

# Extracting Objects and Their Attributes from Tables in Text Documents

© Nikita Astrakhantsev

Institute for System Programming of the Russian Academy of Sciences  
astrakhantsev@ispras.ru

## Abstract

Extracting information from tables is an important and rather complex part of information retrieval.

For the task of objects extraction from HTML tables we introduce the following methods: determining table orientation, processing of aggregating objects (like *Total*) and scattered headers (super row labels, subheaders).

## 1 Introduction

Significant amount of text information has relational structure, which is often represented in a table view. Since this view is designed for humans and contains many ambiguous elements, it follows that automatic processing of tables is rather complex.

For example, table 1 contains scattered header, complex hierarchical header, special objects, and other elements. They play particular roles in the table and, thus, should be treated in a special way. These elements are described in the rest of the paper.

Another non-trivial task is to determine table orientation: it can be horizontal (row wise, table 1) or vertical (column wise, table 2).

We consider tables in structured formats such as HTML and Wiki markup<sup>1</sup>. The main goal of our table processing is to extract objects as collections of attribute-value pairs. Thereupon we focus on determining table orientation and understanding the role of each element in the table.

## 2 Related work

Silva et al [1] distinguish five tasks of extraction information from table in their detailed survey:

1. Location: differentiating the table from other text elements such as body text, titles, lists, etc.
2. Segmentation: identifying table cells, rows, and columns and their relative positions.

<sup>1</sup> Wiki markup is a lightweight markup language used to write pages in wiki websites, such as Wikipedia, and is a simplified alternative/intermediate to HTML.

[http://en.wikipedia.org/wiki/Wiki\\_markup](http://en.wikipedia.org/wiki/Wiki_markup)

3. Functional analysis: classifying a table area (cell, column, row) to data or attribute area.
4. Structural analysis: connecting each data cell to all characterizing attribute cells.
5. Interpretation: understanding and further using extracted information.

First of them occurs mostly in plain text and image formats. The last task depends on kind of table usage: generating ontology from tables [2], mapping extracted information to predefined scheme [3], and so on.

Our work is devoted to the three last tasks while structural analysis is a principal one.

Table 1

Model	Version	Year	Power		Production
			hp	kW	
<i>Sports cars</i>					
SVT	9	1993	240	179	5,276
		1994			2,280
		1995			
		<b>Total</b>			7,556
	10	1999	360	268	4,000
		2000			4,966
		2001			6,381
		<b>Total</b>			28,124
		<b>Total</b>			35,680

Scattered header      Special objects      Complex header      Empty cell

Table 2

	A310-200	A310-200F
Crew	Two	
Length	46.66 metres (153 ft 1 in)	
Height	15.8 metres (51 ft 10 in)	
Wingspan	43.9 metres (144 ft)	
Wing area	219 square metres (2,360 sq ft)	
Wing sweep	28 °	
Cross section	5.64 metres (18 ft 6 in)	
Passengers (2-cl)	240	33t cargo
Empty weight	80,142 kg	72,400 kilograms
Max fuel	55,200 l (14,603 US g)	
Cruise speed	0.80 (850 km/h.)	

## 2.1 Plain text format

Early works deal with tables in plain text, usually in ASCII format [4-7]. The main problem here is to detect table, to recognize table delimiters (spaces, tabs, special characters like hyphens, etc), and thereby to understand table structure.

Rus and Summers [4] consider a text block to be a table if it consists of columns separated by white space and if cells of such columns are lexically persistent. A similar approach is used in the work of Douglas et al. [5]; they group text blocks surrounded by white spaces and use heuristic to determine whether these blocks are parts of tables.

There are many works concerned plain text and most of them use approaches inapplicable for HTML tables. However there are some useful ideas, which were explored in these works and inspired by linguistic characteristics of table content. For example, similarity/cohesion among different table elements (cells/lines/columns) became a common feature in future works including HTML tables oriented ones. Hurst and Douglas [6] suggest explicit string-based formulas for computing cell values cohesion.

Pinto et al. [7] present Conditional random field algorithm for classification of each row in ASCII table to one of 12 classes such as "data row", "section header", "super header", etc.

## 2.2 Image format

The majority of these works are devoted to performing location and segmentation of a table (Tupaj et al. [8], Wang et al. [9, 10], inter alia).

At a distance, Gatterbauer et al. [11] process HTML tables treating them as images. More precisely, they distinguish two representations of web pages: DOM tree representation and visual box (topological) representation. Our work is based on the first representation, while their approach is based on the second one, and thereby it cannot be applied to our work.

## 2.3 HTML format

A main source of HTML tables is Web pages, but many Web-tables are created just for layout, not for representing relational information. Wang and Hu [12] call such tables *non-genuine*. They suggest machine learning classifiers (SVM and decision tree) to determine whether the table is genuine or non-genuine. Features are divided into three types: length consistency of the cell contents, type of cell content, and word group.

Chen et al. [13] apply string and content type similarity to this task. They also use cell similarity for determining table orientation.

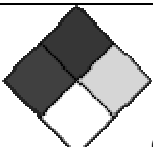
Yoshida et al. [14] suggest Expectation Maximization algorithm for classifying each table into one of 9 predefined types. Ontological knowledge are used as a parameter for the model, e.g. "Name" and "Birthday" are more often attributes than values.

Cafarella et al. [15] process the corpus of 14 billion Web tables. They introduce a tool for synonymic attributes searching (Attribute Correlation Statistics), which can be useful in the task of attribute/value classifying. Also the statistics gathered by authors corroborates the assumption that there is a small set of schemas that most tables in the world conform to.

## 3 Table orientation

What is an object in a table? Often the answer is obvious. For example, in table 2 objects are two airbuses: A310-200 and A310-200F. Table 1 is a little bit vaguer. It contains SVT vehicles manufactured in different years. Sometimes tables with nondefinable objects are found, e.g. multiplication table or table 3.

Table 3

NFPA 704	
	0
Fire diamond for aluminium shot	

As a rough definition, table object is a real world entity, whose attributes are set in the table. We assume that either row or column represents an object. So, table has either horizontal or vertical orientation.

We use 2 machine learning algorithms with the same features to determine table orientation: decision tree and naïve Bayes. All features have common nature: some function is computed on horizontal table as well as on vertical one and the difference between obtained values is found.

First feature is the difference in header depth. The function is very simple; it returns the depth of table header hierarchy. E.g. the header depth for horizontal table in figure 1 is 2, due to Power, hp, and kW cells; the header depth for the vertical table is 1. The header depths for both orientations of table 2 are 1. The motivation is following: orientation with deeper header is more likely to be correct.

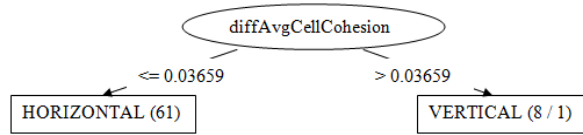
Second feature is difference in maximum cell cohesion. Cell cohesion is an average string similarity of all cells in a row or in a column. Average string similarity is computed by summation of all pairwise string similarities/metrics of cells and normalization (dividing by square of total count of cells). Maximum cell cohesion for the horizontal table is simply a maximum of cell cohesion in all rows; the same stands for the vertical one with replacement of rows by columns.

Third feature is the difference in average cell cohesion. The only distinction from the previous feature is that all cell cohesions are summed and then divided by total count of rows or columns.

After experiments with a small set of Wikipedia tables (about 70) we found that the most valuable feature is the difference in average cell cohesion. All

other features aren't presented in decision tree without overlearning (see figure 1).

Figure 1



Also we experimented with introducing the third type of tables, which contain no object, but effectiveness decreased dramatically. This can be easily explained because even humans can't label classify some table into the third type with confidence. So we don't take into account tables without objects.

#### 4 Aggregating objects processing

Some tables contain *aggregating objects*, which actually store information about other objects from the same table. For example, in table 4 the last row isn't an ordinary entity and should be processed in special way.

Table 4

System	Affiliates	%
FONASA	12,248,257	72.69
ISAPRE	2,780,396	16.50
<b>Total pop.</b>	<b>16,849,081</b>	<b>100.00</b>

Recognizing such objects by only presence of a keyword (e.g. *Total*) isn't efficient because of cells like "Total depravity, with convenient grace, does not preclude free will". So we collected statistics on thousands of Wikipedia tables and developed the heuristic for determining aggregating objects on basis of the next features:

1. Type of keyword in the cell content.
2. Number of words in the cell.
3. *Position* of the cell (sequence number of its column).
4. *Decoration* (bold, uppercase, <th>-tag).

All keywords are split into 2 types: *the strong type* (**Total, Totals, Tot., Subtotal**) and *the weak type* (**All, Sum**).

The cells containing words from the weak type must satisfy next conditions:

- exactly 1 word
- decorated
- position is not greater than 2

The cells containing words from the strong type must satisfy next conditions:

- if there is 1 word - no condition
- if there are 2 words - the cell must be decorated
- if there are from 3 to 5 words - the cell must be decorated and its position must be not greater than 2

The heuristic was inspired by the following considerations. The aggregating cells store information about special objects, so they should be noticeable by human readers. If the cell string is long or if it contains common word like *all*, then it must have some decorations in order to attract attention.

In future we plan to develop machine learning approach with these heuristics as features.

## 5 Scattered header processing

Some tables contain special rows, or *scattered headers*, which add structural information and are not table objects. Example is the row *Sports car* from table 1. We called the object extracted from such scattered header row during the initial processing *the scattered header object*.

We extract scattered header objects only from the object set, which corresponds to the horizontal orientation of the table. Vertical objects aren't processed, because width of any table is limited and scattered headers are useless in vertical tables.

### 5.1 Scattered header recognizing

Deciding whether each row is a scattered header (SH) is based on the assumptions that only one cell in the row is nonempty and the row must stand out against other table. In addition, we don't consider empty rows and last rows.

We divide all SHs into three subclasses: *single-cell SH*, *middle SH*, *first-cell SH*.

1) **Single-cell SH** is a row with just one cell in a table where other rows have more than 1 cell (in HTML terms, a row with colspan greater than 1).

The example is given in table 5 (row *Central Asia*).

Table 5

Name of region and territory	Area (km <sup>2</sup> )	Population (1 July 2008 est.)	Capital
<b>Central Asia:</b>			
Kazakhstan	2,724,927	15,666,533	Astana
Kyrgyzstan	198,500	5,356,869	Bishkek

We consider such row to be an SH without other criteria.

2) **Middle SH** is a row with just one nonempty cell, located in the middle of the table.

The 4th row *Bonus track* of table 6 is a middle SH.

Table 6

#	Title	Songwriters	Length
1.	"Mixing pot" ("Tacho")	Hemeto Pascoal	9:18
2.	"Pica pau (Take 1)"	Hemeto Pascoal	4:19
3.	"Just listen" ("Escuta meu piano")	Hemeto Pascoal	7:08
	<b>Bonus tracks</b>		
4.	"Open field"	Hemeto Pascoal	4:25

We require the nonempty cell of such row to be decorated.

In addition, we check the table to be non-sparse. Sparse table is a table with many empty cells, e.g. table 7. It is evident that the heuristic for determining middle SH doesn't work correctly with sparse tables.

Table 7

Km	Exit	Junctions	To	Remarks
		SMS Kota Tinggi	Sains	
		SMK Bandar	Sekolah Menengah	
0		Bulatan Desaru	<b>NORTH JALAN</b> Sedili	Roundabout
		<b>Desaru guardpost</b>		
		<b>DESARU</b>	Desaru Impian	
		Kampung Baharu		

So we check the rows near the concerned middle SH row (3 rows above and 3 below). If they have empty cells, then the row under review isn't a middle SH. Of course, the row with empty cells can be another middle SH; to address this issue we don't take such rows into account while checking their cells. But previous and next rows must differ from the concerned one, because scattered headers never have the same structure and content with another SH.

Thereby table 7 contains no SH.

3) **First-cell SH** is a row whose only nonempty cell is first.

For example, the third row *Cities (10 Largest)* of table 8 belongs to this type.

**Table 8**

<b>Census Metropolitan Areas:</b>	<b>2006</b>	<b>2001</b>	<b>1996</b>
Calgary CMA	1,079,310	951,395	821,628
Edmonton CMA	1,034,945	937,845	862,597
<b>Cities (10 Largest):</b>			
Calgary	988,193	878,866	768,082

The criteria for this type are the same as for the previous type.

## 5.2 Scattered header processing

Scattered header object are removed from the original set before the aggregating objects processing. Therefore created aggregating objects have no information about scattered header objects.

We update attributes of all objects by adding the new field; currently its name is *Type*. Every table object located below the current scattered header and above the next scattered header (if it exists) is updated by adding a new value, which is the content of the corresponding scattered header.

## 6 Conclusions and future work

In this paper we concerned on the task of objects extraction from HTML tables, gave a short survey of occurring problems, and introduced methods (mostly heuristics) for their solving. Of course, there are many cases uncovered by this paper, e.g. accurate detection of table header, processing of non-aggregating special objects and so on. It's the scope of future research.

The most common usage of extracted objects is to map them to predefined relational scheme. Embley et al. [3] worked towards this task, but we believe that fully automatic processing will be ineffective. Gatterbauer et al. [11] make a similar note: "domain-independent table interpretation cannot result in unambiguously structured information because of existing inherent domain-specific ambiguities that can sometimes not even be resolved by humans". Therefore, we consider the computer-aided way to be more promising for the task of objects mapping and, in general, for table interpretation.

## References

- [1] A.C. Silva, A.M. Jorge, L. Torg. Design of an end-to-end method to extract information from tables, IJDAR(8), No. 2-3, pp. 144-171, 2006.
- [2] Y. A. Tijerino, D. W. Embley, D. W. Lonsdale, Y. Ding, and G. Nagy. Towards ontology generation from tables. World Wide Web, 8(3):261-285, 2005.
- [3] D.W. Embley, C. Tao, S.W. Liddle. Automating the Extraction of Data from HTML Tables with Unknown Structure, 2003.
- [4] D. Rus, K. Summers. Using white space for automated document structuring. Workshop on the Principles of Document Processing, 1994.
- [5] S. Douglas, M. Hurst, D. Quinn. Using Natural Language Processing for Identifying and Interpreting tables in Plain Text. In: Fourth Symposium on Document Analysis and Information Retrieval, pp. 535-545, 1995.
- [6] M. Hurst, S. Douglas. Layout and Language: Preliminary investigations in recognizing the structure of tables. In: Proceedings of International Conference on Document Analysis and Recognition (ICDAR'97), pp. 1043-1047, 1997.
- [7] D. Pinto, A. McCallum, X. Wei, and W.B. Croft, Table Extraction Using Conditional Random Fields, in Proc. DG.O, 2003.
- [8] S. Tupaj, Z. Shi, C.H. Chang, A. Hassan. Extracting tabular information from text files, EECS Department. Tufts University, 1996.
- [9] Y. Wang, T.P. Ihsin, H. Robert. Improvements of zone content classification by using background analysis. In: Proceedings of Document Analysis Systems, 2000.
- [10] Y. Wang, T.P. Ihsin, H. Robert. Automatic ground truth generation and A background-analysis-based table structure extraction method. In: Sixth International Conference on Document Analysis and Recognition, 2001.
- [11] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, B. Pollak. Towards domain-independent information extraction from web tables. In: Proceedings WWW, 2007.
- [12] Y. Wang, J. Hu. A machine learning based approach for table detection on the web. In: Proceedings of the Eleventh International WorldWideWeb Conference, 2002.
- [13] H.-H. Chen, S.-C. Tsai, S.-C., J.-H. Tsai. Mining tables from large scale HTML texts. In: 18th International Conference on Computational Linguistics (COLING), pp. 166-172, 2000.
- [14] M. Yoshida, K. Torisawa, J. Tsujii. A method to integrate tables of theWorldWideWeb. In: First International Workshop on Web Document Analysis, 2001.
- [15] M.J. Cafarella, A. Halevy, Y. Zhang, D.Z. Wang, E. Wu. WebTables: Exploring the Power of Tables on the Web. Proceedings of VLDB, 2008.