Efficient Acquisition of Web Data through Restricted Query Interfaces

Simon Byers    Claudio Silva    Juliana Freire
AT&T Labs    Bell Labs

WWW10 Hong Kong, May/2001
The Hidden Web
Web Databases

- 80% of Web data is accessible through limited query interfaces
- Form interfaces provide a simple way to query and filter data, but...
- they can be too restrictive
- Too many queries may be needed to retrieve the desired information, e.g., find me all the stores in NJ, and
- it may take too long
Materializing Web Databases

- Enable data exploration - richer queries
- Improve performance
- Useful in a number of applications:
  - Comparison shopping services, job search sites, etc
- Back-door approach: content provider may create a special, more powerful interface
- What if the content provider does not cooperate?
  - E.g., vendor A wants to track the expansion strategy of vendor B

How to scan the database through the limited interface?

How to do that efficiently?
Some simple useful strategies

Pose the most general queries allowed by the interface to minimize the number of queries:
- leave optional attributes unspecified
- choose most general values for obligatory attributes (e.g., all property types)
- given a choice of obligatory attributes, select the one with smallest domain (e.g., state instead of zipcode)

Not always enough to guarantee full coverage...
Generating covers of hidden databases accessible through NN-query interfaces

- NN-query: find the 10 closest stores to zipcode 07974
- Naïve strategy: pose one query per zipcode
  - over 10,000 queries
  - it takes from hours to days to retrieve all the data

Is it possible to find a smaller set of zipcodes that return the same answers?
**Algorithm**

- Our algorithm is quite simple and is composed of two parts:
  - We use a spatial data structure to keep track of which parts of a region R that have already been covered by previous queries.
  - At any given point in time, we use the coverage information obtained thus far to determine where to perform the next query as to minimize the overlap of queries.
Coverage (snapshot I)
Coverage (snapshot II)
Coverage (snapshot III)
Coverage (snapshot IV)
Datasets

Dataset 1

Dataset 2

Acquisition by querying zip codes takes over 10,000 queries!
Experimental results

Let $\text{QUAD}(\mathcal{D})$ be the number of queries performed by our technique, where in general $\text{OPT}(\mathcal{D}) \leq \text{QUAD}(\mathcal{D})$. We can define an approximation factor, $\rho(\mathcal{D})$, to be the ratio between our algorithm and the optimum, that is,

$$\rho(\mathcal{D}) = \frac{\text{QUAD}(\mathcal{D})}{\text{OPT}(\mathcal{D})}.$$  \hspace{1cm} (1)

Note that by definition $\rho(\mathcal{D}) \geq 1$; one being the best possible query schedule for retrieving all sites with a $k$-NN query interface.

<table>
<thead>
<tr>
<th></th>
<th>$k = 5$</th>
<th>$k = 10$</th>
<th>$k = 20$</th>
<th>$k = 30$</th>
<th>$k = 40$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>QUAD 411</td>
<td>QUAD 191</td>
<td>QUAD 94</td>
<td>QUAD 60</td>
<td>QUAD 42</td>
</tr>
<tr>
<td></td>
<td>$\rho$ 2.7</td>
<td>$\rho$ 2.6</td>
<td>$\rho$ 2.5</td>
<td>$\rho$ 2.4</td>
<td>$\rho$ 2.2</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>QUAD 435</td>
<td>QUAD 207</td>
<td>QUAD 99</td>
<td>QUAD 65</td>
<td>QUAD 54</td>
</tr>
<tr>
<td></td>
<td>$\rho$ 2.6</td>
<td>$\rho$ 2.4</td>
<td>$\rho$ 2.3</td>
<td>$\rho$ 2.3</td>
<td>$\rho$ 2.6</td>
</tr>
</tbody>
</table>

Table 1: Query performance with varying $k$. In the table, we show: (1) $\text{QUAD}$, the number of queries necessary to find all the sites in a given dataset; (2) $\rho$, the approximation factor of the queries. See text for details.
Experimental results (cont.)

Dataset 1

Dataset 2
### Acquiring Web Data

- **Automate Web navigation and data retrieval**
  - WebVCR, W4F, Perl scripts

- **Optimize the execution of Web queries**
  - many queries may be required for a full scan over the hidden database
  - optimize wrapper (access and extraction)
  - exploit parallelism, smart source selection, re-ordering, etc

- **Disguise requests**
  - anonymizing proxies, random time intervals,
Related Work

- **Mediators and restrictive query interfaces** (e.g., Information Manifold, TSIMMIS, Web Integrator)
  - previous works did not consider the *coverage problem*, or nearest-neighbor interfaces

- **Xyleme project from INRIA**: build a world-wide XML warehouse
  - considers only documents that can be reached through traditional crawling - blind to the hidden Web

- **Optimization of Web queries**
  - query scrambling, adaptive techniques

- **Wrapper creation**
  - WebVCR, W4F, NoDoSe, Ariadne
Full technical report available at

http://www.research.att.com/~csilva/papers