Uncovering the Relational Web

Michael J. Cafarella[*]
University of Washington
mjc@cs.washington.edu

Alon Halevy
Google, Inc.
halevy@google.com

Daisy Zhe Wang
U.C. Berkeley
daisy@cs.berkeley.edu

Eugene Wu
MIT
eugene@csail.mit.edu

Yang Zhang
MIT
zhang@csail.mit.edu

ABSTRACT

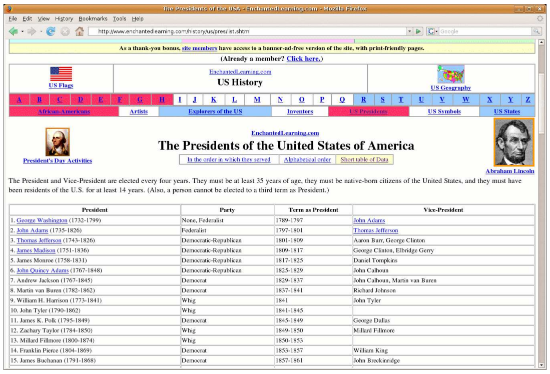
The World-Wide Web consists of a huge number of unstructured hypertext documents, but it also contains structured data in the form of HTML tables. Many of these tables contain both relational-style data and a small "schema" of labeled and typed columns, making each such table a small structured database. The WebTables project is an effort to extract and make use of the huge number of these structured tables on the Web. A clean collection of relational-style tables could be useful for improving web search, schema design, and many other applications.

This paper describes the first stage of the WebTables project. First, we give an in-depth study of the Web's HTML table corpus. For example, we extracted 14.1 billion HTML tables from a several-billion-page portion of Google's general-purpose web crawl, and estimate that 154 million of these tables contain high-quality relational-style data. We also describe the crawl's distribution of table sizes and data types.

Second, we describe a system for performing *relation recovery*. The Web mixes relational and non-relational tables indiscriminately (often on the same page), so there is no simple way to distinguish the 1.1% of good relations from the remainder, nor to recover column label and type information. Our mix of hand-written detectors and statistical classifiers takes a raw Web crawl as input, and generates a collection of databases that is five orders of magnitude larger than any other collection we are aware of. Relation recovery achieves precision and recall that are comparable to other domain-independent information extraction systems.

1. INTRODUCTION

Search engines, Web browsers, and most users consider the Web as a corpus of unstructured *documents*. While hierarchical URL names and the hyperlink graph pose some structure, the basic unit for reading or processing is the unstructured document itself. However, Web documents often contain large amounts of relational data. For example, the Web page shown in Figure 1 contains a table that lists American presidents[1]. 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. Relational operators like selection and projection would make obvious sense on this data. This Web page essentially contains a small relational database, even if it lacks the explicit metadata traditionally associated with a database.



The Presidents of the USA - EnchantedLearning.com - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.enchantedlearning.com/history/us/pres/list.shtml
Google
US Flags
EnchantedLearning.com
US History
US Geography
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
African-Americans Artists Explorers of the US Inventors US Presidents US Symbols US States
President's Day Activities
Abraham Lincoln
In the order in which they served
Alphabetical order
Short table of Data

As a thank-you bonus, site members have access to a banner-ad-free version of the site, with print-friendly pages.

(Already a member? Click here.)

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. The relation here has a schema that is never explicitly declared but is obvious to a human observer, consisting of several typed and labeled columns. The navigation bars at the top of the page are also implemented using the `table` tag, but clearly do not contain relational-style data. WebTables **attempts to extract only true relations, including appropriate column labels and types.**

This paper describes the first stage of the WebTables system, whose goal is to study and leverage the collection of useful HTML tables on the Web. Our first contribution is a large-scale study of the HTML table corpus from which we extract the relational web. We began from a portion of the google.com Web crawl (consisting of several billion pages) and extracted approximately 14.1 billion raw HTML tables. We show that the number of high-quality relations is a tiny

[*] Work done while all authors were at Google, Inc.

[1] `http://www.enchantedlearning.com/history/us/pres/list.shtml`

Proceedings of the 11th International Workshop on Web and Databases (WebDB 2008), June 13, 2008, Vancouver, Canada

fraction of the number of HTML tables, but its absolute size, which we estimate to be 154 million, is still very large. Hence, we can consider the Web to be not only the world's largest corpus of documents, but a vast corpus of databases as well.

Our second contribution is a set of techniques for **relation recovery**, which distills the original raw HTML tables into the much smaller set of high-quality relations. Most HTML tables are used for page layout, form layout, or other non-relational data presentation (such as "property sheet" lists that focus on a single data item); these non-relational tables must be filtered out. Further, even the correctly-detected relations lack explicit metadata such as column labels and types. WebTables detects this metadata when it is embedded in the HTML.

There are a number of WebTables applications enabled by these extracted tables, which we describe elsewhere [2]. One application is a search engine that takes keywords as input, but returns relevant extracted databases instead of URLs. Because each result is structured, the search system can automatically provide structured services (*e.g.*, data visualizations) as part of the search results. For example, a search for `us states` can elicit not just relevant tables, but an automatic map visualization.

Another set of WebTables applications are based on the schema statistics that we can derive from the extracted tables. For example, attribute label statistics allow us to perform *schema autocomplete*, a service inspired by word processors' word autocompletion features. A novice database designer can input one or two domain-relevant attributes (say, *instructor*), and the system returns a set of other useful attributes for the database (*e.g.*, *course-title*, *time*, and *credits*). We cover the extracted schema statistics in more depth in Section 3.3.

It is important to distinguish the WebTables data from the deep web. WebTables considers only HTML tables that are already surfaced and crawlable. The deep web refers to content that is made available through filling HTML forms. The two sets of data overlap, but neither contains the other. There are many HTML tables that are not behind forms (only about 40% of the URLs in our corpus are parameterized), and while some deep-web data is crawlable, the vast majority of it is not (or at least requires special techniques, such as those described in [7]). In contrast to the work we describe in this paper, deep web research questions focus on identifying high quality forms and automatically figuring out how to query them in a semantically meaningful fashion.

In addition to HTML tables and the deep web, there are many kinds of structure on the Web, including lists, tagged items, ontologies, XML documents, spreadsheets, and even extracted language parses [9]. In this paper we will only consider the `table` tag.

In the next section we characterize the web's raw HTML Table corpus, a novel dataset that has not been studied previously. Section 3 shows how we recover high-quality relations from the huge number of low-quality tables. We evaluate relation recovery in Section 4, and conclude with discussions of related and future work (Sections 5 and 6).

2. THE HTML TABLE CORPUS

Our goal is to extract from the raw Web a corpus of high-quality relations. An HTML `table` that is also a good relation should contain data arranged in tuple-centric rows, and have a coherent set of domain-appropriate attributes in each column. These relations make up a small percentage of the overall tables in our crawl. This section describes the `table` crawl, and the first steps that WebTables takes to find the good relations.

We applied an HTML parser to a multi-billion-page crawl of the English-speaking web to obtain about 14.1B instances of the `table` tag. Presenting relational-style data is perhaps the most "obvious" use of the tag, but non-relational uses are far more common.

Table type	**% total**	**count**
Extremely small	88.06	12.34B
HTML forms	1.34	187.37M
Calendar	0.04	5.50M
Obviously non-relational, total	89.44	12.53B
Other non-relational (est.)	9.46	1.33B
Relational (est.)	**1.10**	154.15M

Table 1: Various types of HTML tables found in the crawl. Types are non-overlapping; we only examine tables that were not eliminated by tests higher on the list. The rate of *other non-relational* tables is estimated from a human-judged sample.

A few use cases make up the bulk of all HTML tables and are easy to detect:

- *Extremely small* tables are those with fewer than two rows or two columns. We assume that these tables carry no interesting relational information.
- Many tables are embedded inside *HTML forms* and are generally used for visual layout of user input fields.
- Some tables are used to draw a *calendar* onscreen, and consist of nothing but a column for each day of the week and a number in each cell.

We wrote parsers that reliably detect each of these use cases. As seen in Table 1, these three table types make up more than 89% of the HTML tables in our raw corpus[2]. Any web-embedded relations must be found in the remaining portion.

However, even this remainder consists primarily of non-relational tables. These non-relations include tables used for page layout, tables with an enormous number of blank cells, tables that are really simple lists presented in two dimensions, and "property sheets" that consist of attribute value pairs for a single data entity (e.g., MySpace personal preference lists).

[2] Remarkably, 0.88% of all HTML tables in our raw crawl (more than 122M) are "no-ops," containing *zero rows and zero columns.*

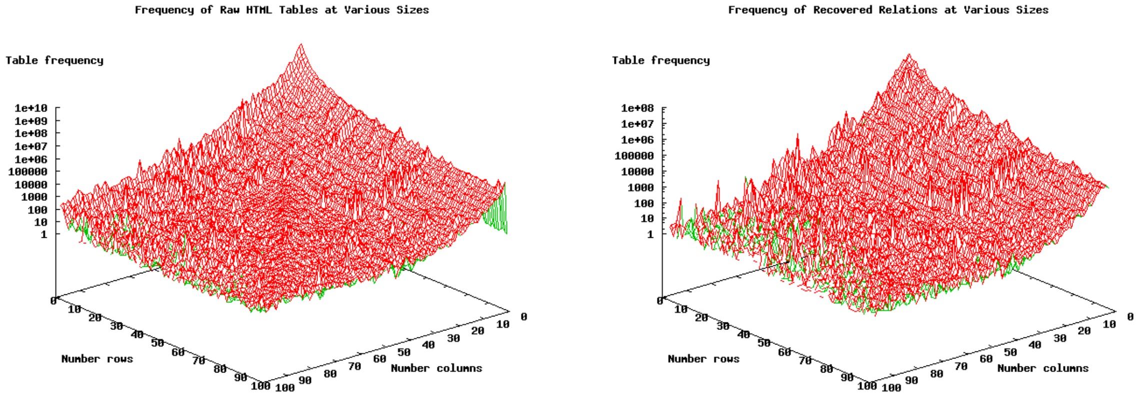


Figure 2: Frequency of HTML tables/relations of various row and column sizes. The figure on the left shows tables from the raw crawl; the figure on the right shows only the high-quality recovered relations. The most frequently-seen table in the raw crawl (seen 3.8B times, accounting for 27.32% of all tables) contains a single cell, and is represented in the left-hand plot by the point at `rows=1, cols=1`. The right-hand plot shows no tables with fewer than 2 rows or 2 columns, and many fewer tables in general.

Unlike the HTML forms and calendars listed above, it is difficult to automatically detect these non-relational types. Two tables may have identical HTML structure, but only one may contain good relational data. Nor can traditional tests for relational well-formedness (*e.g.*, testing whether a table obeys schema constraints, or testing a schema to see whether it is in a certain normal form) be applied here. Not only is there no explicit metadata associated with the tables, many traditional relational constraints (*e.g.*, foreign key constraints) make no sense in the web-publishing scenario.

Answering whether a table contains relational data usually means understanding the table's data, and thus is unavoidably one of human judgment. So we asked two human judges to examine a sample of the remaining tables and mark each as *relational* or *non-relational*. As we describe in Section 3.1, we trained a classifier based on this training set and applied it to the entire corpus.

The results (in Table 1) show that 1.1% of the original raw crawl are classified as relational. Thus, our 14B-table crawl contains roughly 154M high-quality relations. While the percentage is relatively small, the vast number of tables in our crawl means that the resulting set of relations is still enormous.

Figure 2 shows the relative number and sizes of tables in the crawl. The left-hand image shows the frequency of the raw HTML tables, and the right-hand image shows the frequency of the relations filtered from the raw corpus by the trained classifier.

Cols	Raw %	Recovered %
0	1.06	0
1	42.50	0
2-9	55.00	93.18
10-19	1.24	6.17
20-29	0.19	0.46
30+	0.02	0.05

Rows	Raw %	Recovered %
0	0.88	0
1	62.90	0
2-9	33.06	64.07
10-19	1.98	15.83
20-29	0.57	7.61
30+	0.61	12.49

Table 2: Frequency of tables/relations at selected sizes, as a percentage of each dataset. Non-relational tables are filtered out using a combination of handwritten parsers and a trained classifier.

Table 2 compares a few selected pieces of data from Figure 2. Note that tables with between 2 and 9 columns make up 55% of the raw corpus, but more than 93% of the recovered relations; as intuition would suggest, there are relatively few high-quality relations with a very large number of attributes.

In contrast, there is a much greater diversity of row counts among the recovered relations.

3. RELATION RECOVERY

Recovering relations from the raw HTML tables consists of two steps. First, as described in the previous section, WEBTABLES attempts to filter out all non-relational tables. Second, WEBTABLES attempts to recover metadata (in the form of attribute labels) for the now-filtered relations. Recovering the metadata involves a novel technique that uses an imperfectly-extracted set of schema statistics to improve accuracy and recall for metadata recovery.

3.1 Relation Filtering

# rows
# cols
% rows w/mostly NULLS
# cols w/non-string data
cell strlen avg. μ
cell strlen stddev. σ
cell strlen $\frac{\mu}{\sigma}$

(a) Relational Filtering

# rows
# cols
% cols w/lower-case in row_1
% cols w/punctuation in row_1
% cols w/non-string data in row_1
% cols w/non-string data in *body*
% cols w/$|len(row\_1) - \mu| > 2\sigma$
% cols w/$\sigma \leq |len(row\_1) - \mu| \leq 2\sigma$
% cols w/$\sigma > |len(row\_1) - \mu|$

(b) Header Detection

Figure 3: Selected features used in relational ranking and header detection. Relational Filtering requires statistics that help it distinguish relational tables, which tend to contain either non-string data, or string data with lengths that do not differ greatly. Header detection relies on both syntactic cues in the first row and differences in data type and string length between the first row and the remainder of each column.

After applying the hand-written filters described above in Section 2, we can treat relational filtering as a machine learning classification problem, similar to the work of Wang and Hu [12]. We asked two human judges to classify a large number of HTML tables as *relational* or not. We paired these classifications with a set of automatically-extracted features for each table (listed in Figure 3(a)) to form a supervised training set for a statistical learner. Like Wang and Hu, our features consider both table layout (*e.g.*, # rows, # columns, average cell length) and content consistency (*e.g.*, # columns containing non-string basic datatypes, such as integer and date).

Because the true set of relations is so large, we will need some kind of downstream search system even after the WebTables relational filter has done its work. But only tables that our classifier declares to be *relational* will make it to downstream systems; good relations that are incorrectly labeled as *non-relational* will be lost. So, we tuned the relational classifier to give very high recall at the cost of lower precision, trusting that any downstream processor will need to perform some kind of search in any case.

Our experimental results in Section 4 show that we can recover the estimated set of 154M relations with acceptable precision, and with recall that is comparable to other domain-independent web-data extraction systems. The resulting corpus is roughly five orders of magnitude larger than the second-largest that we found (emitted by Wang and Hu, consisting of 1,740 relational tables, generated from a semi-curated input set).

3.2 Metadata Recovery

Even the correctly-filtered relations still lack formal metadata. However, attribute names and types in a good relation will often be obvious to a human observer (often because labels are directly embedded in the HTML). WebTables is only concerned with metadata as far as it consists of these per-attribute labels. Attribute labels are a fraction of the metadata that is standard in a traditional relational database, but much of that either applies only to multiple-relation databases (*e.g.*, foreign-key constraints) or is too restrictive for the somewhat-dirty data we expect to recover (*e.g.*, uniqueness constraints)[3]. Recovering the metadata is still quite difficult, even when limited to these labels.

High-quality labels for each column can have a number of good downstream effects for applications that process WebTables data. In the context of a relation-search tool, good labels allow relations to appear correctly-rendered on-screen, labels may improve rank quality, and labels could be helpful when applying structured data services (such as XML export or data visualization). Further, good labels allow for the very existence of the *Attribute Coocurrence Statistics Database*, or **ACSDb**, a collection of statistics about schema attributes that we can employ in a number of interesting ways (as described in Section 3.3).

There are two cases to consider for meta-data recovery. In the first case, there is already a “header” row in the table with column labels. However, this header is difficult to distinguish, as relations often contain alphabetic string data. A relational column contains strings either because its intended type really is `string`, or because the column was intended to have a different type (say, `numeric`), but includes a few stray strings as a side-effect of HTML extraction (*e.g.*, a column might contain a misplaced copyright notice). Based on a hand-marked sample, we believe that 71% of the true relations have a header row.

To obtain the metadata contained in header rows, we developed the **Detect** classifier that declares whether a relational header is *present* or not. **Detect** uses the features listed in Table 3(b), trained on more than a thousand hand-marked samples by two separate judges. The two most heavily-weighted features for header-detection are the number of columns and the percentage of columns with non-string data in the first row.

As we will see in the experimental results in Section 4, **Detect** is quite successful at recovering header-embedded metadata, especially when combined with schema statistics information that we describe in Section 3.3.

[3] It is interesting to consider whether we can compute a synthetic title for an extracted relation, but for now we settle for the title of the HTML page where the relation was found.

The second case covers the remaining 29% of true relations, where the data is good but there is no header row. In these cases, we can only hope to synthesize column labels that make sense. We tested an algorithm called **Reference-Match**, which attempted to create synthetic column labels by matching the contents of an unlabeled column to a separate dataset where we already know a correct label. For example, an anonymous column that contains Casablanca, Vertigo, and other movies may match find a large number of entries to a preexisting movie database, allowing us to apply the **movie** label. Unfortunately, we found extremely few tables with clean enough string data to match our controlled database of 6.8M tuples in 849 separate domains. For now, synthetic schema generation is still an area for future work.

3.3 Schema Statistics

The sheer size of our corpus also enables us to compute the first large-scale statistical analysis of how attribute names are used in schemas. We can leverage these statistics in various ways, including refining the **Detect** algorithm.

We created an *Attribute Cooccurrence Statistics Database*, or **ACSDb**, using metadata recovered with the **Detect** algorithm. The **ACSDb** simply contains counts of how many times each attribute occurs with other attributes. It allows us to compute the probability of seeing various attributes in a schema. For example, $p(address)$ is simply the number of times that "address" appears in the schema corpus, divided by the total number of schemas. Each attribute is a raw string - we do not perform any synonym resolution or similar preprocessing. We can detect relationships between attribute names by conditioning an attribute's probability on the presence of a second attribute. For example, we can compute $p(address|name)$ by simply counting the number of times "address" appears in the same schema as "name" (and normalizing appropriately). We created an **ACSDb** with 5.4M unique attribute names and 2.6M unique schemas.

The **ACSDb** has many applications, as mentioned briefly above and described in Section 6. One simple application is to compute a *schema coherency score*, which we can use to improve metadata recovery.

We compute the schema coherency score $S(R)$ for relation R by averaging all the Pointwise Mutual Information (or PMI) scores for every pair of distinct attributes A and B in a relation R:

$$S(R) = \frac{\Sigma_{A,B \in R, A \neq B} log(\frac{p(A,B)}{p(A)p(B)})}{|R|(|R|-1)}$$

PMI is a measure often used in computational linguistics and web text research, and is designed to give a sense of how strongly two items are related [5, 4, 11]. PMI will be large and positive when two variables strongly indicate each other, zero when two variables are completely independent, and negative when variables are negatively-correlated. Put another way, when the joint probability of two attributes is "surprisingly large" when compared to the product of their marginal probabilities, the PMI will be high.

For example, our corpus of relations shows that schema attributes "make" and "model" have a high positive PMI, while "make" and "zipcode" have a slightly negative one. By taking an average across all such PMI scores in R, we hope to reward schemas that have highly-correlated attributes, while not overly-penalizing relations with just one "bad" attribute.

true class	Precision	Recall
relational	0.41	0.81
non-relational	0.98	0.87

Table 3: Test results for filtering true relations from the raw HTML table corpus.

Detector	header?	Precision	Recall
Detect	has-header	0.79	0.84
	no-header	0.65	0.57
Detect-ACSDb	has-header	0.89	0.85
	no-header	0.75	0.80

Table 4: Test results for detecting a header row in true relations. We correctly detect most of the true headers, but also mistakenly detect headers in a large number of non-header relations. Incorporating ACSDb information improves both precision and recall.

4. EXPERIMENTAL RESULTS

We now present experimental results for the two main tasks associated with relation recovery: filtering and metadata detection.

4.1 Filtering

We asked human judges to classify several thousand HTML tables as *relational* or not. We then trained a rule-based classifier using the extracted features listed in Table 3(a), using the WEKA package [13]. We cross-validated the trained classifier by splitting the human-judged data into five parts, using four parts for training and the fifth for testing. We trained five different classifiers, rotating the testing set each time, and averaged performance numbers from the resulting five tests.

Table 3 shows the results. As mentioned previously, we tuned the training procedure to favor recall over precision, trusting that most downstream applications will need to perform the relevance-ranking in any case. The rule-based classifier retains 81% of the truly relational tables, though only 41% of the output is relational. These results mean we retain about 125M of the 154M relations we believe exist in the raw crawl, at a cost of sending 271M tables to the WebTables search indexer. Our filter thus raises the "relational concentration" from 1.1% in the raw HTML table corpus up to 41%. Except for the true-relation precision number (which we tuned to be relatively low), these results are consistent with other domain-independent extraction systems, such as KnowItAll and Snowball [5, 1].

4.2 Metadata Recovery

Recovering metadata entails first detecting when there is a header for the relation, and then generating a synthetic header when the raw HTML does not give one.

We created two different header-detectors. The first, **Detect**, uses the features from Figure 3(b), as described in Section 3.2. The second, **Detect-ACSDb**, also incorporates data from the **ACSDb**. We took the filtered output from the previous stage, and marked a large sample of 1000 relations as either *has-header*, *no-header*, or *non-relational* (in case of a mistake made by the relational filter). We then used all of the *has-header* or *no-header* relations to train and test our rule-based classifier, again using five-fold cross-validation (as above).

Table 4 shows the results. Unlike the relational-filtering case, there is no obvious recall/precision bias we should impose. Both precision and recall are good for metadata recovery, and are consistent with other published extraction results [5, 1].

5. RELATED WORK

We are not aware of any other effort to extract relational tables from the Web at a scale similar to WebTables. We have seen no comparable study of structured data elements on the Web. A number of authors have studied the problem of information extraction from a single table, some of which serve a role similar to that of the WebTables relation filter [3, 6, 10, 14, 15]. As discussed in Section 3.1, Wang and Hu detected "true" tables with a classifier and features that involved both content and layout [12]. This last paper processed the most tables, taking as input about 11,000 HTML tables, and emitting a corpus of 1,740.

The idea of leveraging a corpus of schemas to solve database schema problems was first considered in [8], which considered only collections of 40-60 schemas.

6. FUTURE WORK AND CONCLUSIONS

The extracted WebTables table corpus suggests a large number of interesting applications. In Section 1, we discussed a structured database search engine and a tool for performing schema autocomplete. Another possible use of the schema statistics is automatic attribute synonym computation, for use in schema matching. Users could build novel datasets by joining and unioning WebTables data. Another interesting system might automatically compose multi-table structures with just a single keyword query as user input. For all of these projects, the vastness of the Web allows the user to avoid much of the burden associated with constructing a new relational database. Some of these ideas describe research that we have already started, while others are still in the future.

Another important area of future work is increasing the number of structured datatypes that WebTables processes. We mentioned several unhandled types in Section 1, including the HTML list. HTML lists are often used to display structured data; each bulleted or numbered entry is a row, and columns are communicated with punctuation marks or sometimes simply whitespace. Because column boundaries must be recovered, lists pose an additional challenge beyond tables. However, they are so common that list-derived data would probably be a very useful addition to a WebTables corpus.

Web-embedded structured data is a huge and largely untapped resource. This paper described a large crawl of HTML tables, and offered an extraction system that recovered the largest corpus of databases that we know of. It is the first stage of the WebTables project, which we believe will yield many interesting and novel applications enabled by the relational Web.

7. REFERENCES

[1] E. Agichtein, L. Gravano, V. Sokolovna, and A. Voskoboynik. Snowball: A prototype system for extracting relations from large text collections. In *SIGMOD Conference*, 2001.

[2] M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. Webtables: Exploring the power of tables on the web. In *VLDB*, 2008.

[3] H. Chen, S. Tsai, and J. Tsai. Mining tables from large scale html texts. In *18th International Conference on Computational Linguistics (COLING)*, pages 166–172, 2000.

[4] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th Annual Association for Computational Linguistics*, 1989.

[5] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-scale information extraction in knowitall (preliminary results). In *Thirteenth International World Wide Web Conference*, 2004.

[6] 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 World Wide Web Conference (WWW 2007)*, pages 71–80, 2007.

[7] B. He, Z. Zhang, and K. C.-C. Chang. Knocking the door to the deep web: Integration of web query interfaces. In *SIGMOD Conference*, pages 913–914, 2004.

[8] J. Madhavan, P. A. Bernstein, A. Doan, and A. Y. Halevy. Corpus-based schema matching. In *ICDE*, 2005.

[9] J. Madhavan, A. Y. Halevy, S. Cohen, X. L. Dong, S. R. Jeffery, D. Ko, and C. Yu. Structured data meets the web: A few observations. *IEEE Data Eng. Bull.*, 29(4):19–26, 2006.

[10] G. Penn, J. Hu, H. Luo, and R. McDonald. Flexible web document analysis for delivery to narrow-bandwidth devices. In *International Conference on Document Analysis and Recognition (ICDAR01)*, pages 1074–1078, 2001.

[11] P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the Twelfth European Conference on Machine Learning*, 2001.

[12] Y. Wang and J. Hu. A machine learning based approach for table detection on the web. In *Eleventh International World Wide Web Conference*, 2002.

[13] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufman, San Francisco, 2nd edition edition, 2005.

[14] Y. Yang and W. Luk. A framework for web table mining. In *Proceedings of the 4th International Workshop on Web Information and Data Management*, pages 36–42, 2002.

[15] R. Zanibbi, D. Blostein, and J. Cordy. A survey of table recognition: Models, observations, transformations, and inferences, 2003.