End-to-End Conversion of HTML Tables for Populating a Relational Database

George Nagy
Rensselaer Polytechnic Institute, nagy@ecse.rpi.edu

David W. Embley
Brigham Young University, embley@cs.byu.edu

Sharad C. Seth
University of Nebraska-Lincoln, seth@cse.unl.edu

Follow this and additional works at: http://digitalcommons.unl.edu/cseconfwork

Part of the Computer Engineering Commons, Electrical and Computer Engineering Commons, and the Other Computer Sciences Commons

Nagy, George; Embley, David W.; and Seth, Sharad C., "End-to-End Conversion of HTML Tables for Populating a Relational Database" (2017). CSE Conference and Workshop Papers. 296.
http://digitalcommons.unl.edu/cseconfwork/296

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Conference and Workshop Papers by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.
End-to-End Conversion of HTML Tables for Populating a Relational Database

George Nagy
Rensselaer Polytechnic Institute
Troy, NY, USA

Sharad Seth
University of Nebraska, Lincoln
Lincoln, NE, USA

David W. Embley
Brigham Young University
Provo, UT, USA

Abstract—Automating the conversion of human-readable HTML tables into machine-readable relational tables will enable end-user query processing of the millions of data tables found on the web. Theoretically sound and experimentally successful methods for index-based segmentation, extraction of category hierarchies, and construction of a canonical table suitable for direct input to a relational database are demonstrated on 200 heterogeneous web tables. The methods are scalable: the program generates the 198 Access compatible CSV files in ~0.1s per table (two tables could not be indexed).

Keywords—table segmentation, table index, Wang category, header factoring, header cross-product, canonical relational table

I. INTRODUCTION

Several successful methods have been demonstrated for locating and delimiting HTML tables in spite of the common use of “table” tags for non-table page layout [1]. For many years spreadsheet software has offered built-in functions to import HTML tables in either a proprietary format that preserves both structure and formatting or as comma-separated values (CSV files) that retain the fundamental grid structure but not most cell formatting.

Table segmentation, i.e., the separation of table headers, notes and footnotes from data cells, has proved more difficult. It typically relies on column-headers located at the top, and row headers at the left of the table, and on only moderately reliable heuristics based on alphabetic header content and numerical data [2,3]. Beyond segmentation, it is even more difficult to assign a unique row and column index (that may span several header rows) to every data cell. Finally, we do not know of any research that attempts to analyze and extract category headers as prescribed by X. Wang [4].

Yet all of the above is necessary in order to populate a database with web table content. The advantage of doing so has long been recognized [5,6]. A DBMS provides query and retrieval functions that allow combining information from several tables in the form of new tables [7]. Although printed and HTML tables are logically symmetric in row and column organization, relational tables are not: rows are records (or tuples), and columns are fields (or entities). This organization opens the way for a wealth of useful operations based on predicate calculus and governed by the well-established concepts of relational algebra and calculus [8]. With the proposed transformation, the uniqueness of table indices is reflected by the notion of keys, and the grid structure by the fixed arity of tuples.

We demonstrate below three contributions, all of which we consider essential for effective computer search of HTML table contents:

1. Table segmentation based on the fundamental indexing property rather than on appearance features. Our experiments on segmentation of indexing (i.e., minimal) headers and data regions yield 99.5% agreement with human ground truth, as opposed to 85% with appearance features [9]. The single error on 200 tables reflect ground truth judgments and will not hamper database operations.

2. Extraction of category structure based on “factoring” prefixed headers. We detect all 21 multi-category headers in the 200 tables, with no false positives. An example of a multi-category rowheader column is: [Blois, Male, Female, Tours, Male, Female]T. The two mutually exclusive and exhaustive (or orthogonal) categories are {Blois, Tours} and {Male, Female}. In real tables the categories may be spread over several header rows or columns and their extraction is considerably more complicated.

3. Resolution of the apparent conflict between the symmetric structure of ordinary tables and the asymmetric organization of relational tables by transforming the CSV tables imported from the web into a canonical format suitable for immediate input to a database. The canonical CSV table’s rows are assembled by concatenating every row and column header path of the original table with the data cell string that it indexes. This format preserves the category structure. Furthermore it allows, by means of a pivot operation, to use any category for fields (attribute names) and the remaining categories as keys, regardless of the original layout of the table.
Our work leading up to the current process includes examination of egregious tables [10] and studies of table semantics [11,12]. Our 2006 survey [13] is largely obsolete in view of recent work by teams sponsored by Google [14], Yahoo [15], Citeseer [16, 17] and by other academic groups [18, 19, 20]. Current research by Adelfio and Samet on table semantics [11,12]. Our 2006 survey [13] is largely obsolete in view of recent work by teams sponsored by Google [14], Yahoo [15], Citeseer [16, 17] and by other academic groups [18, 19, 20]. Current research by Adelfio and Samet on table semantics [11,12].

Because of space limitations, only brief descriptions and examples of our approach to segmentation, category extraction and canonical transformation are given in Sections 2, 3, and 4. In Section 5 we present our experiments and results on tables culled from the web. Section 6 summarizes this research and proposes topics for further investigation.
of a pair or header rows with the length of the header. The two rows
of the column header of Fig. 1 constitute two categories because the
Cartesian product \((\{\text{Schools, Pupils}\} \times \{\text{Less than} 100, 100-299, 300 \text{ or more}\})\) of the two rows has 6
elements, which is exactly the length of the column header.

The detection of multiple header categories based on the
cross-product has been implemented only for pairs or rows or
columns. For greater generality we use header factoring [24],
which can handle arbitrary sized headers.

For example, the initial algebraic expression used for
"factoring" the column header of the table in Fig. 1, is obtained
by tracing the header paths from the left to right:

\[
\text{Sch}*<100 + \text{Sch}*100-299 + \text{Sch}*>300 + \text{Pup}*<100 + \text{Pup}*100-299 + \text{Pup}*>300
\]

Note that the * and + operations in the expression represent
vertical and horizontal concatenation, respectively. After
factoring, the non-singleton sum terms in the top-level product
correspond to the categories:

\[
(\text{Sch + Pup})*(<100 + 100-299 + >300)
\]

IV. CANONICAL RELATIONAL TABLE

The contents of the table of Fig. 1 can be laid out by the
table designer in 3x2 different ways, (besides permuting rows
and columns), because each of the three categories can form a
complete row or column header. Fig. 2 shows an alternative
layout for the same table.

It is also possible to lay out the table in with only a single
column of data. Part of such a 36-row table is rendered in Fig.
5. Any of the six configurations can be obtained from this M×1
form by pivoting [7] one of the key fields of the first three
columns. This is, therefore, our choice for converting the
information into a canonical relational form. The CSV file of
Fig. 5 was generated by looping through the rows and columns
of the (possibly prefixed) headers, and appending the data
values from the original table. The headers of the canonical
table, which will become field names in the database, preserve
the category structure of the original table. The canonical table
is ready for loading into Access or any other database.

Aggregates like TOTAL and Percent Change are position-
dependent, therefore it would be best to omit them from any
web table before its transformation into canonical form. Any
DBMS worth its salt can produce all kinds of aggregates: only
the data itself must be preserved. We don’t yet have a reliable
method for removing aggregates, but others, especially V.
Long, have pointed the way [19].
### Table 4. Patents granted in Finland by IPC section in 2008

<table>
<thead>
<tr>
<th>Industry</th>
<th>Patents granted in Finland</th>
<th>European patents validated in Finland</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Finnish avg</td>
</tr>
<tr>
<td>A: Human necessities</td>
<td>121</td>
<td>58</td>
</tr>
<tr>
<td>B: Performing operations,</td>
<td>195</td>
<td>179</td>
</tr>
<tr>
<td>C: Chemistry, metallurgy</td>
<td>127</td>
<td>47</td>
</tr>
<tr>
<td>D: Textiles, paper</td>
<td>119</td>
<td>99</td>
</tr>
<tr>
<td>E: Fixed constructions</td>
<td>31</td>
<td>27</td>
</tr>
<tr>
<td>F: Mechanical engineering</td>
<td>78</td>
<td>69</td>
</tr>
<tr>
<td>G: Physics</td>
<td>137</td>
<td>118</td>
</tr>
<tr>
<td>H: Electricity</td>
<td>190</td>
<td>132</td>
</tr>
</tbody>
</table>

Fig. 6. A web page from Finland with a common configuration of a two-category column header. The total row contains aggregates.

Fig. 7. CSV rendition of the three-category HTML table of Fig. 6. Empty cells will be left filled. Same-row prefixing is necessary for the seventh row. The table title is in the fifth row, which is unusual.

### Table A-3: Top 10 U.S. Land Ports for Land Trade with Canada and Mexico: 2003 and 2004

<table>
<thead>
<tr>
<th>U.S. Port</th>
<th>All land modes (Thousands of current U.S. dollars)</th>
<th>Truck</th>
<th>Rail</th>
</tr>
</thead>
<tbody>
<tr>
<td>U.S.-Canada Trade</td>
<td>2003</td>
<td>2004</td>
<td>Percent change</td>
</tr>
<tr>
<td>Top 10 ports</td>
<td>362,319,129</td>
<td>408,612,969</td>
<td>12.8</td>
</tr>
<tr>
<td>Detroit, MI</td>
<td>101,815,113</td>
<td>113,668,714</td>
<td>11.6</td>
</tr>
<tr>
<td>Buffalo-Niagara, NY</td>
<td>59,275,775</td>
<td>68,283,239</td>
<td>12.3</td>
</tr>
<tr>
<td>U.S.-Mexico Trade</td>
<td>200,457,309</td>
<td>224,949,742</td>
<td>12.2</td>
</tr>
<tr>
<td>Top 10 ports</td>
<td>190,980,524</td>
<td>211,303,066</td>
<td>10.5</td>
</tr>
<tr>
<td>Laredo, TX</td>
<td>78,762,959</td>
<td>89,510,852</td>
<td>12.6</td>
</tr>
<tr>
<td>Buffalo-Niagara, NY</td>
<td>59,369,091</td>
<td>68,351,546</td>
<td>13.6</td>
</tr>
<tr>
<td>Detroit, MI</td>
<td>101,815,113</td>
<td>113,668,714</td>
<td>11.6</td>
</tr>
<tr>
<td>Port Huron, MI</td>
<td>62,244,347</td>
<td>65,879,966</td>
<td>12.5</td>
</tr>
</tbody>
</table>

Fig. 8. Excerpts from a 36-row table that requires a two-column row index because of duplicate ports under each of three trade headings. There is only one row category because the replication is incomplete. But there are two column categories {All land modes, Truck, Rail} and (2003, 2004, Percent change).

The canonical table produced has (36 × 9) + 1 = 325 rows. The first two rows are shown below the table.

### Table 9: U.S.-Canada Trade

| Last modified: 2007-10-30 |

TS1

<table>
<thead>
<tr>
<th>Industry</th>
<th>Patents granted in Finland</th>
<th>European patents validated in Finland</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Finnish avg</td>
</tr>
<tr>
<td>A: Human necessities</td>
<td>121</td>
<td>58</td>
</tr>
<tr>
<td>B: Performing operations,</td>
<td>195</td>
<td>179</td>
</tr>
<tr>
<td>C: Chemistry, metallurgy</td>
<td>127</td>
<td>47</td>
</tr>
<tr>
<td>D: Textiles, paper</td>
<td>119</td>
<td>99</td>
</tr>
<tr>
<td>E: Fixed constructions</td>
<td>31</td>
<td>27</td>
</tr>
<tr>
<td>F: Mechanical engineering</td>
<td>78</td>
<td>69</td>
</tr>
<tr>
<td>G: Physics</td>
<td>137</td>
<td>118</td>
</tr>
<tr>
<td>H: Electricity</td>
<td>190</td>
<td>132</td>
</tr>
</tbody>
</table>

Fig. 9. Excerpt from a table where the row header requires three columns because of duplicate entries.
Algorithmic, rather than heuristic, methods were demonstrated for entering CSV tables imported from the web into a relational DBMS. The experiments show that the proposed approach can handle large, complex and heterogeneous tables, from diverse sources, fast enough for production operation. To the best of our knowledge, this is the first end-to-end method for converting arbitrary web tables into an Access-compatible format that preserves the data cell indexing and category structure defined by the headers.

The only heuristics that are part of the program are those used to detect non-data rows above and below the data region. It is possible that these will need modification for new configurations of web tables. The current method depends on the detection of some empty cells or uniform rows (e.g. units) below the header or at the bottom of the table. Fortunately, enlarging the data regions to include these cells does not prevent conversion of the table into a useful database format.

The last two decades have seen many publication describing the conversion of printed, and even-hand-printed, tables to their underlying grid structure. Some of the proposed methods include heuristics for detecting at least column headers. The method suggested here could be applied to imperfectly OCR’d tables provided that the exact cell-content string-matching used in our indexing and category detection schemes is modified to use approximate string-matching. The number of induced indexing errors would depend, of course, on the accuracy of the OCR’d cell contents.

We intend to experiment next with web tables imported into Access to discover what type of useful new information can be queried by combining tables from the same or different sources. We would like to combine factoring and cross-products to spot aggregates, which often give rise to multi-category headers. Another goal is to discover how to make full use of the header hierarchies and auxiliary information (like footnotes and footnote references) that we can already extract. The exploitation of auxiliary items is also an issue in documents without tables.

ACKNOWLEDGMENT
Professor M.K. Krishnamoorthy (RPI) made significant contributions to our analysis of tables and checking of results.

REFERENCES