Table Extraction Using Spatial Reasoning on the CSS2 Visual Box Model [*]

Wolfgang Gatterbauer and **Paul Bohunsky**
Database and Artificial Intelligence Group
Vienna University of Technology, Austria
{gatter, bohunsky}@dbai.tuwien.ac.at

Abstract

Tables on web pages contain a huge amount of semantically explicit information, which makes them a worthwhile target for automatic information extraction and knowledge acquisition from the Web. However, the task of table extraction from web pages is difficult, because of HTML's design purpose to convey visual instead of semantic information. In this paper, we propose a robust technique for table extraction from arbitrary web pages. This technique relies upon the positional information of visualized DOM element nodes in a browser and, hereby, separates the intricacies of code implementation from the actual intended visual appearance. The novel aspect of the proposed web table extraction technique is the effective use of spatial reasoning on the CSS2 visual box model, which shows a high level of robustness even without any form of learning (F-measure $\approx$ 90%). We describe the ideas behind our approach, the tabular pattern recognition algorithm operating on a double topographical grid structure and allowing for effective and robust extraction, and general observations on web tables that should be borne in mind by any automatic web table extraction mechanism.

Introduction

The huge amount of information on the Web makes it an ideal source for collecting facts (Etzioni *et al.* 2004) and deducing semantic relationships from the extracted data (Pivk 2006) with one important application of comparison shopping (Bilenko, Basu, & Sahami 2005).

This process of knowledge acquisition from the Web can be divided conceptually into three consecutive steps (Gatterbauer 2006): *Information Retrieval* (IR) which aims to locate documents containing data and information relevant to a certain query; *Information Extraction* (IE) which aims to extract relevant information or data like keywords that appear in certain semantic or syntactic relationships; and finally, *Information Integration* (II) which aims to match individual pieces of information from heterogeneous sources and to build a consistent view in the form of a knowledge base. One is tempted to argue that these three steps form a processing chain that leads from data and information to knowledge.

With the rise of the WWW to the biggest single pool of information, focus of IE research gradually shifted from *unstructured* documents to *semi-structured* documents like HTML web pages. Within semi-structured documents, however, the content itself again can appear in natural language context (such as in weblogs) or in structured context (such as in tables and lists). Especially within tables, the notions of *syntax* and *semantics* blend into each other because each logical cell of a table derives its semantic meaning from its relative positional information, which can be regarded as syntax. Such table syntax is rather strict and less expressive than that of natural language, at least when focusing on such tables used to convey relations between data elements ("relational databases").

The task of *table understanding* can conceptually be separated into three consecutive steps (Hurst 2001): *location* of a table within a document; *table recognition* which aims to identify the relative spatial positions between a table's composing logical units; and *table interpretation* which aims to infer the reading paths of semantic information encoding. We refer to the first two steps as *table extraction*.

Web tables contain visual cues like background color and font metadata to help the human reader distinguish individual building blocks or logical cells of the contained relational information. Our observation is that the individual steps of table understanding become easier by taking advantage of such information in the representation that web browsers use in their internal DOM tree to correctly display or render the tables.

With this idea in mind, we propose an innovative approach to web table extraction (and in a future step web table understanding) that relies upon the positional and metadata information of element nodes within the DOM tree of a web browser. The pure approach proves to deliver very robust results and shows high potential for significant improvements in the future by adding heuristics and machine learning techniques on the available features. In our opinion, the robustness of our approach can be contributed to its chosen representation for reasoning that very well reflects the nature of semantic information in tables.

[*]This research has been supported in part by the Austrian Academy of Sciences through a DOC scholarship, and by the Austrian Federal Ministry for Transport, Innovation and Technology under the FIT-IT contract FFG 809261.

Overall, the main contributions of this paper can be summarized as follows: (1) We propose the visualized element nodes as basis for robust table extraction. (2) We introduce our visual table extraction model by which we define such structures that contain semantically relevant information in tabular form. (3) We introduce a bottom-up circulating expansion algorithm that reasons on the visualized element nodes and report the experimental results of our prototype implementation.

Related work

The tasks of table extraction and interpretation originate from the document understanding community. Approaches can basically be divided into two categories: top-down like (Nagy & Seth 1984) and bottom-up like (Kieninger 1998), depending on where the algorithms start. Hurst (2000) introduced an abstract table model for table understanding and separated the task into 5 steps. Web table recognition systems conventionally consider only the source code of HTML for extracting tabular data. Wang & Hu (2002) defined 16 features deduced from the source code and applied ML algorithms to distinguish "genuine" (Penn *et al.* 2001) from "non-genuine" tables. Both publications assume that interesting tables can only be found in leaf table elements, which are `<TABLE>` nodes that do not contain any other nested `<TABLE>` nodes.

To our best knowledge, the idea of actually *rendering* or "executing" HTML code in order to use the results for detecting relational information in web tables is first mentioned in (Cohen, Hurst, & Jensen 2002) within the context of Wrapper Learning. The described approach, however, does not actually render documents in a browser, but rather infers relative positional information of table nodes in an abstract table model with relative positional information deduced from the source code. In contrast, (Krüpl, Herzog, & Gatterbauer 2005) described a top-down table extraction mechanism working exclusively on rendered information obtained from the Open Source Mozilla Browser. In a forthcoming publication, Krüpl & Herzog (2006) report an approximate success rate of 70% with a bottom-up clustering algorithm of visualized word bounding boxes. The approach of our paper, in contrast, reasons on visualized element nodes with text projected into end results, which avoids problems with multiline cell detection inherent to both top-down and bottom-up table recognition approaches based on word bounding boxes (Handley 2000).

In a related research area, Cai *et al.* (2003) for the first time described an approach of page segmentation by obtaining DOM structure and visual information from the Internet Explorer web browser. Later, (Zhao *et al.* 2005) and (Zhai & Liu 2005) proposed to augment HTML code based approaches to record boundary detection with visual cues. Recently, Simon & Lausen (2005) described an approach that exclusively considers visual rendering of a web page for record boundary detection with the help of global multiple sequence alignment techniques.

The Visual Table Model

When HTML documents are laid out on the screen, CSS (Cascading Style Sheets) represents the elements of the document by rectangular boxes that are laid out one after the other or nested inside each other in an ordering that is called a flow. Each box has a content area and optional surrounding padding, border and margin areas according to the CSS2 visual formatting model (Wium Lie *et al.* 1998). We refer to such rendered or *visualized element nodes* as *element boxes*, use their border edges as our defining edges, and retrieve their positional information from the Mozilla browser as described in (Krüpl, Herzog, & Gatterbauer 2005).

Given all element boxes, we superimpose a minimal grid which covers each of their borders. In contrast to a 2-dimensional grid, as explained in (Hurst 2000) or used to reference fields of a chess board and cells of a spreadsheet, we use a *double topographical grid structure* with 4 dimensions (x_1, y_1, x_2, y_2) for each of the 4 cardinal directions of the visual plane: right, down, left, up (Figure 1). Despite not being more expressive in saving structures, we found this data structure built upon 4 lists of lists to be very time-effective in the process of detecting structures on a non-perfect grid with partly overlapping or empty blocks.

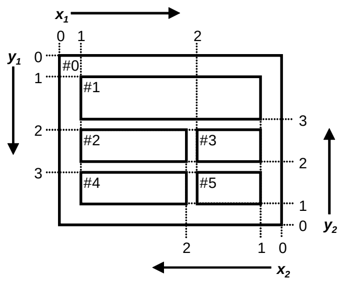
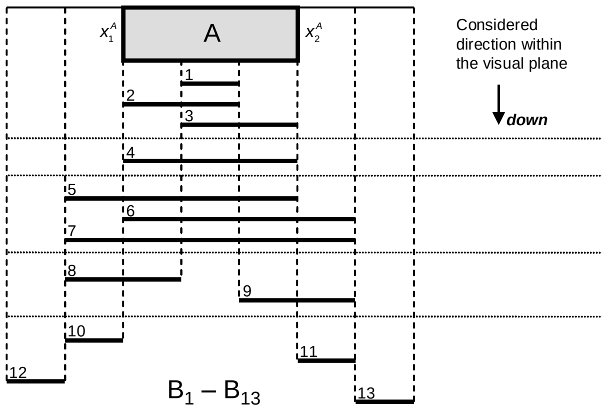


Figure 1: Minimal *double topographical grid*, superimposed on element boxes, allowing for efficient spatial reasoning.

We use two notions to characterize the spatial relations between any two boxes on the grid both vertically and horizontally: alignment and adjacency. *Alignment* compares horizontal and vertical projections of boxes. Aiello (2002) introduced rectangle relations based on the 13 temporal interval relations from Allen (1983) together with a notion of thick boundary for the purpose of reading order detection from document images. For our purpose, clustering into 5 distinct types of *neighbor relations* proved useful, as they express the 5 possible spatial relations between table cells for a given direction (Figure 2).

Adjacency is a term closely related to distance. Theoretically, two cells in the grid are adjacent if they are flush with each other. In reality, however, the actual distance between adjacent boxes is not always 0 and a long list of parameters would have to be recorded and calculated in order to correctly "reverse engineer" whether any two boxes are meant to be adjacent according to the CSS2 visual box model (Wium Lie *et al.* 1998). To avoid the resulting time performance reductions, we currently consider two boxes with a distance between 0 and 3 pixels to be adjacent.



	Lower neighbor relations of B(A)	*Allen's interval relations (Allen 1983; Aiello 2002)*	*X-coordinates' relations*	*Semantic relations*
I	B = smaller neighbor (A)	1 B during A 2 B starts A 3 B finishes A	$\left(x_1^A \leq x_1^B \wedge x_2^A > x_2^B\right) \vee$ $\left(x_1^A < x_1^B \wedge x_2^A \geq x_2^B\right)$	B is sub-ordinate of A
II	B = twin neighbor (A)	4 B equal A	$x_1^A = x_1^B \wedge x_2^A = x_2^B$	direct relation
III	B = bigger neighbor (A)	5 A finishes B 6 A starts B 7 A during B	$\left(x_1^A \geq x_1^B \wedge x_2^A < x_2^B\right) \vee$ $\left(x_1^A > x_1^B \wedge x_2^A \leq x_2^B\right)$	B is super-ordinate of A
IV	B = step neighbor (A)	8 B overlaps A 9 A overlaps B	$\left(x_1^A > x_1^B \wedge x_2^A > x_2^B\right) \vee$ $\left(x_1^A < x_1^B \wedge x_2^A < x_2^B\right)$	relation not well defined
V	B = no neighbor (A)	10 B meets A 11 A meets B 12 B before A 13 A before B	$x_1^A \geq x_2^B \vee x_2^A \leq x_1^B$	no relation

Figure 2: Five relevant lower neighbor relations, downward looking from box A, and corresponding semantic relations.

Within the pool of positional box information, we look for certain patterns we refer to as *hyperBoxes* which are a objects composed of rectangular areas on the double topographical grid together with a list of boxes that show the following distinguishing characteristics: the boxes together tile these encompassing rectangles in such a way that the whole area is covered and no box overlaps another box (*MECE* = Mutually Exclusive, Collectively Exhaustive). We call tables of interest (TOI) the subset of hyperBoxes that contain relevant tabular semantic information. Such regions show the following necessary, but not sufficient characteristics:

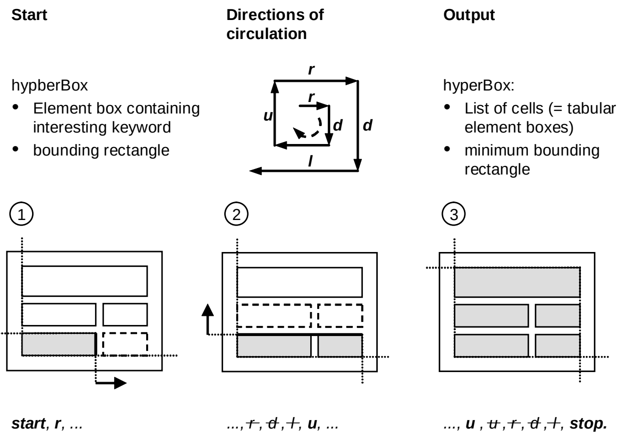
1. *HyperBox.* Like any hyperBox, a TOI has one bounding box as circumference.
2. *MECE.* Like any hyperBox, the boxes composing a TOI completely tile the encompassing hyperBox without any overlaps.
3. *Keyword.* A TOI contains a semantically relevant keyword (see next chapter for details).
4. *No step neighbors.* A TOI does not contain any step neighbors. This alignment relationship applies to any two cells within the TOI, not only adjacent boxes. The reason is that step neighbors do not have a well-defined categorical relation to each other.
5. *2 dimensions.* A TOI has at least 2 columns and 2 rows to ensure two-dimensional relationships.
6. *Cleanness.* The table contains no empty cells that could be merged into a subset of their neighbors without changing the semantic relationship between cells.

Despite or perhaps because of its simplicity, this visual table extraction model for TOIs works very well in practice. As HTML is not as expressive for visual tables as paper, we do not have to consider some peculiarities like diagonal subheadings that could be found on printed tables. HTML documents also usually do not align and place units of information adjacent to each other if there is no relevant semantic relation between them, as such a visual relation would also mislead the human reader. In total, this set of assumptions serve well for automatic extraction of TOIs from web pages.

Spatial Reasoning Algorithm

The table extraction stage is one module of an intended web information gathering system. Starting with seed knowledge (keywords), an IR module retrieves web pages containing relevant data. Information can be derived from the relationships between individual pieces of data. The syntax that is necessary to supplement this semantics is derived from positional information. Additional information can be extracted for data that appears in a similar syntactic relationship, analogous to the Dual Iterative Pattern Relation Expansion method of Brin (1998).

Such defined keywords are one input to our tabular pattern recognition and extraction module to which we refer to as **VENTrec** for **V**isualized **E**lement **N**ode **T**able **rec**ognition. 3 types of boxes on the grid are relevant: element boxes; hyperBoxes; and word bounding boxes.



Start
Directions of circulation
Output
hypberBox
• Element box containing interesting keyword
• bounding rectangle
r
r
u
d
d
l
hyperBox:
• List of cells (= tabular element boxes)
• minimum bounding rectangle
1
2
3
start, r, ...
..., ~~r~~, ~~d~~, ~~l~~, u, ...
..., u , ~~u~~, ~~r~~, ~~d~~, ~~l~~, stop.

Figure 3: Working of the *expansion algorithm*.

We refer to our tabular pattern recognition and extraction algorithm as the *expansion algorithm*, a recursive and circulating algorithm that tries to expand from any given hyperBox on the grid. Expansion into one direction is possible only if an adjacent hyperBox forms a twin neighbor and does not contain any element boxes introducing step neighbors.

The algorithm is circulating clockwise around the 4 cardinal directions of the visual plane and stops when no expansion is possible any more (Figure 3). Algorithm 1 shows the idea for one of the 4 directions.

Algorithm 1 EXPANDRIGHT($hBox$): tries to expand a given hyperBox to the right by one additional hyperBox

```
Input: $hBox$:hyperBox
Output: $\langle expandResult$:hyperBox, $expandSuccess$:boolean$\rangle$
1: $candidateList \leftarrow$ list of upper aligned and right adjacent boxes of $hBox$
2: for all $candidate \in candidateList$ do
3:     if $candidate$ = TwinNeighbor($hBox$) then
4:         return $\langle$Union($hBox$, $candidate$), $true\rangle$
5:     else if $candidate$ = SmallerNeighbor($hBox$) then
6:         $tempHBox \leftarrow candidate$
7:         repeat
8:             $\langle tempHBox, tE\rangle \leftarrow$ EXPANDDOWN($tempHBox$)
9:             if $tempHBox$ = TwinNeighbor($hBox$) then
10:                return $\langle$Union($hBox$, $tempHBox$), $true\rangle$
11:            end if
12:        until $tempHBox$ = BiggerNeighbor($hBox$) $\vee$ $tE = false$
13:    end if
14: end for
15: return $\langle hBox, false\rangle$
```

In the beginning, the keywords are projected into possibly containing element boxes which serve as starting hyperBoxes. At the end, the system projects all word boxes into the composing element boxes of the final hyperBox, cleans the hyperBox by replacing semantically non relevant empty element boxes, verifies that the result consists of at least 2 columns and 2 rows, and finally saves the results in XML.

Evaluation

We tested our algorithms with three different test sets in order to provide maximum transparency of its performance.

First, we gathered a number of different web pages with interesting information for our information gathering task in tabular form at the beginning of our project. We were especially interested in obtaining a maximum number of peculiarities in web tables in order to create a simple web table phenomenology. We added web pages that showed interesting characteristics and were challenging to our approach in theory. These web pages contain some very unusual web tables, sometimes even difficult to read for the human viewer. We do not provide quantitative analysis on this set of documents as the content was gathered – by logic of its selection process – to be challenging and the performance on this set is not representative for our approach. However, we describe the kind of problems and possible improvements of our approach deduced from this set at the end of this section.

Second, we defined a couple of search engine keywords from the electronic goods domain which are likely to return web tables. We retrieved the first 100 search results for the keywords "Canon Ixus IIs resolution" from the public search engine Google[1] with emptied cache and deleted cookies on February 19th, 2006. Out of these 100 results, only 96 were HTML pages we could retrieve and 42 contained tables of interest. If we did not find relevant web tables on the page, we additionally followed links to web pages from the same website likely to contain relevant web tables (e.g. "specifications"). As a result, we found additional 14 web TOIs. We followed this procedure as it emulates the way our knowledge acquisition system crawls for semantically relevant information. We applied our algorithm to these 110 web pages together with the keywords (we chose "resolution") and manually checked whether the resulting grid structure actually represents a TOI.

Third, we used the public web table ground truth used in (Wang & Hu 2002) and available on the web page of Yalin Wang[2]. This set contains 1,393 web pages with in total 11,477 leaf tables, manually sorted into 1,740 genuine and 9,373 non-genuine tables, where the term "genuine" resembles but does not match our notion of a TOI. This database was compiled more than 4 years ago and does not necessarily still represent current methods to build web tables. As an example, the affiliated publication still assumed that relevant semantic information can only be found in leaf `<table>` nodes. This assumption neither reflects our original hypothesis nor our findings (see discussion), but the database still provides us with an transparent and reproducible procedure to test our mechanism. We randomly ordered the list of annotated 1,393 web pages and chose the first 50 from the list that contained leaf table nodes. Whenever the specific web page contained at least one genuine table, we chose a keyword from the the first one to test our algorithm. Otherwise, we chose a keyword contained in a non-genuine, text-containing table.

Testing approach	Recall	Precision	F($\beta = 1$)	#Web pages	#TOIs	#False positives	#False negatives
(2a) 100 Google results	97.6%	82.0%	89.8%	96	42	9	1
(2b) 14 linked Web tables	92.9%	100%	96.4%	14	14	0	1
(3) (Wang & Hu 2002)	84.2%	94.1%	89.2%	50	19	1	3
Total without heuristics	93.0%	87.4%	90.2%	160	75	10	5
Total with heuristics	89.0%	96.7%	92.8%	160	75	2	8

Table 1: Results of our experiments.

Table 1 contains the quantitative evaluation of our system with regard to objectively chosen test sets. The list of web pages used, details of the quantitative evaluation, and an online VENTrec test facility can be found on the web page accompanying this publication [3].

In the following, we qualitatively explain the challenges to our approach with regard to all web pages tested.

1. *ASCII tables.* Tables that are formatted with spacing and line breaks in order to encode horizontal and vertical alignments cannot be detected by our approach. In our opinion, the occurrence of this kind of tables on the Web is nowadays very limited (1 occurrence in the quantitative test set) and will decrease even further.

[1]http://www.google.com

[2]http://www.math.ucla.edu/~ylwang/

[3]http://www.dbai.tuwien.ac.at/staff/gatter/ventrec

2. *Determinism.* In a number of cases, the algorithm currently expands into the wrong direction for seemingly aligned cells and then stops, because no further expansion is possible. We estimate this occurrence to be small (2 occurrences in the quantitative test set), but still relevant as it shows the current limits of our visual table model. This problem can, in theory, be solved with heuristics which are in their current conception still too time expensive.

3. *Semi-alignment.* A portion of false negative (though no occurrence in the quantitative test set) comes from boxes that are not completely aligned. This happens either because of semantic tables created by `<li>` element nodes, or because of nested tables whose content is not even aligned on the screen. We estimate this occurrence to be small but to be of increasing relevance with more tables implemented as two-dimensional lists.

4. *Distance between boxes.* Some negative positives (1 occurrence in the quantitative test set) result from spacing between the individual cells of a TOI to be bigger than our arbitrarily value of 3 pixel. After choosing the value of 5 pixel, this table could also be detected. However, as the second value can be interpreted as consequence of learning, we report the test results with the first value. As mentioned before, this issue could be solved by using more of the available DOM tree data. However, the time performance of the system would suffer significantly.

5. *Semantically non relevant tables.* The biggest portion of false positives (11 occurrences in the quantitative test set) occurs because of detected tabular structures that do not fulfill our requirements of having semantic relations between composing elements using spatial relations. Most of these extracted patterns are actually web tables with NLP text in them, e.g. multi-column blog entries, or descriptions of vendor and price within one single cell. These tables actually contain relevant information, but are not useful for our larger information integration system without detailed parsing of the content of each individual cell. The smaller part are two-dimensional structures that fulfill the purpose of lists. We have tested the simple heuristic of not considering tabular structures, when the number of words within one cell exceeds 20. By adding this heuristic, we could quickly increase the total performance of our system (F-measure $\approx$ 93%).

The time for analyzing one single web page composes three parts: time for downloading a web page, time for retrieving the necessary positional data from Mozilla, and reasoning on the data. Downloading and positional data gathering scales in the order of $\mathcal{O}(n)$ and our reasoning algorithm has time complexity $\mathcal{O}(kn^{0.5})$ where n is the number of element nodes and k the number of appearances of a keyword on a web page. In total, the time needed for the last two steps is on average below one second, but has reached a maximum of 32 seconds for a complicated web page with several tables on a Pentium M computer with 1.6 GHz. In this case, our current implementation returns all individual tables separately.

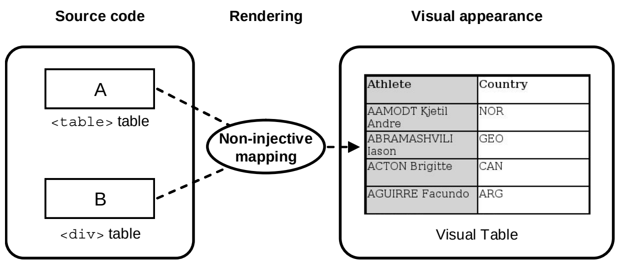
Discussion and Future Work

We started our work under the hypothesis that tabular information on the Web would be encoded in an increasing number with `<div>` instead of `<table>` tags as a result of spread of CSS usage for web page implementation. To our surprise, however, only a small number of encountered web tables (e.g. 1 occurrence in the quantitative test set) use one of, three possible alternatives to `<table>` tables.

1. *ASCII tables.* Our approach fails by its design with this kind of tables, because their logical units are not bound to an element node. The only approaches we know of that could deal in principle with this kind of tabular information are the top-down and bottom-up versions of a word bounding box approach mentioned in (Krüpl, Herzog, & Gatterbauer 2005) and (Krüpl & Herzog 2006). After our analysis of several hundred web tables, we estimate the importance of ASCII tables to be minimal and to decrease even further in the future (e.g. 1 occurrence in the quantitative test set).

2. *LIST tables.* An alternative, which is still used very seldom, consists of vertically arranged horizontal lists. At the moment, our system cannot detect these tables because adjacent list elements are not necessarily aligned on the y_2 dimension. Adding heuristics to our approach can remedy this shortcoming in the future.

3. *DIV tables.* To our surprise, we did not find any single web page that used absolutely or relatively placed `<div>` tags to encode relevant semantic tabular information. As a consequence and in order to test our system, we created our own `<div>` test web TOI listing the international athletes competing in the 2006 Winter Olympics. Our system extracts the tabular information without any problem.



Athlete	Country
AAMODT Kjetil Andre	NOR
ABRAMASHVILI Iason	GEO
ACTON Brigitte	CAN
AGUIRRE Facundo	ARG

Figure 4: Non injection of HTML rendering “renders” reliable table pattern recognition within source code difficult.

We also created a `<table>` version of the previously mentioned `<div>` table that exactly maps to the same visual appearance when rendered in the web browser Firefox (Figure 4). This example proves rendering of HTML code to be a non-injective mapping. As a consequence, tabular structure pattern recognition approaches that operate merely on the source code are inherently more difficult than visual based approaches. Another argument for rendering web pages for web table extraction and interpretation is the point that the relative positions of HTML tags on the screen can not be reliably calculated from the source code without actually rendering a web page as a whole, a bit analogous to the thought that functioning of individual pieces of code within a complex system can not be reliably understood without simulating or "executing" the code as a whole.

Another important observation is that a major part of relevant tabular information (16% in the quantitative testset from Google; 9 out of 56 `<table>` tables) is contained in either non-leaf tables or several vertically arranged tables that only together build the relevant visual table. In both cases, standard approaches that only analyze leaf table nodes actually miss significant parts of relevant tabular information available on the Web.

Our current research goals focus on improving our table extraction performance with heuristics and machine learning techniques, and subsequent table understanding on the extracted tables. In addition to the currently used features for table extraction, we can use the actual size of element nodes as rendered on the screen in addition to specific metadata information like background color. Such background information together with the relative spatial relations play an important role in determining the actual semantic relations between the logical components of tables and the crucial step to extracting not only raw data, but actual information (Figure 2). Please refer to the previously referenced VENTrec web page for information on ongoing work.

Conclusions

Tables on web pages contain a lot of semantically explicit information, which makes them a predestined target for automatic information extraction and knowledge acquisition from the Web. In this paper, we present an innovative and robust approach for finding and extracting tabular data from arbitrary web pages. We propose spatial reasoning on the Visualized Element nodes for table extraction and show that the raw approach produces good results even without any form of training (F-measure $\approx$ 90%). We estimate that applying heuristics and machine learning will bring significant performance improvements and will lay the foundations for robust table understanding in the future.

Acknowledgements

We would like to thank Professor Georg Gottlob for overall motivation of this research, our colleagues Bernhard Krüpl and Marcus Herzog for (at times passionate) discussions of our distinct approaches, Bernhard Krüpl and Tobias Dönz for partial code sharing for the C++ Mozilla component, and the anonymous reviewers for hints that helped us improve the clarity of the paper.

References

Aiello, M. 2002. *Spatial Reasoning: Theory and Practice*. Ph.D. thesis, ILLC, University of Amsterdam.

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11):832–843.

Bilenko, M.; Basu, S.; and Sahami, M. 2005. Adaptive product normalization: Using online learning for record linkage in comparison shopping. In *Proc. ICDM'05*, 58–65. IEEE.

Brin, S. 1998. Extracting patterns and relations from the world wide web. In *Proc. WebDB at EDBT'98*, 172–183. Springer.

Cai, D.; Yu, S.; Wen, J.-R.; and Ma, W.-Y. 2003. Extracting content structure for web pages based on visual representation. In *Proc. APWeb'03*, 406–417. Springer.

Cohen, W. W.; Hurst, M.; and Jensen, L. S. 2002. A flexible learning system for wrapping tables and lists in html documents. In *Proc. WWW'02*, 232–241. ACM.

Etzioni, O.; Cafarella, M. J.; Downey, D.; Popescu, A.-M.; Shaked, T.; Soderland, S.; Weld, D. S.; and Yates, A. 2004. Methods for domain-independent information extraction from the Web: An experimental comparison. In *Proc. AAAI'04*, 391–398. AAAI / MIT.

Gatterbauer, W. 2006. Estimating required recall for successful knowledge acquisition from the Web. In *Poster Proc. WWW'06*. ACM. (to appear).

Handley, J. C. 2000. Table analysis for multi-line cell identification. *Proc. SPIE Vol. 4307, Document Recognition VIII:* 34–43.

Hurst, M. 2000. *The Interpretation of Tables in Texts*. Ph.D. thesis, University of Edinburgh.

Hurst, M. 2001. Layout and language: Challenges for table understanding on the Web. In *Proc. WDA at ICDA'01*, 27–30. IEEE.

Kieninger, T. G. 1998. Table structure recognition based on robust block segmentation. *Proc. SPIE Vol. 3305, Document Recognition V:* 22–32.

Krüpl, B., and Herzog, M. 2006. Visually guided bottom-up table detection and segmentation in web documents. In *Poster Proc. WWW'06*. ACM. (to appear).

Krüpl, B.; Herzog, M.; and Gatterbauer, W. 2005. Using visual cues for extraction of tabular data from arbitrary HTML documents. In *Poster Proc. WWW'05*, 1000–1001. ACM.

Nagy, G., and Seth, S. C. 1984. Hierarchical representation of optically scanned documents. In *Proc. ICPR'84*, 347–349. IEEE.

Penn, G.; Hu, J.; Luo, H.; and McDonald, R. 2001. Flexible web document analysis for delivery to narrow-bandwidth devices. In *Proc. ICDAR'01*, 1074–1078. IEEE.

Pivk, A. 2006. Automatic ontology generation from web tabular structures. *AI Communications* 19(1):83–85.

Simon, K., and Lausen, G. 2005. ViPER: augmenting automatic information extraction with visual perceptions. In *Proc. CIKM'05*, 381–388. ACM.

Wang, Y., and Hu, J. 2002. A machine learning based approach for table detection on the Web. In *Proc. WWW'02*, 242–250. ACM.

Wium Lie, H.; Bos, B.; Lilley, C.; and Jacobs, I. 1998. Cascading Style Sheets, level 2. Technical report, World Wide Web Consortium. See http://www.w3.org/TR/REC-CSS2.

Zhai, Y., and Liu, B. 2005. Web data extraction based on partial tree alignment. In *Proc. WWW'05*, 76–85. ACM.

Zhao, H.; Meng, W.; Wu, Z.; Raghavan, V.; and Yu, C. 2005. Fully automatic wrapper generation for search engines. In *Proc. WWW'05*, 66–75. ACM.