Semantic query answering in digital repositories: Semantic Search v2 for DSpace

Dimitrios A. Koutsomitropoulos* and
Georgia D. Solomou

High Performance Information Systems Lab,
Department of Computer Engineering and Informatics,
University of Patras,
Building B, Second Floor, 26500 Rion, Patras, Greece
Email: kotsomit@hpclab.ceid.upatras.gr
Email: solomou@hpclab.ceid.upatras.gr
*Corresponding author

Theodore S. Papatheodorou

Deceased; formerly of: University of Patras

Abstract: The added value the Semantic Web has to offer can find fertile ground in querying collections with rich metadata, such as ones often occurring in digital libraries and repositories. A relevant such effort is the semantic search service for the popular DSpace digital repository system. Semantic Search v2 introduces a structured query mechanism that makes query construction easier as well as several improvements in system design, performance and extensibility. Queries are targeted towards the dynamically created DSpace ontology, containing constructs that enable knowledge acquisition among available metadata. Both an empirical and a quantitative evaluation suggest that the system can bring semantic search closer to inexperienced users and make its benefits evident in the context of digital repositories, such as new querying dimensions, thus forming a paradigm production services can built upon.

Keywords: Semantic Web; semantic search; reasoning; digital libraries; digital repositories; information search and retrieval; knowledge acquisition; query answering; ontology design.

Reference to this paper should be made as follows: Koutsomitropoulos, D.A., Solomou, G.D. and Papatheodorou, T.S. (2013) 'Semantic query answering in digital repositories: semantic search v2 for DSpace', *Int. J. Metadata, Semantics and Ontologies*, Vol. 8, No. 1, pp.46–55.

Biographical notes: Dimitrios A. Koutsomitropoulos is an Adjunct Assistant Professor at the Department of Computer Engineering and Informatics and a Research Fellow at the High Performance Information Systems Laboratory (HPCLab), University of Patras. He has received a fellowship-awarded PhD and an MSc from the same department, specialising in knowledge management and discovery on the web. For over ten years he has lead and participated in many R&D projects with European and national funding. His research interests include knowledge discovery, ontological engineering, semantic interoperability and the semantic web, where he has published an important number of research articles and papers.

Georgia D. Solomou is currently pursuing a doctoral degree in Semantic Web technologies at the University of Patras in Greece. She holds a Master's degree in Computer Science from the same University. She is a member of the High Performance Information Systems Laboratory (HPCLab) since 2005 and she has taken part in many European research projects. Her research interests are related to the Semantic Web standards and technologies, mainly focusing in the field of interoperability and educational resources in digital libraries.

Theodore S. Papatheodorou was Professor at the Department of Computer Engineering and Informatics, University of Patras, from 1984 to 2012. He served as the Department's Chair between 2005 and 2009. He was Head of the High Performance Information Systems Laboratory (HPCLab). He received a PhD in Computer Science in 1973 and an MSc in Mathematics in 1971 from Purdue University as well as a BSc in Mathematics from University of Athens in 1968. He authored hundreds of scientific publications in several areas of computer engineering and computer science.

1 Introduction

Instead of relying on traditional data retrieval, the Semantic Web can present digital content repositories with novel pathways in accessing and discovering information. By combining the semantics of resource descriptions, either explicitly annotated or implied, reasoning-based query answering can enable and produce results that would be very hard or even impossible to deduce otherwise. For such an approach to be appealing and thus useful to end users, it has to withhold its inherent complexity and be as intuitive as possible.

DSpace is an up-to-date and popular digital repository software, backed by one of the largest communities of users and developers worldwide (http://www.dspace.org). On top of DSpace we have designed, developed and deployed a complete and integrated semantic search mechanism that facilitates users to perform reasoning-based queries to the repository's contents.

To evaluate this mechanism, we have prepared and conducted an online survey that explores both the service's usability and performance. Our goal is first to discover the service's impact and appeal to all kinds of users (those familiar and non-familiar with Semantic Web concepts), and then to detect possible areas for improvement and collect user suggestions. In addition to the survey, an analysis of several real queries based on logged data is also considered, in accordance to current recommendations for the evaluation of semantic search systems (e.g. Wrigley et al., 2010; Uren et al., 2011).

Semantic search in digital libraries and repositories has also been attempted before; however, added-value in terms of entailment-based knowledge acquisition and discovery is not always clear. A recent overview of tools and systems to support semantic search in digital repositories can be found in the work of Koutsomitropoulos et al. (2010b).

In this paper, we focus on the Semantic Search v2 plugin for the DSpace digital repository system, initially introduced by Koutsomitropoulos et al. (2011). In comparison, this paper reflects the evolution and several enhancements in the DSpace ontology as well as various improvements in the system design, including support for new reasoners; above all, an extensive evaluation has been conducted, both quantitative as well as empirical, taking also into account the end user perspective and tackling with scalability issues.

The rest of this paper is organised in six sections: in Section 2, we give an overview of the DSpace ontology and outline its enhancements and their purpose. The actual system architecture is described, along with the main features that result from design improvements, in Section 3. Section 4 includes specific query examples made possible with the system and compares them to traditional search options. Section 5 shows and comments on the findings of the user survey, presents a quantitative analysis of queries performed and discusses current scalability potential. Finally, Section 6 summarises our conclusions.

2 The DSpace ontology

The DSpace ontology is the knowledge base of the Semantic Search plugin. The main idea is to enable export of as much available DSpace metadata as possible and then to translate them into full-fledged RDF/OWL triples. In this process, certain implicit entities (like items, collections, authors, sponsors) are reified and become individual nodes themselves instead of mere literals .

This reification is performed on the fly, following a sophisticated procedure fully described by Koutsomitropoulos et al. (2010b) and based on the interoperable system's mechanisms for exporting resources' metadata through OAI-PMH. In essence, metadata are first harvested and then transformed by means of an XSLT to the required OWL/RDF representation. The vocabulary used in these triples is specified by the DSpace ontology.

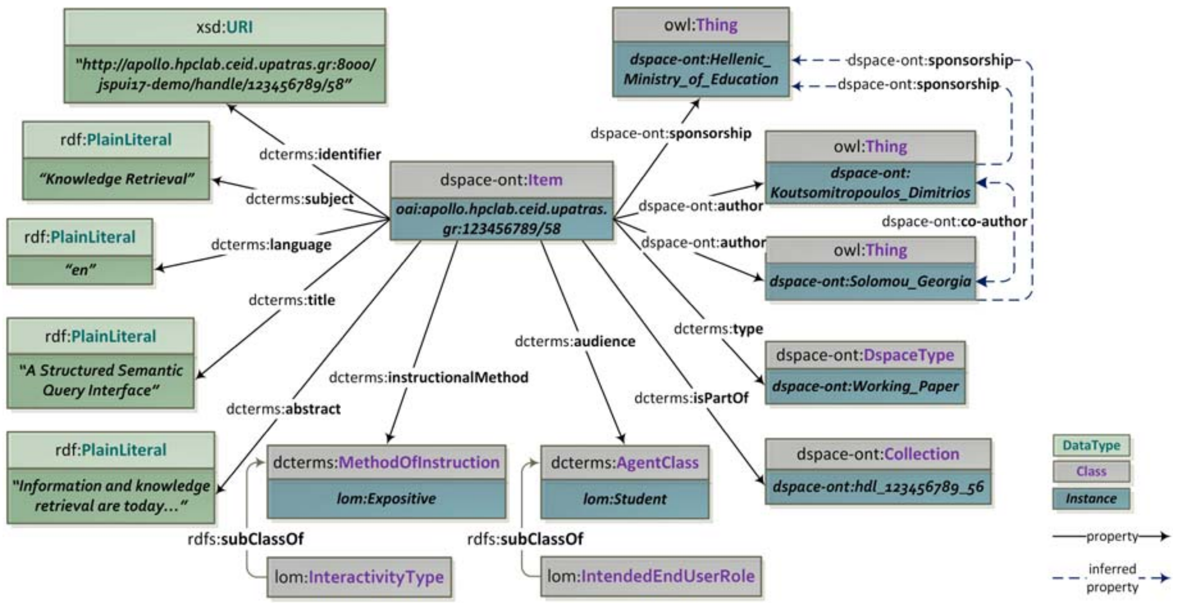
The DSpace ontology is based on the qualified Dublin Core (QDC) RDF schema and is in fact comprises several ontology documents that import each other, thus forming an incremental semantic application profile. All these documents are kept and maintained separately from the application's code base and are available at: http:// swig.hpclab.ceid.upatras.gr/dspace-ont/, which serves also as the main vocabulary namespace. Figure 1 shows how a sample DSpace metadata record may look like after its triplification and mapping to the DSpace ontology.

Since DSpace allows the extension and configuration of its metadata schema, we have also extended it with several Learning Object (LOM) metadata. At least for the part used in the implementation, a reference LOM ontology has also been devised. As a result, the translation to OWL takes also into consideration LOM metadata and their mapping to QDC.

The main idea behind this mapping is to consider first those LOM elements having an explicit counterpart in the DC elements set and map them accordingly. For example, LOM elements like *Title*, *Language* and *Contribute* were directly mapped to *dc.title*, *dc.language* and *dc.contributor*, respectively. The exact correspondence between these elements can also be found in IEEE LTSC (2002). Remaining LOM elements, for which no such explicit correspondence exists, are carefully mapped to those QDC properties that seem as a best match for their interpretation. The decision was made after considering both the IEEE LOM specification (IEEE LTSC, 2002) and the work suggested by IEEE LTSC (2008), which proposes a potential LOM to DC Abstract Model (DCAM) mapping. A more detailed account of the mapping procedure and the LOM ontology can be found in the work of Koutsomitropoulos et al. (2010a).

Semantic Search v2 includes several enhancements and/or fixes throughout the DSpace ontology creation process as well as in the involved ontology documents, aiming for better resource discovery, reasoner compatibility and extended inference capabilities. For example, the Unique Name Assumption (UNA) is now enforced, thus allowing queries that constrain the number of entities related through a certain property, e.g. the number of items produced by the same author (number restrictions). In addition, the output of the translation is now serialised in XML syntax (Motik et al., 2010) to ensure explicit assertion of property triples (whether datatype- or object-) instead of RDF, which appeared to cause problems with punning.

Figure 1 A sample record in the DSpace ontology (see online version for colours)



3 The Semantic Search v2 plugin

With Semantic Search v2, a structured querying mechanism and interface is introduced, which guides users in building semantic queries. The main idea of this new searching mechanism is to aid the user in breaking down his intended query expression into several atoms. These atoms now span several input fields in the UI and are then combined to form allowed expressions in Manchester Syntax (Horridge and Patel-Schneider, 2012). At the same time, the interface tries to be as intuitive as possible by making suggestions, checking user input and enabling or disabling subsequent fields accordingly. Next, Section 3.1 summarises new features. Section 3.2 gives a deeper insight into the precomputation and caching philosophy and discusses potential trade-offs.

3.1 New features

The novel semantic search interface is backed by an updated *DSpace Semantic API* that uses the new OWL API v.3.2, coming with additional capabilities for reasoner and ontology handling (Horridge and Bechhofer, 2009). The overall system architecture and interface is depicted in Figure 3. Design improvements aim at leveraging extensibility and performance and include:

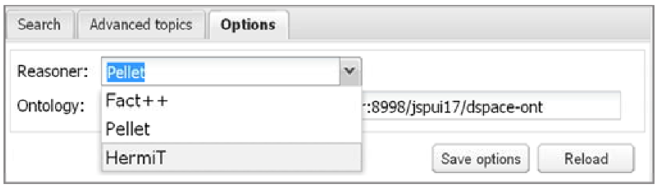
Hot-swap between reasoners: One of the design principles of semantic search was extensibility and support for different reasoners and ontologies. Therefore, the new semantic API gives users the ability to 'hot-swap' between reasoners dynamically by implementing a pluggable reasoner wrapper. For the time being, any OWL API compliant reasoner can be supported, including out-of-the-box support for Pellet, FaCT++ and HermiT (see Figure 2).

Ontology caching: Semantic Search runs by default on top of the DSpace ontology. Nevertheless it can work against any OWL ontology accessible through the web. Loaded ontologies can be reused, thus avoiding the overhead of reloading and parsing the whole ontology definition. In particular, when a user asks for a new ontology, this is loaded and stored only once in an internal registry. When another user asks for the same ontology, no reparsing is needed, and the ontology is served from the registry.

Precomputing inferences: An important feature implemented in current releases of semantic search v2 is inferences' precomputation. Precomputation is an initialisation phase where the reasoner classifies the ontology, and computes in advance certain reasoning operations, such as instance checking. Precomputation results are then readily available in memory and, instead of engaging the inference engine each and every time, new queries can take advantage of them, resulting in a significant performance increase.

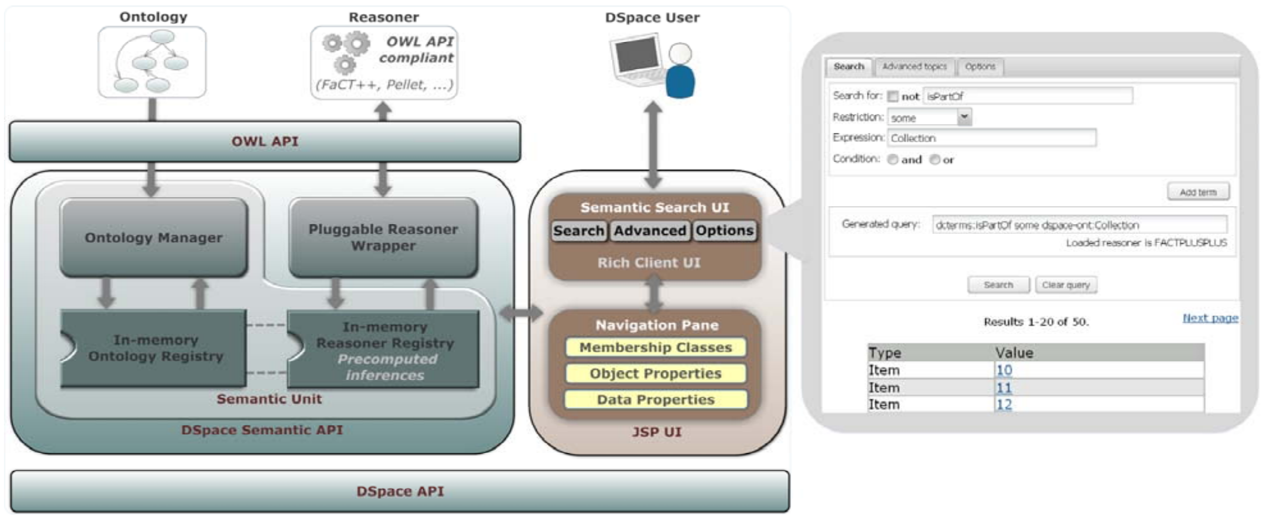
Reasoner caching: Just like ontologies, reasoner objects among different users can also be cached in an internal registry and reused. Combined with inference precomputation, this can also improve average performance of multiple individual requests on the same ontology. Finally, a '*Reload*' button that clears registry and reloads both the ontology and the reasoner anew is provided.

Figure 2 Hot-swap between reasoners



Reasoner: Pellet
Ontology: :8998/jspui17/dspace-ont

Figure 3 The semantic search architecture and interface (see online version for colours)



Search for: not isPartOf
Restriction: some
Expression: Collection
Condition: and or

Generated query: dcterms:isPartOf some dspace-ont:Collection
Loaded reasoner is FACTPLUSPLUS

Results 1-20 of 50.

Type	Value
Item	10
Item	11
Item	12

3.2 *Reasoner precomputation and cache*

Figure 4 depicts the actual precomputation algorithm we follow in our implementation. Precomputation is actually delegated to the reasoner, as a standard reasoning operation in Description Logics systems (Nardi and Brachman, 2002). The set of available inferences types includes certain reasoning tasks like the computation of the ontology class hierarchy, the class membership of individuals (*class assertions*), the property hierarchies and the relationships between individuals (*property assertions*). There is no point in having a reasoner without an ontology and a reasoner object can only be instantiated by providing the ontology it would operate on. We represent such a pair of a reasoner *r* and an ontology *o* with <*r*, *o*>.

Figure 4 Reasoner precomputation procedure

```
T: the set of requested inference types
o: active ontology
<r, o>: selected reasoner

initialize <r,o> {
  T ≡ {class hierarchy, class assertions}
  for each t in T {
    if not isPrecomputed (t, <r,o>)
      precompute (t, <r, o> )
      isPrecomputed (t, <r,o>) = true
    }
  }
}
```

Not all reasoners support all available inference types. In addition, certain inference types (e.g. data property assertions) may take too long or fail and it is unclear whether they contribute to query performance, since our query policy is class oriented. Furthermore, some inference types include the others. For example, the computation of property hierarchies is a prerequisite for discovering property assertions. Therefore, we cause precomputation for some inference types only, the ones we found that contribute the most to our strategy and that are actually implemented by the reasoners.

It should be noted that there are no substantial trade-offs related to inference precomputations. Most Semantic Web reasoners require an initialisation phase (classification) in order to become operable, i.e. there are not capable of answering queries on demand.

Further reasoning tasks, like the computation of data property assertions can be demanded, but are often impractical: they take too long (e.g. in the case of HermiT several minutes), but do not cut-off query times accordingly.

Figure 5 shows the algorithm for implementing the logic of the reasoner and the ontology registry. Notice also the call to the *initialise* algorithm explained previously, which implements precomputation.

Figure 5 The logic of caching

```
$\mathcal{L}$: Ontology registry, o: active ontology
$\mathcal{R}$: Reasoner registry, <r,o>: selected reasoner

if $\{o\} \nsubseteq \mathcal{L}$ or new-user or reload-button
then load o
    create <r,o>
    initialize <r,o>
    $\mathcal{L} \to \mathcal{L} \cup o$
    $\mathcal{R} \to \mathcal{R} \cup$ <r,o>
    return <r,o>
if $\{o\} \subseteq \mathcal{L}$ and <r,o> $\nsubseteq \mathcal{R}$
then create <r,o>
    initialize <r,o>
    $\mathcal{R} \to \mathcal{R} \cup$ <r,o>
    return <r,o>
if $\{o\} \subseteq \mathcal{L}$ and <r,o> $\subseteq \mathcal{R}$
then return <r,o>
```

Ontology caching is especially useful in case of large ontologies that take too long to fetch and/or for knowledge bases that grow slowly. For other cases, incremental reasoning is currently open research (e.g. Grau et al., 2007), but a viable alternative can be offered by triple stores.

For example, OWL 2 RL (Motik et al., 2011) is an OWL profile specifically designed to be amenable to rule-based implementations that are known to be scalable, as is often the case with triple stores. As such, it imposes some restrictions to the full set of OWL 2. The DSpace ontology includes the following axiom:

$$\text{Item} \sqcap \exists\ \text{dcterms:hasPart.Item} \sqsubseteq \text{Collection}$$

stating that everything that has part an item is a collection, except if it is an item itself. This is useful, for example, to automatically classify DSpace collections correctly, to convey the information that collections are consisting of items and to distinguish between collections and items that may be related to other items. However, this particular axiom is not allowed in OWL 2 RL, since the negation in the left part of subclass axioms is forbidden. In order to maintain soundness and completeness of reasoning, this axiom should be dropped, and inferences like the above are therefore compromised.

4 Conducting queries

In this section, we present some of the different types of queries that one can pose through the semantic search service. They are categorised into three groups according to their knowledge retrieval capabilities as compares to traditional search mechanisms. Table 1 summarises indicative examples for each query group.

By traditional search we mean text-based search, i.e. search for the occurrences of particular text strings within available metadata. On the other hand, semantic search operates on additional knowledge that is implied by the combination of these metadata and their semantic interpretations in the ontology. This combination is made possible by inference engines that implement automated reasoning techniques based on formal logics (Horrocks, 2008), which are the essential underpinnings of the Semantic Web.

When a user submits a query, a parser maps the assembled Manchester Syntax expression into a concept expression (class). Consequently, all instances classified by the reasoner under this particular concept are retrieved as query answers.

Group A represents string-based queries that are easy to formulate and conduct also from within the standard repository's searching facilities (simple search, advanced search or browsing). For example, query *A*1 asks for items authored by a person with a specific surname. Examples *A*2 and *A*4 show that conjunctions and string-pattern searches (e.g. wildcards) are also possible.

Group B contains semantic queries which, even though they might be attempted with traditional search, manage to retrieve additional results, because of the several ontology-specific features (like reification of implicit entities and the utilisation of role-chains). For example, query *B*1 obtains all entities that draw sponsorship from the 'European Commission'. The keyword-based counterpart of this query would only return items (and just items), with this particular keyword in their 'sponsorship' metadata field. Semantic search, on the other hand, retrieves also the contributors of these items, aside from the items themselves, due to a role-chain in the ontology that transfers an item's sponsorship also to its contributors.

Table 1 Example types of semantic queries organised in groups

#	*Semantic Query*	*DSpace equivalent*	*Ask for:*
	Group A: Queries also possible through traditional search		
A1	dspace-ont: author some (foaf: surname value "Solomou")	"Solomou" (simple search) Author: "Solomou" (advanced search)	… all items having as author a person with surname "Solomou"
A2	dspace-ont:author some (foaf:surname value "Solomou") and dcterms:type value dspace-ont:Working_Paper	Author: "Solomou" AND Type: "Working Paper" (advanced search)	… all items of type "Working Paper" also having as author a person with surname "Solomou"
A3	dcterms:issued value "2010"	Issue Date (browsing)	… all entities that have been issued during 2010
A4	(HermiT only) dcterms:subject some rdf:PlainLiteral [pattern ".*[sS]emantic.*"]	Subject: "semantic" (advanced search)	…all items that contain "semantic" in their subject
	Group B: Queries that obtain more results		
B1	dspace-ont:sponsorship value dspace-ont:European_Commission	Sponsor: "European Commission" (advanced search)	… all items/authors that are sponsored by a specific institution
B2	dcterms:type value dspace-ont:Presentation	Type: "Presentation" (advanced search)	… all items of type "Presentation"
	Group C: Queries impossible through traditional search		
C1	dcterms:type min 2 dspace-ont:DspaceType		… all items that have been characterised with at least two types (e.g. Article, Book Chapter, …)
C2	dspace-ont:author min 4		… all items having 4 or more authors
C3	dcterms:instructionalMethod value lom:Mixed		… all learning objects that use a mixed instructional method
C4	dcterms:hasPart some (dcterms:type value dspace-ont:Learning_Object)		… all collections that contain learning objects

The most important aspect of semantic search is that it enables unique, entailment-based queries that take advantage of the underlying ontological model. This is outlined by the examples of Group C, which lists some queries that are impossible to express with traditional DSpace search. Queries *C*1 and *C*2 make use of number restrictions, asking for items that have a minimum number of types or contributors, respectively. In addition, query *C*3 is only possible due to the mapping between qualified DC and LOM that occurs during the population of the ontology. Finally, *C*4 is an example of a nested class expression that allows retrieving precisely those collections that contain learning objects.

Limitations of Semantic Search compared to traditional mechanisms include the inability to take advantage of automatic stemming or to express queries with negations on property restrictions. General negation does not hold for semantic search due to the Open Word Assumption (OWA) in the OWL world. Therefore, a query such as *not (dspace-ont:author value dspace-ont: Solomou_Georgia)* would return no results, although this can be easily reproduced with traditional search. A prominent exception is the functional datatype property *dspace-ont:uniqueName*, which can be successfully negated exactly because of the fact that it is functional, i.e. can have only one value.

5 Evaluation

Our evaluation methodology consists of three phases: (a) an end user survey aiming to capture user satisfaction as well as user reaction to system-specific behaviour (b) a series of metrics and statistics about actual queries coming from various logged data and (c) system performance tests under high ontology loads. The following results reflect collected replies and statistics for a six-month period (May–October 2011).

5.1 User reception

In the following, we analyse the results of the user survey.

User replies were collected anonymously using a publicly available online questionnaire (still active at http://goo. gl/Zt0aP) that includes mostly multiple-choice as well as 3- and 5-point Likert scale questions.

The majority of people that took part in this survey appear as being averagely familiar with Semantic Web concepts. As expected, their Manchester Syntax level of knowledge is a bit lower, but still most of them can at least recognise queries written in this language and possibly can construct some (mean average 2.94 in a 5-point scale with 5 corresponding to ‘expert’).

The average percentage of successful queries reached the very promising ratio of 61%. This increases as users become more Manchester Syntax literate (see Table 2).

The percentages have been computed by taking into account the answers given by participants that had actual hands-on experience with the system (85% of all participants).

Nearly half (44%) did not show preference to the ‘generated query’ box (they used it ‘only a few times’) and about one quarter (22%) used exclusively the step-by-step procedure. This suggests that the semantic search interface came in handy for a significant majority of users, especially those inexperienced with the specific-query syntax.

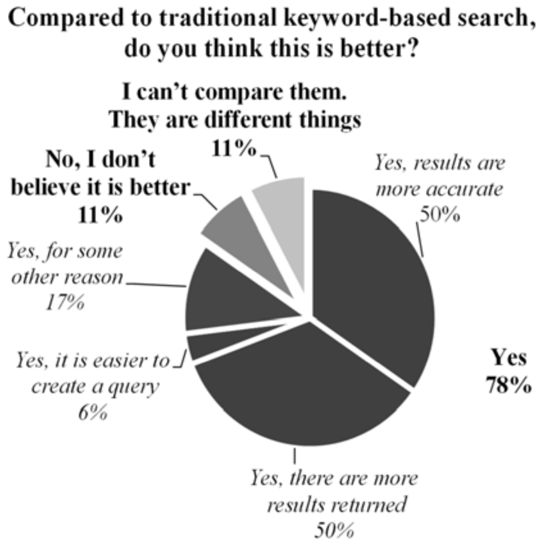
Overall interface intuitiveness has some room for improvement, gaining a mean score equal to 3.28/5, which is kept consistent with user expertise. Auto-complete suggestions is the most liked feature (rated 4/5) and the most expected are SPARQL support as well as friendlier explanations for the interface parts.

Compared to the traditional keyword-based search, semantic search is considered to be ‘better’, as shown by the number of positive votes (78% of the respondents indicated at least one positive reason, see Figure 6). The observation that Semantic Search outclasses traditional search in terms of improved precision and recall comes exactly as a consequence of the service’s capability to perform entailment-based query answering.

Table 2 Survey findings correlated with Manchester syntax expertise level

Manchester syntax expertise	*Percentage of successful queries*	*Did you have to edit the ‘generated query’ box manually? (the two most frequent replies)*		*Overall, how easy was for you to understand and/or use the interface? (1: ‘Very hard to understand and use’ 5: ‘Very easy and straightforward’)*
1–2 (28%)	40%	Never. I was more comfortable creating the query step-by-step.	40%	2.80
		Only a few times.	20%	
3 (39%)	62%	Only a few times.	57%	3.43
		Never. I was more comfortable creating the query step-by-step	29%	
4–5 (33%)	72%	Only a few times.	50%	3.50
		More often than not, it is easier for me this way.	50%	
Total	61%	Only a few times.	44%	3.28
		Never. I was more comfortable creating the query step-by-step.	22%	

Figure 6 Comparison between traditional keyword-based search and the semantic search service, % of respondents. Multiple replies allowed



Finally, the vast majority of users (90%) found that the semantic search interface makes semantic searching 'very much' or at least 'somehow' easier and only 10% disagrees with this viewpoint, answering 'not at all' (total mean average 2.24/3). It is worth mentioning that by collapsing overall interface intuitiveness (Table 2) into a 3-point scale, both these questions have almost the same average score (2.24 for facilitation of semantic searching and 2.17 for overall intuitiveness, out of 3). This indicates that there exists a direct relationship between semantic searching and the user friendliness of the underlying interface and suggests that semantic search can benefit from interface intuitiveness.

5.2 *Query analysis*

Data used below were recorded in the system log files of our publicly available demo installation (http://bit.ly/dspace-ss-demo). The demo runs on a 64 bit Intel Xeon at 2.8 GHz and Java is allowed a maximum heap memory of 1024 MB.

First, Table 3 gives overall statistics about the demo set-up and the queries performed. From a technical point of view, 'successful' queries are those that achieve reasoner response without errors, i.e. succeed in fetching results unabruptly, even zero ones. Query analysis indicates an almost 70% of such queries. Remaining queries failed mostly due to syntax errors and reasoner limitations (unsupported facets and data types). This is in accordance to the findings of the user survey (61% success rate, see Section 5.1), though leaning a little on the positive side by also including queries that may have worked, but did not come up to the user expectations.

Table 3 Query success rates and statistics

Total DSpace Items	50
Total ontology individuals	304
Total ontology triples	2138
Total queries	561
Successful	391 (69.7%)
Unsuccessful	170 (30.3%)
Average query time	405.62 ms
Average number of results	37 (12.2% of total individuals)
Average query length (in characters)	31
Average reasoning time	
With Pallet 2.2.2	430.93 ms
With FaCT++ 1.5.0	56.28 ms

Reasoning time corresponds to inferences precomputation by the reasoner, while query time corresponds to query evaluation, i.e. the time it takes for the reasoner to evaluate the input class expression and to decide its instance membership (what individuals belong to the evaluated class).

These times are essential, since users may be accustomed to rapid response times from their everyday experience with traditional search engines. On the other hand, systems that employ semantics to offer enhanced search services can be somewhat more burdened. A recent study on the evaluation of such systems reports average response times between a few *ms* and several seconds (Elbedweihy et al., 2012). In this setting, the fact that Semantic Search for DSpace exhibits average query (and reasoning) times of less than half a second can be considered satisfactory.

Average query length is 31 characters (Table 3), which is larger than usual web search queries. For example, average query length in Google is around 20 characters (Tsotsis, 2010) and similarly in Bing, new queries appear to be almost 19 characters long (Tyler and Teevan, 2010). This is due to the fact that Semantic Search offers several enhancements that reduce users' workload, such as prefix auto-completion, pull-down menus and auto-complete suggestions for entity names, all making easier for users to construct longer and expressive queries.

Figure 7 partitions logged queries into groups based on their types and characteristics and shows how the number of results returned as well as query evaluation times are directly dependent on these query types. This partitioning is explained in Table 4, where indicative examples are also given. The last column includes the correspondence of the query examples in Section 4 (Table 1) with these query types.

Figure 7 Distribution of queries per type and their average performance and results

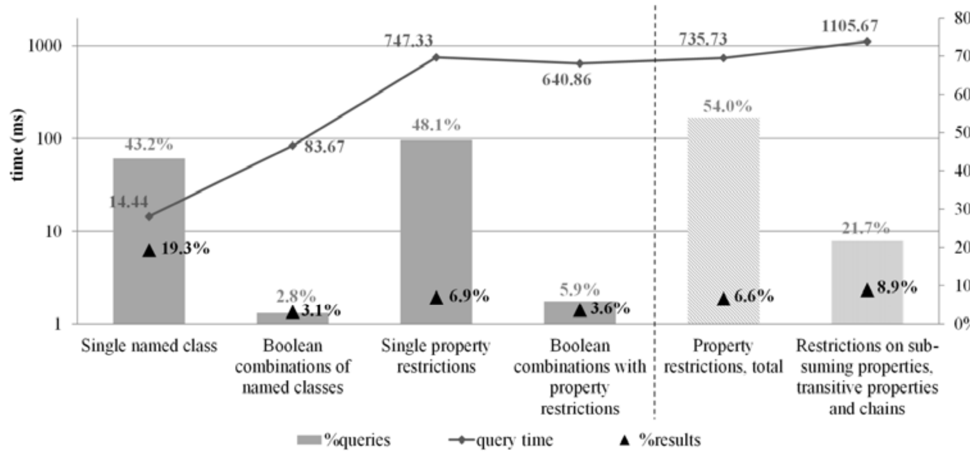


Table 4 Query types according to their structural elements

#	*Query Type*	*Example/Description*	*Matching examples of retrieval capabilities*
T1	Single named class	e.g. 'dspace-ont:Item'	
T2	Boolean combinations of named classes	e.g. 'dspace-ont:Item or dspace-ont:Collection	
T3	Single property restrictions	e.g. 'dcterms:type value dspace-ont:Article'	A1, A2, A3 B1, B2 C1, C2, C3, C4
T4	Boolean combinations with property restrictions	e.g. 'dspace-ont:Item and dcterms:type value dspace-ont:Article'	A2
T5	Total property restrictions	Aggregates the two above categories, i.e. all queries that involve a property restriction	
T6	Restrictions on subsuming properties, transitive properties and chains	Special case regarding the following properties: dspace-ont:sponsorship, dspace-ont:co_author, dcterms:relation, dcterms:description, dcterms:contributor dcterms:hasPart dcterms:isPartOf	B1 C4

Clearly, since a precomputation has preceded, query evaluation may range from very low (practically insignificant) times to several milliseconds, depending on whether query results are already in memory or a recomputation is required, as is often the case with property restrictions and certain combinations thereof. This is especially true in cases where properties subsume long hierarchies of other properties or role chains (with transitivity being a special case of chain), since the corresponding DL algorithms have to investigate exhaustively all possible paths connecting instances, based on the role hierarchy.

It should be noted that the syntactic characteristics of queries, as they are generally described in Table 4 (*T*1–*T*5), do not have any direct effect on their knowledge retrieval capabilities. For example, most examples in Section 4 are concentrated within *T*3 (property restrictions) regardless of their grouping, and there is hardly any observable relationship between types and groups. Instead, it is the semantic qualities of the vocabulary and expressions used within the queries that determine their ability to perform knowledge acquisition.

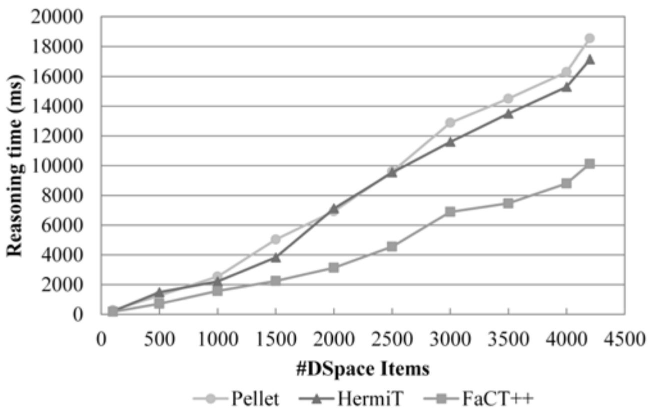
A notable exception is *T*6 (last row of Table 4), which actually considers semantically 'intense' properties, due to their manipulation in the ontology. Queries of this type are expected to exhibit improved retrieval capabilities, by design; however, the opposite does not necessarily hold.

More than half of user queries (54%) consist of property restrictions. However, these queries resulted on average in a small percent (6.6%) of total individuals. On the other hand, the simpler single-named class expressions are somewhat less frequent, but are accountable for a far greater number of results (19.3%). This confirms that users were able to take advantage of the new features of the interface and construct also complex and 'educated' queries filtering effectively the result space. Besides, more than half of the users reported that only rarely did they opt for manual query construction and liked better to utilise the guided query mechanism, according to the user survey (Section 5.1).

5.3 Scalability

To test scalability we also experimented with a series of larger data sets. These are in fact ontologies containing far more DSpace items than our demo installation and correspond to real data from the University of Patras theses archive (4200 items total). We measure reasoning time, defined as in Section 5.2, for various ontology sizes incrementally, at a step of 500 items, ranging from 23 K to 200 K RDF triples. Measurements were performed on a 64 bit Intel Quad Core@ 2.40 GHz with 2048 MB available heap space for Java. Figure 8 summarises test results.

Figure 8 Reasoning times of semantic search using various reasoners and incremental ontology data sets



It goes without saying that scalability of semantic search purely depends on the scalability of the underlying reasoning mechanism. At least for FaCT++, it appears that this approach scales well even for several thousands of triples. It should be noted though that precomputation cannot guarantee (equal) speed-up in all cases, as this depends greatly on the type of reasoner used and, to a lesser extent, on the query itself. For example, FaCT++ after precomputation demonstrates almost insignificant query times, irrelevant of ontology size or query type, in stark contrast to the other reasoners that can take up to several thousands of milliseconds for certain types of queries.

Still, precomputing reasoning results is clearly a major optimisation and is feasible in cases of repositories that exhibit slow but steady growth. In other cases, a more viable solution has to be sought and we are already experimenting with triple store mechanisms as a possible alternative, for example, OWLim. However, since current ontology expressivity falls within the full set of OWL 2, the trade-off between efficiency and expressive power needs to be carefully evaluated before such a migration takes place.

6 Conclusions

Semantic Search for DSpace is under ongoing development as an open-source Google Code project (http://code.google.com/p/dspace-semantic-search/) and is periodically updated to include fixes and new features as well as to reflect compatibility with latest DSpace versions. The system at its current stage is fully operational and is listed as an official DSpace add-on (http://www.dspace.org/addons/#semantic).

In accordance to the evaluation results, next steps will focus on UI fine-tuning by including advanced Manchester Syntax constructs like facets and inverse properties; tighter integration with the Linked Data cloud, by assigning resolvable IRIs to entities; adding support for expressive queries in other syntaxes as well, e.g. SPARQL-DL (Sirin and Parsia, 2009); and considering more scalable architecture alternatives, like the ones based on entailment-capable triple stores.

The open architecture of Semantic Search combined with the fact that it remains agnostic to the rest of the DSpace business logicor even to the underlying ontology implies that this approach can equally be reused in other repository configurations or libraries. Indeed, Semantic Search can accommodate any web ontology document (the *Options* tab) and simply use DSpace as a wrapper UI.

The overall methodology followed for equipping DSpace with a semantic search mechanism is built on interoperability principles and may serve as a generic paradigm for future development of digital libraries. Although it can by no means be considered as a replacement to traditional search, this approach offers added-value that appears to be well-appreciated by end users. In addition to experts, the structured query mechanism and the other enhancements introduced with v2 is a step forward towards improving its appeal to everyday users too.

Acknowledgements

In loving memory of Prof. Theodore S. Papatheodorou, who passed away unexpectedly during preparation of this paper. His inspiring motivation and guidance will always be remembered.

References

Elbedweihy, K., Wrigley, S.N., Ciravegna, F., Reinhard, D. and Bernstein, A. (2012) 'Evaluating semantic search systems to identify future directions of research', Proceedings of 2nd International Workshop on Evaluation of Semantic Technologies, Vol. 843, pp.25–36.

Grau, B.C., Halaschek-Wiener, C. and Kazakov, Y. (2007) 'History matters: incremental ontology reasoning using modules', Proceedings of the 6thInternational Semantic Web Conference (ISWC2007), pp.183–196.

Horridge, M. and Bechhofer, S. (2009) 'The OWL API: a Java API for working with OWL 2 ontologies', Proceeding of 6th OWL Experienced and Directions Workshop (OWLED 2009), 23–24 October, Chantilly, VA, USA.

Horridge, M. and Patel-Schneider, P.F. (2012) OWL 2 Web Ontology Language Manchester Syntax, 2nd ed., W3C Working Group Note. Available online at: http://www.w3.org/TR/owl2-manchester-syntax/

Horrocks, I. (2008) 'Ontologies and the Semantic Web', Communications of the ACM, Vol. 51, No. 12, pp.58–67.

IEEE LTSC (2002) Draft Standard for Learning Object Metadata, IEEE 1484.12.1-2002. Available online at: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

IEEE LTSC (2008) Draft Recommended Practice for Expressing IEEE Learning Object Metadata Instances Using the Dublin Core Abstract Model, IEEE P1484.12.4/D1. Available online at: http://dublincore.org/educationwiki/DCMIIEEELTSCTask for ce?action=AttachFile&do=get&target=LOM-DCAM-new draft.pdf

Koutsomitropoulos, D.A., Alexopoulos, A.D., Solomou, G.D. and Papatheodorou, T.S. (2010a) 'The use of metadata for educational resources in digital repositories: practices and perspectives', D-Lib Magazine, Vol. 16, No. 1.

Koutsomitropoulos, D.A., Borillo Domenech, R. and Solomou, G.D. (2011) 'A structured semantic query interface for reasoning-based search and retrieval', Proceedings of 8thExtended Semantic Web Conference (ESWC 2011), Part I, pp.17–31.

Koutsomitropoulos, D.A., Solomou, G.S., Alexopoulos, A.D. and Papatheodorou, T.S. (2010b) 'Semantic Web enabled digital repositories', International Journal on Digital Libraries, Vol. 10, No. 4, pp.179–199.

Motik, B., Grau, B.C., Horroks, I., Wu, Z., Fokoue, A. and Lutz, C. (Eds) (2012) OWL 2 Web Ontology Language Profiles, 2nd ed., W3C Recommendation. Available online at: http://www.w3.org/TR/owl2-profiles/#OWL_2_RL

Motik, B., Parsia, B. and Patel-Schneider, P.F. (Eds) (2010) OWL 2 Web Ontology LanguageXML Serialization, 2nd ed., W3C Recommendation. Available online at: http://www.w3.org/TR/owl2-xml-serialization/

Nardi, D. and Brachman, R.J. (2002) 'An introduction to description logics', in Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D. and Patel-Schneider, P.F. (Eds): Description Logic Handbook, Cambridge University Press, Cambridge.

Sirin, E. and Parsia, B. (2009) 'SPARQL-DL: SPARQL Query for OWL-DL', Proceedings of 3rd OWL Experiences and Directions Workshop (OWLED 2007), 6–7 June, Innsbruck, Austria.

Tsotsis, A. (2010) 'Google instant will save 350 million hours of user time per year', TechCrunch. Available online at: http://techcrunch.com/2010/09/08/instant-time/

Tyler, S.K. and Teevan, J. (2010) 'Large scale query log analysis of re-finding', Proceedings of 3rd ACM International Conference on Web Search and Data Mining (WSDM'10), pp.191–200.

Uren, V., Sabou, M., Motta, E., Fernandez, M., Lopez, V. and Lei, Y. (2011) 'Reflections on five years of evaluating semantic search systems', International Journal of Metadata, Semantics and Ontologies, Vol. 5, No. 2, pp.87–98.

Wrigley, S., Reinhard, D., Elbedweihy, K., Bernstein, A. and Ciravegna, F. (2010) 'Methodology and campaign design for the evaluation of semantic search tools', Proceedings of the Semantic Search 2010 Workshop (SemSearch 2010), ACM, New York, NY, USA.