Applying WebTables in Practice

Sreeram Balakrishnan, Alon Halevy, Boulos Harb, Hongrae Lee,
Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, Cong Yu
Google Research
{sreevb,halevy,harb,hrlee,jayant,rostami,whshen,wilder,wufei,congyu}@google.com

ABSTRACT

We started investigating the collection of HTML tables on the Web and developed the WebTables system a few years ago [4]. Since then, our work has been motivated by applying WebTables in a broad set of applications at Google, resulting in several product launches. In this paper, we describe the challenges faced, lessons learned, and new insights that we gained from our efforts.

The main challenges we faced in our efforts were (1) identifying tables that are likely to contain high-quality data (as opposed to tables used for navigation, layout, or formatting), and (2) recovering the semantics of these tables or signals that hint at their semantics. The result is a semantically enriched table corpus that we used to develop several services. First, we created a search engine for structured data whose index includes over a hundred million HTML tables. Second, we enabled users of Google Docs (through its Research Panel) to find relevant data tables and to insert such data into their documents as needed. Most recently, we brought WebTables to a much broader audience by using the table corpus to provide richer tabular snippets for fact-seeking web search queries on Google.com.

1. INTRODUCTION

A few years ago we started investigating the collection of HTML tables on the Web [4], a vast resource that also inspired several other research efforts, e.g., [10, 3, 13]. Our goal was twofold. First, we wanted to characterize the size and quality of this untapped source of structured data. Second, we wanted to create services that would expose this content to Google users.

In the past few years, we have been tackling the main challenges concerning this collection: (1) extracting a high-quality corpus of HTML data and (2) recovering signals that provide semantic clues about the content of these tables. Based on our high-quality corpus, we demonstrated that structured data from WebTables is relevant to a broad set of services. First, we created a search engine for structured data whose index includes over a hundred million HTML tables and several million tables made public by the users of Google Fusion Tables, a tool for easily managing and visualizing data [6]. Second, we enabled users of Google Docs to tap into the table collection and insert data into their documents as needed. Most recently, we used the table corpus to present tabular data in the Google.com search result page, thereby reaching a much wider audience.

This paper describes the challenges we faced to create the table corpus and use it in the aforementioned applications. While some of the ideas underlying this work appeared in our previous research papers [4, 12], here we focus on the developments that were made and insights that were gained by applying these ideas in our products.

Section 2 describes how we built the table corpus. Section 3 describes how we enriched the corpus with semantics. Section 4 describes each of the applications and the additional innovations needed to launch them.

2. EXTRACTING HIGH QUALITY TABLES

The Web contains tens of billions of HTML tables, even when we consider only pages in English. However, over 99% of the tables do not contain data, but rather use the HTML table tag as a formatting mechanism for presentation (see Figure 1(b)). A more subtle problem we discovered while looking at many tables is that there is no clear definition of what makes a *good* table and the classification can be subjective. It was not uncommon to have disagreements among the evaluators of table quality, even when these evaluators had been studying HTML tables for quite a while. For example, while most people will agree that the table in Figure 1(a) provides valuable information, tables in Figure 2 may have more borderline importance. Such tables tend to be useful for an extremely small group of people, e.g., Figures 2(a) or (b), or have useful information in navigational links outside the table itself, e.g., Figure 2(c).

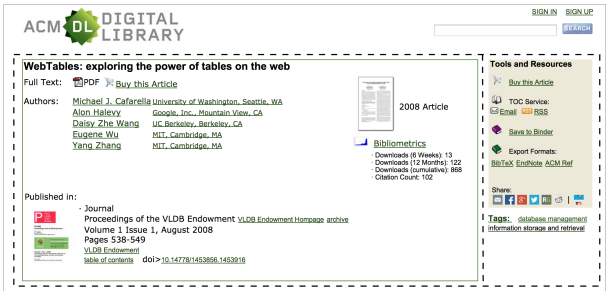
These characteristics of Web tables created unique challenges in identifying the (relatively) tiny subset (albeit huge in absolute numbers) of good tables among the sea of bad tables. The first concrete technical challenge is that there is an extreme imbalance of bad to good table examples, thereby making it difficult to train a machine learning classifier to recognize good tables. A random sample of tables will invariably consist of mostly negative examples, and hence any technique that classifies every table as bad would have performed quite well. The second challenge was that goodness of a table could be highly dependent on the application we were building.

7th Biennial Conference on Innovative Data Systems Research (CIDR '15) January 4-7, 2015, Asilomar, California, USA..

Table of liquid–vapor critical temperature and pressure for selected substances [edit]

Substance[7][8]	Critical temperature	Critical pressure (absolute)
Argon	−122.4 °C (150.8 K)	48.1 atm (4,870 kPa)
Ammonia[9]	132.4 °C (405.5 K)	111.3 atm (11,280 kPa)
Bromine	310.8 °C (584.0 K)	102 atm (10,300 kPa)
Caesium	1,664.85 °C (1,938.00 K)	94 atm (9,500 kPa)
Chlorine	143.8 °C (416.9 K)	76.0 atm (7,700 kPa)
Ethanol	241 °C (514 K)	62.18 atm (6,300 kPa)
Fluorine	−128.85 °C (144.30 K)	51.5 atm (5,220 kPa)
Helium	−267.96 °C (5.19 K)	2.24 atm (227 kPa)
Hydrogen	−239.95 °C (33.20 K)	12.8 atm (1,300 kPa)
Krypton	−63.8 °C (209.3 K)	54.3 atm (5,500 kPa)
Chlorine	−82.3 °C (190.8 K)	45.79 atm (4,640 kPa)
CH_4 (methane)		

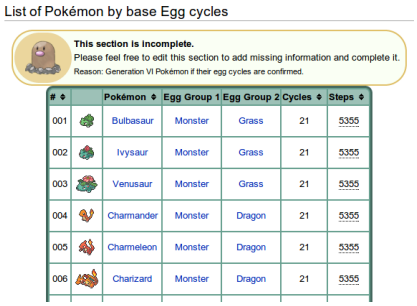
(a) The table shows critical temperature and pressure for various substances as a tabular data.



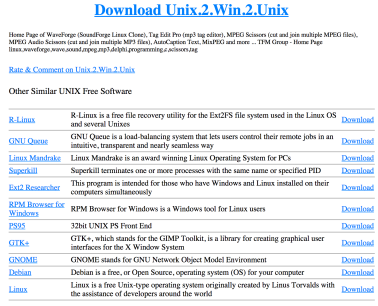
(b) The table marked by dotted line includes contents, but its primary goal is to control the layout.

Figure 1: Example data and formatting tables

(a) Personal note



(b) Game information with images



(c) Download links

Figure 2: Example tables with subjective value

To address these challenges, we followed a 2-step approach:

1. Simple rules: We manually gathered the most frequent bad table patterns and designed simple heuristic rules that filtered tables that were most certainly bad. Examples of those heuristic patterns include tiny tables (less than 3 rows and 2 columns), calendar tables, password tables, table-of-content tables. These heuristics eliminate about 90% of bad tables. After this step, bad tables still dominate good ones by factor of 9 to 1, which is much better than the 99 to 1 with which we started.

2. Machine learning classifier: At this point, the decision on which tables are considered good is dependent on the application for which we intend to use the table corpus. Here is a real example of such a choice. Originally, we considered tables to be good if they are similar in spirit to relational tables: they should contain multiple rows, where each row corresponds to data about a particular entity in the world (e.g., a location or a movie), and the columns of the table describe attributes of the row's entity. When we started developing tabular snippets for google.com search (see Section 4.3), however, we realized that a large collection of *vertical tables* turned out to be useful as well. Vertical tables (see Figure 3) typically have only two columns and a small number of rows. The vertical table lists the properties of a *single* entity, and the values in the first column are the names of the properties.

There are three main aspects in the development of our machine learning classifiers: feature design, training example generation, and model selection. The production features we eventually adopted encode both syntactic and semantic information. Semantic features include such signals as to whether the table content falls into a boilerplate section of the page as well as labeling the topic of each column in the table; Section 3 describes how we generated some of these semantic features. The structural features include the number of rows and columns, mean/variance of the number of characters per cell, the fraction of non-empty cells, the fraction of cells that are generated using <th> tags and the number of distinct tokens in a table.

In order to generate training examples we used two sources of data: a uniform sampling of tables taken after the heuristic filter stage (which includes only a small fraction of good tables, i.e., about 10%) and an additional uniform sample taken from tables that were marked good by a simplistic decision-tree classifier developed earlier [4], which makes use of the above features. Since the simplistic classifier is only used as a mechanism to collect training examples for a more advanced classifier, not as a production model to be deployed for real use, we did not rigorously evaluate its performance. It only has the property that a significant fraction (%50+) of the tables it labels as good are indeed good. Thus, once the two sources are combined, the final training set contains a larger proportion of good tables (about 35%), which we found useful for training a higher recall classifier.

After the training examples are generated, we train two high-accuracy classification models, one which classifies a table as "good horizontal" (e.g., Figure 1(a)) or not, and a second which classifies a table as "good vertical" (e.g., Figure 3) or not. Both models are trained using the multi-kernel SVM described in [5], which uses an alignment based metric to

Figure 3: An example vertical table

combine three Gaussian kernels with varying bandwidth parameters. We note the use of a non-linear model is important as linear models simply did not perform well on this task. On a benchmark test set containing 26% horizontal tables the horizontal classifier achieves a precision/recall of 83%/91% and overall accuracy of 93%, while for a benchmark test set containing 13% vertical tables the vertical classifier achieves a precision/recall of 89%/85% and overall accuracy of 96%.

The efforts described above focus on HTML tables. We note that there are a significant number of tabular data sets that are not in HTML tables, such as HTML lists and repeated patterns on the Web. We developed techniques for extracting repeated patterns with the goal of including them in our corpus. However, such data are often noisier than tables and pose extra challenges in their extraction and annotation. Using such data in production requires further refinement to meet the quality bar.

In addition to classifying tables, a row-level classifier was used to distinguish between header and non-header rows. The features for the row-level classifier are similar to those for the table-level classifiers. They are, however, computed based on the content of the row. There are also additional features that are based on (dis)similarity to neighboring rows and absolute information about the row, (e.g., whether this is the first row in the table). A standard SVM model with an RBF kernel is trained base on those features. On a benchmark dataset containing 4% header rows the model achieves a precision/recall of 96.6%/85.1%, overall row-level accuracy of 99.5%.

In summary, we constructed a high quality table corpus of more than a hundred million tables beginning with tens of billions HTML tables we originally found. In the next section, we describe how we annotate these tables with some of their semantics.

3. EXTRACTING TABLE SEMANTICS

In order to serve tables to users, we need to be able to match tables to relevant queries. However, matching tables to queries is challenging because most of the semantics of the table are implicit or embedded in the surrounding text. Tables on the Web are typically designed so casual readers (as opposed to machines) can understand their content. Hence, even when tables do contain header rows, the attribute names are often obscure, useless, or require understanding the context of the page. We developed several techniques to partially recover table semantics by enriching the table with additional annotations.

Detecting subject columns: We observed that over 75% of the tables in our corpus contain a column that lists the entities the table is about, while the other columns describe properties of these entities. For example, in Figure 1(a), the table is about substances and properties are critical temperature and critical pressure. Hence, we developed an algorithm for detecting that column, which we refer to as the *subject* column. Unlike primary keys in relational databases, the subject column in a Web table need not be a key of the table and may contain duplicate values. We note that it is possible that the subject of the table is represented by more than one column, but we currently do not attempt to identify these cases which are also relatively rare in practice.

We model the subject detection as a binary classification problem and trained an SVM classifier using one thousand manually labeled tables. This classifier achieved 94% accuracy in our experiments. After identifying the subject column, we treat the set of header rows in the table as an ad-hoc schema.

Class labels for columns: Our next annotation is to attach classes to columns of tables. For example, a column can contain entities such as countries, presidents, or values like phone numbers. Identifying the class can help eliminate spurious table matches and aid subsequent analysis steps on the table.

Our method is based on the following intuition: if a substantial number of cells in a column A belong to a class C, we attach C as a class label to A. However, the challenge is that for many cell values we may not know which classes they belong to, and they may belong to multiple classes.

We used the Google Knowledge Graph (KG) [1] (whose schema, and hence set of classes is identical to that of Freebase) to map cell values to entities, and then to the classes in the KG to which they belong. Specifically, we derived two databases from KG. The first is of the form (value, entity, score) which maps a string value to a KG entity with confidence score. The same string value usually maps to multiple entities due to ambiguity. The second is an isA database of the form (entity, class) that maps KG entities to their classes. For each cell value v_j in column A, we first look it up in the first database to get all possible KG entities paired with confidence scores, i.e., (v_j, e_i, s_i), where e_i is an entity ID in the KG and s_i is the confidence that v_j maps to e_i. Then we replace each candidate entity with each of its classes by consulting the isA database. The results are tuples of the form (v_j, c_i, s_i), where c_i is a class name.

Next, we aggregate all such cell-level candidate classes to collectively determine the class labels for the whole column. When we originally developed this general method [12] we experimented with several aggregation methods because the isA database we were using was extracted from the Web using Hearst patterns [9], and hence noisy. Classes in KG have narrower coverage but are manually curated and of high quality. Thus, by switching over to the KG classes, we were able to apply a simple and more computationally efficient

majority voting algorithm, i.e., any class C that applied to more than 50% of the cells in a column was associated with the column.

After getting class labels for columns, we use them to refine the linking from cell values to KG entities. For each cell we filter those candidate entities which does not belong to any of those column-level classes. If there are still multiple candidate entities left, we pick the one with the highest mapping score. An important observation of our work is that considering the structure of the table was extremely important. To prove the point, we also tried to treat table content as plain text and apply entity linking techniques, e.g., [8]. However the performance was not as good mainly because they do not utilize the table structure information as effectively. For columns which hold location data, we use Google Maps API [2] to determine the latitude and longitude coordinates of each cell.

An interesting observation was that being able to assign a label to a column for the subject column was a very effective signal for table quality. When semantics can be discovered for a table, it is more likely that the table contains high-quality data. Hence, the percentage of cells in a column that were mapped to the same KG class was one of the important semantic features for our classifier.

Detecting binary relationships between columns: The semantic class of the subject column tells us quite a bit about the contents of the table and sometimes suffices for matching tables to queries. However, it is also important to know what properties of the subject-column entities are described in the table, i.e., what are the binary relationships between the subject column and each of the other columns in the table.

These relationships are typically expressed in text surrounding the tables. However, the text can be quite long and detecting the specific phrases that refer to these relationships can be tricky. To that end, we have recently developed a dictionary of attributes that exist in search queries and Web text [7]. For example, for the class COUNTRIES we mine thousands of attributes that may be associated with countries, ranging from common ones such as GDP and CAPITAL to longer-tail ones, such as COFFEE PRODUCTION and CORRUPTION INDEX. We then use this collection to find prominent words in the surrounding text that could refer to the attributes of the subject column. These annotations are already in our corpus but are not applied to production applications yet due to their recent addition. Our preliminary analysis shows that they enable us to recover much more of the table semantics. We expect the new addition to have production impact in the near future.

Finally, we extract captions, and the text surrounding a table. Often times, captions and text surrounding the table describe some of the semantics of the table such as the units used for numerical columns, relationships between columns as described above, or some constraint that applies to the entities in the table (e.g., that the table has data collected in 2013, or contains only countries in the southern hemisphere). To extract surrounding text we obtain a window of text before and after the table. To extract the caption of the table, we analyze the DOM structure of the page to identify prominent fragments (e.g., header text) and use syntactic signals such as the distance from the table.

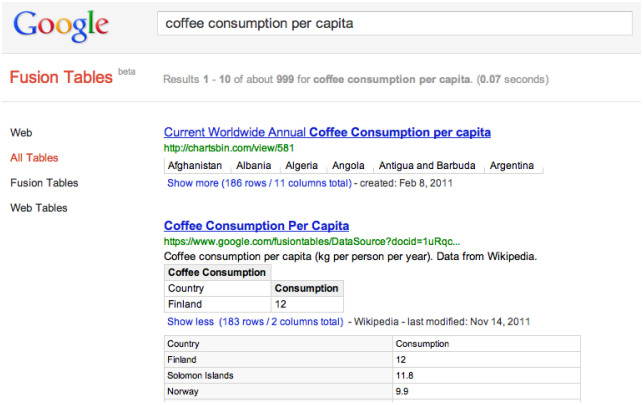


Figure 4: Current table search results page

4. APPLICATIONS OF WEB TABLES

Now that we have a high-quality corpus, we set out to incorporate it into Google services and we were pleasantly surprised at the breadth of applications for WEBTABLES at Google. We describe these services in the order they were created.

4.1 Table Search

Our first major application was the one we anticipated at the start of the project: we want to make all the high quality tables available to the users through a data-specific search engine. That led us to Google Tables[1], a highly customized table search engine that is designed to provide our users the ability to quickly find tabular data on the Web relevant to their information needs.

The system extensively leverages Google's scalable search infrastructure, including compartmentalized indexing and scoring infrastructure, which allows for easy injection of data and scoring signals into the serving system. The infrastructure also provides scalable keyword based retrieval and processing infrastructure that allows a large number of documents to be fetched based on the query keywords and scored. The key contributions of this system, however, lie in two major aspects. The first is the table-specific ranking signal that we designed and incorporated into the serving system that ensures relevant table results are returned to users. The second is the auxiliary data and functionalities we added to the search engine that provide an enhanced experience.

The ranking signals come from three main aspects. The page-level signals include the usual suspects that are borrowed from Web document ranking and are not specific to tables. Those signals include pagerank, link analysis, and we simply apply the existing (internal) techniques from the Web document ranking.

The second major set of signals come from the structure of the tables. Starting from the corpus of over $100M$ high-quality tables, we create an index that identifies each token that appears in a table. This indexing structure pinpoints the tables and cells in which the token appears in, and more importantly, whether the token appears in a header row or a subject column (or both) and whether the token appears in some relevant context of the table (such as table captions or titles). This indexing structure not only allows us to retrieve the set of tables containing tokens that match the user's keyword query, but also allows us to rerank the resulting

[1]https://research.google.com/tables

tables based on where the query hits appear in the table. During scoring time, we consider hits on the schema, hits on table cells that are in columns deemed important, and the context surrounding the table. The contributions from different hits are combined via a linear weighted sum model, where the weights are learned via simple linear regression.

The third major set of signals come from the semantic annotations (Section 3). In addition to the original tokens from the table content, we further index tokens that come from annotations on the content. To leverage those annotation tokens, we also perform semantic annotations on the queries to supplement raw query tokens with corresponding rewritten annotation query tokens. Similarly to IR-style text normalization, both table content token and query token are *normalized* via common annotations and matched together and those matches provide this third major set of ranking signals.

Presenting table search results turns out to be more challenging than we expected. Snippets, i.e., the little piece of text comes after each result title/URL, are integral parts of modern search results. One of the major differentiators of our table search engine is that it provide auxiliary information that significantly enhances the user experience beyond simply finding relevant table results. These auxiliary information are reflected in two ways: table snippet and table import. Once a table is retrieved, instead of simply showing the first few rows or columns of the table, we use query hits on the table to generate a preview of the table that is most relevant to the query, by projecting and selecting the rows and columns where the hits occur. Each hit is ranked based on how much it matters to the table snippet. The signals we use include the number of cells the same hit appears in, whether the hit is in a header row or subject column, and whether the hit belongs to a group of hits in the same cell, row or column.

Table snippets (Figure 4) gives our users a nice preview over the table. Just previewing the data, however, may not be the end goal of users; they often wish to import the data and perform analytical tasks. Thus, table search aims to serve as a launching pad for completing such tasks using the found data. As a first step in that direction, we added support to import tables found in table search into Google Fusion Tables or Google Sheets. Once imported, the user can query and visualize the data, combine it with other sources and export it to other tools for further manipulation.

Today, Google Tables is used by hundreds of thousands of users and table import is a very popular feature.

4.2 Docs and Presentation

Google Research Tool, as part of Google Docs and Google Presentation, is a tool that provides users a convenient way to search for information while they are working on their documents. For example, 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 for Research Tool and was requested by our users.

We integrated Google Tables into the Research Tool by incorporating table search as part of the Everything Search tab as well as the Table Search tab, collectively called Tables in Research Tool. For the Everything Search tab, which presents results returned by regular Google Web search, we inject a table snippet into a Web result's summary (i.e., snippet) if the Web result contains a high quality table that

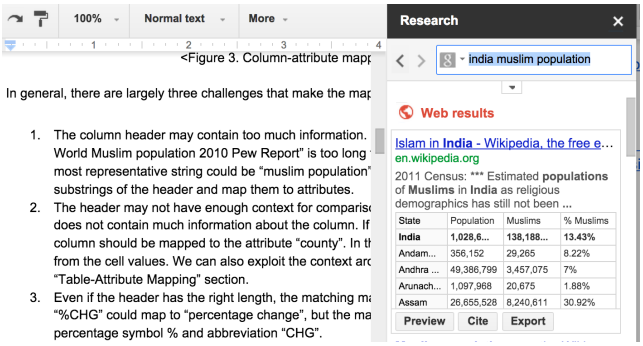


Figure 5: Research tool in Google Docs

would have been returned by Google Tables for the same query. The Table Search tab, on the other hand, is essentially another interface for Google Tables. Furthermore, the user can then view the entire content of the table by clicking on the table snippet and easily drag it into the document being edited. The same import functionality is similarly available in Research Tool as well if the user would like to add the table to Google Fusion Tables or Spreadsheets.

Compared to users of Google Tables, who intentionally came to the table search engine to find relevant data, typical users of Research Tool are likely to be 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 a much higher goodness or verticalness scores, presence of both header rows and subject columns, and, for non-vertical data tables, presence of numerical columns or columns with sufficient semantic annotations.

The other challenge for Tables in Research Tool is to select the best sub-table to return to the user, since the space in the Research Tool is much more limited. We used the same method described in Section 4.1, but adopted a higher bar for hit selection and augmented the table snippet by also highlighting the cell that is relevant to the user's query when appropriate. For example, for the query "india muslim population", we highlight the cell, which contains the requested information, at the intersection of the "India" row and "Muslim" column (Figure 5).

Tables in Research Tool is very well received by the community at its launch[2] nearly a year ago. Today, it is one of the most commonly used features in the Research Tool of Google Docs and Google Spreadsheet.

4.3 Tabular Results in Web Search

To bring a much larger benefit from Web tables, we embarked on direct integration of WebTables into Google.com search results. Our key goal was to highlight that certain Web results include tabular data and help guide our users to those pages. We mainly focus on fact seeking queries, i.e., entity-property queries, such as "literacy rate of Malaysia", "american airlines baggage fees", or "critical temperature of krypton". The queries impacted typically belong to the longer-tail content that are not suitable for curation and

[2]https://plus.google.com/+GoogleDrive/posts/e7qsYrUC9ur

may not be stored in the KG.

The integration with Google search raises multiple challenges. First, we need to identify queries that are fact seeking. Second, when a query is fact seeking, we need to find pages that contain relevant tables and when there are multiple such tables, choose which one to highlight in the result. Finally, we need to design a UI that is most effective in the limited space.

Given a web query, when appropriate, we create a tabular snippet on top of the search results with selected rows and columns from a Web table. This enables people to easily find pages that contain tables that are very likely to answer their question. Currently, we examine the top few results in the search ranking and decide whether any of them contain a Web table that is relevant to the query. Note that restricting to the top results already imposes a very high quality bar on the possible tables we consider and also about the relevance of the table to the query.

To decide whether a table is relevant, we perform a structured matching between the query and tables where the entity in the query identifies relevant rows in the table and the property in the query constraints the relevant columns. As an example, given a query "critical temperature of krypton" and the Web table in Figure 1(a), we can potentially match "krypton" to a cell in the subject column and "critical temperature" to the CRITICAL TEMPERATURE column, and thus identify the table cells that are most relevant to the query. We can then present, to the user, a subset of rows and columns that include those most relevant cells. Performing such matching with high degree of accuracy requires good understanding of both the query and table. As mentioned in Section 3, Web tables schema information is inherently vague and we therefore consider various sources of information including page context (title, caption) to understand the table as well as the table itself.

5. RELATED WORK

There has been a rich body of work on extracting HTML tables from the Web, annotating their schema, and searching Web tables e.g., [4, 12, 3, 13, 10]. Pimplikar and Sarawagi proposed a structured search engine based on Web tables which returns a multi-column table in response to a query consisting of keywords describing each of its columns [10]. They defined a query segmentation model for matching keywords to table columns, and a mechanism of exploiting content overlap across table columns. Yakout et al. presented three core operations, namely entity augmentation by attribute name, entity augmentation by example and attribute discovery, that are useful for information gathering tasks on a Web table corpus [13]. To achieve higher precision and coverage, they considered indirectly matching tables as well. Adelfio and Samet improved the schema extraction task and proposed a classification technique based on conditional random fields in combination with a novel feature encoding method called logarithmic binning [3].

Several recent studies aim to respond to user queries with Web tables or a knowledge base system. Sarawagi and Chakrabarti developed techniques to present a ranked list of quantity distributions for queries whose target is a quantity with natural variation [11]. Yang et al. studied how to find highly relevant patterns in a knowledge base for user-given keyword queries to compose table answers [14].

6. CONCLUSIONS

Since we started building the WEBTABLES repository, we have made significant efforts to apply WEBTABLES in practice. Table data now appears in three different products and more is coming soon. In this paper, we discussed challenges faced, lessons learned, and new insights that we gained.

While these applications are a great success, they also highlighted major areas for developments for the future. First, we aim to build an even larger corpus based on other structured data and implicitly structured data. Second, we need better understanding of table semantics and query semantics to improve coverage.

7. REFERENCES

[1] Google Knowledge Graph. `http://www.google.com/insidesearch/features/search/knowledge.html`.

[2] Google Maps API. `https://developers.google.com/maps/`.

[3] M. D. Adelfio and H. Samet. Schema extraction for tabular data on the web. *Proc. VLDB Endow.*, 6(6):421–432, Apr. 2013.

[4] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: exploring the power of tables on the web. *VLDB*, 2008.

[5] C. Cortes, M. Mohri, and A. Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13(1):795–828, 2012.

[6] H. Gonzalez, A. Y. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, and W. Shen. Google fusion tables: data management, integration and collaboration in the cloud. In *SoCC*, 2010.

[7] R. Gupta, A. Halevy, X. Wang, S. Whang, and F. Wu. Biperpedia: An ontology for search applications. *VLDB*, 2014.

[8] X. Han, L. Sun, and J. Zhao. Collective entity linking in web text: a graph-based method. In *SIGIR*, pages 765–774, 2011.

[9] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLIN*, 1992.

[10] R. Pimplikar and S. Sarawagi. Answering table queries on the web using column keywords. *VLDB*, 2012.

[11] S. Sarawagi and S. Chakrabarti. Open-domain quantity queries on web tables: Annotation, response, and consensus models. In *SIGKDD*, 2014.

[12] 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. *VLDB*, 2011.

[13] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *SIGMOD*, 2012.

[14] M. Yang, B. Ding, S. Chaudhuri, and K. Chakrabarti. Finding patterns in a knowledge base using keywords to compose table answers. In *VLDB*, 2015.