Ten Years of WebTables

Michael Cafarella
University of Michigan
michjc@umich.edu

Alon Halevy
Megagon Labs
alon@megagon.ai

Hongrae Lee
Jayant Madhavan
Cong Yu
Google, Inc.
{hrlee, jayant, congyu}@google.com

Daisy Zhe Wang
University of Florida
daisyw@cise.ufl.edu

Eugene Wu
Columbia University
ew2493@columbia.edu

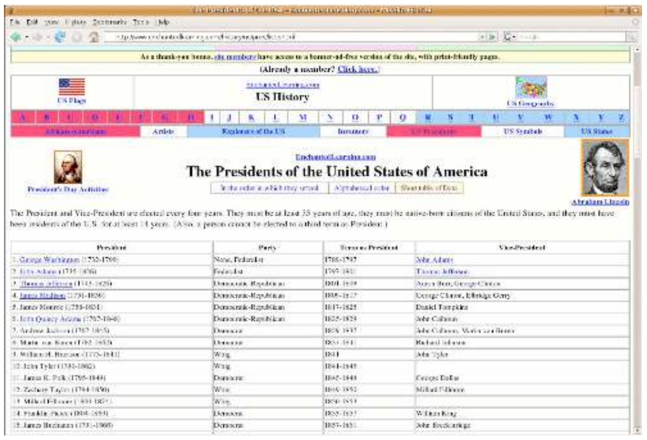
ABSTRACT

In 2008, we wrote about WebTables, an effort to exploit the large and diverse set of structured databases casually published online in the form of HTML tables. The past decade has seen a flurry of research and commercial activities around the WebTables project itself, as well as the broad topic of informal online structured data. In this paper, we[1] will review the WebTables project, and try to place it in the broader context of the decade of work that followed. We will also show how the progress over the past ten years sets up an exciting agenda for the future, and will draw upon many corners of the data management community.

PVLDB Reference Format:
Michael Cafarella, Alon Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. Ten Years of WebTables. *PVLDB*, 11 (12): 2140-2149, 2018.
DOI: https://doi.org/10.14778/3229863.3240492

1. INTRODUCTION

In 2008, the Web had been traditionally modelled as a corpus of unstructured documents. Some structure was imposed by hierarchical URL names and the hyperlink graph, but the basic unit for reading or processing was the unstructured document itself. That is mostly still true in 2018. However, Web documents often contain large amounts of relational data. For example, the Web page shown in Figure 1 (from the original paper [7]) contains a table that lists American presidents. The table has four columns, each with a domain-specific label and type (e.g., **President** is a person name, **Term as President** is a date range, etc) and there is a tuple of data for each row. This Web page essentially contains a small relational database, even if it lacks the explicit metadata traditionally associated with a database.



EnchantedLearning.com
The Presidents of the United States of America

The President and Vice-President are elected every four years. They must be at least 35 years of age, they must be native-born citizens of the United States, and they must have been residents of the U.S. for at least 14 years. (Also, a person cannot be elected to a third term as President.)

President	Party	Term as President	Vice-President
1. George Washington (1732-1799)	None, Federalist	1789-1797	John Adams
2. John Adams (1735-1826)	Federalist	1797-1801	Thomas Jefferson
3. Thomas Jefferson (1743-1826)	Democratic-Republican	1801-1809	Aaron Burr, George Clinton
4. James Madison (1751-1836)	Democratic-Republican	1809-1817	George Clinton, Elbridge Gerry
5. James Monroe (1758-1831)	Democratic-Republican	1817-1825	Daniel Tompkins
6. John Quincy Adams (1767-1848)	Democratic-Republican	1825-1829	John Calhoun
7. Andrew Jackson (1767-1845)	Democrat	1829-1837	John Calhoun, Martin van Buren
8. Martin van Buren (1782-1862)	Democrat	1837-1841	Richard Johnson
9. William H. Harrison (1773-1841)	Whig	1841	John Tyler
10. John Tyler (1790-1862)	Whig	1841-1845	
11. James K. Polk (1795-1849)	Democrat	1845-1849	George Dallas
12. Zachary Taylor (1784-1850)	Whig	1849-1850	Millard Fillmore
13. Millard Fillmore (1800-1874)	Whig	1850-1853	
14. Franklin Pierce (1804-1869)	Democrat	1853-1857	William King
15. James Buchanan (1791-1868)	Democrat	1857-1861	John Breckinridge

Figure 1: A typical use of the `table` tag to describe relational data. From the original WebTables paper.

The goal of the original WebTables paper [7] was to automatically detect these "database-like" HTML tables, use them to construct the largest corpus of databases to date, and then build novel applications out of the resulting corpus. We extracted 14.1B HTML tables from Google's general-purpose web crawl, and then used a trained classifier to identify the estimated 154M tables that were database-like. The percentage of raw tables that described databases was small — about 1.1% — but the number of raw tables was so large that the resulting corpus of databases was still larger than any previously-known collection, by at least five orders of magnitude. The paper described how we constructed the corpus, and described a number of novel applications that the corpus enabled.

The WebTables work at Google was very exciting because it combined the Web's enormous scale and topical reach with questions that have been traditionally associated with relational databases, such as how to understand and

[1] One of the things that has made the WebTables project exciting is its long lifespan: the project began in 2007, and is still ongoing. The authors of this paper took an especially large role in WebTables, but not everyone who worked on the project is an author of this paper or even necessarily known to us. We are very grateful to everyone who has contributed over the years.

Proceedings of the VLDB Endowment, Vol. 11, No. 12
Copyright 2018 VLDB Endowment 2150-8097/18/8.
DOI: https://doi.org/10.14778/3229863.3240492

recover schema information. However, we did not imagine the many positive events that would follow. Many subsequent research papers, from both academia and industry, improved the original extraction methods, then applied the resulting datasets to novel problems such as attribute discovery and entity extension. There was also industrial software engineering effort – most notably at Google and Microsoft – that built real products similar to the use cases described in the original WebTables paper, as well as entirely novel ones.

However, we also believe there are still tremendous opportunities around extracting and manipulating structured data on the Web. Indeed, we think the next decade holds even more promise for WebTables-style work than the last.

In this paper, we will offer a brief summary of the original WebTables work, and attempt to describe and organize much of the intellectual work that followed. We will also describe the practical engineering efforts that were necessary to turn the WebTables vision into real products. Finally, we will sketch a vision of what we believe are open opportunities around WebTables-like work, both in the intellectual sphere and the practical engineering one.

1.1 WebTables: A Brief Recap

In this section we provide a short overview of the original WebTables paper [7] and its core contributions.

1.1.1 Extraction and Data Model

We collected roughly 14.1 billion HTML tables from the Google search web crawl, and applied a trained *is-relational* classifier to identify the tables as *relational* or *non-relational.* The *relational* label is somewhat informal, defined driven by human judgmements: the human judges wanted tables where rows clearly represent separate tuple-like objects, and columns represent different dimensions of each tuple. Extraction details were described in Cafarella, *et al.* [8].

A "header row" of attribute labels at the top of the table is optional, but if recovered offers a small amount of schema-like information about the extracted data table. We trained a second classifier to detect this header row. Of course the schema information is extremely informal, and even if recovered is not very expressive. For example, even attribute typing information is not explicit, and many traditional relational schema elements such as key constraints are missing.

We applied the *is-relational* classifier to the collection of raw HTML tables and obtained a corpus $\mathcal{R}$ of databases (where the classifier returned a verdict of *relational*). Each database consists of a single relation. For each relation $R \in \mathcal{R}$, we have:

- The url and page offset where R was recovered; these uniquely identify R.
- The header row, or "schema" $R_{\mathcal{S}}$, which is an ordered list of attribute labels. For example, the table in Figure 1 has the attributes $R_{\mathcal{S}}$ = [**President, Party**, ...]. One or more elements of $R_{\mathcal{S}}$ may be empty strings (*e.g.*, if the table's schema cannot be recovered).
- A list of tuples, $R_{\mathcal{T}}$. A tuple t is a list of data strings. The size of a tuple t is always $|R_{\mathcal{S}}|$, though one or more elements of t may be empty strings.

1.1.2 Schema Statistics

Extracting a large collection of database-like tables and their relational attribute labels allowed us to build the *attribute correlation statistics database*, or **ACSDb**. It contains statistics about general WebTables schema use.

The **ACSDb** listed each unique schema $\mathcal{S}$ found in the set of all $R_{\mathcal{S}}$, along with a count that indicates how many relations contain the given $\mathcal{S}$. We assume two schemas are identical if they have the same set of attributes (regardless of order). The **ACSDb** $\mathcal{A}$ is a set of pairs of the form $(\mathcal{S}, c)$, where $\mathcal{S}$ is a schema of a relation in $\mathcal{R}$, and c is the number of relations in $\mathcal{R}$ that have the schema $\mathcal{S}$. We only count one schema per internet domain name, to prevent a single site with many similar pages from swamping the counts.

The resulting **ACSDb** contained 5.4M unique attribute labels, and 2.6M unique schemas. Unsuprisingly, a relatively small number of schemas appear very frequently, while most schemas are rare.

The **ACSDb** was simple, but critically allowed us to compute the probability of seeing various attributes in a schema. For example, $p(address)$ is simply the sum of all counts c for pairs whose schema contains *address*, divided by the total sum of all counts. We can also detect relationships between attribute names by conditioning an attribute's probability on the presence of a second attribute. For example, we could compute $p(address|name)$ by counting all the schemas in which "address" appears along with "name", and normalizing by the counts for seeing "name" alone.

1.1.3 Applications

We used this corpus of databases to build several applications. The first was a simple keyword search tool: it would take a user's search query as input, and return a ranked list of WebTables-derived databases to answer the query. Our initial paper used "city population" as an example query. Subsequent work at Google deployed this idea in the core search engine; Figure 2 shows that query from the original paper's screenshot, and that same query on the Google search engine of today.

We also built several applications on top of the **ACSDb**. Two of the most interesting are *schema autocomplete* and *attribute synonym finding.*

The *schema autocomplete tool* helped database designers build a schema, by suggesting the most-likely next attributes to add to a schema. For example, if the designer inputs the attribute **stock-symbol**, the **ACSDb**-powered schema auto-complete tool will suggest **company**, **rank**, and **sales** as additional attributes. This tool worked by using **ACSDb** statistics to find the most probable attributes, conditioned on seeing attributes the designer has already entered.

The *attribute synonym finding tool* automatically computed pairs of schema attributes that seem to be used synonymously in the WebTables corpus. For example, the tool could find that **hr** and **home run** are used synonymously when representing baseball data, even though this would be an extremely unlikely pair of words to find in a traditional thesaurus or other linguistic resource. This tool worked by identifying pairs of attribute labels where (1) the labels never appeared together in the same WebTables schema, and (2) the labels frequently share similar co-attributes, among other sources of evidence.

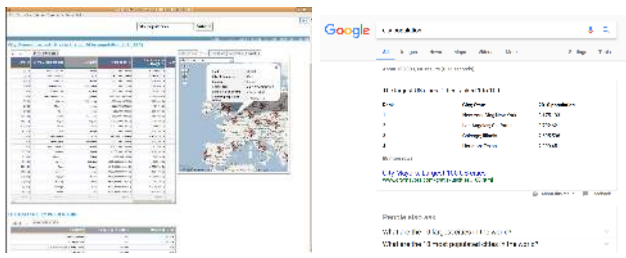


Figure 2: A query for "city population", from the original WebTables paper's demonstration table search system (left), and Google search results in 2018 (right).

1.2 Previous Work

In 2008, WebTables was not the first effort to identify databases from HTML tables. Other systems [10, 37, 41], especially that of Gatterbauer, *et al.* [18], had tried a number of interesting approaches for extracting databases from HTML pages. WebTables was the first system we know of to obtain a database corpus at its scale.

Keyword search over structured data was a known problem in the database literature, but tended to focus on returning tuples from a single large database, as with the DBXplorer [2] and DISCOVER [22] projects. Keyword-driven web search engines sometimes returned structured results, but were usually limited to a tiny number of domains, such as weather. In 2008, most other access paths for popular consumption of structured data, such as mobile device app stores and voice assistants, did not yet exist.

2. A DECADE OF RESEARCH

The WebTables project was one example of research into online structured data – a long-lasting line of intellectual and engineering work that includes Semantic Web [4], Web of Linked Data [5], and many other projects. However, dedicated research into Web-embedded data tables *per se* has became quite popular in the last decade.

2.1 Extraction

One early direction was to extend table extraction beyond HTML tags. The original paper focused on identifying very conventional tables of data, with attribute-oriented columns and tuple-oriented rows. However, relational data are also encoded as attribute-value pairs in the form of vertical tables (e.g., Wikipedia infoboxes), or as entries in lists [16, 13]. For example, a list of cartoons may have entries such as "Duck Amuck (Warner Bros./1953)". This string encodes a structured record containing the title, the producer, and the release year. Elmeleegy *et al.* [16] introduced a pipeline that splits individual list entries into candidate rows, performs attribute column alignment across rows, and finally refines the table to address inconsistencies.

Chu *et al.* [13] further introduced syntactic and semantic coherence measures for list extraction, and learned semantic coherence measures by leveraging column co-occurence statistics from existing web tables. Several researchers produced web tables from the public Common Crawl [1, 24, 15], thereby making them available to a broad audience outside the large Web companies. Wang, *et al.* [36] improved extraction quality by leveraging curated knowledge bases.

2.2 Table Search

Table Search was a core and exciting potential application in the WebTables paper, and it is not surprising that it attracted substantial research attention.

In general, keyword-based table search ranks extracted tables based on a combination of cell contents, attribute names, and surrounding text (*e.g.*, page title). However, this approach may not be sufficient for several important classes of search queries, encountered by Chakrabarti, *et al.* when deploying a web table service on the Bing search engine [9]. One is *row-subset queries*, which only request a subset of rows in a larger table. For example, the query "largest software companies in USA" may only match a table of large software companies, and must be filtered to the subset of USA rows. Another is *entity-attribute queries*, such as "aberdeen population", which match some keywords to entities ("aberdeen") and other keywords to attribute names ("population").

Another way to navigate a set of tables is via *column search*. This method does not use keywords, except perhaps in the initial user interaction. Instead, users begin with an "initial table," then navigate through a "concept space" of tables that would be good join candidates for the original. Different methods built this space using knowledge bases [17], isA databases [32], or crowd sourcing [17].

2.3 Table Enhancement

In several "table enhancement" projects, the user submits a *query table* to the system, which then attempts to "complete" it by filling in attribute values, recommending novel attributes, or adding more entities of the same type.

Several operators have been proposed for specific types of table enhancement tasks [6, 39, 43]. In *Augmentation by attribute name (ABA)* (also called `EXTEND()` in [6]), the system fills in attribute values given an explicit user-provided attribute. In *augmentation by example (ABE)*, the system fills in attribute values given other examples of desired attribute values (but no attribute name). The *Attribute Discovery (AD)* operator recommends important attributes given a list of entity names.

Cafarella, *et al.* [6] initially implemented `EXTEND()` with a combination of schema matching and search engine results to find candidate completion values from extracted data tables. InfoGather [39] observed that precision and coverage of the filled-in values can be improved by exploiting *indirect match* tables that are a "hop away" from the original query table. This method was used to implement *ABA*, *ABE*, and *AD*. The follow-on InfoGather+ [43] system ensured that the web tables used for `EXTEND()` contained attributes were semantically consistent. For instance, extending a table of companies with "revenue" should not draw from columns with names such as "2010 revenue", "2011 revenue", and "revenue (euros)" even though they are textually quite similar to the original attribute name.

Another example of table enhancement is *fact lookup*, in which the query table contains a single entity and single attribute, and the goal is to retrieve the attribute value (the fact). To answer such queries, FACTO [40] identified web pages about individual entities (*e.g.*, a wikipedia page about Barack Obama), then found attribute-value tables from the relevant pages, and finally retrieved answers from them.

Concept expansion takes as input a high level concept (*e.g.*, **rock stars**) and an example list of entities within the concept (*e.g.*, **freddy mercury**, **yoshiki**, and **prince**) and expands the set of entities within the concept. Wang, *et al.* [34] identified possible source-data web tables for this task by examining the surrounding text (e.g., table captions), and employing only *exclusive tables* to avoid semantic drift. The user-provided examples serve to seed this iterative process. Chen, *et al.* attempted to solve a similar task, with a system tailored for long-tail infrequently-observed items [11].

One important aspect of table improvement is to improve the quality of table-contained data. Along this line, Wang et. al. [35] proposed a framework based on functional dependencies (FDs). Unlike in traditional database design, where FDs are specified as statements of truth about all possible instances of the database; in web environment, FDs are not specified over the data tables. Instead, FDs are extended with probabilities to capture their inherent uncertainty, and are generated using counting-based algorithms over many data sources. These probabilistic FDs can improve data and schema quality by (1) pinpointing dirty data sources and (2) normalizing large mediated schemas.

Many of the above techniques fundamentally leverage relationships among columns, rows, and tables, often with a machine learning component. Limaye, *et al.* [25] studied how to annotate web tables with the presence of *entities* in cells, attribute types and concepts; and the presence of *relationships* between attributes. They used a graphical model that jointly learns these annotations within a single model, and leverages the YAGO [31] knowledge base as a source of attribute concepts and entities. Pimplikar, *et al.* [27] employed a similar joint-labeling technique by modeling table enhancement as a graphical model.

The relationship between tables can be further leveraged to synthesize tables that are the results of *combining* multiple related tables and thereby generating data tables that are not present anywhere on the Web. Ling, *et al.* [26] explored how tables with the same semantics can be discovered and vertically combined into a table with a more complete set of rows. Das Sarma, *et al.* [30] looked at tables that share core identifying attributes and yet complement each other on other attributes; these therefore can be horizontally combined to provide a table with a more complete set of columns.

3. MAKING SOME REAL IMPACT

The publication of the initial WebTables paper inspired a long line of product work at both Google and Microsoft. At Google, as described by Balakrishnan, et al. [3], this included efforts to bring the core WebTables machinery up to production quality, integrating pieces of the original WebTables application vision into existing products, and launching novel WebTables-driven applications, such as Tables in Featured Snippets and Structured Snippets. At Microsoft, the team embarked on its own web tables effort, which became part of several interesting products. This section will mostly focus on efforts at Google with some overview of efforts at Microsoft in Section 3.6.

3.1 Production Corpus Construction

Extracting High Quality Tables: Subsequent engineering work used the same basic scan-and-classify architecture as the original WebTables paper, with a few important additions:

- The extractor used **simple filtering rules** that are written by hand to identify a huge number of non-relational HTML tables, such as degeneratively small ones, calendars, and tables-of-contents. These simple rules are effective at dramatically shifting the class imbalance of the *is-relational* task. Instead of 99 non-relational tables for each relational one, the ratio is more like 9 to 1, making a trained classifier much easier to build.

- The system aimed to additionally obtain **vertical tables** — that is, lists of attribute/value pairs, often seen in Wikipedia infoboxes. This entire class of tables were not considered as useful as horizontal tables originally, but their inclusion proved to be crucial for some applications.

- The team designed and implemented **machine learning classifiers** to make predictions on *is-relational-vertical* and *is-relational-horizontal*.

Recovering Table Semantics: A substantial amount of the initial WebTables paper focused on understanding the semantics of the table corpus as a whole, in the form of the **ACSDb**. However, a surprisingly small amount of the initial system described by that paper focused on understanding the semantics of a particular table; it attempted to identify whether the first row contained a schema, and used some crude ranking features for table search, but not much else. However, table-level semantic understanding is key for many applications. Some examples of recovering table-level semantics include:

- **Subject-column discovery:** Researchers observed that over 75% of the tables in the extracted WebTables corpus contained a subject-like column that describes the main entities of the table, while other columns describe properties of these entities. For example, the table in Figure 3 describes literacy rates in UN-listed countries, and contains a **subject column** of **Country**. The team developed a high-quality classifier to identify subject columns. Interestingly, unlike the primary keys in relational databases, the subject column in a Web table may not be a key of the table, and may contain duplicate values.

- **Column class-label annotation:** Many WebTables' columns contain values drawn from a relatively small number of classes, such as countries. Others are drawn from a small number of "algorithmic" classes, such as phone numbers. These class labels are quite different from the table's schema labels; the former are drawn from a fixed set, while the latter can be anything the data authors wishes to write. It can be useful for downstream applications if the tables' columns are annotated with their relevant class labels. Note that a column can belong to multiple classes.

  Balakrishnan, *et al.* [3] used the Google Knowledge Graph to first match cell values to KG entities (such as the KG entity that represents **Barack Obama**). They then recovered KG entity classes (such as the class that represents **Presidents** or the class that represents **People**) for each cell. Finally, they aggregated the recovered KG classes to provide a label for the overall table column. This column-level verdict can be further leveraged to clean up ambiguous cell-level references (*e.g.*, it may not initially be clear whether the string "Obama" refers to the entity **Barack Obama** or **Michele Obama**, but a column-level class-label of **President** resolves the issue). Additional work [14] made concept detection more scalable and accurate.

  Interestingly, being able to identify a class label for the subject column proved to be a very effective signal of overall table quality.

- **Binary relationships between columns**: The relationships between the *subject column* (say, **President**) and other columns of the table (say, **Vice-President** or the more obscure **Chief-of-Staff**), are often described using complicated text in the source page. These descriptions are often bundled along with general text in the page, and it can be hard to disentangle the column-specific description.

  Researchers [19] developed a dictionary of pairwise attributes found in search queries and web text (*e.g.*, **Countries** and **GDP** or **Coffee Production**). They used that dictionary to identify the parts of the surrounding page that likely refer to a specific column, and annotate each column with the resulting text.

3.2 Google Tables

The keyword-driven table search application was one that we found exciting from the very start of the WebTables project. The prototype from the initial paper became available to users as Google Tables[2] (now in maintenance mode), an experimental public site for finding tables on the Web. While it was not the killer application that we had hoped, it demonstrated the value of WebTables and paved the way for the follow-up applications at Google.

Google Tables was built on Google's internal scalable search infrastructure. It integrated table-specific ranking signals, including (1) some traditional page ranking information, such as link analysis; (2) detailed information about which portion of each extracted table received the keyword match (such as whether on a cell value, or the table's caption text); and (3) signals derived from the enhanced semantic information. This latter information also enabled semantics-powered query expansion.

Google Docs Integration: Google Tables was used for finding useful tables, but also as the first step in finding data import targets for other Google applications. Google Docs' Explore panel — originally called the Google Research Tool — provides users a convenient way to search for information while they are working on their documents. Users can insert citations, links, and images into the document directly from the tool. Searching for tabular data turned out to be a natural new feature.

From November 2013 to September 2016, table search functionality was integrated into various parts of Google Docs. Compared to users of the Google Tables product, who intentionally came to the table search engine to find relevant data, Google Docs users are typically less familiar with tabular data and may not want to scroll over many results pages to find the data they want. Thus, we are more selective in the search result and apply a higher bar for search quality. We considered only a subset of our table corpus, which is roughly 10% of the table search corpus. We selected those higher-quality tables by applying a few simple rules, such as requiring much higher quality scores, or requiring the presence of both header rows and subject columns.

3.3 Fusion Tables Integration

Google Fusion Tables [20] was introduced in June 2009 as a lightweight data management tool for the lay person. Users could upload structured data from spreadsheets or CSV files and see the data easily on maps and other visualizations. In a sense, Fusion Tables complemented WebTables – instead of creating an HTML table on a Web page, users can directly create the data in Fusion Tables, and we would not have to extract it from a Web page. Fusion Tables was used in many applications, the most notable ones were in journalism and in disaster response where professionals needed quick ways of making data visualizations available to many people.

Fusion Tables was later integrated into table search. Users could point to an HTML table on a Web page and have it directly imported into Fusion Tables, thereby making it easier to use the HTML table for visualizations or to be joined with other tables.

3.4 Tables in Featured Snippets

The team eventually found two important applications for WebTables in Google's core search application—Tables in Features Snippets described in this subsection and Structured Snippets in the next subsection. While Google Tables and Fusion Tables demonstrated the potential of WebTables, users of those products were a rather focused group. In contrast, these next applications brought WebTables to billions of users.

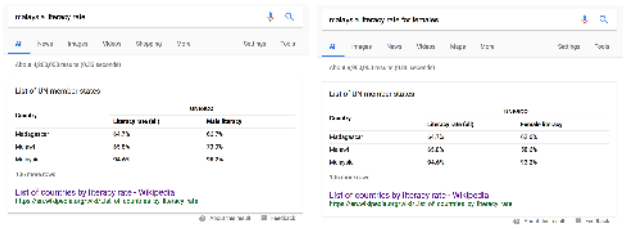
Featured Snippets[3] is a special block at the top of the search results page that includes a summary of the answer, extracted from a webpage, and is one of most prominent features in Google's search results. The team noticed user questions could often be best answered by tabular form data, and integrated WebTables into the Search results, launching *Tables in Features Snippets* in May 2014. The right side of Figure 2 shows an example.

The queries that are most amenable to Tables in Features Snippets (e.g., "literacy rate of Malaysia" in Figure 3) typically belong to the longer-tail content that would not be suitable for curation and storage in the Google Knowledge Graph. We return to this point in Section 4.

The integration with Google search raised multiple challenges beyond dedicated Google Tables. First, we needed to identify "fact-seeking" queries where the user could conceivably benefit from a table of data. Second, when a query is fact-seeking, we need to identify the pages that contain relevant tables; when there are multiple such tables, we had to choose which one to highlight in the result page. Finally, we needed to design a UI that is most effective in the very limited space. A challenging task in presenting table search results lies in generating a helpful result summary. Engineers used the keyword hits on the table to automatically choose projections and selections to generate a query-specific snippet for each table result. Figure 3 shows two similar search

[2]http://research.google.com/tables

[3]https://www.blog.google/products/search/reintroduction-googles-featured-snippets/



malaysia literacy rate

List of UN member states

	UNESCO	
Country	Literacy rate (all)	Male literacy
Madagascar	[illegible]	[illegible]
Malawi	[illegible]	[illegible]
Malaysia	94.6%	[illegible]

List of countries by literacy rate - Wikipedia

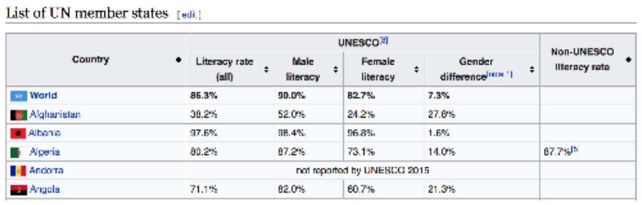
malaysia literacy rate for females

List of UN member states

	UNESCO	
Country	Literacy rate (all)	Female literacy
Madagascar	[illegible]	[illegible]
Malawi	[illegible]	[illegible]
Malaysia	94.6%	[illegible]

List of countries by literacy rate - Wikipedia

Figure 3: Tables in Featured Snippets will modify the projected columns in response to different user search terms.



List of UN member states [edit]

Country	UNESCO[3]				Non-UNESCO literacy rate
	Literacy rate (all)	Male literacy	Female literacy	Gender difference[note 1]	
World	86.3%	90.0%	82.7%	7.3%	
Afghanistan	38.2%	52.0%	24.2%	27.8%	
Albania	97.6%	98.4%	96.8%	1.6%	
Algeria	80.2%	87.2%	73.1%	14.0%	87.7%[5]
Andorra	not reported by UNESCO 2015				
Angola	71.1%	82.0%	60.7%	21.3%	

Figure 4: The original Wikipedia table that provides tabular data for the Malaysia queries.

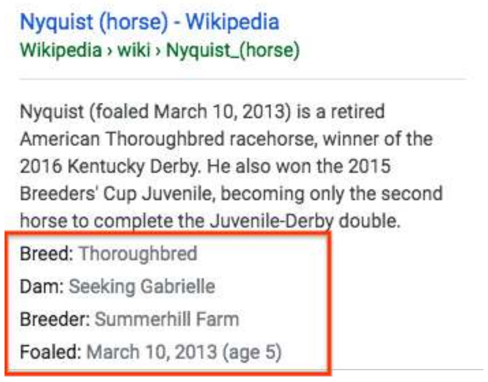
queries that retrieve the same table, but generate different query-sensitive table snippets.

3.5 Structured Snippets

While Tables in Features Snippets produced a subset of a single table and were displayed at the top of the search result page, we discovered another opportunity for using tables *within* the snippets[4] of the regular search results. Launched in August 2014, Structured Snippets[5] has become one of the main search result snippet features. This was a pleasant surprise in two main aspects. First, we had not envisioned that table data could be used to impact search result snippets, a fundamental feature for search. Second, the feature was powered by vertical tables, a class of tables that were originally considered much less interesting and rich than the horizontal ones. This new application posed new challenges, the primary one being fact quality, i.e., to identify which facts from the vertical table are relevant and interesting to search users. For example, as shown in Figure 5, for a Kentucky Derby winning horse, the facts **breeder** will be more interesting than **Country**. This is in some ways similar to identifying high quality columns of the table, except that the columns and rows are transposed and the signals for identifying interesting facts may come from outside of the table. We leveraged a number of ranking techniques to achieve very high fact quality. We note that since Structured Snippets applied to all the results on the page and did not have to appear as prominently as Tables in Featured Snippets, they appeared much more often.

Richer snippet presentation using "knowledge carousels" constructed from horizontal tables was explored in [12]. Knowledge carousels found a surprisingly powerful application in

[4]Snippets are small samples of content that gives search users an idea of what's in the webpage for each page in the search result.
[5]https://ai.googleblog.com/2014/09/introducing-structured-snippets-now.html



Nyquist (horse) - Wikipedia
Wikipedia › wiki › Nyquist_(horse)

Nyquist (foaled March 10, 2013) is a retired American Thoroughbred racehorse, winner of the 2016 Kentucky Derby. He also won the 2015 Breeders' Cup Juvenile, becoming only the second horse to complete the Juvenile-Derby double.

Breed: Thoroughbred
Dam: Seeking Gabrielle
Breeder: Summerhill Farm
Foaled: March 10, 2013 (age 5)

Figure 5: Structured Snippets providing long tail structured data as part of Google search result's snippet

news, where readers of a news article can easily get access to structured data knowledge that are super-relevant to the content. The main challenge here is to understand the matching between the news article and the structured table. The feature was launched in the Google News Android app in August 2017.

3.6 Microsoft Products

At Microsoft, Chakrabarti, *et al.* [9] described how web tables have been used in their synonym and web tables services. For the synonym service, web tables were used to filter spurious synonyms (if they appear in the same columns) and identify novel synonyms (if they appear in the same rows) [21]. This service is used in Bing Snapp, Ask Cortana, Bing Knowledge API, and Bing synonym API. Vertical-specific search engines on customer websites often use the synonym API to disambiguate keyword searches. The web tables service has been integrated into Excel PowerQuery (in 2013) as a keyword table search feature, where users can search for tables to populate excel sheets through a search interface, and into Bing Search (in 2015), where tables are returned for 2% of search queries with 98% precision.

4. THE LANDSCAPE OF CONSUMER STRUCTURED DATA

One major change over the past ten years has been the explosion of consumer demand for structured data. Mobile apps, voice assistants, structured "infobox-style" web search results, and other *consumer structured data applications* were all either nonexistent or in their infancy at the time of the original WebTables paper. Today, they are all very popular and powered by structured data. These diverse consumer applications are powered by an equally diverse range of structured data resources.

In this section we review this landscape and its characteristics with the goal of motivating future opportunities.

We can think of at least four different classes of sources that power consumer structured data applications, ordered roughly from most to least structured:

1. **Relational Data** with a standard schema, sometimes in stream form, such as weather or stock quotes. These data products have been commercially available for

many years. The number of unique topics is fairly small, but the data is of extremely high-quality.

2. **Knowledge Graph Data** is information that usually has a relational-like schema[6]. Like applications of the relational data mentioned above, the contents of the knowledge graph is maintained with very high levels of accuracy. KGs cover a large range of popular topics either through human curation or automatic extraction. The KG content grows relatively slowly and is often maintained by developers in an "entity-centric" rather than a "schema-centric" manner. Coverage of KGs is highly biased to popular entities, relations and events, resulting in a very sparse graph. Google's Knowledge Graph[7], YAGO [31], Microsoft's Satori[8], and WikiData [33] are all good examples.

3. **WebTables** offers "long-tail" structured tables found on Web pages online. Unlike most knowledge graphs, the WebTables corpus is curated entirely automatically. As a result, the WebTables corpus can cover far more topics than the products above, including some topics that are quite obscure.

4. **Web Documents** may not appear structured at all, but many of them have enough detectable structure that they can be used in certain consumer data products. Traditional web search, of course, is the most well-known and popular system. But these "unstructured" texts can sometimes be used in search-driven question answering (such as medical question answering), and in voice assistant use cases. Of course, web documents cover a vast number of topics.

The points in the spectrum above also differ in other ways.

Query access: Relational data generally permits the most expressive queries, enabling full SQL. Knowledge graph and WebTables data can support single-record queries, but due to limited coverage generally cannot provide accurate aggregations. Finally, raw web documents only support NLP questions.

Conforming to integrity constraints: Data systems at the top of the list have comparatively more machinery for checking model compliance than systems at the bottom. A relational database, of course, will not maintain data that violates its schema. Knowledge graphs will enforce attribute types (e.g., that *height* is an integer), but generally rely on non-technical mechanisms to ensure good modeling (e.g., that *people* have a *height*, *weight*, and *birthday*) and to avoid missing values. WebTables applies probabilistic methods for attribute typing, and relies on indirect hints to detect good modeling (such as distribution of rows and columns). Web documents have no real data model at all, apart from being a "high-quality" page.

Diversity of topics: The diversity of data topics increases as we go down the list of web data resources, perhaps because the costs of building model-compliant data decreases. It is much harder to build a relational database than to type some text into a page.

Objective vs. subjective facts: Web query results carry some amount of implicit responsibility for the facts they present; single-answer systems (such as QA or voice assistants) tend to carry more, while search result pages — filled with links to the original publisher — carry less. In the former case, the search engine is perceived as giving a definite answer, while in the latter, it is just an intermediary between the user and relevant content. Data systems thus require more objective, neutral sources for the most sensitive user applications, but can use potentially-subjective sources for other applications. Applications using relational databases and KGs are engineered to contain primarily objective facts, making them suitable for returning factual answers. In contrast, WebTables and Web documents today are mixtures of objective and subjective information, making them appropriate for a broader range of information needs.

It's important to note that whether data is objective vs. subjective is a *schema level* decision. For example, the attribute *cute* of an animal would be subjective – there is no ground truth. In contrast, the number of people attending an event (e.g., a presidential inauguration) is an objective fact whose value may or may not be known and could be wrong in a particular database.

To date, knowledge graphs have simply not modeled attributes that are subjective thereby limiting their potential coverage of facts of interest to users quite a bit, but at the time avoiding more urgent and thorny issues.

Factual Accuracy: The accuracy of purely factual claims also generally declines as one goes down the list. One reason is that wider populations of users can contribute to, and edit, the corpora lower in the list, thus opening more room for casual fact-checking or vandalism.

Consumer structured data applications benefit from data resources that are *highly-structured*, *high-coverage*, and *accurate*. More structure means more distinct applications can use the data; more coverage means more user queries can be answered; and more accuracy means users can trust the application results. Applications also need *objective* input data, so that users do not receive a false sense of confidence about the application. At the same time, the role of *subjective* data is important to note — users often seek opinions which guide their decisions. Modeling subjective data and its provenance is an exciting direction for future research.

The current landscape of data resources partially fills these needs, but there is lots of room to do better. In the next section we will discuss how researchers can potentially improve these resources, and thereby makes things better for the huge numbers of people now using consumer structured data applications.

[6]It may seem surprising to say that knowledge *graph* data has a relational schema rather than a graph-oriented data model. It is true that most knowledge graph products do not have a single administrator-designed relational schema, and also true that graph-style measures such as centrality can be well-defined on knowledge graphs. However, in practice, knowledge "graphs" act more like "collections of relational tables with informal schema enforcement": entities tend to exhibit a small number of fixed attributes, and computing graph-style measures is a rare use case.

[7]`https://www.google.com/intl/en_us/insidesearch/features/search/knowledge.html`

[8]`https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/`

5. FUTURE OPPORTUNITIES

As we discussed, *highly-structured* resources are often not *high-coverage*, and *high-coverage* resources may not be *accurate*. Improving these resources is a hard challenge and would have a direct positive impact on the huge number of people now using consumer structured data applications.

Unfortunately, the technical path for improving these data resources is arduous. Knowledge graphs are very useful, but because of high standards for factual accuracy, they are generally thought to be extremely expensive to construct and expand their coverage. WebTables offer better coverage, but their unclear objectivity and accuracy means they can only power certain consumer applications. Meanwhile, relational databases and Web documents have the same vices (low coverage for the former; low objectivity and accuracy for the latter), but are far older, better understood, and to our mind offer worse prospects for near-term improvement in their impact on consumer data systems than knowledge graphs or WebTables.

Moreover, there is substantial work still to be done in exploring possible software architectures for consumer data systems, and in pursuing applications for deeper engagement with tables on the Web. In this section, we propose a few exciting directions for continuing these lines of work.

5.1 Improving Structured Data Resources

We need structured data resources that have the structure, accuracy, and objectivity of relational databases and KGs, but the coverage (and accompanying cost-effectiveness) of WebTables and web documents.

The WebTables project was a major step in the right direction, by creating a class of data that has web-like cost and coverage properties, but can still be used in some structured applications. Later projects developed signals for detecting extremely high-quality tables. For instance, Biperpedia [19] leveraged an existing ontology to identify high quality web tables whose semantics can be identified. Similar approaches towards identifying high quality tables have used highly-structured corpora [25, 17]. Even so, there is still opportunity to develop better tools for structured extraction [42, 11], data cleaning [28], and objectivity-testing [38, 29].

Another opportunity is to spark a virtuous cycle that encourages users to create and publish data in web documents in a structured form. Projects such as Fusion Tables [20] and Exhibit [23] are structured data authoring tools for end users that provide value-add in the form of visualization and presentation. Similarly, the presence of WebTables in search results that conform to a standard structure bring awareness to millions of potential data authors.

Finally, it may be possible to grow ”effective coverage” by rethinking how query systems show information lineage to users. The existence of a hyperlink on a search result page is now popularly understood to mean the search engine itself has no control over the link target's content, and has limited responsibility in surfacing the link at all. It may be possible for consumer data applications to use WebTables information more in more cases, if applications communicate that a particular query answer may be subjective or unreliable. Solving this problem effectively will require a combination of expertise in HCI, social science, and data management.

5.2 Enabling More Diverse Architectures

Despite the problems we listed above, if you had told us 10 years ago that in 2018 there would be a flourishing system of (1) general-purpose structured data resources and (2) consumer applications that consume those resources, we would have been thrilled! However, we probably would have predicted that the software architecture itself would more closely resemble the open Web. Instead, the data resources and consumer clients (such as structured search and voice assistants) have become tightly integrated and usually under the control of individual organizations.

These monolithic systems — in which the data resources and clients are tied to each other and cannot be easily modified by users or third parties — have some real advantages. It is easier to ensure semantic compatibility between the clients and structured datasets. For example, a voice assistant engineer can know in advance the label that the backing knowledge graph uses for the *height* attribute. It may also make it easier to hide known data quality or coverage problems, through clever client query rewriting.

However, monolithic architectures also pose clear downsides. It is difficult for individual users to modify the backing data resources, so users cannot easily add idiosyncratic or small-audience datasets. For example, a user might want a voice assistant that can answer questions about her employee's org chart, but today's knowledge graphs do not accept arbitrary contributions from third parties. Publishing the data via WebTables is possible, but WebTables' data quality issues and the fact that their semantics is not fully understood by the agent mean it is challenging to use for voice applications.

Similarly, it is difficult for users to swap out new client software without also swapping the back-end KG or WebTables resources. A startup that specializes in, say, medical question understanding will also need to take on the burden of building the entire data backend.

Building a non-monolithic open consumer data stack — in which anyone can contribute a novel dataset and specify the breadth of its possible use, and users can always rely on objective, accurate, high-coverage answers — poses serious technical challenges. Client engineers cannot assume that the data resources will use the clients' preferred attribute names. Objectivity and accuracy checking probably has to be entirely automated, in order to keep up with data contributions from every corner of the internet. Data authors cannot assume they know the query workload in advance, because there will always be novel and specialized clients to service. In many ways, the technical challenges here represent the ”cost curve” ones in the section above, but at an extreme scale.

5.3 Supporting Human-Scale Use Cases

The growth of popular consumer data applications is an absolutely massive benefit for researchers interested in structured data online, because it enables workload- and metrics-driven technical improvement: percentage of queries answered, percentage of facts correctly recognized as correct, and so on. Many of these applications are not only exciting but were part of the early vision for WebTables and other structured data projects, so it is entirely natural to focus on them. There is lots of opportunity in improving support for consumer data applications, and the database community should seize it.

However, there has also been a thread of this work that focuses on workloads that will never see a billion users or a billion dollars: analysts with limited resources who are trying to solve problems however they can. One of the motivations behind WebTables was to dramatically lower the cost of data discovery and acquisition for domain experts who were familiar with data, but did not have a large company or university to back them. This motivation came to fuller fruition in the form of Fusion Tables, and in Microsoft Excel's PowerBI table search features, which combined WebTables' inexpensive data search with usable and inexpensive analytical tools.

We believe these applications are extremely important and would like to see the database community support individual analysts even more. To a large extent, this work is taking place as part of the broader effort to build tools that support data scientists. That effort could be made even more effective by focusing on the data acquisition toolchain.

6. REFERENCES

[1] Common crawl. http://commoncrawl.org/.

[2] S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, pages 5–16, 2002.

[3] S. Balakrishnan, A. Y. Halevy, B. Harb, H. Lee, J. Madhavan, A. Rostamizadeh, W. Shen, K. Wilder, F. Wu, and C. Yu. Applying webtables in practice. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*, 2015.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[5] C. Bizer. The emerging web of linked data. *IEEE Intelligent Systems*, 24(5):87–92, Sept. 2009.

[6] M. J. Cafarella, A. Y. Halevy, and N. Khoussainova. Data integration for the relational web. *PVLDB*, 2(1):1090–1101, 2009.

[7] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *PVLDB*, 1(1):538–549, 2008.

[8] M. J. Cafarella, A. Y. Halevy, Y. Zhang, D. Z. Wang, and E. Wu. Uncovering the relational web. In *11th International Workshop on the Web and Databases, WebDB 2008, Vancouver, BC, Canada, June 13, 2008*, 2008.

[9] K. Chakrabarti, S. Chaudhuri, Z. Chen, K. Ganjam, Y. He, and W. Redmond. Data services leveraging bing's data assets. *IEEE Data Eng. Bull.*, 2016.

[10] H.-H. Chen, S.-C. Tsai, and J.-H. Tsai. Mining tables from large scale html texts. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*, COLING '00, pages 166–172, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[11] Z. Chen, M. J. Cafarella, and H. V. Jagadish. Long-tail vocabulary dictionary extraction from the web. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*, pages 625–634, 2016.

[12] F. Chirigati, J. Liu, F. Korn, Y. Wu, C. Yu, and H. Zhang. Knowledge exploration using tables on the web. *PVLDB*, 10(3):193–204, 2016.

[13] X. Chu, Y. He, K. Chakrabarti, and K. Ganjam. TEGRA: table extraction by global record alignment. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 1713–1728, 2015.

[14] D. Deng, Y. Jiang, G. Li, J. Li, and C. Yu. Scalable column concept determination for web tables using large knowledge bases. *PVLDB*, 6(13):1606–1617, 2013.

[15] J. Eberius, K. Braunschweig, M. Hentsch, M. Thiele, A. Ahmadov, and W. Lehner. Building the dresden web table corpus: A classification approach. In *2nd IEEE/ACM International Symposium on Big Data Computing, BDC 2015, Limassol, Cyprus, December 7-10, 2015*, pages 41–50, 2015.

[16] H. Elmeleegy, J. Madhavan, and A. Y. Halevy. Harvesting relational tables from lists on the web. *PVLDB*, 2(1):1078–1089, 2009.

[17] J. Fan, M. Lu, B. C. Ooi, W. Tan, and M. Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 976–987, 2014.

[18] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. Towards domain-independent information extraction from web tables. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 71–80, 2007.

[19] R. Gupta, A. Y. Halevy, X. Wang, S. E. Whang, and F. Wu. Biperpedia: An ontology for search applications. *PVLDB*, 7(7):505–516, 2014.

[20] A. Y. Halevy. Data publishing and sharing using fusion tables. In *CIDR 2013, Sixth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 6-9, 2013, Online Proceedings*, 2013.

[21] Y. He, K. Chakrabarti, T. Cheng, and T. Tylenda. Automatic discovery of attribute synonyms using query logs and table corpora. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 1429–1439, 2016.

[22] V. Hristidis and Y. Papakonstantinou. DISCOVER: keyword search in relational databases. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pages 670–681, 2002.

[23] D. F. Huynh, D. R. Karger, and R. C. Miller. Exhibit: lightweight structured data publishing. In *Proceedings of the 16th international conference on World Wide Web*, pages 737–746. ACM, 2007.

[24] O. Lehmberg, D. Ritze, R. Meusel, and C. Bizer. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*, pages 75–76, 2016.

[25] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *PVLDB*, 3(1):1338–1347, 2010.

[26] X. Ling, A. Y. Halevy, F. Wu, and C. Yu. Synthesizing union tables from the web. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 2677–2683, 2013.

[27] R. Pimplikar and S. Sarawagi. Answering table queries on the web using column keywords. *PVLDB*, 5(10):908–919, 2012.

[28] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.

[29] T. Rekatsinas, M. Joglekar, H. Garcia-Molina, A. Parameswaran, and C. Ré. Slimfast: Guaranteed results for data fusion and source reliability. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1399–1414. ACM, 2017.

[30] A. D. Sarma, L. Fang, N. Gupta, A. Y. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu. Finding related tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 817–828, 2012.

[31] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[32] P. Venetis, A. Y. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. *PVLDB*, 4(9):528–538, 2011.

[33] D. Vrandecic and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014.

[34] C. Wang, K. Chakrabarti, Y. He, K. Ganjam, Z. Chen, and P. A. Bernstein. Concept expansion using web tables. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 1198–1208, 2015.

[35] D. Z. Wang, L. Dong, A. D. Sarma, M. J. Franklin, and A. Halevy. Functional dependency generation and applications in pay-as-you-go data integration systems. In *WebKB*, 2009.

[36] J. Wang, H. Wang, Z. Wang, and K. Q. Zhu. Understanding tables on the web. In *Conceptual Modeling - 31st International Conference ER 2012, Florence, Italy, October 15-18, 2012. Proceedings*, pages 141–155, 2012.

[37] Y. Wang and J. Hu. A machine learning based approach for table detection on the web. In *Proceedings of the Eleventh International World Wide Web Conference, WWW 2002, May 7-11, 2002, Honolulu, Hawaii, USA*, pages 242–250, 2002.

[38] Y. Wu, P. K. Agarwal, C. Li, J. Yang, and C. Yu. Toward computational fact-checking. *PVLDB*, 7(7):589–600, 2014.

[39] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 97–108, 2012.

[40] X. Yin, W. Tan, and C. Liu. FACTO: a fact lookup engine based on web tables. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, pages 507–516, 2011.

[41] R. Zanibbi, D. Blostein, and J. R. Cordy. A survey of table recognition. *IJDAR*, 7(1):1–16, 2004.

[42] C. Zhang, J. Shin, C. Ré, M. J. Cafarella, and F. Niu. Extracting databases from dark data with deepdive. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 847–859, 2016.

[43] M. Zhang and K. Chakrabarti. Infogather+: semantic matching and annotation of numeric and time-varying attributes in web tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 145–156, 2013.