

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, ...

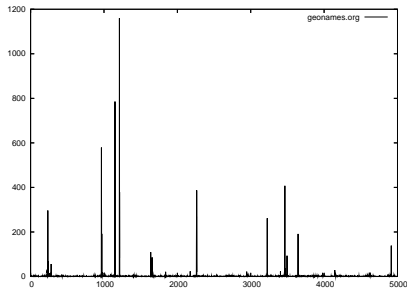
Web service response times

Service	Protocol	\bar{t}_r	\tilde{t}_r	t_r^{min}	t_r^{max}	$\sigma_{t_r}^2$
Amazon	REST	0.8	0.3	0.2	663.5	150.2
Dbpedia	SPARQL	0.9	0.5	0.1	301.2	42.7
Del.icio.us	REST	0.6	0.4	0.1	24.3	0.5
Geo	REST	1.8	0.1	0.0	1160.4	771.4
Google	Web	0.3	0.2	0.1	10.3	0.2
Swoogle	Web	35.8	1.6	0.2	101022.2	1762682.4
Wikipedia	Web	0.4	0.2	0.1	60.9	1.3

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

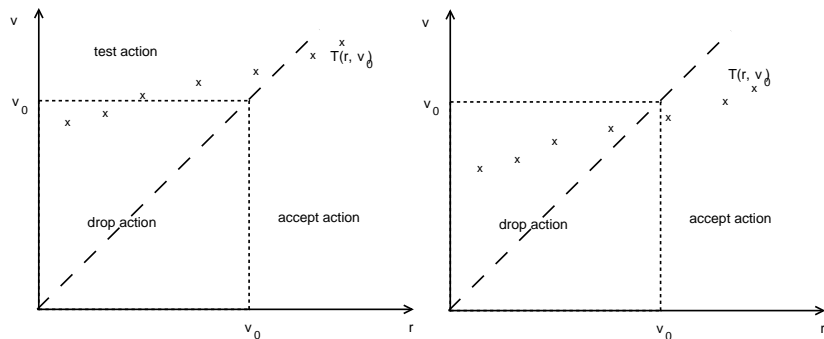


Figure: The Optimal Policy.

The “Search Test Stop” Framework

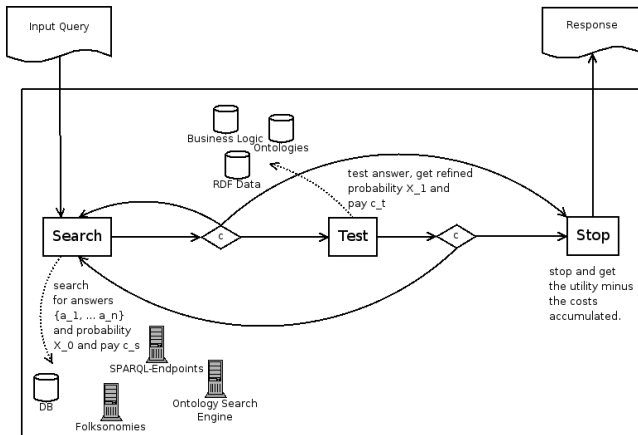


Figure: The Search Test Stop approach.

Application - Cost Functions

For instance:

- ▶ CPU-time
- ▶ **request time**
- ▶ bandwidth
- ▶ storage cost

This research:

- ▶ $\rightarrow c \propto t_r$

Application - Utility Functions

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 $h(x_0, x_1, u)$

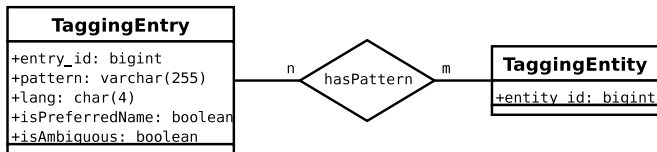


Figure: Geo-Tagger Database Schema

Utility Function:

$$n_{Entities} = |TaggingEntity| \quad (3)$$

$$n_{Mappings} = |hasPattern| \quad (4)$$

$$n_{ambiguous} = |\sigma_{[isAmbiguous='true']}(TaggingEntry * hasPattern)| \quad (5)$$

$$P_{incorr} = 1 - \frac{n_{Entities}}{n_{Mappings} + n_{ambiguous}} \quad (6)$$

Evaluation - Normal Distribution

Search Cost (c_s)	u_{min}	Quality (\bar{u})		Quantity ($\frac{\Delta u}{\Delta t}$)	
		STS	Conv	STS	Conv
low	2	6.62	5.58	3.47	7.79
low	4	6.64	6.13	3.56	6.93
low	6	6.69	6.55	3.57	5.95
low	8	6.66	6.39	3.55	2.75
medium	2	4.99	4.84	1.88	3.22
medium	4	5.02	5.15	1.92	2.76
medium	6	5.01	5.32	1.89	2.27
medium	8	5.00	3.86	1.87	0.79
high	2	2.81	3.20	0.78	1.05
high	4	2.75	3.25	0.76	0.88
high	6	2.84	2.81	0.80	0.59
high	8	2.81	-0.91	0.76	-0.09

Table: Tagging performance.

Evaluation - Swoogle

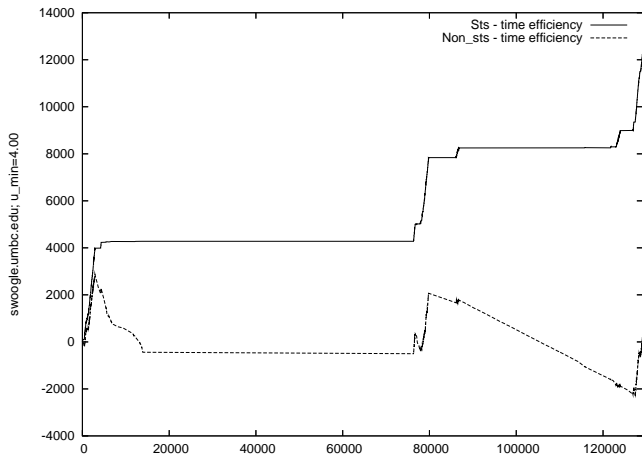


Figure: Swoogle; $\tilde{t}=1.6$

Evaluation - Google

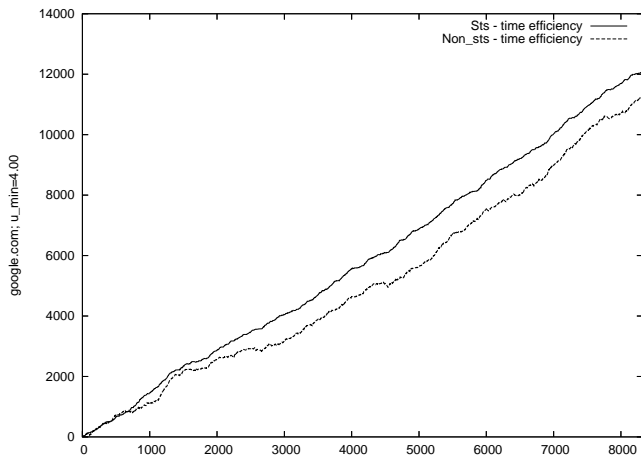


Figure: Google; $\tilde{t}=0.2$

Evaluation - GeoNames.org

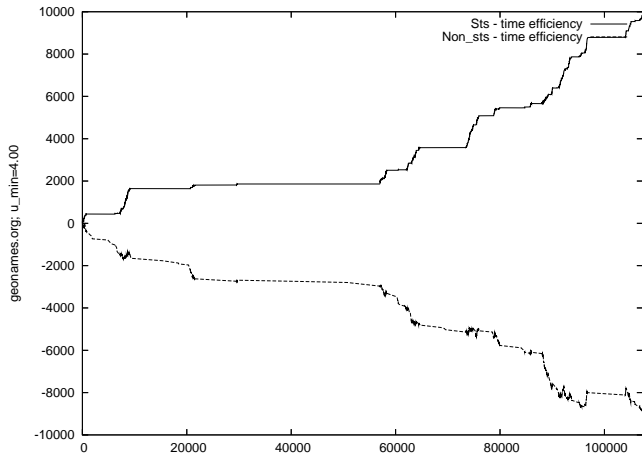


Figure: geonames.org; $\tilde{t}=0.1$

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 minimize 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

