Semantic Web in a Constrained Environment

Laurens Rietveld and Stefan Schlobach

Department of Computer Science, VU University Amsterdam, The Netherlands
{laurens.rietveld,k.s.schlobach}@vu.nl

Abstract. The semantic web is intrinsically constrained by its environment. These constraints act as a bottlenecks and limit the performance of applications in various ways. Examples of such constraints are the limited availability of memory, disk space, or a limited network bandwidth. But how do these bounds influence Semantic Web applications? In this paper we propose to study the Semantic Web as part of a constrained environment. We discuss a framework where applications adapt to the constraints in its environment.

Keywords: downscaling, ranking, constraints, resource bounds

1 Introduction

Agreement that the Semantic Web has become a huge success story is widening: standardised languages exist for representing web data and ontologies, together with tools to deal with such semantic information. This facilitates applications that publish and consume *triples* in the billions, in domains as various as life-sciences, cultural heritage, e-business and journalism. Additionally, the number of robust commercial triple stores is constantly increasing, and successful DBMS such as Oracle become RDF aware. Parallelization and distribution have made expressive semantic reasoning massively scalable, and the movement towards storage and reasoning in the cloud suggest that even the last technological barriers can soon be overcome.

Although we partially share this optimistic view, it only paints a part of a larger picture. In fact, the apparent scalability critically depends on a computational infrastructure that is out of reach for the vast majority of the human population. Powerful servers, supercomputers and clusters are the privilege of few. Often, users of the Semantic Web cannot rely on a constant and reliable Internet connection with sufficient bandwidth. In more remote regions, even electricity supply is not always guaranteed, with restricted battery runtime having to be considered when building applications. But it is not just the infrastructure, or its lack of, that provides unsurpassable boundaries for the Semantic Web to be used and useful. Think, e.g., of real-time user interaction interaction that restricts the computation time to the often very short attention span of humans. After all, who wants to wait more than a few seconds for search results? Additionally, such interaction has to take the human processing bandwidth into

account. Anybody wants to see more than 10 search results? No. In short: the Semantic Web is bound by resource availability.

Although those bounds look diverse at first glance, we suggest to study information access and processing on the Semantic Web in the context of these bounds more systematically. The purpose of such an analysis is not an end in itself: such an analysis of the relation between Semantic Web applications and the way they are resource bound can help when building better applications, or to build good applications more easily. A promising approach is to study the explicit and intrinsic orderings and rankings in data and the information need. This information can then be used to deal with the resource-bounds. Take as example the human attention span, which requires applications to produce results extremely fast. This is often impossible in computationally expensive representation languages and for applications involving complex data, unless the intrinsic rankings are used to produce *good* results first and fast. Ranking of goodness is at the basis of such any-time approaches.

This paper is a first attempt to investigate the dependencies between particular resource bounds and the type of orderings and rankings used to overcome the boundaries. Is there a more generic relation between ranking and resources? And if so, can we explicate this relation, and use it to guide the process of building Semantic Web applications in a resource bound world? The rest of this paper introduces a number of assumptions underlying our approach (section 2). Following, we study those assumptions in the context of two Semantic Web applications(section 3). We conclude with future work and open questions(section 4).

2 Constraints & Ranking

In the introduction we claim that resource bounds and rankings are related. This claim constitutes the foundations of an unifying idea for more easily building good application for the Semantic Web: the idea of using orderings in the data and application requirements to deal with explicitly defined resource bounds. This section will introduce those claims more systematically. We will first elaborate on these statements. Afterwards, in the next section, we will study them in light of two very different use-cases.

(1.) The world is resource bound. Our environment is full of constraints. These bounds constrain us, applications, and the way we interact with, and have to build, these applications.

(2.) Deal with these constraints. Applications have to (implicitly) deal with these constraints, which often means trading functionality of the application to remain within the given bounds (e.g. switch to off-line mode when there is no connectivity).

(3.) We need ranking Applications can deal with these constraints by ranking results and/or tasks. This enables the application to take a sub-selection (top-k) part of the results or tasks, and process them. In doing so, the application

does not process the complete result or task set. An example where the ordering of tasks is used to adapt to changing bounds, is that of an operating system. Most operating systems will rank the running tasks in importance, and give the higher ranked tasks precedence. This ensures the most important tasks will still run whenever there is a high CPU load (resource bound).

(4.) Ranking: it's all in the data Ranking of results and/or tasks depends on rankings of data. This data often contains explicit ordering such as recency. Or there is a more implicit ordering covering items such as importance or relevance. For this framework we selected a set of ranking measures which are easy to retrieve and calculate:

- *Recency:* How old is this entity
- *Size:* How large is this entity
- *Frequency:* How often is this entity used or accessed
- *Similarity:* How similar is this entity compared to others

(5.) We need explicit bounds In doing so, an application is able to use these bounds as input, and link these bounds to other functionality in the application. The following list is a first attempt at distinguishing between the generic types of constraints relevant for Semantic Web applications:

Hardware Constraints We consider any machine (e.g. server, pc, laptop) hardware restriction as a machine constraint. Examples of such constraints are *Memory*, *Hard disk space* or *Battery power*. Not all constraints are applicable to every domains. For example, the hardware restriction *Battery power* only applies to environments where laptops are used.

Network Constraints The Semantic Web needs network connections. The connections between the nodes in such a network are constrained by for example *Network Bandwidth* or *available connectivity*.

Interaction Constraints We consider interaction constraints as the constraints imposed by the limits of the agent using the application. These constraints are more difficult to obtain, but they might be retrievable from access logs or user profiles. Examples are *reaction time*[1] or *Processing Bandwidth*, i.e. the maximum information an agent is capable of processing in a given situation.

(6.) Tell me your bounds and ranking, and I deal with them. This requires an explicit link between the constraints and the ranking measures. By using this link, applications can adapt their ranking measures to the changing resource bounds. The ability to adapt is particularly useful, because resource bounds are not static: Network bandwidth might fluctuate, just as available memory can change. Applications and the way they use the ranking measures should change as well.

The picture shown in figure 1 illustrates the plausible connections between the constraints and the ranking measures. The case study descriptions in the next section will elaborate on these connections in more detail.

[1] The maximum time limit between a *request* of the agent to the SW application, and the desired *response* from the application.

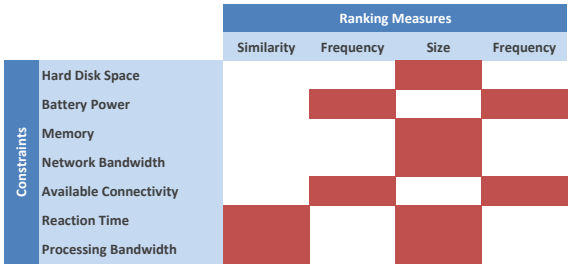

Fig. 1: Ranking Measures vs. Constraints

3 Case Studies

We discuss two very different case studies: the SemanticXO and visualization of census data, and the way rankings and resources bounds relate in them.

3.1 SemanticXO

An environment with obvious constraints is given in the SemanticXO project. The XO laptop is part of the One Laptop per Child (OLPC) project. The aim of OLPC is to create educational opportunities for the worlds poorest children by providing each child with a 'rugged, low-cost, low-power, connected laptop'. The SemanticXO is a project which aims "to provide an infrastructure that is needed to integrate semantic information from the Web of Data into programs (...) running on an XO computer" [2].

Software Stack Currently, the SemanticXOs software stack contains [2]:

- A triple store, in charge of storing triples with meta-data.
- An HTTP server, which serves as a public interface to the triple store, provide de-referenceable URIs for the resources, and serve the files. The HTTP server is accessible by other XOs.
- A common interface (API) for accessing the triples stored locally, on another SemanticXO, or elsewhere on the Web of Data.

The SemanticXO uses this software stack to store information as a Data Graph in the triple store. After the graph is stored in the triple store, it is available to other XOs via the HTTP Server or the API.

Problem description XO's are often used in a network with similar XO's, where a school server functions as an access point with internet connection. Currently, distributing information within such difficult. Moving information from one XO to another requires both XOs to be on-line, and run the same program (called 'activity'). Situations where the recipient of information is off-line, where information may be shared asynchronously, or where information needs to be shared regardless of any running activity, are currently impossible to deal with.

The SemanticXO offers a framework with which to approach this problem. Currently, objects containing meta-information are stored locally as a data store object (DS-Object). These objects contain meta-information, and contain (a combination of) attribute-value pairs. More complex information such as images are stored separately. Generic shipping of objects requires a DS-Object to be wrapped as a Semantic DS-Object (SDS-Object), which is stored as a graph in the triple store. Via the HTTP server or the API, this information is automatically available to other XO's in the network. Using this infrastructure, the nodes can share and sync objects. This creates a network of loosely coupled triple stores between which information has to be distributed.

Constraints & Ranking We will discuss the constraints and rankings in this case study, using the claims from the previous section.

(1.) The world is resource bound Internet connections are unreliable or have a very limited bandwidth, and hardware may not be on-par with regular laptops and desktops, to name but a few.

(2.) Deal with these constraints Not dealing with constraints such as network bandwidth will most likely result in an inadequate distribution of objects. Not all objects may be distributed to all nodes, which means SDS-Objects run a risk of not being delivered to the intended recipient.

(3.) We need ranking Constraints may limit the number of SDS-Objects to be shared, which means a subsection of the SDS-Objects should be synced. This is essentially a top-k ranking problem, where only the most useful objects are cached and synced, and others are ignored.

(4.) Ranking: it's all in the data Relevant ranking measures are *Recency* (i.e. how long ago is this graph created or modified), *Size* (i.e. how big (in bytes) is the graph), and *Frequency* (i.e. how often is this graph synced)

(5.) We need explicit bounds The resource bounds of the SemanticXO are either caused by the hardware specification[1] or the XO network connections. We consider the following resource bounds for the SemanticXO (a subset of the bounds described in section2): Available memory, available hard disk space, battery power, network bandwidth and available connectivity.

(6.) Tell me your bounds and ranking, and I deal with them By combining the resource bounds **(5)**, the rankings **(4)** and the links between both (fig 1), the application can deal and adapt to its resource bounds. We consider the following relations between the constraints and these rankings measures:

1. **Hard Disk Space:** With limited disk space available, giving precedence to smaller (*size*) objects will allow the SemanticXO to store a larger quantity of objects.
2. **Battery Power:** When battery power is running low, giving precedence to *recency* and *frequency* will make sure the old objects which have not been synced that often are ordered higher than others, because older objects in the queue might indicate these objects are not spread through the network.
3. **Available Memory:** Processing large graphs might require more memory. When available memory is low, precedence should be given to smaller graphs.

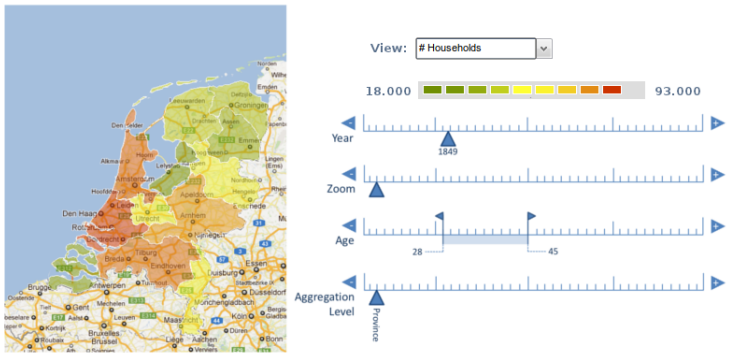

Fig. 2: Census Visualization

4. **Network Bandwidth:** A low network bandwidth should decrease the importance of the *size* of SDS-Objects. This way, at least the smaller objects will still be distributed via this node.
5. **Available Connectivity:** When the SemanticXO is connected to a very limited number of other nodes, the graphs which are not very well distributed in the network should be given precedence. This means objects which are not often synced (*frequency*) and older object (*recency*) are ranked higher than others.

This framework is particularly useful for the SemanticXO, because hardware may differ between XO's. An XO server has completely different hardware (bounds) than a regular XO laptop. Using the described framework, the server and laptop can both optimize their performance based on their own resource bounds.

3.2 Visualizing Census Data

To show that resource bounds are not limited to the environment of developing countries we describe a totally different application where Dutch historical census data is visualized. This use case is part of the Data2Semantics project[2], which focusses on the problems of 'how to share, publish, access, analyse, interpret and reuse scientific data'.

Visualizing Census Data This dataset contains Dutch census counts from between 1795 and 1971, and covers demographic information, such as gender, age, location, marital status, occupation, household and religion. An example of an application making use of the (convert to RDF) dataset is shown in figure 2. This visualization has several selectors:

1. *Variable selection:* The variable of which the distribution is shown on the map
2. *Time:* The year of the census count

[2] `www.data2semantics.org`

3. *Zoom:* Zoom in or out of the map
4. *Age:* Age range for which to visualize the data for
5. *Aggregation Level:* The aggregation level for the visualization. Changing this value changes the granularity of the colors on the map.

Because this visualization makes use of a SPARQL endpoint, these selectors correspond to SPARQL queries. In this respect, we can consider the *variable selection*, *time*, and *age* as filters, and the *aggregation level* as a 'group by'.

Constraints & Ranking We will discuss the constraints and rankings in this case study, using the claims from section 2.

(1.) The world is resource bound This visualization application is clearly resource bound. Users will expect near to any-time responses when changing the sliders. Additionally, the application will have to deal with current (consumer) hardware and network constraints.

(2.) Deal with these constraints Not adapting to the constraints leads to situations where the response time (when changing a slider) is too long.

(3.) We need ranking A way to deal with these constraints is by pre-executing queries. Executing all possible setting combination will not be feasible, however: there are too many possible combinations of queries. Deciding which queries to pre-execute, though, is a ranking problem/

(4.) Ranking: it's all in the data Relevant ranking measures are:

1. *Query similarity:* Difference in similarity between current view, and the possible query. The bigger the difference, the wider the 'window' becomes which is being pre-executed.
2. *Frequency of usage:* Selectors often used are more interesting to pre-execute. Therefore, queries where the filter value of a frequently used selector is different from the filter value in the SPARQL query of the current view, should have precedence.
3. *Size of expected number of results for a given query:* e.g., decreasing the aggregation level increases the retrieved number of triples.

(5.) We need explicit bounds The resource bounds of this visualization involve: available memory, network bandwidth, reaction time of the user, and processing bandwidth of the user.

(6.) Tell me your bounds and ranking, and I deal with them By combining the resource bounds **(5)**, the rankings **(4)** and the links between both (fig 1), the application can deal and adapt to its resource bounds. We consider the following relations between these constraints and the rankings measures:

1. **Memory:** Limited memory implies limited available space to store the query results. This should give precedence to queries where the expected query result is low.
2. **Network bandwidth:** The smaller the network bandwidth is, the smaller the expected *size* of query results should be.
3. **Reaction time:** More time might allow for a smaller *query similarity*, which results in a bigger window size for the pre-executed queries. A larger interval allow for a larger query result, as there is more time available for the response to be generated and processed.

4. **Processing bandwidth:** The more information a user is able to process, the larger the *size* of the query result may be, and the bigger the allowed windows size may be (*query similarity*).

4 Future Work & Conclusion

This paper showed a perspective with which to consider applications in a (dynamic) resource bound environment. The presented framework should allow semantic web applications to adapt to changing constraints, by making use of the explicit and intrinsic orderings and rankings in the data. We showed the generic link between ranking and constraints, and put these into context of two very different case studies. This framework is not complete though. A roadmap towards completion requires discussion on some unanswered questions, and an implementation/evaluation of this framework. Current open questions are:

- The ranking measures mentioned in section2 are very broad: recency, size, frequency and similarity. Can we keep these notions on such a generic level? Or are these ranking measures open to interpretation, depending on the application domain?
- What are the generic relations between constraints and ranking measures? For both use cases we show some links. Are these generic? And does dealing with one constraints influence other constraints? Is there a cost in dealing with constraints.
- How do the ranking measures relate to each other in such a dynamic system? How is the final ranking score determined?
- How to define interaction constraints? Different users interact differently with a system.
- How does this approach hold up against other methods such as the use of access control, differentiated services, or formal maximization under constraint?

The main question seems to be: does explicating these concepts and relations work in practice? This requires further implementation and evaluation of this framework in one of the use cases.

Acknowledgements. This publication was supported by the Dutch national program COMMIT.

References

1. CL1 Hardware Design Specification (2008)
2. Guéret, C., Schlobach, S.: SemanticXO : connecting the XO with the World's largest information network. In: Proceedings of the First International Conference on e-Technologies and Networks for Development (ICeND2011). "Communications in Computer and Information Science", Springer LNCS, Dar-es-Salaam, Tanzania (2011)