" and concept T_2:"Arrival City". Therefore, the values of s_1 to s_5 are 1, and 0 for s_6, and 1 for w_h and w_v. The total relation score of T_1 is

*score (T_1) = 1 * (0+0+1+0+0) + 1 * (1+1+1+1+1+1+1+1) = 9,*

and the total relation score of T_2 is

*score (T_2) = 1 * (0+0+1+0+0) + 1 * (0+1+1+1+1+1+1+1) = 8.*

Since *score (T_1) > score (T_2),* the "City" in cell (1, 1) should be tagged as the concept "Departure City".

3.4. An Illustrative example

Figure 14(a) shows a hospital outpatient timetable and the corresponding domain knowledge is shown in Figure 15. Figure 14(b) shows the concept tagging result of each cell. In this table, cells are grouped into four monotonic areas as shown in the table of Figure 14(c). Applying the transformation mechanism for 2-dimension tables, our system rearranges the cells in the form of Figure 14(d). Finally the table in Figure 14(a) is transformed into the database format table in Figure 14(e), where each row is a target content.

4. EXPERIMENTATIONS[4]

We conducted four experiments on four independent domains: 1. interest rate, 2. transportation information, 3. faculty list, and 4. hospital service. We collect tables from different websites, and manually edit the domain knowledge. With the domain knowledge, the system recognizes the table structure and performs table

[3] The coordinate of the top-left corner is (0, 0)

[4] All example tables are in Chinese originally. We translate them in to English for reader's convenience.

transformation automatically. For each table, we evaluate the precision and recall according to the expected number of rows in the output database table, which should be the number of cells with target content in the original table. For each domain, the experimental results are divided into 2 groups, where group 1 are tables that our system can recognize the structure of the table and group 2 are tables that our system cannot recognize the table.

The Table Reading Application is implemented using Microsoft Visual Studio. We use C++.Net, HTML Tidy Library[http://tidy.sourceforge.net/] to build our kernel, Visual Basic.Net to build our Windows application, and C#.Net to run as standard Microsoft Windows ASP Web application.

4.1. Tables in four different domains

(1) For the interest rate domain, we collect all available interest rate tables from web sites of 15 banks in Taiwan, 14 of them are in HTML.

We manually read the first three tables and built the domain knowledge. There are five important concepts: interest rate, fix interest or non-fix interest, period, type, and mixed of type and period. Each concept has several instances, the interest rate's instance are all real numbers between 0% and 100%. The period concept's instances are 1 month to 36 months. The type concept's instances are a personal deposit account or a special account. All the concepts are common sense to human, however, in various wordings.

Our program recognizes 11 tables out of the 14 tables. The precision and recall are 96% and 82%, respectively. It is very interesting that no two tables shared the same layout in this experiment, but our system can recognize most of them.

The detailed result is shown in Table 3. The tagging detail shows that there are many omissions and some ambiguity of the meaning of cells. There are three tables

that our program cannot read, they are complicated tables as we mentioned in section 2.

(2) For the Transportation Information domain, we collect 368 transportation information tables from web pages of 4 highways in the Real Time Transportation Information System (http://road.iot.gov.tw/). We also manually read the first three tables and built the domain knowledge.

The recall and precision is 100%. We believe that they are machine-generate tables. Therefore, they are regular and easier to understand. The result is shown in Table 4.

(3) For faculty domain, we select 28 faculty tables from the web sites of 7 colleges. 26 tables are recognizable. The precision is 94% and the recall is 98.4%. The result is shown in Table 5. In this domain we cannot report how many tables are read manually to build the domain knowledge. Since our editor uses reference from educational web pages other than the ones used in the experiment.

(4) For the hospital domain, we select 683 outpatient service timetables from web pages of 17 medical service centers. 16 tables are recognizable. The record level precision is 94% and the recall is 82%. The result detail is shown in Table 5I. In this domain we also cannot report how many tables are read manually to build the domain knowledge. Since our editor uses reference from medical web pages other than the ones used in the experiment.

4.2. Discussion on the experimental result

According to the analysis in section 2. If the structure of a table is recognized as 1-dimainsional or 2-dimansional, the table can be transferred fully automatically. However, since the domain knowledge cannot cover all possible wordings, in some cases, the system tag the concept and instance based on heuristics.

The experiments show that the precision and recall rate are quit high, it support our selection of the heuristics. The precisions and recalls of the four domains in our experiment are above 93% and 82% respectively. There is a clear boundary between Group 1 (we can recognize the structure of the table) and Group 2 (we can not recognize the table). Our approach can (1) extract information into database from common layout structures once the semantic characteristic of data are identified, and (2) identify the semantic characteristic of data if it is supported by a well-defined knowledge model.

In faculty tables, a lot of tables use special repeated layouts. Since there is no general rule to define the layouts of these tables, and data in these tables has repeated semantic characteristics, we believe a frame-based approach should work better. Also, our approach cannot process tables containing sentences with footnotes since the program could misunderstand the information in a cell when it contains too many words.

5. CONCLUSIONS & FUTURE WORKS

We have described a semantic approach to extract tabular information from the web. In this approach, we transform HTML tables into a relational database table.

Our semantic approach can extract tabular information from web tables that have various layouts. In order to recognize the content in each cell, we build templates to describe possible representations of concepts and instances. These templates help to label semantic tags on each cell. In many cases, there are multiple tags for a cell because several templates are matched. However, by comparing the semantic relationships between neighbor cells, we can often determine a unique tag for the cell.

We design layout syntax grammar to denote the layout descriptions so that we can transfer various layouts of tables into a relational database. The program recognizes these layout descriptions automatically. We also design a semantic preserving transition to rearrange the information into database tables for the 1-dimsional and

2-dimsional tables.

The experimental results show a high success rate in different domains. A systematic post-processing can fix most of the fail cases that our approach cannot recognize the table layout. For tables containing multiple items in cells, we need to split each multi-concept cell into several cells. To deal with complex cells, such as footnote cells, we will need a preprocessing filter to isolate them.

An important future work is automatic ontology acquisition. Due to the requirement of table recognition, concept hierarchy is important. There are two possible ways to acquire the hyponym relationship. 1.) A prior approach: given domain documents, find all the concept-instance pairs.[9] We have done some preliminary work in this area [35]. 2.) Online testing approach: given a table, enumerate all possible concept-instance pairs and then check the concept-instance pairs by the search engine. Which approach should be adopted depends on the application.

REFERENCES

[1] N. Ashish and C. A. Knoblock, Wrapper generation for semi-structured Internet sources, ACM SIGMOD Record (Special Issue on Managment of Semi-Structured Data) (1997) 8-15

[2] R. Baumgartner, S. Flesca and G. Gottlob, Visual Web information extraction with Lixto, in proceedings of 27th International Conference on Very Large Data Bases (VLDB 2001), (2001), 119-128

[3] H.-H. Chen, S.-C. Tsai and J.-H. Tsai, Mining Tables from Large Scale HTML Texts, in proceedings of 18th International Conference on Computational Linguistics, (2000), 166-172

[4] H. H. Chen and G. W. Bian, White Page Construction from web Pages for finding People on the Internet, International Journal of Computational Linguistics and Chinese Language Processing 3 (1) (1998) 75-100

[5] S. Douglas and M. Hurst, Layout and Language: Lists and Tables in Technical Documents, in proceedings of ACL SIGPARSE Workshop on

Punctuation in Computational Linguistics, (1996), 19-24
[6] D. Embley, D. Campbell, R. Smith and S. Liddle, Ontology-based extraction and structuring of information from data-rich unstructured documents, in proceedings of the Conference on Information and Knowledge Management (CIKM98), (1998), 52-59
[7] D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y.-K. Ng and R. D. Smith, Conceptual-model-based data extraction from multiple-record web pages, Data & Knowledge Engineering 31 (1999) 227-251
[8] D. W. Embley, C. Tao and S. W. Liddle, Automating the extraction of data from HTML tables with unknown structure, Data & Knowledge Engineering 54 (1) (2005) 3-28
[9] M. Fleischman, E. Hovy and A. Echihabi, Offline Strategies for Online Question Answering: Answering Questions Before They Are Asked, in proceedings of ACL-2003, (2003), 1-7
[10] D. Freitag, Information Extraction from HTML: Application of a General Machine Learning Approach, in proceedings of AAAI, (1998), 517-523
[11] E. Green and M. Krishnamoorthy, Model-based analysis of printed tables, (Springer-Verlag, 1996)
[12] C.-N. Hsu, Initial Results on Wrapping Semi-structured web Pages with Finite-States Transducers and Contextual Rules, Technical Report WS-98-01, (Menlo Park, CA, AAAI Press., 1998)
[13] C.-N. Hsu and C.-C. Chang, Finite-State Transducers for Semi-Structured Text Mining, in proceedings of IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications, (1999),
[14] C.-N. Hsu and C. A. Knoblock, Semantic Query Optimization for Query Plans of Heterogeneous Multi-Database Systems, IEEE Transactions on Knowledge and Data Engineering 12 (6) (2000) 959-978
[15] J. Y.-j. Hsu and W.-t. Yih, Template-Based Information Mining from HTML Documents, in proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97), (1997), 256-262
[16] W.-L. Hsu, Chinese parsing in a phoneme-to-character conversion system based on semantic pattern matching, International Journal on Computer Processing of Chinese and Oriental Languages 40 (1995) 227-236
[17] W.-L. Hsu, Y.-S. Chen and Y.-K. Wang, A Context sensitive model for concept understanding, in proceedings of ITALLC 98, (1998),

161-169
[18] W.-L. Hsu, Y.-S. Chen and Y.-K. Wang, Natural language agents - An agent society on the Internet, in proceedings of PRIMA, (1999),
[19] W.-L. Hsu, S.-H. Wu and Y.-S. Chen, Event Identification Based On The Information Map - INFOMAP, in proceedings of Natural Language Processing and Knowledge Engineering Symposium of the IEEE Systems, Man, and Cybernetics Conference, (2001), 1661-1672
[20] J. Hu, R. Kashi, D. Lopresti, G. Nagy and G. Wilfong, Why table ground-truthing is hard, in proceedings of Proc. 6th International Conference on Document Analysis and Recognition (ICDAR01), (2001), 129-133

[21] M. Hurst, Layout and Language: A Corpus of Documents Containing Tables, in proceedings of AAAI Fall Symposium Using Layout for the Generation, Understanding and Retrieval of Documents, (1999),
[22] M. Hurst, Layout and Language: Beyond Simple Text for Information Interaction - Modeling The Table, in proceedings of The 2nd International Conference on Multi-modal Interfaces, (1999),
[23] M. Hurst and S. Douglas, Layout & Language: Preliminary experiments in assigning logical structure to table cells, in proceedings of the Fifth Conference on Applied Natural Language Processing, (1997),
[24] M. Hurst and S. Douglas, Layout and Language: Preliminary investigations in recognizing the structure of tables, in proceedings of the Fourth International Conference on Document Analysis and Recognition, (1997),
[25] N. Kushmerick, Wrapper Induction for Information Extraction, Ph.D. thesis, Department of Computer Science and Engineering, University of Washington, (1997)
[26] L. Liu, C. Pu and W. Han, XWrap: An XML-enabled Wrapper Construction System for Web Information Sources, in proceedings of International Conference on Data Engineering (ICDE 2000), (2000),
[27] D. Lopresti and G. Nagy, A tabular survey of automated table processing, (Springer Verlag, 2000)
[28] I. Muslea, S. Minton and C. Knoblobk, STALKER: Learning Extraction Rules for Semi-structured web-based Information Source, in proceedings of AAAI-98 Workshop on AI and Information Integration, (1998),
[29] D. Pinto, A. McCallum, X. Wei and W. B. Croft, Table extraction

using conditional random fields, in proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, (2003), 235 - 242
[30] A. Sahuguet and F. Azavant, Building Intelligent Web Applications Using Lightweight Wrappers, Data and Knowledge Engineering (2000)
[31] M. Thompson, A tables manifesto, in proceedings of SGML Europe 1996, (1996), 151-153
[32] H.-L. Wang, S.-H. Wu, I.-C. Wang, C.-L. Sung, W.-L. Hsu and W. K. Shih, Semantic Search on Internet Tabular Information Extraction for Answering Queries, in proceedings of Proceedings of Ninth International Conference on Information and Knowledge Management, (2000),
[33] X. Wang, Tabular abstraction, editing, and formatting, PhD Thesis thesis, Department of Computer Science, University of Waterloo, (1996)
[34] P. Wright, A user oriented approach to the design of tables and flowcharts, The Technology of Text, (Educational Technology Publications, 1982)
[35] S.-H. Wu and W.-L. Hsu, SOAT: A Semi-Automatic Domain Ontology Acquisition Tool from Chinese Corpus, in proceedings of 19th International Conference on Computational Linguistics (Coling-02), (2002), 1313-1317
[36] M. Yoshida, K. Torisawa and J. Tsujii, A method to integrate tables of the world wide web, in proceedings of International Workshop on Web Document Analysis, (2001), 31-34
[37] R. Zanibbi, D. Blostein and J. R. Cordy, A Survey of Table Recognition: Models, Observations, Transformations and Inferences, International Journal on Document Analysis and Recognition 7 (1) (2004) 1 - 16

Train Number						Start Time	Arrival Time	Remark
CHU KUANG	11		TAI-TUNG NEW	To	KAO-HSIUNG	05:15	07:52	Every Day
FU HSING	101	M	KEE-LUNG	To	PING-TUNG	06:03	09:08	Every Day
TZE CHIANG	1003	M	SUNG-SHAN	To	KAO-HSIUNG	07:00	09:16	Every Day
TZE CHIANG	1005	M	SUNG-SHAN	To	KAO-HSIUNG	07:40	09:46	FRI SAT SUN only
TZE CHIANG	1007	M	SUNG-SHAN	To	KAO-HSIUNG	07:55	10:11	Every Day
CHU KUANG	41	M	KEE-LUNG	To	CHIA-I	08:23	11:06	Every Day
TZE CHIANG	1009	M	KEE-LUNG	To	KAO-HSIUNG	08:40	10:40	Every Day
TZE CHIANG	1011	M	SUNG-SHAN	To	KAO-HSIUNG	09:00	11:15	Every Day

Figure 2. A train timetable

C_1	…	C_n
I_1^{*c}	…	I_n^{*c}

(a)

Flight	Day	Departs	Arrives	Stops/Vi
LM202	Sun	11:10	13:20	0
LM208	Mon	20:20	21:45	0
LM208	Thu	20:45	22:10	0
LM208	Tue	20:00	22:10	0
LM451	Mon	10:40	12:05	0
LM963	Tue	10:40	12:05	0

(b)

Figure 3. (a) an abstract 1-dimensional table, (b) a 1-dimensional table example of (a). C_X denotes a string representing concept X in a cell, and I_X denotes a string that representing an instance of concept X in a cell. ^{*c} denotes a repetition in a column and ^{*r} denotes a repetition in a row. *,* denotes a repetition in a rectangular area. See the next sub-section for detail.

?	I_Y^{*r}
I_X^{*c}	$I_Z^{*,*}$

(a)

C_1	…	C_n	I_Y^{*}
I_1^{*c}	…	I_n^{*c}	$I_Z^{*,*}$

(b)

Time	Morning	Afternoon
Mon	John Wang (2002)	Indy Lai (2005)
Tue	Jimmy Lin (2007)	Wendy Lee (2001)
Wen	Indy Lai (2005)	John Wang (2002)
Thr	Jimmy Lin (2007)	John Wang (2002)
Fri	Wendy Lee (2001)	Indy Lai (2005)
Sat	John Wang (2002)	

(c)

	Room	Mon	Tue	Wed	Thu	Fri	Sat
Morning	1	Joe Jiang	Jean Tasi	Joe Huang	Hellon Yuo	Hellon Yuo	Collin Lee
	2		Joe Jiang	Collin Lee	Jean Tasi	Joe Jiang	
	3	Jean Tasi	Collin Lee	Jean Tasi	Joe Huang	Collin Lee	Hellon Yuo
Afternoon	1	Collin Lee	Jean Tasi	Hellon Yuo	Joe Jiang	Joe Huang	
	2		Joe Jiang			Jean Tasi	
	3		Joe Huang	Collin Lee	Hellon Yuo	Collin Lee	

(d)

Figure 4. Tables (a) & (b) are abstract 2-dimensional tables, (c) is an example of (a), and (d) is an example of (b).

C_{X1}	…	C_{Xn}
I_P		
I^{*c}_{X1}	…	I^{*c}_{Xn}
I_P		
I^{*c}_{X1}	…	I^{*c}_{Xn}

(a)

C_X	C_Z
I_Y	I_Z
I_X I_Y	I_Z
I_Y	I_Z
I_X I_Y	I_Z

(b)

C_X	C_Y	C_X	C_Y
I^{*c}_X	I^{*c}_Y	I^{*c}_X	I^{*c}_Y
C_X	C_Y	C_X	C_Y
I^{*c}_X	I^{*c}_Y	I^{*c}_X	I^{*c}_Y

(c)

C_X	C_Y C_Z
I_X	I_Y I_Z
I_X	I_Y I_Z
I_X	I_Y I_Z
I_X	I_Y I_Z
I_X	I_Y I_Z

(d)

A D
B E
C F

(e)

Date	Room	Doctor
Mon	1	Jenny
	2	Penny
Tue	1	David
	1	Josh

(f)

Date	Room	Doctor
Mon	1	Jenny
	2	Penny
Tue	1	David
	1	Josh

(g)

Figure 5. Complex tables. (a) Partition labels. (b) Over-spanned labels. (c) Combination. (d) Multiple items in a cell. (e) Vertical writing. (f) Forward reduction. (g) Original form of (f).

Morning						
Room	Mon	Tue	Wed	Thu	Fri	Sat
1	Yi-Chung Chen 10201	Hong-Tai Hur 10201	Hsi-Sang Yang 10201	Yi-Chung Chen 10201	Hsi-Sang Yang 10201	Hsi-Sang Yang 10201
2	Jean-Lin Chen 10202	Ching-Hue Lee 10202	Hong-Tai Hur10202	Hsi-Sang Yang 10202	Hong-Tai Hur10202	Ching-Hue Lee 10202
3		Tung-Long Tsai 10213	Tung-Long Tsai 10203	Hong-Tai Hur10203	Jean-Lin Chen 10203	
4						
Afternoon						
Room	Mon	Tue	Wed	Thu	Fri	Sat
1	Tung-Long Tsai 20201	Len-Hue Huang 20201	Jean-Lin Chen 20201	Jean-Lin Chen 20201	Tung-Long Tsai 20201	
2	Ching-Hue Lee 20202	An-Ching Lai 20202			Ching-Hue Lee 20202	
3						
4						

Figure 6. An illustrative example of combination and multiple items in a cell

Items & Periods		Regular	Float
Fixed Deposit	3 Monthes	4.40	4.40
	6 Monthes	4.95	4.95
	9 Monthes	5.05	5.05
	1 Year	5.15	5.15
	2 Years	5.25	5.25
	3 Years	5.25	5.25
Regular Fixed Deposit	1 Year	5.25	5.25
	2 Years	5.35	5.35
	3 Years	5.35	5.35

Figure 7. A table with over-spanned labels

Rate (%)	Regular	Float
Regular Fixed Deposit		
1 Year	5.05	5.05
2 Years	5.10	5.10
3 Years	5.10	5.10
Fixed Deposit		
3 Monthes	4.35	4.35
6 Monthes	4.60	4.60
9 Monthes	4.70	4.70
1 Year	5.00	5.00
2 Years	5.05	5.05
3 Years	5.05	5.05

Figure 8. A table with partition labels

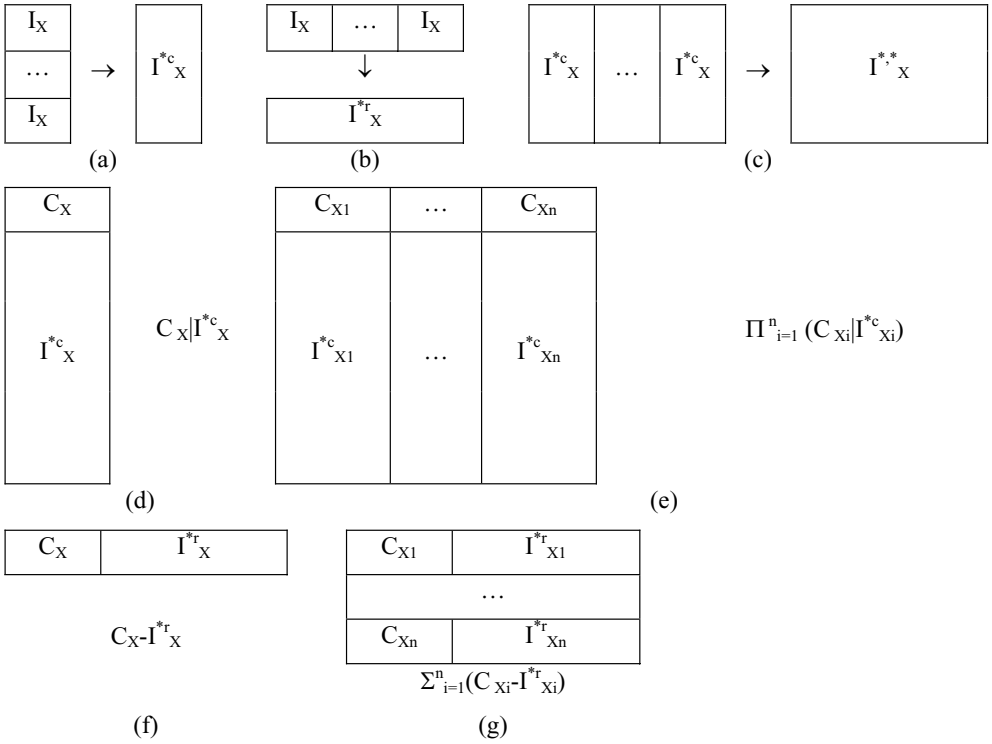


Figure 9. Areas represented by layout syntax grammars.

C_X	$I^{2r}{}_Y$
$I^{6c}{}_X$	$I^{6,2}{}_Z$

(a) Abstract table of Figure 4(c)

C_X	C_Y	C_Z
$C_X(1)$	$C_Y(1)$	$C_Z(1,1)$
$C_X(1)$	$C_Y(2)$	$C_Z(1,2)$
…	…	…
$C_X(6)$	$C_Y(1)$	$C_Z(6,1)$
$C_X(6)$	$C_Y(2)$	$C_Z(6,2)$

Figure 10 (b) Transformed from Figure 10 (a) by the transition rules

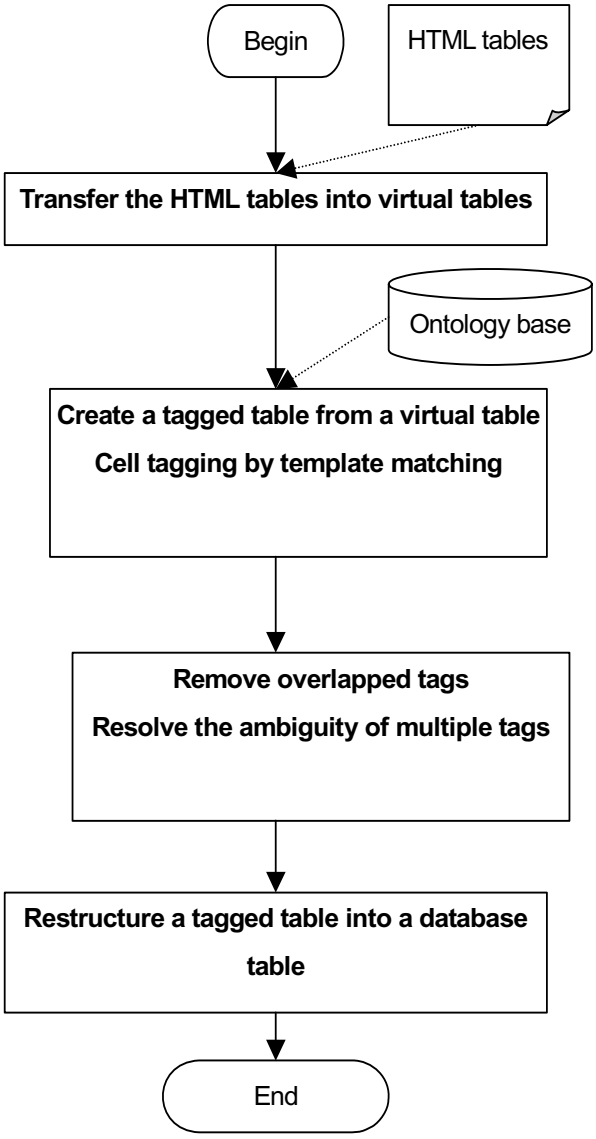
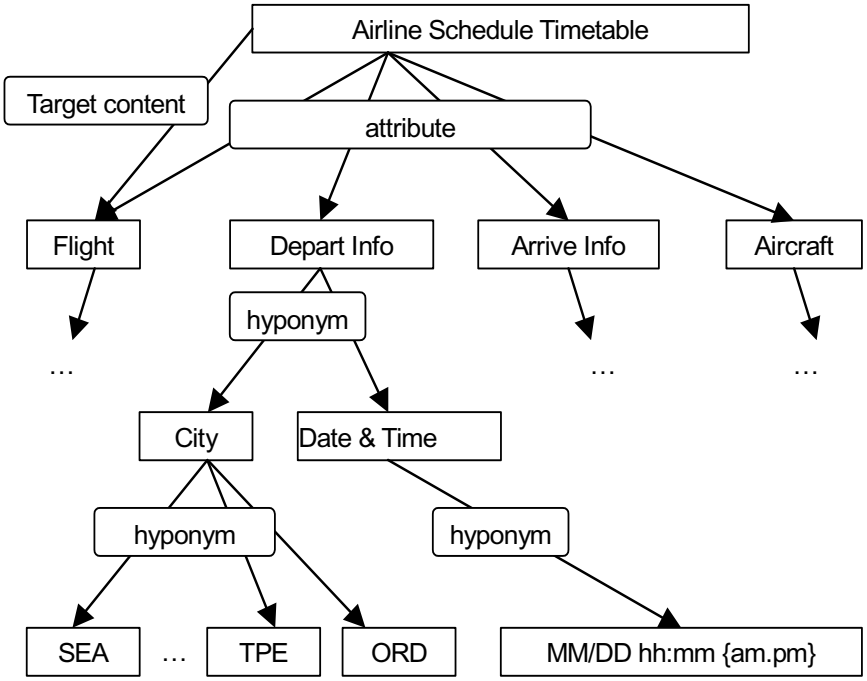


Figure 11. Table reading flowchart

Flight	Departing		Arriving		Aircraft
	City	Date & Time	City	Date & Time	
EVA Airways 10	TPE	07/14 11:50pm	YVR	07/14 07:40pm	744
American Airlines 6647	YVR	07/14 09:00pm	LAX	07/14 11:45pm	737
Air Canada 9800	TPE	07/14 11:50pm	YVR	07/14 07:40pm	744
American Airlines 6501	YVR	07/15 06:35am	LAX	07/15 09:27am	737
China Airlines 61	TPE	07/14 08:10pm	FRA	07/15 06:50am	M11
American Airlines 83	FRA	07/15 10:40am	ORD	07/15 01:05pm	763
American Airlines 473	ORD	07/15 02:30pm	LAX	07/15 04:35pm	738

(a) An airline timetable



(b) The knowledge map for airline timetables (some details are omitted)

Figure 12. An airline timetable and the corresponding knowledge map

```
        Algorithm TagginSelection
Input: tagged table Γ
Output: tagged table Γ with at most one tag in every cell

for c = 0 to Γ.max_columns – 1
      for r = 0 to Γ.max_rows – 1
            best_tag := nil
            for every tag Ti in Γ.cell (c, r)
                  Ti.score := score (Ti)
                  if best_tag = nil or best_tag.score < Ti.score then
                        best_tag = ti
            remove all tags except best_tag in Γ.cell (c, r)
```

Algorithm TaggingSelection processes the tag selection for all cells.

Figure 13. Tagging Algorithm

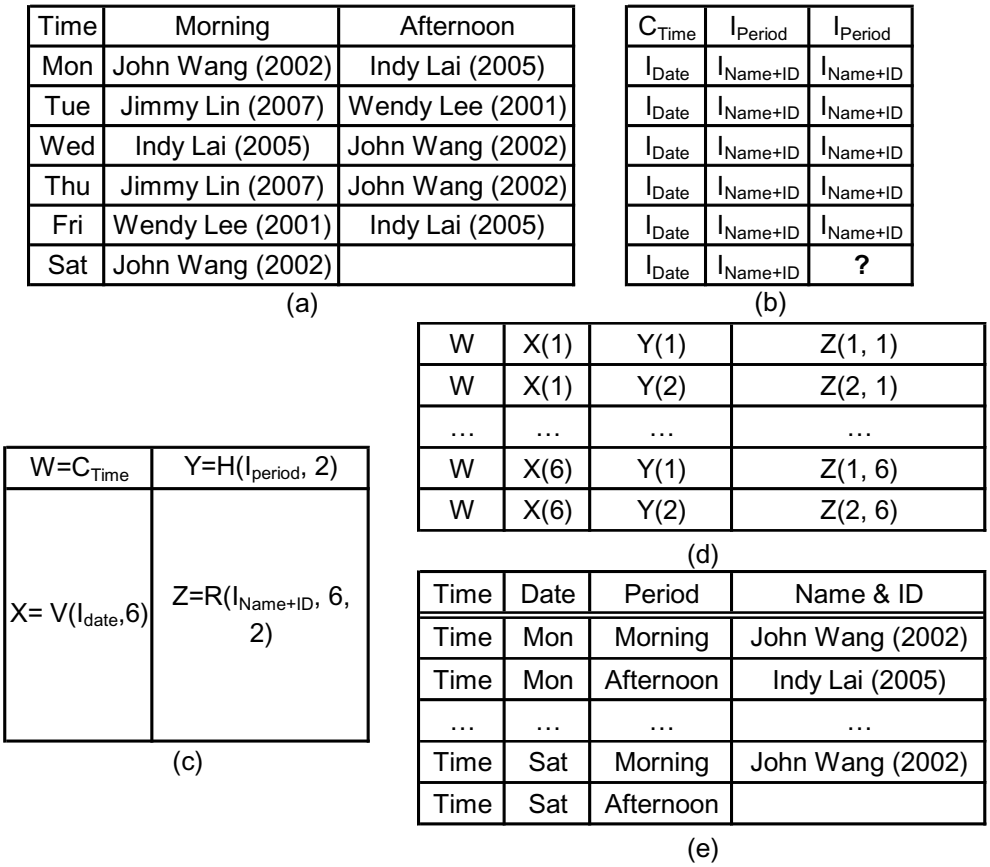


Figure 14. (a) The original table, (b) The outcome after the tagging step, (c) the outcome after the layout recognition step, 4 monotonic areas, (d) the transformed outcome by applying the transformation rules, (e) The outcome in the original strings.

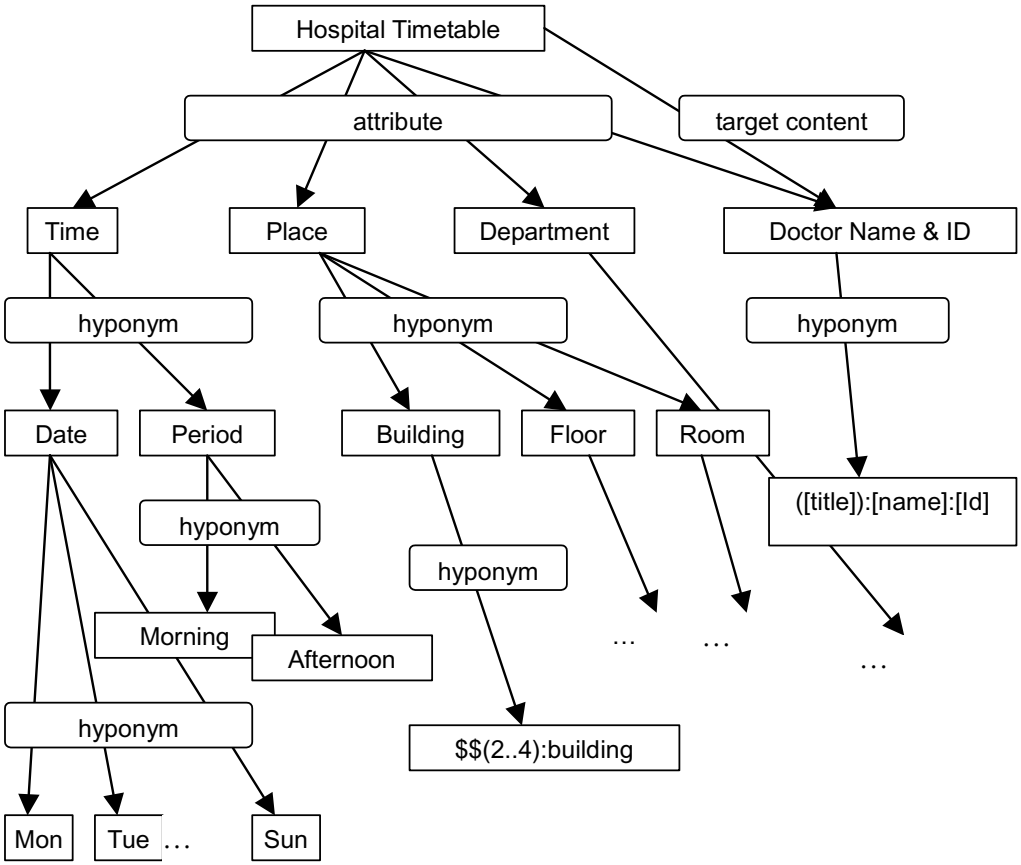


Figure 15. The Domain knowledge for the outpatient service timetables (templates of each concept's synonyms are omitted)

TABLE 1.
Table transformation rules

Table Dimension	Original Layout	Database Layout
1-dim table	C_X-$I^{*r}{}_X$	$C_X\|I^{*r}{}_X$
2-dim table	$(?\|I^{mc}{}_X)$-$(I^{nr}{}_Y\|I^{m,n}{}_Z)$	$(C_X$-C_Y-$C_Z)\|(\Gamma^{m}{}_{i=1,}{}^{n}{}_{j=1} I_X(i)$-$I_Y(j)$-$I_Z(i,j))$

TABLE 2. INFOMAP template syntax

Concept	Template of Instances	Example
Date	[digit](2..2):"/":[digit](2..2)	12/25
Title	{"Mr.", "Dr.", "Prof."}:[family name]	Dr. Wang
Phone number	[digit](3..3):"-"(0..1):[digit](4..4)	594-3388

Table 3
Experimental Results of Interest Rate Table

The second column shows the number of data in the original tables, the fourth column shows the number of records that the program reports and the third column is the number of correct records. The last four columns report the detail of cell tagging, including the number of tagged cells, the number of cells with multiple tags, the number of cells whose tags are decided by our algorithm and the number of cells with empty tag.

ID	# of record	# of correct record	cell with data	Precision	Recall	# of Tagged cell	Cell with Multiple tag	# of cell that the tag is Guessed	# of cell with Empty tag
Group 1									
Bank1	54	52	53	98.1%	96.3%	114	0	0	2
Bank2	30	30	30	100.0%	100.0%	47	0	0	1
Bank3	39	37	38	97.4%	94.9%	113	0	25	60
Bank4	68	39	39	100.0%	57.4%	71	14	22	4
Bank5	72	59	59	100.0%	81.9%	96	0	13	80
Bank6	43	43	43	100.0%	100.0%	68	0	1	4
Bank7	51	30	31	96.8%	58.8%	74	0	0	57
Bank8	38	37	37	100.0%	97.4%	76	0	4	8
Bank9	39	26	28	92.9%	66.7%	54	4	5	31
Bank13	16	10	14	71.4%	62.5%	23	0	0	1
Bank14	44	39	39	100.0%	88.6%	89	0	2	14
average	44.9	36.5	37.4	**96.0%**	**82.2%**	75.0	1.6	6.5	23.8
Group 2									
Bank10	38	0	37	0.0%	0.0%	46	1	1	19
Bank11	48	3	43	7.0%	6.3%	56	0	0	44
Bank15	48	10	48	20.8%	20.8%	72	0	0	40
average	44.7	4.3	42.7	9.3%	9.0%	58.0	0.3	0.3	34.3

Table 4

Experiment results of transportation information tables

ID	Number of tables	Number of tables extracted correctly	recall	Precision
Highway #1	92	92	100%	100%
Highway #2	92	92	100%	100%
Highway #3	92	92	100%	100%
Highway #1 Hsichih-WuKu Branch	92	92	100%	100%
	368	368	**100%**	**100%**

Table 5

Experiment result of faculty tables

Department	# of record	# of correct record	cell with data	Precision	Recall	# of Tagged cell	Cell with Multiple tag	# of cell that the tag is Guessed
Group 1								
NTU FL	101	101	106	95.3%	100.0%	359	59	65
NTU Phil	28	28	28	100.0%	100.0%	99	21	17
NCTU AM	25	25	28	89.3%	100.0%	93	12	16
NCKU LS	13	13	13	100.0%	100.0%	45	2	8
NCKU Aero	43	43	43	100.0%	100.0%	146	4	30
NCKU UP	19	19	21	90.5%	100.0%	65	6	21
NCKU Econ	4	4	4	100.0%	100.0%	17	2	1
NCCU Turkish	7	7	7	100.0%	100.0%	23	0	8
NCCU English	45	44	45	97.8%	97.8%	64	10	32
NCCU RM	25	25	25	100.0%	100.0%	83	6	20
NCCU IB	17	17	17	100.0%	100.0%	55	10	17
NCCU CS	10	10	10	100.0%	100.0%	44	5	9
NCCU AM	15	14	15	93.3%	93.3%	68	3	10
NTNU Arts	45	38	46	82.6%	84.4%	127	15	55
NCNU Edu&Psy	28	28	28	100.0%	100.0%	159	15	14
NCCU Arab	14	12	13	92.3%	85.7%	42	5	12
NCCU RTV	7	7	11	63.6%	100.0%	36	4	5
NTHU Math	32	32	32	100.0%	100.0%	132	5	31
NTHU Mech1	7	7	8	87.5%	100.0%	26	1	7
NTHU Mech2	9	9	10	90.0%	100.0%	39	7	4
NTHU Mech3	8	8	8	100.0%	100.0%	33	4	3
NTHU Mech4	8	8	9	88.9%	100.0%	36	5	3
FJU Psy	12	12	12	100.0%	100.0%	55	4	6
FJU Japanese	8	8	8	100.0%	100.0%	33	4	3
average	22.1	21.6	22.8	**94.6%**	**98.4%**	78.3	8.7	16.5
Group 2								
NTU History	13	0	14	0.0%	0.00%	30	2	13
NTNU Music	50	0	0	0.00%	0.00%	79	10	42
average	31.5	0	7	0%	0%	54.5	6	27.5

Table 6

Experiment results of outpatient service timetable

Hospital	# or record	# of correct record	cell with data	Precision	Recall	# of Tagged cell	Cell with Multiple tag	# of cell that the tag is Guessed	# of cell with Empty tag
Group 1									
ChungShan	347	221	245	90.20%	63.69%	1172	98	14	770
Taipei XiuChuan	46	34	34	100.00%	73.91%	59	3	5	142
ZhuShan XiuChuan	197	172	172	100.00%	87.31%	317	10	3	312
ZhongXiau	19	19	19	100.00%	100.00%	29	17	0	6
GengShen	139	73	125	58.40%	52.52%	302	11	5	300
YuLi CiJi	63	61	62	98.39%	96.83%	210	8	1	493
CiJi	323	265	279	94.98%	82.04%	624	138	36	385
XinGuang	581	560	572	97.90%	96.39%	1348	65	2	803
Taipei	12	12	12	100.00%	100.00%	19	0	0	23
GuanShan CiJi	35	27	34	79.41%	77.14%	161	2	0	304
CMUH	363	347	365	95.07%	95.59%	718	260	1	314
ZhangHua XiuChuan	596	547	554	98.74%	91.78%	1246	93	56	1591
TaiAn	74	54	54	100.00%	72.97%	162	52	1	241
GuangFu XiuChuan	136	90	90	100.00%	66.18%	210	24	1	201
XinLou	13	13	14	92.86%	100.00%	45	14	0	45
CKU	118	80	88	90.91%	67.80%	195	22	39	287
average	181.0	156.9	164.9	**93.6%**	**82.8%**	376.3	47.9	10.0	363.1
Group 2									
TauYuan	44	8	75	18.18%	10.67%	123	55	10	44
average	44	8	75	18.18%	10.67%	123	55	10	44