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]. 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.
proposes topics for further investigation.

analyses are less fine grained.

experiment on much larger data sets than we do, but their
web tables [22] is very much in the spirit of our work. They

Current research by Adelfio and Samet on table
view of recent work by teams sponsored by Google [14],
semantics [11, 12]. Our 2006 survey [13] is largely obsolete in
examination of egregious tables [10] and studies of table

In Section 5 we present our experiments and results on tables
and canonical transformation are given in Sections 2, 3, and 4.

examples of our approach to segmentation, category extraction
and classification of schema extraction [21] and by Lautert, Scheidt and Dorneles

Because of space limitations, only brief descriptions and
elements 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.

II. SEGMENTATION

The table is segmented by first finding the Minimum
Indexing Point (MIP) [23] and then the four Critical Cells
(CC1 and CC2 for the stub header, CC3 and CC4 for the data
region) [9]. Cells unmerged to accommodate CSV are left and
top filled with the preceding cell content. The algorithm of [23]
searches for the MIP (CC2) from the top left corner and
backtracks from redundant (for indexing) rows or columns. The remaining CCs are found heuristically by deleting sparse
rows from the data region below and to the right of the MIP.

The segmentation is exemplified by the small Statistics Norway web table of Fig. 1a. It is shown in Fig. 1b as imported
into a spreadsheet in CSV format.

The segmented table with its indexing headers is displayed
in Fig. 1c. The MIP (or CC2, here A3) is found by searching
from cell A1 for unique column header columns and unique
row header rows. The unnecessary rows above the header are
eliminated by backward sweep from the MIP of the minimum
indexing rows. The CCs in Fig. 1c are A2, A3, B4, and
G10. Therefore the column header is B2:G3 and the row
header is A4:A10.

Rows below CC4 (G10) are eliminated. The footnote in
A11:B11 is linked to its reference marker in A10. Note that
conversion to CSV resulted in demotion of the superscript "1".
Empty rows and rows with no data (there are none here) are
retained and marked, but not used for segmentation. Prefixing
is required when the repetition of cell labels prevents unique
indexing. Prefixing prepends an additional header row, as
required by the column header of Fig. 3.

Because of space limitations, only brief descriptions and
elements 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 (\{Schools, Pupils\} × \{Less than 100, 100-299 pupils, 300 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:

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

IV. CANONICAL RELATIONAL TABLE

The contents of the table of Fig. 1 can be laid out by the table designer in \(3 \times 2\) 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 \(M \times 1\) 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 \[19\].

V. EXPERIMENTS

All of the above operations are part of a Python 2.7 module running under IDLE. The program was used to convert 200 web tables imported earlier into Excel \[25, 26\] to canonical form. Only one segmentation error was detected by comparing the Critical Cells generated with Ground Truth. Whether this particular instance does really constitute an error depends on the acceptability of an empty cell as a column header. The program accepted it, but the Ground Truth did not. (Our earlier attempt at segmentation based on appearance features had a 15% error rate prior to interactive correction \[27\].) Only 20 two-category headers were found using the cross-product, but all 21 were found by factoring. Neither method generated any false positives.

The tables, all from large international statistical websites, vary greatly in layout, size, and content. The distribution of column-header sizes (after prefixing) is as follows: 69% had only one-row headers; 26.5% had at least one two-row header; and 4.5% (9 tables) had at least one three-row header. Two tables could not be indexed because of repeated rows or columns.

All of the footnotes were found in the 33% of the tables that had them. The program detected 218 reference marks to the footnotes within the body of the tables (some had more than a dozen). It missed them in three tables where the footnote reference marks were not near the end of the cell text.

The entire processing for all 200 tables, including writing the 198 files for input to Access, required 27 seconds. A processed web table is shown in Figs. 6-7. Parts of larger tables are rendered from the imported .CSV files in Figs. 8-9.
Table A-3: Top 10 U.S. Land Ports for Land Trade with Canada and Mexico: 2003 and 2004

(Thousands of current U.S. dollars)

<table>
<thead>
<tr>
<th>Excel</th>
<th>CSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>U.S. Port</td>
<td>All land modes</td>
</tr>
<tr>
<td>U.S.-North American Trade</td>
<td>562,776,436</td>
</tr>
<tr>
<td>Top 10 ports</td>
<td>412,424,713</td>
</tr>
<tr>
<td>Detroit, MI</td>
<td>101,889,513</td>
</tr>
<tr>
<td>Laredo, TX</td>
<td>78,762,959</td>
</tr>
<tr>
<td>Buffalo-Niagara, NY</td>
<td>59,369,091</td>
</tr>
<tr>
<td>U.S.-Canada Trade</td>
<td>362,319,128</td>
</tr>
<tr>
<td>Top 10 ports</td>
<td>288,166,879</td>
</tr>
<tr>
<td>Detroit, MI</td>
<td>101,815,113</td>
</tr>
<tr>
<td>Buffalo-Niagara, NY</td>
<td>59,275,775</td>
</tr>
<tr>
<td>Port Huron, MI</td>
<td>62,244,347</td>
</tr>
<tr>
<td>Champlain-Rouses Pt., NY</td>
<td>14,412,634</td>
</tr>
<tr>
<td>Blaine, WA</td>
<td>12,905,376</td>
</tr>
<tr>
<td>Alexandra Bay, NY</td>
<td>10,035,184</td>
</tr>
<tr>
<td>U.S.-Mexico Trade</td>
<td>200,457,309</td>
</tr>
<tr>
<td>Top 10 ports</td>
<td>190,980,524</td>
</tr>
<tr>
<td>Laredo, TX</td>
<td>78,762,959</td>
</tr>
<tr>
<td>El Paso, TX</td>
<td>39,204,331</td>
</tr>
<tr>
<td>El Paso, TX</td>
<td>19,678,318</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 \((36 \times 9) + 1 = 325\) rows. The first two rows are shown below the table.

<table>
<thead>
<tr>
<th>RowCat_1.1</th>
<th>RowCat_1.2</th>
<th>ColCat_1.1</th>
<th>ColCat_2.1</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>U.S.-North American Trade</td>
<td>ditto</td>
<td>All land modes</td>
<td>2003</td>
<td>562,776,436</td>
</tr>
</tbody>
</table>

Fig. 9. Excerpt from a table where the row header requires three columns because of duplicate entries.
VI. CONCLUSION

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