DISCOVERY AND VERIFICATION OF COMPUTED DATA VALUES IN HETEROGENEOUS WEB TABLES

David W. Embley (BYU), Mukkai Krishanmoorthy (RPI), George Nagy (RPI), Sharad Seth (UNL)

Abstract

Printed and HTML web tables often contain redundant data values like totals and percentages. When appropriately labeled, these derived data values can be detected, verified, and optionally deleted either before or after importing these tables into a database or RDF triple store for data analysis. Aggregate analysis based on keyword search has been reported only on financial reports in a simple format. For more complex web tables, exploiting the table structure by segmenting and factoring headers increases the effectiveness of keyword search. Verification of potential computed values via SQL is demonstrated on 200 heterogeneous web tables from international sites that are algorithmically transformed into a canonical format and imported into a database.

1. Introduction

We demonstrate the transformation of human-readable tables into a format better suited for large-scale analysis with special attention to the role of redundant computed data values that help human readers but hamper automated analysis. Web tables constructed from databases or digitized from books, journals and newspapers, are the largest existing source of semi-structured data.[1] They are *structured* in the sense that data values appear in individual grid cells indexed in two layout dimensions by row and column headers that provide the semantic context of the data values. In contrast to relational tables that represent views of a database management system, human-readable web tables are only *semi-structured* because the distinctions between header paths, value cells, footnotes and footnote references, and other table components are not immediately accessible to computer algorithms. Furthermore the multi-dimensional category structure of tables (e.g. the three categories: Industry, Patent Type, and Total/Assignee in Fig. 1) is also algorithmically invisible even if obvious to human readers. Categories are conventionally represented as a multi-dimensional data cube.

Conventional table layout is well suited to looking up the value of items at the intersection of chosen row and column indices. Automated data analysis on an entire collection of tables, however, is more than lookup. It typically requires the application of arithmetic or logic operations (sum, percentage, maximum, equality) to the contents of cells selected via query criteria from the same or different tables. Further algorithmic operations or statistical evaluations may then be performed on the computed values. Data analysis on human-readable tables is severely limited by the amount of human effort required to organize the data for analysis.

We have developed effective methods for importing human-readable tables already available in some grid-friendly file format like HTML and CSV, into Data Base Management Systems (DBMS) and Resource Description Framework (RDF) triples. Some of our queries on one or several tables in Access, Protege and Virtuoso[2] returned anomalous results due to aggregates.

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[3]. 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 column sum in Fig. 1, should be clearly identified as derived values.

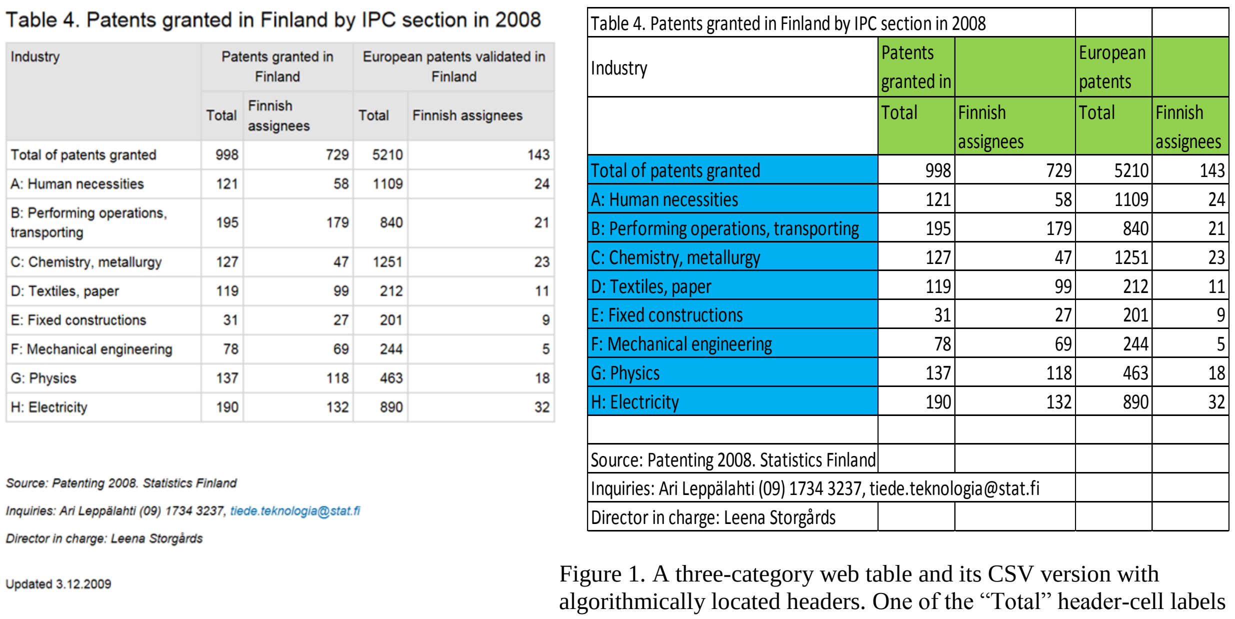


Table 4. Patents granted in Finland by IPC section in 2008

Industry	Patents granted in Finland		European patents validated in Finland	
	Total	Finnish assignees	Total	Finnish assignees
Total of patents granted	998	729	5210	143
A: Human necessities	121	58	1109	24
B: Performing operations, transporting	195	179	840	21
C: Chemistry, metallurgy	127	47	1251	23
D: Textiles, paper	119	99	212	11
E: Fixed constructions	31	27	201	9
F: Mechanical engineering	78	69	244	5
G: Physics	137	118	463	18
H: Electricity	190	132	890	32

Source: Patenting 2008. Statistics Finland

Inquiries: Ari Leppälahti (09) 1734 3237, tiede.teknologia@stat.fi

Director in charge: Leena Storgårds

Updated 3.12.2009

Table 4. Patents granted in Finland by IPC section in 2008				
Industry	Patents granted in		European patents	
	Total	Finnish assignees	Total	Finnish assignees
Total of patents granted	998	729	5210	143
A: Human necessities	121	58	1109	24
B: Performing operations, transporting	195	179	840	21
C: Chemistry, metallurgy	127	47	1251	23
D: Textiles, paper	119	99	212	11
E: Fixed constructions	31	27	201	9
F: Mechanical engineering	78	69	244	5
G: Physics	137	118	463	18
H: Electricity	190	132	890	32
Source: Patenting 2008. Statistics Finland				
Inquiries: Ari Leppälahti (09) 1734 3237, tiede.teknologia@stat.fi				
Director in charge: Leena Storgårds				

Figure 1. A three-category web table and its CSV version with algorithmically located headers. One of the "Total" header-cell labels corresponds to aggregate values and the other two are spurious.
http://tilastokeskus.fi/til/pat/2008/pat_2008_2009-12-03_tau_005_en.html

2. Table Structure Discovery

In preparation for relational queries on a table, including analysis of its computational properties, we take the following steps:

1 Segment the table to identify row and column headers, data values, and ancillary information like table title, footnotes, notes, and empty cells;

2. Extract the logical and possibly hierarchical category structure from the headers;

3. Preprocess numeric fields to facilitate subsequent queries.

4. Create canonical versions of the table that can be directly imported into a database or a triple store.

The success of this procedure is guaranteed for Well Formed Tables (Fig. 2) that account for most human-readable tables. Of our random sample of 200 tables from a larger collection from statistical government sites in 6 countries,

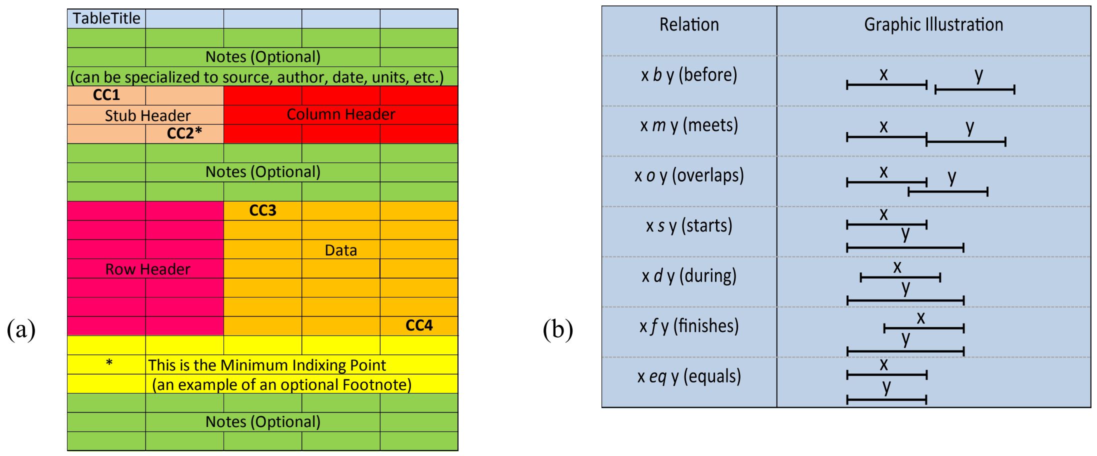


Figure. 2. (a) A WFT with a three-row Column Header, a two-column Row Header, and 7×3 data cells. Segmenting a WFT consists of locating the four *Critical Cells CC1, CC2, CC3* and *CC4* that demark the header and data regions. (b) Permissible horizontal and vertical spatial relations between every pair of regions are specified by Allen's Interval Algebra. For example, the vertical extent of the Stub Header must be EQUAL (eq) to the vertical extent of the Column Header, and horizontally the Stub Header must MEET (m) the Column Header. Additional constraints formalize the Header-Data indexing relationships and the possible locations of Footnotes, Footnote Markers, and Footnote References.

only two tables, with erroneous duplicate columns, failed. Most earlier table segmentation is based on machine learning methods (often conditional random fields or support vector machines), using syntactic, formatting and layout features[4,5,6]. Each row of the table is classified into classes such as table title, header, and data. Further analysis into relational tuples may exploit the distinction between alphanumeric row headers and numeric data values. The logical category structure is ignored. Although corporate teams have harvested and processed millions of tables, they too are limited to verifying their results on a small subset of tables. In contrast, our Minimum Indexing Point Search (MIPS) algorithm exploits the fundamental indexing property of table headers to partition the row, column, and stub header from the data cells.[7] The row and column headers that are found minimally sufficient to index the data cells are then factored to reveal their logical category structure.[8]

The table is segmented by locating the four Critical Cells shown in the WFT template. Atomic CSV cells are filled with the contents of the original merged cell that span multiple rows or columns. The Minimum Indexing Point Search (MIPS) algorithm searches for the MIP (CC2) from the top left corner. After finding a candidate, it backtracks from redundant (for indexing) rows or columns to compensate for the possible decrease in the height of the column header caused by additional columns in the row header. CC1 is found after eliminating unnecessary rows above the header by a backward sweep from the MIP. The remaining CCs are found by deleting sparse rows (usually Notes or Footnotes) from the data region below and to the right of the MIP. The CSV table derived automatically from the HTML table of Fig. 1 is segmented when the Critical Cells at the top-left and bottom-right corners of the row (blue) and column (green) header regions are found.

The next step, factoring each header, reveals the categories and their hierarchical structures. A factorization of a sum-of-products expression E is carried out under the following constraints[8].

1. Only the distributive law and the associative laws are used. The × operation has higher precedence than +. (The commutative law is disallowed, so that ordering is maintained both among header paths for + and within header paths for ×. To avoid changing the number and length of paths, the idempotency laws are also disallowed).
2. The factorization preserves the unique indexing property of E.

The factorization is complete in the sense that none of its terms can be factored further[8]. In the above example, the two rows of the column header constitute two categories because the Cartesian product (Patents granted in Finland + European patents validated in Finland) × (Total + Finnish assignees) of the two rows has 4 elements, which is exactly the length of the column header. The initial algebraic expression E for factoring the column header is obtained by tracing the header paths from the left to right:

Patents granted in Finland × Total + Patents granted in Finland × Finnish assignees + …

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

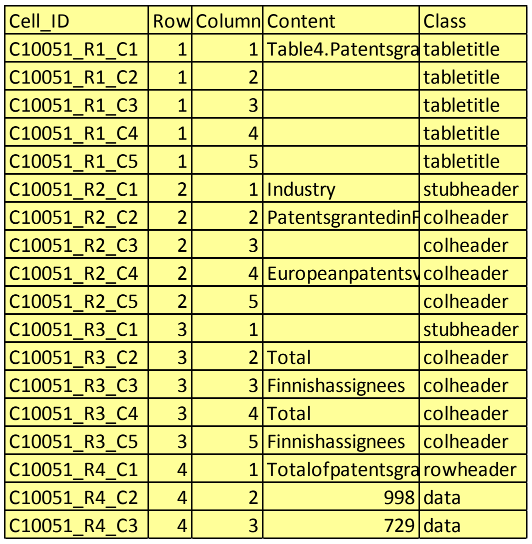
(Patents granted in Finland + European patents validated in Finland) × (Total + Finnish assignees)

Factoring out categories is necessary for queries on tables that share or complement subsets of a category (like Years) with similar data, or on tables that report different data for the same category. The results of the segmentation and factorization are compiled into a canonical CSV Classification Table and a canonical CSV Category Table (Fig. 3). They are both relational tables that can be directly imported and queried in a relational database like Access or MySQL or a triple store like Protege or Virtuoso[2].

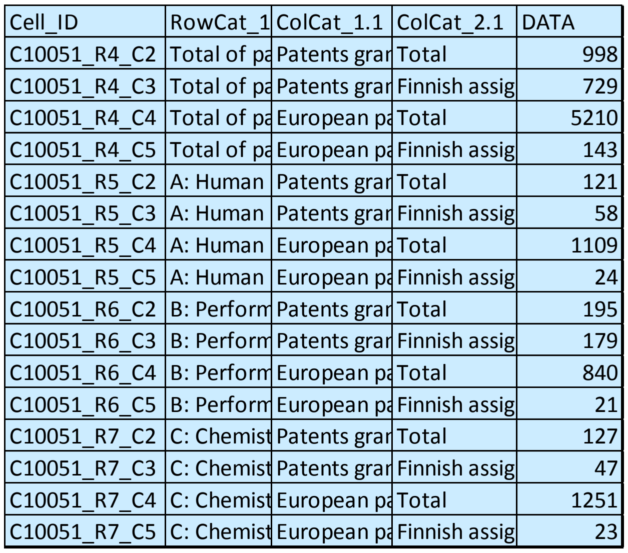
The Classification Table contains all the information in a table, including ancillary data like footnotes and footnote references. Here we use it for global queries like finding all the tables with a single occurrence of the keyword "Total" in any row or column, and for verifying whether the rows or columns above it sum to the reported value. It does not, however, show the category paths. We draw on the Category Table for category-by-category queries on a single table or on several tables with related categories. The header paths in the Category Table also suggest likely configurations for computed values, such as the first or last rows or columns of a category or subcategory. We can formalize the discovery of computations in tables by importing the table into a spreadsheet (e.g. Excel) and replacing computed data values by the formulas that compute them.

3. Discovering Redundant Data

The CSV table in Fig. 4, extracted from *Statistics Finland*, illustrates the complexity of automatically spotting redundant data values that are added to actual data in web tables for ease of human understanding.

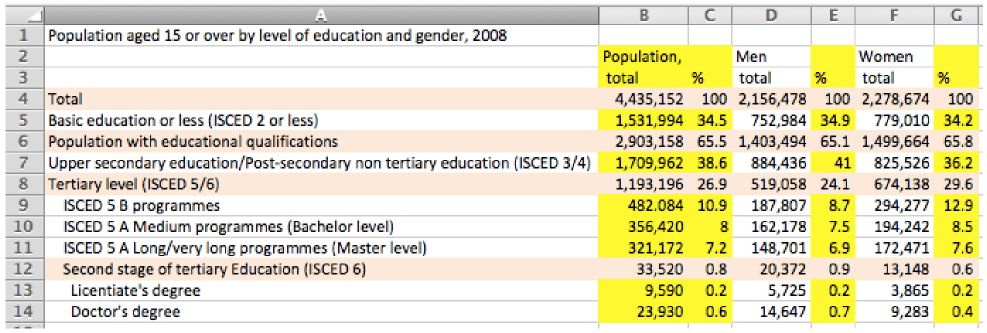


Cell_ID	Row	Column	Content	Class
C10051_R1_C1	1	1	Table4.Patentsgra	tabletitle
C10051_R1_C2	1	2		tabletitle
C10051_R1_C3	1	3		tabletitle
C10051_R1_C4	1	4		tabletitle
C10051_R1_C5	1	5		tabletitle
C10051_R2_C1	2	1	Industry	stubheader
C10051_R2_C2	2	2	PatentsgrantedinF	colheader
C10051_R2_C3	2	3		colheader
C10051_R2_C4	2	4	Europeanpatentsv	colheader
C10051_R2_C5	2	5		colheader
C10051_R3_C1	3	1		stubheader
C10051_R3_C2	3	2	Total	colheader
C10051_R3_C3	3	3	Finnishassignees	colheader
C10051_R3_C4	3	4	Total	colheader
C10051_R3_C5	3	5	Finnishassignees	colheader
C10051_R4_C1	4	1	Totalofpatentsgra	rowheader
C10051_R4_C2	4	2	998	data
C10051_R4_C3	4	3	729	data



Cell_ID	RowCat_1	ColCat_1.1	ColCat_2.1	DATA
C10051_R4_C2	Total of pa	Patents gran	Total	998
C10051_R4_C3	Total of pa	Patents gran	Finnish assig	729
C10051_R4_C4	Total of pa	European pa	Total	5210
C10051_R4_C5	Total of pa	European pa	Finnish assig	143
C10051_R5_C2	A: Human	Patents gran	Total	121
C10051_R5_C3	A: Human	Patents gran	Finnish assig	58
C10051_R5_C4	A: Human	European pa	Total	1109
C10051_R5_C5	A: Human	European pa	Finnish assig	24
C10051_R6_C2	B: Perform	Patents gran	Total	195
C10051_R6_C3	B: Perform	Patents gran	Finnish assig	179
C10051_R6_C4	B: Perform	European pa	Total	840
C10051_R6_C5	B: Perform	European pa	Finnish assig	21
C10051_R7_C2	C: Chemist	Patents gran	Total	127
C10051_R7_C3	C: Chemist	Patents gran	Finnish assig	47
C10051_R7_C4	C: Chemist	European pa	Total	1251
C10051_R7_C5	C: Chemist	European pa	Finnish assig	23

Figure 3: Parts of the two canonical tables for the table in Fig.1. Left: Classification Table with one row for every cell of the table. Right: Category Table with one row for every data cell with its indexing header paths.



	A	B	C	D	E	F	G
1	Population aged 15 or over by level of education and gender, 2008						
2		Population,		Men		Women	
3		total	%	total	%	total	%
4	Total	4,435,152	100	2,156,478	100	2,278,674	100
5	Basic education or less (ISCED 2 or less)	1,531,994	34.5	752,984	34.9	779,010	34.2
6	Population with educational qualifications	2,903,158	65.5	1,403,494	65.1	1,499,664	65.8
7	Upper secondary education/Post-secondary non tertiary education (ISCED 3/4)	1,709,962	38.6	884,436	41	825,526	36.2
8	Tertiary level (ISCED 5/6)	1,193,196	26.9	519,058	24.1	674,138	29.6
9	ISCED 5 B programmes	482.084	10.9	187,807	8.7	294,277	12.9
10	ISCED 5 A Medium programmes (Bachelor level)	356,420	8	162,178	7.5	194,242	8.5
11	ISCED 5 A Long/very long programmes (Master level)	321,172	7.2	148,701	6.9	172,471	7.6
12	Second stage of tertiary Education (ISCED 6)	33,520	0.8	20,372	0.9	13,148	0.6
13	Licentiate's degree	9,590	0.2	5,725	0.2	3,865	0.2
14	Doctor's degree	23,930	0.6	14,647	0.7	9,283	0.4

Figure 4. Aggregate rows and columns (colored) are commonly found in web tables. http://www.stat.fi/til/vkour/2008/vkour_2008_2009-12-04_tie_001_en.html

The derived data appears in both columns and rows. The values in column B (Population total) are the sum of the values in columns D and F. The percentage values in columns C, E, and G are derived from the columns to their left. Thus, the yellow columns contain data derivable from the data in columns D and F. The rows in the tables also exhibit complex dependencies:

$$R4 = R5+R6; \quad R6 = R7+R8; \quad R8 = R9+R10+R11+R12; \quad \text{and} \quad R12 = R13+R14$$

Hence the four pink rows are, also, redundant: 52 of the 66 data values in the table can be derived from the data shown in the 14 white cells.

The table in Fig. 4 also illustrates the challenges posed by human data-entry errors in automated detection of aggregate columns and rows. The error appears in the data in cell B9, where the value “482.084” should have been “482,084” (Table 1.). The error would break the following column dependences: B on D and F and C on B. It will also break the dependence of row 8 on the four rows below it.

Table 1. Some equivalent numeric formats of the data value of 57000000000000.

5.7 x 10^{13}	(US and China)	57,000,000,000,000.00	(commas and dot)
5,7 x 10^{13}	(France and Germany)	57 000 000 000 000.00	(spaces and dot)
57.000.000.000.000,00	(dots and comma)	57 trillion	(US and UK)
57 000 000 000 000,00	(spaces and comma)	57 billion	(Continental Europe)

Computational Keywords: The appearance of keywords, such as *total* and *%*, often suggest both the location and function of 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 2. 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%

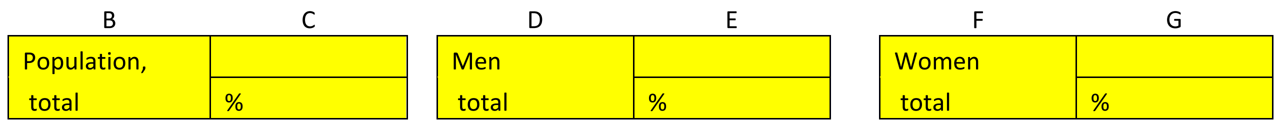
Table 3. Relative frequency of computed values by their function

Function	Associated keywords	Frequency (%)
Sum	*total, all*	72%
Average	*average*	2%
Percentage	*%, percent*	23%
Difference	*change, balance*	3%

Functional role of keywords: 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) and the rest are false positives (as in the column headers of Fig. 1). The functional role of the "true" keywords is shown in Table 3. Conversely, only 70% of the computed values occur with a keyword in the header. We did not, however, find any computed values that were not covered by the four basic functions associated with the keywords. This suggests that even if there is no keyword to guide the search, we need to check only a few of the unlimited number of possible computations.

Category-based Search: Although keywords reduce the search for the computational operations to a small set, for an aggregate function like *Sum* and *Average*, neither the number nor the location of its arguments can be deduced with ease. Generally, the arguments appear contiguously and adjacent to the location of the aggregate (this holds for all computations in Fig. 4, except for the computation in column B). When this condition holds, the search for the arguments of the aggregate function can proceed to an increasing number of contiguous arguments. The search halts when either the aggregate function is verified or the end of the table is reached. The exhaustive nature of this search may be improved by branch-and-bound techniques, if the data values are all of the same sign (as above).

The exceptions to the contiguity and proximity rule substantially raise the algorithmic complexity of finding the arguments[3].**Error! Bookmark not defined.** We have found, however, that the category structure of the header can guide the search for likely candidates. Consider the aggregate in column B in Fig. 4, identified with the *Sum* function by the keyword *total* in its header. The search for contiguous arguments to its right with one through five columns will fail because of the intruding % columns. Instead of then switching to an exhaustive (exponential) search of all subsets of the five columns as arguments, the algorithm checks the category structure of the column header (shown in Fig. 5) and determines that within the three siblings, columns D and F occupy the same position as the computed location B. Therefore, D and F will be tried first as arguments, which will trigger successful termination of the search. As this category-structure based analysis does not rely on keywords or other linguistic clues, it can be adopted in arguments' search even in the absence of the linguistic clues.



B	C	D	E	F	G
Population,		Men		Women	
total	%	total	%	total	%

Figure 5.The category structure of the column header in the table of Fig. 4.

Semantic Analysis: Keyword search obviously fails in keyword-less tables with aggregates or in tables with keywords where an aggregate is not identifiable by a keyword. In one of our tables, a total is labeled Top 10 ports.

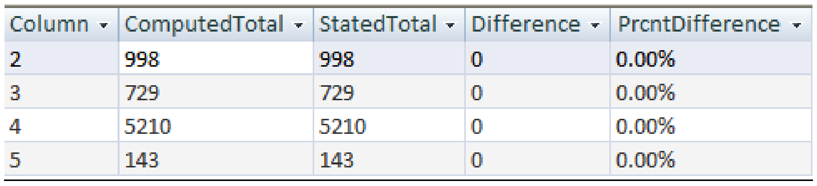
In another, the keyword *total* occurs in 14 of the 15 rows and, as such, is not an effective filter. In that table, *total* refers to external arguments (i.e., each data value is the sum of values that do not appear in the table). In all of these cases, semantic analysis of the header, possibly making use of rapidly growing knowledge bases, such as YAGO2[9], can help narrow the choices for computed data values.

4. Computed Value Verification

Fig. 1 shows a table with computed sums for each data column. Tables with column sums in a row whose row header includes "total" or "Total" are common. As an example of how to find and verify computed values in tables, we wrote a query that checks for the computational pattern in Fig. 1 and also another query that checks the complementary pattern that has a single total column header and data rows whose numeric values sum to the stated total.

As mentioned, the MIPS algorithm produces a relational-database table that classifies each cell. Fig. 3 shows the classification table for the Table in Fig. 1. Every classification table has the same header fields: *Cell_ID*, a unique identifier for a table's cell across all tables; *Row* and *Column* numbers as integers for ease of manipulation; *Content*, which gives a cell's value text; and *Class*, which gives the cell's classification, one of *tabletitle*, *rowheader*, *colheader*, *data*, *footnote*, *note*, and *EMPTY*. All the classification tables were imported into MySQL.

Since all classification tables have the same schema and give all the information contained in a table in a uniform way, we can write a single query to apply to all tables to check for computational patterns and verify their results. To check for the column summation pattern in Fig. 1, an SQL query selects the *data* rows whose row header does *not* include "total" or "Total" in its *rowheader*, groups them by *Column* and sums them to produce a *ComputedTotal* for each column. The query then selects the *StatedTotal*, matches the *ComputedTotal* and *StatedTotal* by column, and computes the *Difference* (if any) and the *PrcntDifference*. Fig. 6 shows the results for the table in Fig. 1. Checking the percent differences, which are all 0.00%, provides confidence that the aggregate sum is intended and indeed that the expected pattern holds.



Column	ComputedTotal	StatedTotal	Difference	PrcntDifference
2	998	998	0	0.00%
3	729	729	0	0.00%
4	5210	5210	0	0.00%
5	143	143	0	0.00%

Figure 6: Total query result for the table of Fig. 1.

We applied this totals check query and its complement query to all 198 tables for which classification tables were generated. The results appear in Table 4. Several web tables did not satisfy the assumption of having all data columns consisting of only recognized numeric values. Some, for example, included string values and some marked empty cells or value-hidden cells with a non-numeric mark such as "x" or "-". In others, tables included sums for a subgroup of columns, or for some but not all columns, or were not intended to be sums but rather something else such as "Total" factor productivity". A surprising number of tables included one or more incorrect stated totals.

Table 4. Results of "Total" query over all 200 tables

Our sample of web tables	200
Well-Formed Tables	198
Tables with a single "Total" or "total" in a row header	72
Specified pattern of aggregates satisfied	23
Exact match of sum computed by SQL query	7
Tables with a single "Total" or "total" in a column header	47
Specified pattern of aggregates satisfied	21
Exact match of sum computed by SQL query	13

5. Conclusion

Ever since the first Roman life expectancy tables, scientists have tabulated whatever facts and observations can be tabulated. It is barely an exaggeration to claim that all the facts known to mankind could be found in tables.

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 keeping independent data values separate 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. Combined with orthogonal approaches based on machine learning and natural language processing, syntactic table analysis can add enormous value to petabytes of machine readable, but not yet machine understandable, information.

The effectiveness of information technology systems is increasingly limited more by the amount and quality of their input data than by the computational resources available to them. The semantic web promises pooling the worldwide treasure of facts, connections between facts, and interconnections between connections. Tables must be part of the treasure hunt.

REFERENCES

[1] A. Halevy, P. Norvig, and F. Pereira, "The Unreasonable Effectiveness of Data," IEEE INTELLIGENT SYSTEMS. March/April 2009.

[2] D.W. Embley, S. Seth, G. Nagy, "Transforming Web tables to a relational database," Proceedings ICPR 2014, Stockholm, Sweden, 2014.

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

[4] W.J. Cafarella,A. Halevy, D.Z. Wang, E. Wu, Y. Zhang, WebTables: "Exploring the Power of Tables on the Web," Proceedings VLDB '08 Auckland, New Zealand, 2008.

[5] P. Venetis, A. Halevy. J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, C. Wu, "Recovering Semantics of Tables on the Web," Proceedings of the LDB Endowment, Vol. 4, No. 9, 2011.

[6] M.D. Adelfio and H. Samet, "Schema Extraction for Tabular Data on the Web," Proceedings of The 39th International Conference on Very Large Data Bases, (Proceedings of the VLDB Endowment, Volume 6, Number 6), Riva del Garda, Trento, Italy 26–30 August, 2013.

[7] S. Seth, and G. Nagy, "Segmenting Tables via Indexing of Value Cells by Table Headers," Proceedings ICDAR 2013, Washington, D.C., August 2013.

[8] 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.

[9] J. Hoffart, F. Suchanek, K. Berberich, G. Weikum, "YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia" ARTIFICIAL INTELLIGENCE JOURNAL, Vol. 194, Jan. 2013, pp. 28-61.