Applying Optimal Stopping for Optimizing Queries to External Semantic Web Resources

Albert Weichselbraun

Vienna University of Economics and Business Administration
Augasse 2-6, Vienna, Austria
albert.weichselbraun@wu-wien.ac.at,
WWW home page: http://www.ai.wu-wien.ac.at/~aweichse

Abstract. The rapid increase in the amount of available information from various online sources poses new challenges for programs that endeavor to process these sources automatically and identify the most relevant material for a given application.
This paper introduces an approach for optimizing queries to Semantic Web resources based on ideas originally proposed by MacQueen for optimal stopping in business economics. Modeling applications as decision makers looking for optimal action/answer sets, facing search costs for acquiring information, test costs for checking these information, and receiving a reward depending on the usefulness of the proposed solution, yields strategies for optimizing queries to external services. An extensive evaluation compares these strategies to a conventional coverage based approach, based on real world response times taken from popular Web services.

1 Introduction

Semantic Web applications provide, integrate and process data from multiple data sources including third party providers. Combining information from different locations and services is one of the key benefits of semantic applications.

Current approaches usually limit their queries to a number of particularly useful and popular services as for instance Swoogle, GeoNames, or DBpedia. Research on automated web service discovery and matching [1] focuses on enhanced applications, capable of identifying and interfacing relevant resources in real time. Future implementations, therefore, could theoretically issue queries spawning vast collections of different data sources, providing even more enhanced and accurate information. Obviously, such query strategies - if applied by a large enough number of clients - impose a considerable load on the affected services, even if only small pieces of information are requested. The World Wide Web Consortium's (W3C) struggle against excessive document type definition (DTD) traffic provides a recent example of the potential impact a large number of clients achieves. Ted Guild pointed out (p.semanticlab.net/w3dtd) that the W3C receives up to 130 million requests per day from broken clients, fetching popular DTD's over and over again, leading to a sustained bandwidth consumption of approximately 350 Mbps.

Service provider like Google restrict the number of queries processed on a per IP/user base to prevent excessive use of their Web services. From a client's perspective overloaded Web services lead to higher response times and therefore higher cost in terms of processing times and service outages.

Grass and Zilberstein [2] suggest applying value driven information gathering (VDIG) for considering the cost of information in query planning. VDIG focuses on the query selection problem in terms of the trade off between response time and the value of the retrieved information. In contrast approaches addressing only the coverage problem put their emphasis solely on maximizing precision and recall.

Optimizing value under scare resources is a classical problem from economics and highly related to decision theory. Applying these concepts to the information systems research domain yields important strategies for optimizing the acquisition of Web resources [3], addressing the trade-off between using resources sparingly and providing accurate and up-to-date information. In this research we apply the search test stop (STS) model to applications leveraging third party resources. The STS model considers the user's preferences between accuracy and processing time, maximizing the total utility in regard to these two measures. In contrast to the approach described by Grass and Zilberstein [2] the STS model adds support for a testing step, designed to obtain more information about the accuracy of the obtained results, aiding the decision algorithm in its decision whether to acquire additional information or act based on the current answer set. Similar to Ipeirotis et al. [4] the resulting query strategy might lead to less accurate results than a "brute force" approach, but nevertheless optimizes the balance between accuracy and costs. Therefore, the search test stop model addresses the trade-off between two of the major software engineering challenges outlined in ISO/IEC 9126-1: (i) *reliability* - the capability of the software product to maintain a level of accuracy according to measures specified in the software design process [5], and (ii) *efficiency* - requiring the software to provide an appropriate performance in terms of processing time and resource allocation, under stated conditions [6].

This paper's results are within the field of AI research facilitating techniques from decision theory to address problems of agent decision making [7].

The article is organized as follows. Section 2 presents known query limits and response times of some popular Web services. Section 3 provides the theoretical background for the search test stop model, and presents its extension to discrete probability functions. Afterwards the application of this method to applications utilizing external resources is outlined in Section 4 and an evaluation of this technique is presented in Section 5. This paper closes with an outlook and conclusions drawn in Section 6.

2 Performance and Scalability

The increased popularity of applications that rely on external data repositories calls for strategies for a responsible and efficient use of these resources.

Extensive queries to external resources increases their share of the program's execution time and may lead to longer response times, requiring its operators to impose limits on the service's use.

Even commercial providers like Google or Amazon restrict the number of accesses to their services. For instance, Google's Web API only allows 1000 requests a day, with exceptions for research projects. Workarounds like the use of Google's public Web interface may lead to blacklisting of the client's IP address[1]. Google's geo coding service imposes a limit of 15,000 queries per day and IP address. Amazon limits clients to 20 queries per second, but restrictions vary between the offered services and might change over time[2]. Other popular resources like GeoNames and Swoogle to our knowledge currently do not impose such limits.

A Web service timing application issuing five different queries to popular Web resources in 30 min intervals over a time period of five weeks yielded Table 2. The services' average response time ($\bar{t}_r$), the response time's median ($\tilde{t}_r$), its minimum and maximum values (t_r^{min}, t_r^{max}), and variance ($\sigma_{t_r}^2$) characterize its potential impact on an application's performance. Due to the timeout value of 60 seconds, specified in the timing application, all t_r^{max} values are equal or below 60. These response times vary, depending on the client's Internet connectivity and location, but adequate values can be easily obtained by probing the service's response times from the client's location.

Table 2 suggests that Google provides a fast and quite reliable service ($\sigma_{t_r}^2 = 0.2$) with only small variations in the response times. This result is not very surprising considering the global and highly reliable infrastructure Google employs.

Service	Protocol	$\bar{t}_r$	$\tilde{t}_r$	t_r^{min}	t_r^{max}	$\sigma_{t_r}^2$
Amazon	REST	0.5	0.2	0.2	31.3	0.6
DBpedia	SPARQL	0.8	0.5	0.1	60.0	4.2
Del.icio.us	REST	0.6	0.4	0.1	24.3	0.5
GeoNames	REST	0.7	0.1	0.0	60.0	19.9
Google	Web	0.3	0.2	0.1	10.3	0.2
Swoogle	Web	4.1	1.6	0.2	60.0	98.4
Wikipedia	Web	0.5	0.2	0.1	60.0	1.3

Table 1. Response times of some popular Web services.

Smaller information providers which cannot afford this kind of infrastructure in general provide good response times (due to fewer requests), but they are more sensitive to sudden peaks in the number of clients accessing their services as visualized in Figure 1. Table 2 reflects these spikes in terms of higher variances and t_r^{max} values.

Our experiments suggest (see Section 5) that especially clients querying services with high variances benefit from implementing the search test stop model.

Another strategy from the client's perspective is avoiding external resources at all. Many community projects like Wikipedia or GeoNames provide database dumps which might be used to install a local copy of the service. These dumps are usually rather large (a current Wikipedia dump including all pages, discussions, but without the edit history comprises approximately 7.8 GB[3]) and often outdated (Wikipedia dumps are sometimes

[1] see p.semanticlab.net/gooso

[2] developer.amazonwebservices.com

[3] download.wikipedia.org; 2008-10-15

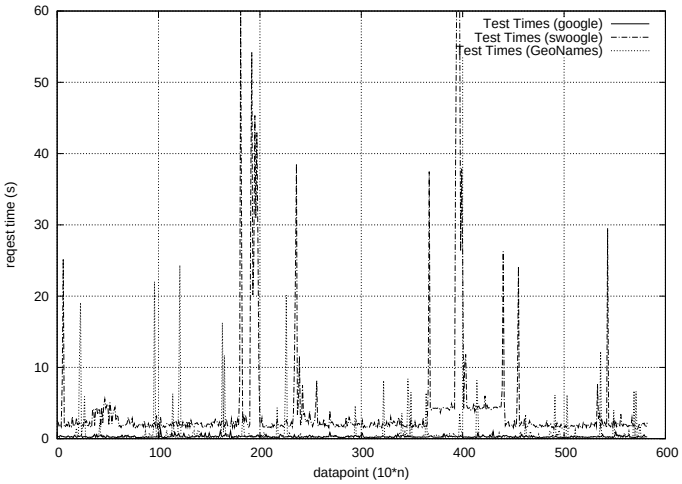

Fig. 1. Selected test times over the time, computed with a timeout of 60 seconds. Every data point accumulates five measurements.

even more than one month old, other services like GeoNames update their records very frequently).

The import of this data requires customized tools (like mwdumper[4]) or hacks and rarely processes without major hassles. In some cases the provided files do not contain all available data (GeoNames for instance does not publish the `relatedTo` information) so that querying the service cannot be avoided at all.

3 The Search Test Stop Model

This section outlines the basic principles of the search test stop (STS) model as found in decision theory. For a detailed description of the model please refer to MacQueen [8] and Hartmann [9].

MacQueen [8] describes the idea of the STS model as follows: A decision maker (a person or an agent) searches through a population of possible actions, sequentially discovering sets of actions (S_A), paying a certain cost each time a new set of actions is revealed (the search cost c_{s_i}). On the first encounter with a set of possible actions, the person obtains some preliminary information (x_0) about its utility (u), based on which he can

1. continue looking for another set of possible actions (paying search cost $c_{s_{i+1}}$),
2. test the retrieved set of actions, to obtain (x_1) - a better estimation of the actions value - paying the test cost (c_{t_i}) and based on this extended information continue with option 1 or finish the process with option 3, or
3. accept the current set of answers (and gain the utility u).

[4] www.mediawiki.org/wiki/MWDumper

The challenge is combining these three options so that the total outcome is optimized by keeping the search (c_{s_i}) and test (c_{t_i}) costs low ($\sum_{i=1}^{m} c_{s_i} + \sum_{i=1}^{n} c_{t_i}$) without jeopardizing the obtained utility u.

Introducing the transformation $r = E(u|x_0)$ yields the following description for a policy *without testing*:

$$v = vF(v) + \int_{v}^{+\infty} rf(r)dr - c_s \tag{1}$$

with the solution $v = v_0$. $F(r)$ represent the cumulative distribution function of the expected utility and $f(r)$ its probability mass function. The constant c_s refers to search cost and v (better v_0) to the utility obtained by the solution of this equation.

Extending Equation 1 to testing yields Equation 2:

$$\begin{aligned} v = {} & vF(r_D) + \\ & \int_{r_D}^{r_A} T(v,r)f(r)dr + \\ & \int_{r_A}^{+\infty} rf(r)dr - c_s \quad \text{and} \end{aligned} \tag{2}$$

$$T(v, r_D) = v \tag{3}$$

$$T(v, r_A) = r_A \tag{4}$$

$T(v,r)$ refers to the utility gained by testing, r_D to the value below which the discovered action set (S_A) will be dropped, and r_A to the minimal utility required for accepting S_A. A rational decision maker will only resort to testing, if the utility gained outweighs its costs and therefore the condition $T(v_0, v_0) > v_0$ holds which is the case in the interval $[r_D, r_A]$.

In the next two sections we will (i) describe the preconditions for applying this model to a real world use case, and (ii) present a solution for discrete data.

3.1 Preconditions

MacQueen [8] defines a number of preconditions required for the application of the STS model. Hartmann [9] eases some of these restrictions yielding the following set of requirements for the application of the model:

1. a common probability mass function $h(x_0, x_1, u)$ exists.
2. The expected value of u given a known realization x_0 ($z = E(U|x_0, y_0)$) exists and is finite.
3. $F(z|x_0)$ is stochastically increasing in x_0. For the concept of stochastically increasing variables please refer to [10, p75].

3.2 The Discrete Search Test Stop Model

This research deals with discrete service response time distributions and therefore applies the discrete STS methodology. Hartmann transferred MacQueen's approach to

discrete models. The following section summarizes the most important points of his work [9].

Hartmann starts with a triple (x_0, x_1, u) of *discrete* probability variables, described by a common probability function $h(x_0, x_1, u)$. From h Hartmann derives

1. the conditional probability function $f(u|x_0, x_1)$ and the expected value $Z = E(u|x_0, x_1)$,
2. the probability function of r, $f(r|x_0)$ and $F(r|x_0)$,
3. the probability of x_0, $f(x_0)$ and $F(x_0)$.

Provided that the conditions described in Section 3.1 are fulfilled only five possible optimal policies are possible - (i) always test, (ii) never test, (iii) test if $u > u_t$, (iv) if $u < u_t$, or (v) if $u_t < u < u'_t$.

The expected utility equals to

1. $E(u|x_0)$ for accepting without testing,
2. $T(r, v)$ with testing, and
3. v_0 if the action is dropped and a new set (S_A) is selected according to the optimal policy.

4 Method

This section focuses on the application of the STS model to Web services. At first we describe heuristics for estimating cost functions (c_s, c_t), and the common probability mass function $h(x_0, x_1, u)$ Afterwards the process of applying search test stop to tagging applications is elaborated.

4.1 Cost functions

In the conventional STS model costs refer to the investment in terms of time and money for gathering information. By applying this idea to software, costs comprise all expenses in terms of CPU-time, bandwidth and storage cost necessary to search for or test certain answers.

Large scale Semantic Web projects, like the IDIOM media watch on climate change [11], process hundred thousands of pages a week. Querying GeoNames for geo-tagging such numbers of documents would add *days* of processing time to the IDIOM architecture.

This research focuses solely on costs in terms of response time, because they are the limiting factor in our current research projects. Other applications might require extending this approach to consider additional cost factors like CPU-time, bandwidth, etc.

4.2 Utility Distributions

Applying the STS model to economic problems yields cash deposits and payments. Transferring this idea to information science is a little bit more subtle, because the utility is highly dependent on the application and its user's preferences. Even within

one domain the notion of an answer set's (S_A) value might not be clear. For instance in a geo context the "correct" answer for a certain problem may be a particular mountain in Austria, but the geo-tagger might not identify the mountain but the surrounding region or at least the state in which it is located (compare Figure 2).

```
1. Austria/Carinthia/Spittal/Heiligenblut/Grossglockner (mountain)
2. Austria/Carinthia/Spittal/Heiligenblut               (village)
3. Austria/Carinthia/Spittal                            (district)
4. Austria/National Park Hohe Tauern                    (national park)
5. Austria/Carinthia                                    (state)
6. Austria/Salzburg (Neighbor)                          (state)
7. Austria/Tyrol    (Neighbor)                          (state)
8. Austria                                              (country)
```

Fig. 2. Ranking of "correct" results for geo-tagging an article covering the "Grossglockner".

Assigning concrete utility values to these alternatives is not possible without detailed information regarding the application and user preferences. Approaches for evaluating the set's value might therefore vary from binary methods (full score for correct answers; no points for incomplete/incorrect answers) to complex ontology based approaches, evaluating the grade of correctness and severe of deviations.

4.3 Application

This work has been motivated by performance issues in a geo-tagging application facilitating resources from GeoNames and WordNet for improving tagging accuracy. Based on the experience garnered during the evaluation of STS models, this section will present a heuristic for determining the cost functions (c_s, c_t) and the common probability mass function $h(x_0, x_1, u)$.

Figure 3 visualizes the application of the search test stop model to Web services. Searching yields an answer set $S_a = \{a_1, \ldots, a_n\}$ and the indicator x_0 at a prices of c_s. Based on x_0 the search test stop description logic decides on whether to (i) accept the current answer set, (ii) drop the answer set and continue searching, or (iii) query another set of resources to retrieve the refined indicator x_1 paying the test cost c_t. Based on x_1 the answer set is dropped or finally accepted.

Cost functions Searching leads to external queries and therefore costs. Measuring a service's performance over a certain time period allows estimating the average response time and variance.

STS fits best for situations, where the query cost c_s is in the same order as the average utility retrieved ($O(c_s) = O(\overline{u})$). In settings with $O(c_s) \ll O(\overline{u})$ the search costs have no significant impact on the utility and if $O(c_s) \gg O(\overline{u})$ no searching will take place at all (because the involved costs are much higher than the possible benefit).

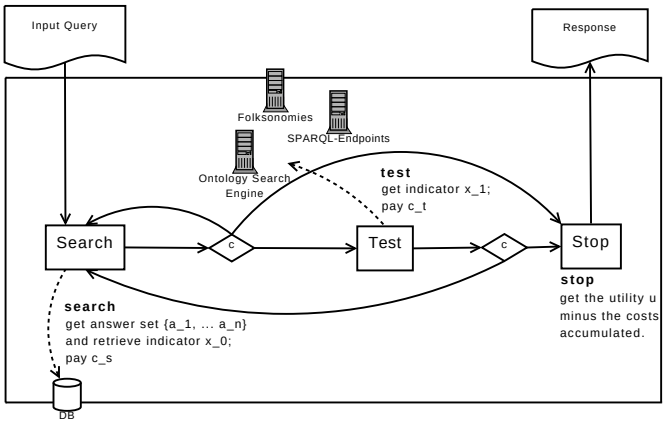

Fig. 3. Applying the search test stop model to Web resources.

In real world situations the translation from search times to costs is highly user dependent. To simplify the comparison of the results, this research applies a linear translation function $c_s = \lambda \cdot t_s$ with $\lambda = const = 1/\tilde{t_s}$ yielding costs of $O(c_s) = 1$. Selecting the median of the response times $\tilde{t_s}$ and specifying a timeout value of 60 seconds for any query operation reduces the influence of service outages on the simulation results.

The performance of the search test stop algorithm is highly dependent on accurate estimations of the query cost, because all decisions are solely based on the common probability mass function and the cost functions. Future research will compute query cost based on use case specific cost functions as demonstrated by Strunk et al. [12], Verma et al. [13], and Yeo and Buyya [14] and evaluate the results yielded by these different approaches.

Utility distribution The discrete common probability mass function h is composed of three components: The probability mass function of (i) the utility u, (ii) the random variable x_0 providing an estimate of the utility and, (iii) the random variable x_1 containing a refined estimate of the answer's utility.

In general a utility function assuming linearly independent utility values might look like Equation 5.

$$u = \sum_{a_i \in S_A} \lambda(a_i) f_{eval}(a_i) \tag{5}$$

The utility equals to the sum of the utility gained by each answer set S_A, which is evaluated using an evaluation function f_{eval}, and weighted with a factor $\lambda(a_i)$. To simplify the computation of the utility we consider only correct answers as useful (Equation 6) and apply the same weight ($\lambda(a_i) = const = 1$) to all answers.

$$f_{eval}(a_i) = \begin{cases} 0 & \text{if } a_i \text{ incorrect;} \\ 1 & \text{if } a_i \text{ correct.} \end{cases} \tag{6}$$

Geo-tagging identifies geographic entities based on a knowledge base as for instance a gazetteer or a trained artificial intelligence algorithm.

After searching the number of identified entries ($|S_a| = x_0$) provides a good estimation of the expected value of the answers utility. Applying a focus algorithm (e.g. [15]) yields a refined evaluation of the entity set ($|S'_a| = x_1$) resolving geo ambiguities. S'_a might still contain incorrect answers due to errors in the geo disambiguation or due to ambiguous terms not resolved by the focus algorithm (e.g. turkey/bird versus Turkey/country). Based on the probabilities of a particular answer $a_i \in S_a / a'_i \in S'_a$ of being incorrect $P_{incorr}(a_i)/P_{incorr}(a'_i)$ the expected value u for a given combination of x_0, x_1 is determined. Evaluating historical error rates yields estimations for $P_{incorr}(a_i)$ and $P_{incorr}(a'_i)$.

If no historical data is available heuristics based on the number of ambiguous geo-entries are useful for providing an educated guess of the probabilities.

A tagger recognizes patterns based on a pattern database table. The relation `hasPattern` translates these patterns to TaggingEntities as for instance spatial locations, persons, and organizations. Figure 4 visualizes a possible database layout for such a tagger.

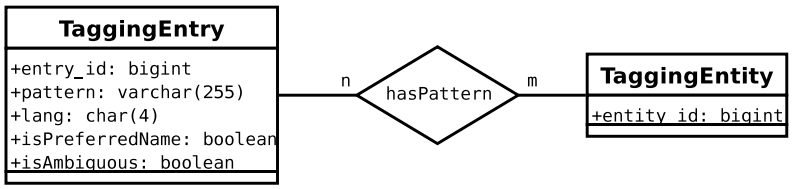

Fig. 4. Database schema of a simple tagger.

The `hasPattern` table often does not provide a unique mapping between patterns and entities - names as for instance Vienna may refer to multiple entities (Vienna/Austria versus Vienna/Virgina/US). On the other side many entities have multiple patterns associated with them (e.g. Wien, Vienna, Vienne, Bech, etc.). Based on the database schema above, $P_{incorr}(a_i)$ for such a tagger is estimated using the following heuristic:

$$n_{Entities} = |TaggingEntity| \tag{7}$$

$$n_{Mappings} = |hasPattern| \tag{8}$$

$$n_{ambiguous} = |\sigma_{[isAmbiguous='true']}(TaggingEntry * hasPattern)| \tag{9}$$

$$P_{incorr} = 1 - \frac{n_{Entries}}{n_{Mappings} + n_{ambiguous}} \tag{10}$$

Extending the database schema visualized in Figure 4 to non geo entries using WordNet and applying Equations 7-10 yields $P_{incorr}(a'_i)$.

5 Evaluation

For evaluating the STS model's efficiency in real world applications a simulation framework, supporting (i) a solely coverage based decision logic and the search test stop model, (ii) artificial (normal distribution) and measured (compare Section 2) distributions of network response times, and (iii) common probability mass functions $h(x_0, x_1, u)$ composed from user defined $P_{incorr}(a_i)$ and $P_{incorr}(a'_i)$ settings have been programmed.

To prevent the coverage based decision logic from delivering large amounts of low quality answers, the simulation controller only accepts answers with an expected utility above a certain threshold (u_{min}). In contrast the search test stop algorithm computes $u_{min} = r_D$ on the fly, based on the current responsiveness of the external service and the user's preferences.

5.1 Performance

Comparing the two approaches at different minimum quality levels (u_{min}), and service response time distributions approximated by a normal distribution $N(\bar{t}, \sigma_t^2)$ yields Table 2. The common probability mass functions has been composed with $P_{incorr}(a_i) = 0.3$, $P_{incorr}(a'_i) = 0.1$. The parameters for the normal distribution are $c_s = N(2, 1.9)$ for high search costs, $c_s = N(1, 0.9)$ for medium search costs, and $c_s = N(0.5, 0.4)$ for low search costs.

Search		Quality ($\overline{u}$)		Quantity ($\frac{\Delta u}{\Delta t}$)	
Cost (c_s)	u_{min}	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 2. Tagging performance.

Table 2 evaluates the two strategies according to two criteria: (i) *answer quality* $\overline{u}$, the average utility of a set (S_A) retrieved by the strategy, and (ii) *answer quantity* $\frac{\Delta u}{\Delta t}$, the rate at which the number of correct answers (and therefore the total utility (u)) grows.

High $\overline{u}$ values correspond to accepting only high quality results, with a lot of correct answers, and dropping low quality answer sets (at the cost of a lower quantity).

The conventional coverage based approach (Conv) delivers the highest quantity for small u_{min} values because virtually all answers are accepted and contribute to the total

utility. This greedy approach comes at the cost of a lower answer quality and therefore low average utility $\overline{u}$ per answer. Increasing u_{min} yields a better answer quality, but lower quantity values. At high search costs this strategy's performance is particularly unsatisfactory, because it doesn't consider the costs of the search operation.

In contrast to the conventional approach STS maximizes answer quality and quantity based on the *current* search cost adjusting queries to the responsiveness of the service and the user's preferences. These preferences formalize the trade-off between quality and quantity by specifying a transformation function between search cost and search times.

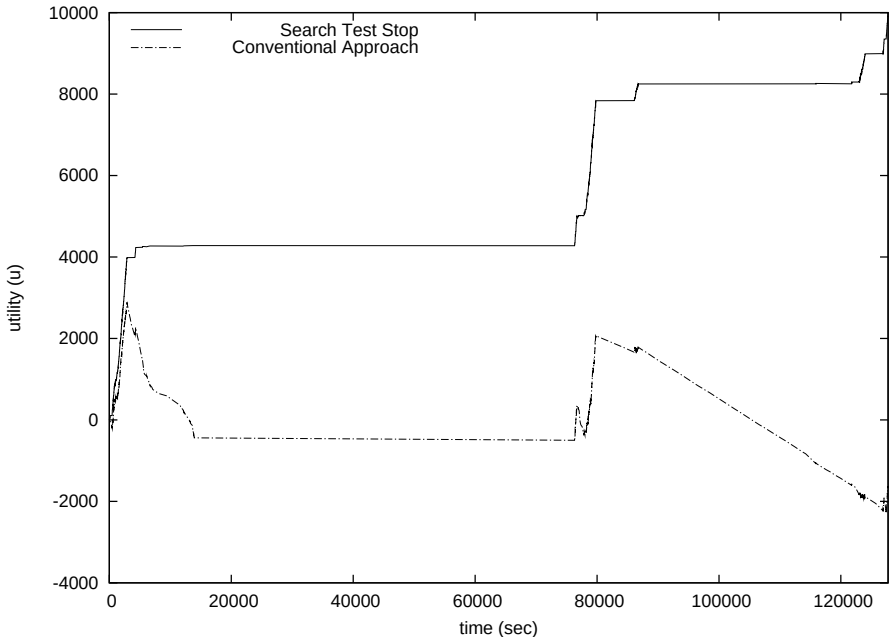

Fig. 5. Search test stop versus conventional decision logic for Swoogle ($\bar{t}$=1.6; $\sigma^2_{t_r} > 10000$).

STS therefore optimizes the agent's behavior in terms of user utility. This does *not* mean that STS minimizes resource usage. Instead STS dynamically adjusts the resource utilization based on the cost of searching (c_s) and testing (c_t), providing the user with optimal results in terms of accuracy *and* response times.

Enforcing a minimal utility u_{min} boosts the average utility $\overline{u}$ of the non STS service, but at the cost of a higher resource utilization, independent from the server's load (leading to extremely high response times during high load conditions). Static limits also do not consider additional queries at idle servers, leading to lower utilities under low load conditions. In contrast to the conventional approach STS (i) utilizes dormant resources of idle servers, and (ii) spares resources of busy servers, maximizing utility according to the user's preferences.

5.2 Web Services

In this section we will simulate the effect of STS on the performance of real world Web services, using search costs as measured during the Web service timing (compare Section 2).

The simulation facilitates the cost and common probability mass functions from Section 5. The figures 5-7 compare the tagger's performance when providing tagged documents corresponding to a utility score of 10,000 based on three different Web services (Swoogle, Google, GeoNames) with a minimum utility (u_{min}) of four.

In all three use cases STS performs well, because the search times are adjusted according to the service's responsiveness. GeoNames and Swoogle experience the highest performance boost, due to high variances in the search cost, leading to negative utility for the conventional query strategy. Using Google as external resource yields the fastest

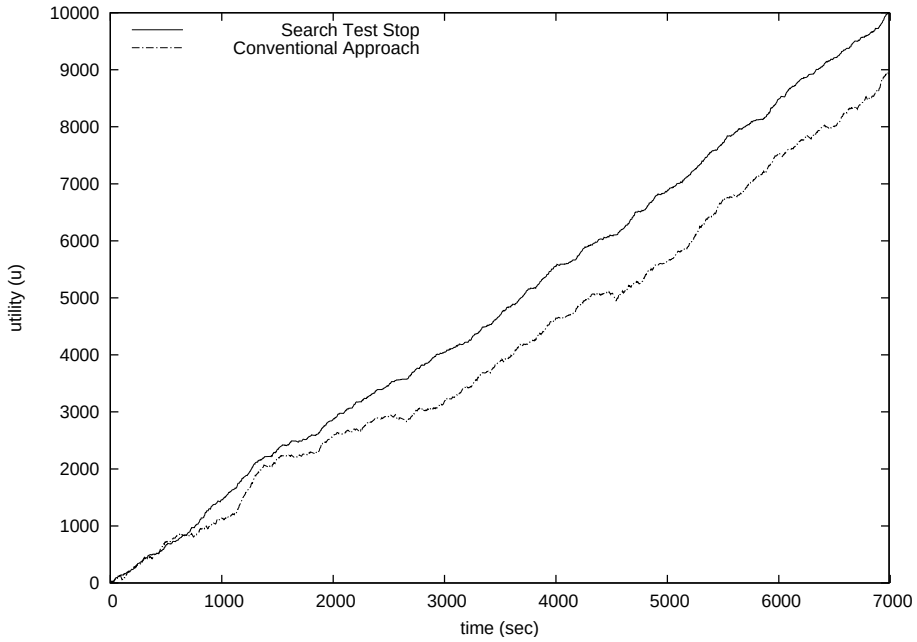

Fig. 6. Search test stop versus the conventional decision logic for Google ($\tilde{t}$=0.2; $\sigma^2_{t_r}$=0.2).

processing time. The algorithm is able to provide documents with the required quality level in around 8,300 seconds in contrast to more than 107,000 seconds for GeoNames and more than 129,000 seconds for Swoogle.

Services with low variances ($\sigma^2_{t_r}$) in their response times as for instance Google, del.icio.us and Wikipedia benefit least from the application of the STS model, because static strategies perform reasonable well under these conditions.

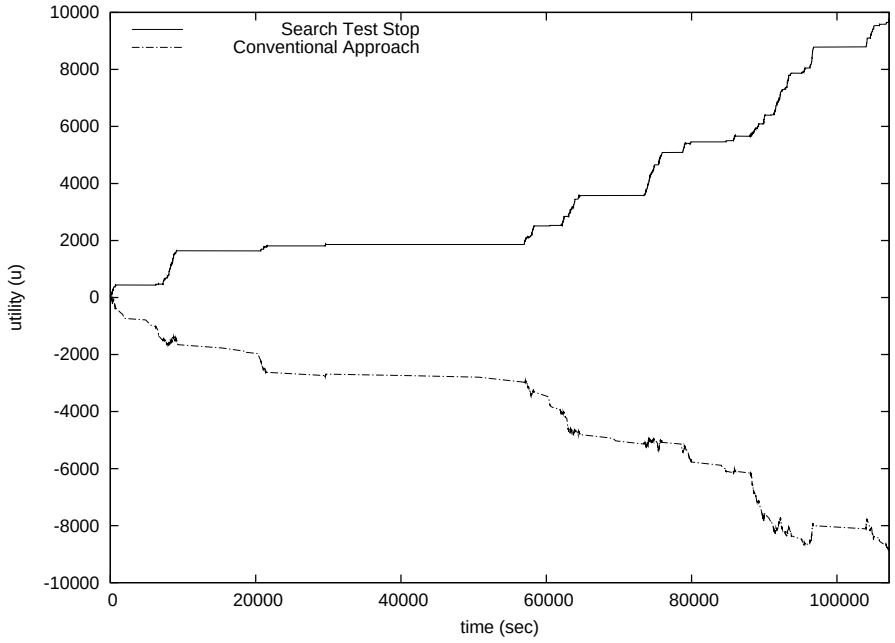

Fig. 7. Search test stop versus the conventional decision logic for GeoNames ($\tilde{t}$=0.1; $\sigma^2_{t_r}$=771.4).

6 Outlook and Conclusions

This work presents an approach for optimizing access to third party remote resources. Optimizing the clients resource access strategy yields higher query performance and spares remote resources by preventing unnecessary queries.

The main contributions of this paper are (i) applying the search test stop model to value driven information gathering, extending its usefulness to domains where one or more testings steps allow refining the estimated utility of the answer set; (ii) demonstrating the use of this approach to semantic tagging, and (iii) evaluating how the search test stop model performs in comparison to a solely value based approach.

The experiments show that search test stop and value driven information gathering perform especially well in domains with highly variable search cost.

In this work we only use one level testing, nevertheless, as Hartmann has shown [9] extending STS to n-levels of testing is a straight forward task. Future research will transfer these techniques and results to more complex use cases integrating multiple data sources as for instance semi automatic ontology extension [16]. The development of utility functions considering partially correct answers and user preferences will allow a more fine grained control over the process's performance yielding highly accurate querying strategies and therefore better results.

Acknowledgment

The author wishes to thank Prof. Wolfgang Janko for his valuable feedback and suggestions. The project results have been developed in the IDIOM (Information Diffusion across Interactive Online Media; www.idiom.at) project funded by the Austrian Ministry of Transport, Innovation & Technology (BMVIT) and the Austrian Research Promotion Agency (FFG).

References

1. Gupta, C., Bhowmik, R., Head, M.R., Govindaraju, M., Meng, W.: Improving performance of web services query matchmaking with automated knowledge acquisition. In: Web Intelligence, IEEE Computer Society (2007) 559–563
2. Grass, J., Zilberstein, S.: A value-driven system for autonomous information gathering. Journal of Intelligent Information Systems **14** (March 2000) 5–27(23)
3. Kukulenz, D., Ntoulas, A.: Answering bounded continuous search queries in the world wide web. In: WWW '07: Proceedings of the 16th international conference on World Wide Web, New York, NY, USA, ACM (2007) 551–560
4. Ipeirotis, P.G., Agichtein, E., Jain, P., Gravano, L.: Towards a query optimizer for text-centric tasks. ACM Trans. Database Syst. **32**(4) (2007) 21
5. Software Engineering Standard Committe of the IEEE Computer Society: IEEE std 830-1999: IEEE recommended practice for software requirements specifications (1998)
6. International Standards Organization JTC 1/SC 7: ISO/IEC 9126-1, 2001. software engineering – product quality – part 1: Quality model (2001)
7. Horvitz, E.J., Breese, J.S., Henrion, M.: Decision theory in expert systems and artificial intelligence. International Journal of Approximate Reasoning **2** (1988) 247–302
8. MacQueen, J.: Optimal policies for a class of search and evaluation problems. Management Science **10**(4) (1964) 746–759
9. Hartmann, J.: Wirtschaftliche Alternativensuche mit Informationsbeschaffung unter Unsicherheit. PhD thesis, Universität Fridericiana Karlsruhe (1985)
10. Lehmann, E.L., Romano, J.P.: Testing Statistical Hypotheses. 3rd edition edn. Springer, New York (2005)
11. Scharl, A., Weichselbraun, A., Liu, W.: Tracking and modelling information diffusion across interactive online media. International Journal of Metadata, Semantics and Ontologies **2**(2) (2007) 136–145
12. Strunk, J.D., Thereska, E., Faloutsos, C., Ganger, G.R.: Using utility to provision storage systems. In: FAST'08: Proceedings of the 6th USENIX Conference on File and Storage Technologies, Berkeley, CA, USA, USENIX Association (2008) 1–16
13. Verma, A., Jain, R., Ghosal, S.: A utility-based unified disk scheduling framework for shared mixed-media services. Trans. Storage **3**(4) (2008) 1–30
14. Yeo, C.S., Buyya, R.: Pricing for utility-driven resource management and allocation in clusters. International Journal of High Performance Computing Applications **21**(4) (November 2007) 405–418
15. Amitay, E., Har'El, N., Sivan, R., Soffer, A.: Web-a-where: geotagging web content. In: SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (2004) 273–280
16. Liu, W., Weichselbraun, A., Scharl, A., Chang, E.: Semi-automatic ontology extension using spreading activation. Journal of Universal Knowledge Management **0**(1) (2005) 50–58 http://www.jukm.org/jukm_0_1/semi_automatic_ontology_extension.