Discovery and Verification of Computed Data Values in Heterogeneous Web Tables

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

Mukkai Krishanmoorthy
Rensselaer Polytechnic Institute
Troy, New York, USA

George Nagy
Rensselaer Polytechnic Institute
Troy, New York, USA

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

***Abstract*—Based on automated analysis of table header structures, keyword labels, and layout patterns, computed data values are detected and checked in a corpus of 200 heterogeneous web tables from international sites.**

Keywords—table processing; table structure; computational keywords, computational layout patterns

I. Introduction

Automated end-to-end table interpretation requires not only discovery of a table's structure—its row and column headers and how they index the data values—but also which of the table's data values are computed and how they are computed from other data values. This was demonstrated by V. Long on simple financial reports [1]. Automated discovery of more complex table structure is non-trivial but once discovered it provides valuable structural and keyword clues that lead to the discovery of a table's computational formulas. Verification of candidate computational formulas requires applying operator functions to check computed values.

For automated data analysis, computed values, including aggregates like the sums in the first two columns in Fig. 1, should be clearly identified as derived values. Our approach to discovering and verifying computational properties of tables is based on definitions of well-formed table structure (row and column headers and their relationship to the data values they index), header keywords such as "total", "average", and "percent difference", and well-formed computational layout patterns such as column aggregates in Fig. 1 and aggregates associated with parent-node labels in Fig. 2.

II. Well-Formed Table Structure

Formally defined (see [2]), well-formed tables have n ($n \geq 2$) header category trees (often degenerate) whose root to leaf header paths uniquely index the table's data values. In Fig. 1, the three category trees are (*Country*, (*Norway*, *Germany*, *total*)), (*Year*, (*2004*, *2005*)), (*Assistance*, (*Million$*, *%GNI*)). The data value *2 199* is indexed by *Country.Norway*, *Year.2004*, *Assistance.Million$*.

The table-structure processing steps are as follows:

Assistance				
	Million$		%GNI	
Country	2004	2005	2004	2005
Norway	2 199	2 786	0.87	0.84
Germany	7 534	10 082	0.28	0.36
total	9 733	12 868	0.33	0.41

Fig. 1. Table 22 in our 200-table corpus, with some shortened text and reduced to two countries.

Assistance		
Country	Norway	Germany
Million$	4 985	17 616
2004	2 199	7 534
2005	2 786	10 082
%GNI	0.86	0.32
2004	0.87	0.28
2005	0.84	0.36

Fig. 2. Table with sums and averages associated with hierarchical parent nodes.

(1) Segment the table to identify row and column headers, data values, and ancillary information like table title, footnotes, notes, and empty cells.
(2) Parse numeric fields to facilitate computations.
(3) Classifies each cell.
(4) Factor each header to reveal category trees.

Fig. 3 shows part of the classification table (as a relational database table) obtained by the first three structure-processing steps for the table in Fig. 1. Note that the *Million$* values in Fig. 1 are formatted with a space-separator for thousands. In general, the parse must recognize numbers in all forms and convert them appropriately.

Cell_ID	Row	Col	Content	Class
T022_R1_C1	1	1	Assistance	tabletitle
…				
T022_R3_C2	3	2	2199	data
…				
T022_R4_C2	4	2	2037	data
…				
T022_R5_C2	5	2	4236	data
…				

Fig. 3. Generated classification table.

In Step 4, the category-trees are discovered by factoring a sum-of-products expression from an indexing column or row header [3]. In column headers, each column is a product of labels, which are summed across the columns. For the column header in Fig. 1, the expression *Million$*×*2004* + *Million$*×*2005* + *%GNI*×*2004* + *%GNI*×*2005* factors as (*Million$* + *%GNI*) × (*2004* + *2005*); and for the row header, the expression (*Norway* + *Germany* + *total*) is already in

factored form. Each of these parenthesized expressions represents a category, which with the addition of a header root node becomes a category tree. In our implementation we add virtual root nodes since we do not yet match the category values with a semantic resource to infer actual root-node label names. Note that for the *Country* category, if we determine that *total* is a header for computed values, we can remove it from the category tree, leaving in this case a clean set of country names for semantic matching; and, as we show in Section IV, a way to identify operand values for the computed values in the *total* row.

Our table-processing system properly segmented and produced a classification table for 198 of the 200 tables in our heterogeneous corpus (the two contained errors, duplicate header labels, making indexing non-unique). All numeric data values were parsed correctly. Factoring succeeded on all 21 of the non-trivial, multi-category row (7) and column (14) headers.

III. Computational Keywords

The appearance of keywords, such as *total* and *%*, often suggest both the location and function of a computation. Table 2 shows the distribution of seven case-independent keyword stems commonly associated with computations in our collection of 200 tables. These keywords appear in 79% of the tables. *Total* and its synonym *All* are by far the most prevalent. Aggregate keywords (*total*, *all*, *average*) appear significantly more often in row headers, while non-aggregate keywords (*change*, *%*, *percent*, *balance*) are much more common in column headers. Among the tables with keywords, 59% include single occurrences and 78% include at most two occurrences.

Table 1. Distribution of 275 computational keywords.

Key-word	Row Headers	Column Headers	Total
total	35%	21%	56%
all	11%	4%	15%
average	3%	1%	4%
%	0%	7%	7%
percent	2%	4%	6%
change	3%	8%	11%
balance	0%	1%	1%

Computational keywords do not necessarily denote computed values. In a sample of 42 tables selected randomly from our 200 tables, only 60% of the keywords correspond to computed values (true positives); the rest are false positives (e.g., "Total patents granted"). Moreover, only 70% of the computed values occur with a keyword in the header. The table in Fig. 2, for example, contains no computational keywords in row headers indexing computed data values.

IV. Well-Formed Computational Patterns

Summation over a column of data values, as in the first two columns in Fig. 1, is a common computational pattern. Formally, we may represent the pattern as

$$(total,Million\$,Year) = \sum_{Country} data_{(Country,Million\$,Year)}$$

where the triples index data cells, the summation over the *Country* category omits the *total* row, and *data* is the designator of a data value (as in Fig. 3).

Fig. 2 shows a common pattern for row-header hierarchies: aggregates at a root over immediate children. *Million$* aggregates are sums across the years, and *%GNI* aggregates are averages across the years. The *%GNI* "totals" in Fig. 1 are computed by a complex pattern of computing the *GNI* for each country for each year, summing these *GNI*'s for each year, and finally computing the *%GNI* as (*GNI*/*Million$total*)×100.

As an example of finding and verifying computed values in tables, we wrote an SQL query that checks for the summation computational pattern in Fig. 1 and also another query that checks the complementary row-total pattern. Fig. 3 shows the relational classification table generated for the Table in Fig. 1. Since all classification tables have the same uniform schema, we can write a single query to find computational patterns in all tables and check the column (row) sum of the non-total column (row) data values against the tabulated total.

We applied this total's check query and its complement query to all 198 tables for which classification tables were generated. The results appear in Table 2. Some web tables included non-numeric string values and some marked empty cells or value-hidden cells with a non-numeric mark such as "x" or "-". Other tables included partial sums, or pseudo-totals like "Total factor productivity". A surprising number of tables included one or more incorrect stated totals.

Table 2. Results of total-pattern SQL query.

Our sample of web tables	200
Well-Formed Tables	198
Tables with a single row header that includes "Total" or "total"	72
Column-sum pattern satisfied	23
Exact match of computed sum	7
Tables with a single column header that includes "Total" or "total"	47
Row-sum pattern satisfied	21
Exact match of computed sum	13

V. Summary and Conclusions

Ordinary tables are wonderfully well suited for human assimilation, but in the world of Big Data facts must often be assembled and agglomerated from multitudes of disparate and heterogeneous tables. Increasing emphasis on data provenance, integrity and verifiability accentuates the desirability of separating independent data values from derived data values. Because tables, and especially table headers, carry considerable structure, syntactic and algorithmic methods can take us surprisingly far towards table understanding and interpretation and can add enormous value to petabytes of machine-readable, but not yet machine-understandable, information.

References

[1] V. Long, An Agent-Based Approach to Table Recognition and Interpretation, Macquarie University PhD dissertation, May 2010.

[2] D.W. Embley, S. Seth, G. Nagy, "Clustering Header Categories Extracted from Web Tables," Procs. DR&R 2015, San Francisco, California, February 2015.

[3] D. W. Embley, M. Krishnamoorthy, G. Nagy, and S. Seth, "Factoring Web Tables," K.G. Mehrotra et al. (Eds.): IEA/AIE 2011, Part I, LNAI 6703, pp. 253–263, 2011. © Springer-Verlag Berlin Heidelberg 2011.