Using Visual Cues for Extraction of Tabular Data from Arbitrary HTML 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

Wolfgang Gatterbauer
Vienna University of Technology
Institute of Information Systems
Database and Artificial Intelligence Group
gatter@dbai.tuwien.ac.at

ABSTRACT

We describe a method to extract tabular data from web pages. Rather than just analyzing the DOM tree, we also exploit visual cues in the rendered version of the document to extract data from tables which are not explicitly marked with an HTML `table` element. To detect tables, we rely on a variant of the well-known X-Y cut algorithm as used in the OCR community. We implemented the system by directly accessing Mozilla's box model that contains the positional data for all HTML elements of a given web page.

1. INTRODUCTION

Most of today's web documents are designed for a human audience. Although many efforts have been undertaken to bring explicit semantics to the Web, the vast majority of pages is designed with a certain visual appearance in mind: authors use HTML rather as a page layout language than for the purpose of semantic markup. However, there is a common misunderstanding that such pages are semantically poor. Instead, the semantics is just shifted from an explicit level (proper HTML or XML tags) to an implicit one: the spatial alignment of the document text on the page.

Documents have come a long way from the mere sequential order of sentences to sophisticated layouts following different typesetting conventions and fashions. Therefore it seems quite natural to exploit this additional information for information extraction applications.

Web layouts can be achieved with different methods ranging from basic HTML markup to fancy CSS stylesheets and dynamic client-side programming. Still, most web information extraction programs operate just on the DOM tree where the spatial information cannot by directly accessed. By utilizing the screen rendering provided by the open source browser Mozilla we are able to exploit this spatial information during the extraction process.

2. WEB PUBLICATION PROCESS

The publication of a web page can be understood as a communication process from persons to persons (see figure 1).

WWW2005, May 10–14, 2005, Chiba, Japan.
.

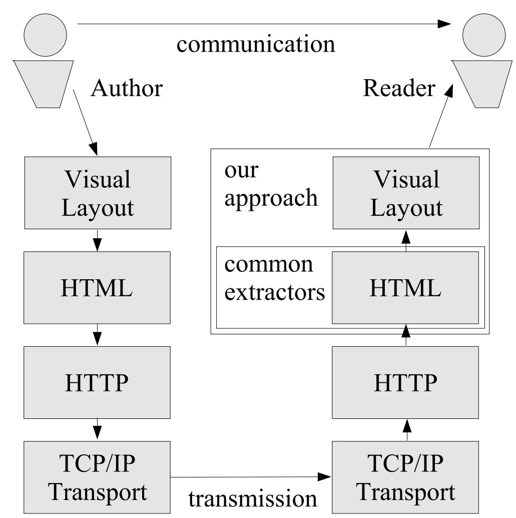

Figure 1: Layers of abstraction in the web publication process

Starting at the left-hand side, an author edits a Web page, often by using a visual editor. The result of this step is a certain HTML (possibly including some CSS) source code. This initial representation of the communication content is then gradually transformed for transmission (going down the stack of communication layers). At the receiver's side, the transformations are applied the other way around, moving the information up the stack. In the last step, a web browser creates a visual rendering from the supplied HTML code by applying a rendering algorithm.

It is quite obvious that the format at the transport layer is not useful at all for information extraction. The lower information moves down in the transformation stack, the more noise and redundancy is added. On the opposite, information in its purest form can be found when it is as close to the receiver as possible, in this case, the visual rendering, not its encoding in various formats.

See [4] for an inspiring discussion how information is transmitted between two persons.

```
<p>Here is some important information:</p>
<table>
  <tr><td>The</td><td>Quick</td></tr>
  <tr><td>Brown</td><td>Fox</td></tr>
</table>

<div style="position: absolute; left:74; top:68;">
<p>Fox</p></div>
<div style="position: absolute; left:74; top:42;">
<p>Quick</p></div>
<div style="position: absolute; left:8; top:1;">
<p>Here is some important information:</p></div>
<div style="position: absolute; left:10; top:42;">
<p>The</p></div>
<div style="position: absolute; left:10; top:68;">
<p>Brown</p></div>
```

Figure 2: Two different chunks of HTML code leading to the same visual rendering

3. BACKGROUND

As a part of a web data extraction project, we need to automatically augment a domain model with additional information extracted from web pages. We employ Named Entity Recognizers (NERs) [2] to find interesting text nodes on a given page. As the analysis of tabular data is most promising for our augmentation task, we try to find out if those particular nodes are part of a table. In the case of a table defined with HTML `table` elements, this is trivial. In the case of other HTML elements like `div`, the hierarchical order of nodes in the document model does not necessarily correspond to the order in the visual representation. In the later case the problem of tabular data extraction is better addressed on the presentational layer, i.e., on the rendering supplied by the web browser (see Figure 2).

When we look at the two HTML source code examples in Figure 2, it is not obvious at all that the visual rendering provided by a web browser is the same, as it is actually the case. In the `div` sample, proximity in the DOM tree does not correspond to proximity in the visual rendering. From a user's point of view it is quite clear that both examples have to be interpreted the same way – users do not care about internal page representations.

4. IMPLEMENTATION

4.1 Positional data

Writing a modern web browser adhering to all standards out there is a very complicated task. As it is not feasible for our extraction system to re-implement all the rendering abilities of such a system, we chose to rely on the Gecko rendering engine that is built-in into Mozilla. Mozilla internally uses the so-called box model, where the bounding boxes of all rendered nodes are stored. We access this positional information from within our program by using XPCOM bindings.

4.2 Detection algorithm

In the OCR communtiy, the well-known X-Y cut algorithm [3] is used for page segmentation. The algorithm works by projecting the document bitmap (i.e., summing up all the pixels in a line) to the sides of document page. By this method, a white space density graph is produced, with peaks for vertical or horizontal lines surrounded by whitespaces. These peaks define the cuts of the document and are used to segment the document into smaller pieces. In a variant [1], the algorithm does not operate on the document bitmap itself but rather on the bounding boxes of connected components (typically characters). While this variant was developed for speed optimization reasons, we came to the conclusion that it is a valuable method to process the positional information we gain from the Mozilla browser, which is also encoded using bounding boxes. (With the notable distinction that the Mozilla bounding boxes are defined for DOM nodes, not for single characters.) In the X-Y cut algorithms, the cuts are recursively applied to the document, and the found segments are stored hierarchically in a X-Y tree – the X-Y tree shows the hierarchical subdivision of a page by recursive (X-horizontal and Y-vertical) cuts. (The root node is the bounding box of the full page.)

4.3 Table detection

Since we apply the algorithm on the bounding boxes of HTML nodes, we are especially interested in ancestor nodes of leaf nodes (which correspond to the HTML nodes) of the X-Y tree. If we can find an ancestor node that contains a certain number of leaf nodes which in turn contain named entities (as found by our NERs), we assume that the X-Y node found represents a table. (Note that the NERs of course operate on the DOM tree.) Part of the named entity recognition process is a simple heuristics that is applied to make sure that a node does not contain too much other content besides the recognized entity. This helps us to identify the data-centric tables we are actually looking for – the assumption is that with growing table cell content, the semantic relations between table cells get weaker.

5. FUTURE WORK

We plan to further investigate our combined visual and DOM tree approach to the extraction problem by applying it to the real world scenario of extracting product attributes from web tabular data in the digital camera domain. Also, we currently investigate how spatial distance measures can help us to address the problem of the table structure recognition [5] of nested tables.

6. REFERENCES

[1] Ha Jaekyu, R.M. Haralick, and I.T. Phillips. Recursive x-y cut using bounding boxes of connected components. In *Proc. of the 3rd Int. Conf. on Document Analysis and Recognition*, pages 952–955, 1995.

[2] A. Mikheev, M. Moens, and C. Grover. Named entity recognition without gazetteers. In *Proc. of the 9th Conf. of the European Chapter of the Assoc. for Computational Linguistics*, 1999.

[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.

[4] Tor Norretranders. *The User Illusion*. Penguin Books, 1998.

[5] R. Zanibbi, D. Blostein, and J.R. Cordy. A survey of table recognition: Models, observations, transformations, and inferences, 2003.