Strategies for Optimizing Querying Third Party Resources in Semantic Web Applications

Albert Weichselbraun

Vienna University of Economics and Business Administration
Department of Information Systems and Operations
Augasse 2-6, 1090 Vienna
albert.weichselbraun@wu-wien.ac.at

July 6, 2008
Agenda

Problem & Motivation

The “Search Test Stop Model”

Application to Semantic Web Services
  Cost Functions
  Utility Functions
  Common Utility Mass Function

Evaluation

Conclusions
Motivation

- IDIOM Media Watch on Climate Change
- Semantic Web → use third party services, interoperability
- compare: OpenCalais, Google Web Services, GeoNames.org, ...
# Web service response times

<table>
<thead>
<tr>
<th>Service</th>
<th>Protocol</th>
<th>( \bar{t}_r )</th>
<th>( \tilde{t}_r )</th>
<th>( t^\text{min}_r )</th>
<th>( t^\text{max}_r )</th>
<th>( \sigma^2_{t_r} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>REST</td>
<td>0.8</td>
<td>0.3</td>
<td>0.2</td>
<td>663.5</td>
<td>150.2</td>
</tr>
<tr>
<td>Dbpedia</td>
<td>SPARQL</td>
<td>0.9</td>
<td>0.5</td>
<td>0.1</td>
<td>301.2</td>
<td>42.7</td>
</tr>
<tr>
<td>Del.icio.us</td>
<td>REST</td>
<td>0.6</td>
<td>0.4</td>
<td>0.1</td>
<td>24.3</td>
<td>0.5</td>
</tr>
<tr>
<td>Geo</td>
<td>REST</td>
<td>1.8</td>
<td>0.1</td>
<td>0.0</td>
<td>1160.4</td>
<td>771.4</td>
</tr>
<tr>
<td>Google</td>
<td>Web</td>
<td>0.3</td>
<td>0.2</td>
<td>0.1</td>
<td>10.3</td>
<td>0.2</td>
</tr>
<tr>
<td>Swoogle</td>
<td>Web</td>
<td>35.8</td>
<td>1.6</td>
<td>0.2</td>
<td>101022.2</td>
<td>1762682.4</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>Web</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
<td>60.9</td>
<td>1.3</td>
</tr>
</tbody>
</table>

**Table:** Response times of some popular Web services.
Lessons learned

- Web service response times are highly volatile especially from smaller service providers.
- Average response times are usually okay.
- Querying such services at peak times is extremely costly.
Idea

- introduce the notion of utility and cost
  → apply methodology from economic theory
The “Search Test Stop” Model

Decision maker searches through a population
→ retrieves \((S_a, x_0)\); Cost: \(c_{s_i}\)

Choices
- discard answer and continue searching → \(c_{s_{i+1}}\)
- test answer → \((S_a, x_0, x_1)\); Cost: \(c_{t_i}\)
- accept answer → retrieves \(u\)

Goal
- maximize \(u - (\sum c_{s_i} + \sum c_{t_i})\)
The “Search Test Stop” Model

Figure: The Optimal Policy.
The “Search Test Stop” Framework

![Diagram showing the search test stop framework with steps involving query input, search, test, and stop with associated data sources and probabilities.]

**Figure:** The Search Test Stop approach.
Application - Cost Functions

For instance:

- CPU-time
- request time
- bandwidth
- storage cost

This research:

- $\rightarrow c \propto t_r$
A generic example:

\[ u = \sum_{S_A} \lambda(i) f_{eval}(i) \quad (1) \]

\[ f_{eval}(i) = \begin{cases} 
0 & \text{if } a_i \text{ incorrect;} \\
1 & \text{if } a_i \text{ correct.} 
\end{cases} \quad (2) \]

Condition:

\[ O(c_s) = O(\bar{u}) \]
\[ O(c_s) \ll O(\bar{u}) \quad \text{search costs have no significant impact} \]
\[ O(c_s) \gg O(\bar{u}) \quad \text{no searching will take place} \]
Use Case - Common Utility Mass Function $h(x_0, x_1, u)$

Utility Function:

\[
\begin{align*}
    n_{\text{Entities}} &= |TaggingEntity| \\ 
    n_{\text{Mappings}} &= |hasPattern| \\ 
    n_{\text{ambiguous}} &= |\sigma[\text{isAmbiguous='true'}](TaggingEntry \ast hasPattern)| \\ 
    P_{\text{incorr}} &= 1 - \frac{n_{\text{Entries}}}{n_{\text{Mappings}} + n_{\text{ambiguous}}} 
\end{align*}
\]
## Evaluation - Normal Distribution

<table>
<thead>
<tr>
<th>Search Cost ($c_s$)</th>
<th>$u_{min}$</th>
<th>Quality ($\bar{u}$)</th>
<th>Quantity ($\frac{\Delta u}{\Delta t}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>STS</td>
<td>Conv</td>
</tr>
<tr>
<td>low</td>
<td>2</td>
<td>6.62</td>
<td>5.58</td>
</tr>
<tr>
<td>low</td>
<td>4</td>
<td>6.64</td>
<td>6.13</td>
</tr>
<tr>
<td>low</td>
<td>6</td>
<td>6.69</td>
<td>6.55</td>
</tr>
<tr>
<td>low</td>
<td>8</td>
<td>6.66</td>
<td>6.39</td>
</tr>
<tr>
<td>medium</td>
<td>2</td>
<td>4.99</td>
<td>4.84</td>
</tr>
<tr>
<td>medium</td>
<td>4</td>
<td>5.02</td>
<td>5.15</td>
</tr>
<tr>
<td>medium</td>
<td>6</td>
<td>5.01</td>
<td>5.32</td>
</tr>
<tr>
<td>medium</td>
<td>8</td>
<td>5.00</td>
<td>3.86</td>
</tr>
<tr>
<td>high</td>
<td>2</td>
<td>2.81</td>
<td>3.20</td>
</tr>
<tr>
<td>high</td>
<td>4</td>
<td>2.75</td>
<td>3.25</td>
</tr>
<tr>
<td>high</td>
<td>6</td>
<td>2.84</td>
<td>2.81</td>
</tr>
<tr>
<td>high</td>
<td>8</td>
<td>2.81</td>
<td>-0.91</td>
</tr>
</tbody>
</table>

**Table:** Tagging performance.
Evaluation - Swoogle

Figure: Swoogle; \( \tilde{t}=1.6 \)
Evaluation - Google

Figure: Google; $\tilde{t} = 0.2$
Evaluation - GeoNames.org

Figure: geonames.org; $\tilde{t} = 0.1$
Conclusions

- The Search Test Stop algorithm *dynamically* allocates resource utilization based on *cost* and *utility* → optimizes results in terms of *accuracy* and *response times*
- Does not provide the most accurate results (brute force)
- Does not minimizes resource usage
- Provides the best trade-off of both
Outlook

- Provide publicly available libraries
- Develop more fine grained notions of utility for geo-taggers
- Application to more complex use cases
  - ontology learning
  - user input