WebTables & Octopus

Michael J. Cafarella
University of Washington

CSE454
April 30, 2009

This page contains 16 distinct HTML tables, but only one relational database.

Each relational database has its own schema, usually with labeled columns.

WebTables system automatically extracts dbs from web crawl:
[WebDB08, “Uncovering…”, Cafarella et al]
[Vldb08, “WebTables: Exploring…”, Cafarella et al]

- An extracted relation is one table plus labeled columns
- Estimate that our crawl of 14.1B raw HTML tables contains ~154M good relational dbs
The Table Corpus

<table>
<thead>
<tr>
<th>Table type</th>
<th>%</th>
<th>total</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small tables</td>
<td>88.06</td>
<td>12.34B</td>
<td></td>
</tr>
<tr>
<td>HTML forms</td>
<td>1.34</td>
<td>187.37M</td>
<td></td>
</tr>
<tr>
<td>Calendars</td>
<td>0.04</td>
<td>5.50M</td>
<td></td>
</tr>
<tr>
<td>Obvious non-rel</td>
<td>89.44</td>
<td>12.53B</td>
<td></td>
</tr>
<tr>
<td>Other non-rel (est.)</td>
<td>9.46</td>
<td>1.33B</td>
<td></td>
</tr>
<tr>
<td>Rel (est.)</td>
<td>1.10</td>
<td>154.15M</td>
<td></td>
</tr>
</tbody>
</table>

Relation Recovery

Step 1. Relational Filtering
Recall 81%, Precision 41%
- 271M databases, about 125M are good
- Five orders of magnitude larger than previous largest corpus [WWW02, "A Machine Learning...", Wang & Hu]
- 2.6M unique relational schemas

What can we do with those schemas? [VLDB08, "WebTables: Exploring...", Cafarella et al]

Schema Statistics

Recovered Relations

<table>
<thead>
<tr>
<th>Schema</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>make, model, year</td>
<td>1</td>
</tr>
<tr>
<td>make, model, year, color</td>
<td>1</td>
</tr>
<tr>
<td>name, addr, city, state, zip</td>
<td>1</td>
</tr>
<tr>
<td>name, last-modified</td>
<td>1</td>
</tr>
</tbody>
</table>

Step 2. Metadata Detection
Recall 85%, Precision 89%
- \{make, model, year\} 1
- \{name, size, last-modified\} 1
- \{name, addr, city, state, zip\} 1
- \{make, model, year, color\} 1

App #1: Relation Search

Problem: keyword search for high-quality extracted databases
- Output depends on both quality of extracted tables and the ranking function

Output depends on both quality of extracted tables and the ranking function
**App #1: Relation Search**

- Schema statistics can help improve both:
  - Relation Recovery (Metadata detection)
  - Ranking

- By computing a schema coherency score \( S(R) \) for relation \( R \) and adding it to feature vector

- Measures how well a schema "hangs together"
  - High: \{make, model\}
  - Low: \{make, zipcode\}

- Average pairwise Pointwise Mutual Information score for all attributes in schema

\[
S(R) = \frac{\sum_{A,B \in R, A \neq B} \log \frac{p(A,B)}{p(A)p(B)}}{|R|(|R| - 1)}
\]

**App #1: Experiments**

- Metadata detection, when adding schema stats scoring
  - Precision 0.79 ⇒ 0.89
  - Recall 0.84 ⇒ 0.85

- Ranking: compared 4 rankers on test set
  - Naïve: Top-10 pages from google.com
  - Filter: Top-10 good tables from google.com
  - Rank: Trained ranker
  - Rank-Stats: Trained ranker with coherency score

<table>
<thead>
<tr>
<th>k</th>
<th>Naïve Filter</th>
<th>Rank</th>
<th>Rank-Stats</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.70 (89%)</td>
<td>0.41 (80%)</td>
<td>0.47 (80%)</td>
</tr>
<tr>
<td>20</td>
<td>0.33 (42%)</td>
<td>0.56 (70%)</td>
<td>0.59 (79%)</td>
</tr>
<tr>
<td>30</td>
<td>0.34 (43%)</td>
<td>0.66 (90%)</td>
<td>0.68 (100%)</td>
</tr>
</tbody>
</table>

**App #2: Schema Autocomplete**

- Input: topic attribute (e.g., make)
- Output: relevant schema
  - \{make, model, year, price\}
  - "tab-complete" for your database

For input set \( I \), output \( S \), threshold \( t \)

\[
\begin{align*}
&\text{newAttr} = \max_{\text{newAttr}} \{S-I | I\} \\
&S = S \cup \text{newAttr} \\
&\text{emit newAttr}
\end{align*}
\]

**App #2: Experiments**

- Asked experts for schemas in 10 areas
- What was autocompleter's recall?

**App #3: Synonym Discovery**

- Input: topic attribute (e.g., address)
- Output: relevant synonym pairs
  - phone = tel-

- Used for schema matching
  - [VLDB01, "Generic Schema Matching...", Madhavan et al]

- Linguistic thesauri are incomplete; hand-made thesauri are burdensome

\[
syn(a, b) = \frac{\sum_{z \in C} p(z|a,C) - p(z|b,C)}{1 + \sum_{z \in C} p(z|a,C) - p(z|b,C)}
\]

For attributes \( a, b \) and input domain \( C \), when \( p(a,b) = 0 \)
App #3: Synonym Discovery

<table>
<thead>
<tr>
<th>Name</th>
<th>e-mail</th>
<th>e-mail</th>
<th>phone</th>
<th>telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>e-mail_address</td>
<td>email_address</td>
<td>date</td>
<td>last_modified</td>
</tr>
<tr>
<td>Instructor</td>
<td>course</td>
<td>course-#</td>
<td>course</td>
<td>course-title</td>
</tr>
<tr>
<td>Selected</td>
<td>candidate</td>
<td>name</td>
<td>presiding-officer</td>
<td>speaker</td>
</tr>
<tr>
<td>ab</td>
<td>team, cabin, angrily, sailor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>right</td>
<td>candidate, last-prise, bed</td>
<td>code, price</td>
<td>count</td>
<td></td>
</tr>
</tbody>
</table>

App #3: Experiments

- For each input attr, repeatedly emit best synonym pair (until min threshold reached)

WebTables Contributions

- Largest collection of databases and schemas, by far
- Large-scale extracted schema data for first time; enables novel applications

Outline

- WebTables
- Octopus

Multiple Tables

- Can we combine tables to create new data sources?
  - Data integration for the Structured Web
  - Many existing “mashup” tools, which ignore realities of Web data
    - A lot of useful data is not in XML
    - User cannot know all sources in advance
    - Transient integrations

Integration Challenge

- Try to create a database of all “VLDB program committee members”
Octopus

- Provides "workbench" of data integration operators to build target database
  - Most operators are not correct/incorrect, but high/low quality (like search)
  - Also, prosaic traditional operators

Walkthrough - Operator #1

- SEARCH("VLDB program committee members")

Walkthrough - Operator #2

- Recover relevant data

Walkthrough - Union

- Combine datasets

Walkthrough - Operator #3

- Add column to data
  - Similar to "join" but join target is a topic

- User has integrated data sources with little effort
  - No wrappers; data was never intended for reuse
CONTEXT Algorithms
- Input: table and source page
- Output: data values to add to table
  - SignificantTerms sorts terms in source page by “importance” (tf-idf)

Related View Partners
- Looks for different “views” of same data

CONTEXT Experiments

EXTEND Algorithms
- Recall: EXTEND ("publications", col=0)
- JoinTest looks for single “joinable” table
- E.g., extend a column of US cities with “mayor” data
- Algorithm:
  1. SEARCH for a table that is relevant to topic (e.g., “mayors”); rank by relevance
  2. Retain results with a joinable column (to col=0). Use jaccardian distance fn between columns

EXTEND Algorithms
- MultiJoin finds compatible “joinable tuples”; tuples can come from many different tables
- E.g., extend PC members with “publications”
- Algorithm:
  1. SEARCH for each source tuple (using cell text and “publications”)
  2. Cluster results, rank by weighted mix of topic-relevance and source-table coverage
  3. Choose best cluster, then apply join-equality test as in JoinTest
- Algorithms reflect different data ideas: found in one table or scattered over many?

EXTEND Experiments
- Many test queries not EXTENDable
- We chose column and query to EXTEND

<table>
<thead>
<tr>
<th>Join column description</th>
<th>Topic query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Countries</td>
<td>Universities</td>
</tr>
<tr>
<td>Us states</td>
<td>Governors</td>
</tr>
<tr>
<td>Us cities</td>
<td>Mayors</td>
</tr>
<tr>
<td>Film titles</td>
<td>Characters</td>
</tr>
<tr>
<td>UK political parties</td>
<td>Member of parliament</td>
</tr>
<tr>
<td>Baseball teams</td>
<td>Players</td>
</tr>
<tr>
<td>Musical bands</td>
<td>albums</td>
</tr>
</tbody>
</table>

- TestJoin: 60% of src tuples for three topics; avg 1 correct extension per src tuple
- MultiJoin: 33% of src tuples for all topics; avg 45.5 correct extensions per src tuple
1. For $i = 1..\text{MAX}$, find breakpoints, ranked by co-location score.

2. Choose $i$ that yields most-consistent columns.

Our current consistency measure is avg std-dev of cell strlens.

Octopus Contributions

- Basic operators that enable Web data integration with very small user burden
- Realistic and useful implementations for all three operators
Future Work

- **WebTables**
  - Schema autocomplete & synonyms just few of many possible *semantic services*
  - Input: schema; Output: tuples
    - *database autopopulate*
  - Input: tuples; Output: schema
    - *schema autogenerate*

- **Octopus**
  - Index support for interactive speeds

---

Future Work (2)

- “The Database of Everything”
  [CIDR09, “Extracting and Querying…,” Cafarella]
  - Is domain-independence enough?

<table>
<thead>
<tr>
<th>Text-embedded</th>
<th>Table-embedded</th>
</tr>
</thead>
<tbody>
<tr>
<td>be/is</td>
<td>&lt;web access log&gt;</td>
</tr>
<tr>
<td>ask/call</td>
<td>&lt;file listing&gt;</td>
</tr>
<tr>
<td>arrive/come/go</td>
<td>&lt;forum posts&gt;</td>
</tr>
<tr>
<td>join/lead</td>
<td>&lt;album listing&gt;</td>
</tr>
<tr>
<td>born-in</td>
<td>&lt;phone numbers&gt;</td>
</tr>
</tbody>
</table>

- Multi-model, multi-extractor approach probable
- Vast deduplication challenge
- New tools needed for user/extractor interaction