Visually Guided Bottom-Up Table Detection and Segmentation in Web Documents [*]

Bernhard Krüpl
Vienna University of Technology
Institute of Information Systems
Database and Artificial Intelligence Group
kruepl@dbai.tuwien.ac.at

Marcus Herzog
Vienna University of Technology
Institute of Information Systems
Database and Artificial Intelligence Group
herzog@dbai.tuwien.ac.at

ABSTRACT

In the AllRight project, we are developing an algorithm for unsupervised table detection and segmentation that uses the visual rendition of a Web page rather than the HTML code. Our algorithm works bottom-up by grouping word bounding boxes into larger groups and uses a set of heuristics. It has already been implemented and a preliminary evaluation on about 6000 Web documents has been carried out.

Categories and Subject Descriptors: H.3.4 [Information Storage and Retrieval]: Systems and Software; I.7.5 [Document and Text Processing]: Document Capture

General Terms: Algorithms, Experimentation.

Keywords: Web information extraction, table detection.

1. INTRODUCTION

Identifying tables containing product data in Web documents is more difficult than it appears to be at first sight. By just selecting all the HTML table elements, not only data tables are returned, but also tables that are only relevant for formatting reasons (non-genuine tables). On the other hand, authors do not have to use table elements for tabular data presentation (genuine tables) at all, as more and more people choose to do all their formatting based on CSS. While both extremes are not really according to Web standards, they appear abundantly in the real world and should be taken care of by a robust table extraction algorithm.

We based our work on the idea of using the visual rendition of a Web page rather than the HTML code [1] and implemented a genuine table extraction architecture (details in section 2) to produce results as shown in Fig. 1. Section 3 will explain the bottom-up table extraction.

2. ARCHITECTURE

Our table extraction architecture has been built to enable the extraction of visually salient entities. Because most Web pages are designed for an average human audience, the formatting and visual presentation usually also contribute to the semantics of the content.

Analysis of just the HTML (or even XML) source code of a Web document is not sufficent, because the final format of a page is also determined by embedded media and the application of associated style sheets. In the world of Web 2.0, pages can even be constructed dynamically by executing client-side Javascript code. We therefore decided to use a standard Mozilla Web browser that comes with all the interpretation and rendering capabilities already built in.

The extraction algorithms in our architecture operate on the pixel positions of all the words on a Web page: after some investigation, we found that extracting just the rendered bounding boxes for all text elements was not sufficient. We therefore implemented a simple word tokenisation as a first step. Following these considerations, our group implemented a C++ Mozilla component that, for any given URL, returns the bounding boxes of all the words on the corresponding page as rendered by the browser.

In order to come up with a truly robust system, we chose to completely isolate the rendering process from the extraction process. Therefore, we run our Mozilla component as a TCP/IP server that waits for requests and returns all word bounding boxes of a page in the form of an XML document. These results are then read by the actual extraction component that is implemented in Java and allows different algorithms to be plugged in.

3. TABLE EXTRACTION

Because of our approach of using the visual rendition of a Web page, we can base our work on lessons learnt from document image analysis literature. The methods there can be classified along two different dimensions: the direction the algorithms take to construct document hierarchies (bottom-up or top-down), and the strategies they employ in constructing the hierarchies (rule-based or statistical-based) [2]. After experimentation with variants of the classical X-Y cut algorithm [3] (which is top-down) as described in [1], we are now presenting an algorithm that operates bottom-up, starting from the word box document entities retrieved from the browser component. As statistical based methods require an additional offline training process for the estimation of the extraction parameters, we chose rather to implement a set of heuristics that can run completely unsupervised.

Table Types. We are interested in the extraction of tables that we consider to be eupeptic, i.e., easy to digest (understand) for our system. This means that we concentrate on those occurences where we are able to reconstruct the underlying semantics: items in the same row are somehow related to each other, as are items in the same column. While we do not deal with nested tables at this stage, we do handle tables that are divided into sections, usually with

[*]This research is supported in part by the Austrian Federal Ministry for Transport, Innovation and Technology under the FIT-IT contract FFG 809261 and by the REWERSE Network of Excellence.

```
<Attribute>
  <path>Image Quality</path><name>Camera Resolution</name>
  <value>3.3 Megapixel</value>
</Attribute>
<Attribute>
  <path>Image Quality</path><name>Image Resolutions</name>
  <value>640x480, 2048x1536, 1600x1200, 1024x768</value>
</Attribute>
<Attribute>
  <path>Video</path><name>Video Resolutions</name>
  <value>640x480 (VGA), 320x240 (QVGA), 160x120</value>
</Attribute>
```

Figure 1: XML output of the table extraction algorithm.

the occurence of intermediate headings.

Common Extraction Functions. Some functions are of universal use to all of the extraction heuristics we apply later on in the process. We defined a set of common methods that can function as a base for more specialised algorithms.

A basic need for all bottom-up heuristics is the ability to group single word boxes into larger clusters. By doing this clustering first, we can not only reduce the computing time needed, but also identify groups of words belonging together early on in the extraction process. This clustering is done by simply looking at word box neighbourhoods, joining adjacent boxes into larger cells (see step ① in Fig. 2). Since there is inherently no noise in our data (we can rely that the browser rendering is pixel exact; there is no skew), this process is much easier than the equivalent process in a more traditional OCR document understanding context.

We also define some additional distance measures for word boxes and cells. As well as the pixel distance, we can count how many boxes lie in between two given coordinates. This enables us to rule out outliers and account for intermediate headings in a data table.

Table Extraction Heuristics. Our extraction algorithm applies a set of extraction heuristics in two stages.

First, we try to identify possible genuine table columns. In our simplified approach, we define genuine table columns as sets of word cells that, with some small tolerance, share a common coordinate on the horizontal axis. Since table cells can be left-, centre- or right-aligned within their column, we try to map all three properties (left, centre, right coordinate) of all cells to other cells and thus derive column candidates (② in Fig. 2). If any two column candidate cells are separated by more than one cell from the next column candidate cell, the column candidate is broken apart to form two independent columns. If there is exactly one cell separating two column candidate cells, we find out if the separating cell is an intermediate table heading. We collect all intermediate heading candidates of a column candidate and test if they themselves share a common coordinate (③). If they do, we have found a column candidate; if not, we separate the column candidate at the now-imaginary intermediate headings. At this stage we can report back a set of column candidates that must also to fulfil a minimal length requirement.

In the second stage, we try to find the best column candidate combination that could possibly form a data table. We have developed a strategy that we term *comb alignment of columns* (step ④): we look for adjacent columns where there is cell overlap only in one direction. This means that have detected a genuine table, and that we can extract the contents of the segmented table into XML as shown in Fig. 1.

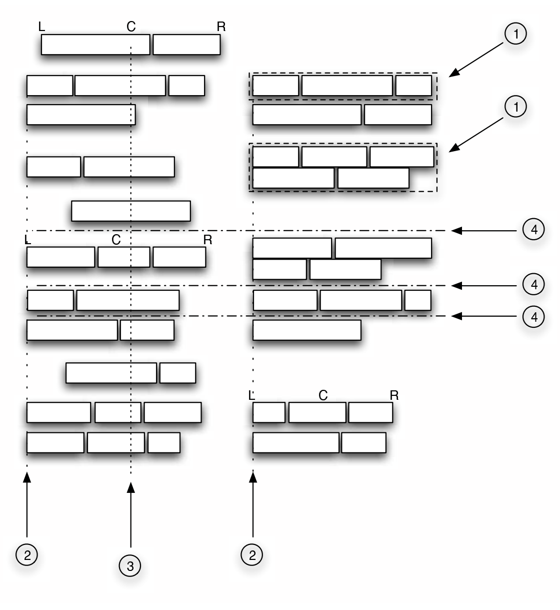

Figure 2: Table extraction algorithm in action.

4. EVALUATION

We performed a preliminary evaluation of our method on about 6000 automatically retrieved Web pages containing keywords from a target product domain. On manual inspection, about 40 per cent of these pages contained genuine data tables. Our system was able to extract these tables correctly in about 70 per cent of all cases.

Limitations. Only those table types that are known in advance are supported by our algorithm. E.g., if there is a vertical gap in table cells, the algorithm currently does not deliver correct results. Also, the process is relatively slow, because it has to wait for Mozilla to render a page.

5. CONCLUSION AND FUTURE WORK

We have introduced a method for genuine table detection and segmentation that follows a visual approach and works completely unsupervised. We believe this to have advantages over methods that just operate on the HTML code level, because we operate on the (visual) layer actually meant to be read.

For the future, we plan to formalise the heuristics using an external representation, thus enabling reasoning and a more systematic way of finding the best combination of the different rules. We also believe that it may be useful to look at additional rendered page characteristics, such as borders and separator graphics.

6. REFERENCES

[1] B. Krüpl, M. Herzog, and W. Gatterbauer. Using Visual Cues for Extraction of Tabular Data from Arbitrary HTML Documents. In *Proc. of the 14th Int. World Wide Web Conf.*, pages 1000–1001, 2005.

[2] J. Liang, I. Phillips, R. Haralick. An Optimization Methodology for Document Structure Extraction on Latin Character Documents. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 7, pages 719–734, 2001.

[3] G. Nagy and S. Seth. Hierarchical representation of optically scanned documents. In *Proc. of the 7th Int. Conf. on Pattern Recognition*, pages 347–349, 1984.