Ontology-driven Keyword-based Search on Linked Data

Carlos Bobed, Guillermo Esteban, and Eduardo Mena

IIS Department
University of Zaragoza
50018 Zaragoza, Spain
{cbobed,529679,emena}@unizar.es

Abstract. Nowadays, the Web is experiencing a continuous change that is leading to the realization of the Semantic Web. Initiatives such as Linked Data have made a huge amount of structured information publicly available, encouraging the rest of the Internet community to tag their resources with it. Unfortunately, the amount of interlinked domains and information is so big that handling it efficiently has become really difficult for the final users. DBPedia, one of the biggest and most important Linked Data repositories, is a perfect example of this issue.
In this paper, we propose an approach to provide the users with different domain views on a general data repository, allowing them to perform both keyword and navigational searches. Our system exploits the knowledge stored in ontologies to 1) perform efficient keyword searches over a specified domain, and 2) refine the user's domain searches. We focus on the case of DBPedia, as it mirrors the information stored in the Wikipedia, providing a semantic entry to it.

Keywords: Ontologies, Linked Data, Semantic Web, Keyword Search

1 Introduction

The Web has made a huge and ever-growing amount of information available to its users. The appearance of new interaction paradigms that the so-called Web 2.0 introduced, in which the Web users become part of the information providers, has made this amount even bigger and more difficult to handle. At this point, the Semantic Web [15] has been proposed in order to relieve the user of the burden of processing the available information. By sharing definitions and tagging resources, the Internet is being made understandable to computers, thus, allowing them to process the information on behalf of users.

This progressive structuring is being made using ontologies (which offer a formal, explicit specification of a shared conceptualization [9]) and a set of different technologies built around them, promoted and supported by the W3C [1]. Initiatives such as Linked Data [5] have already made a huge amount of structured information publicly available, providing schemas and data in machine-readable

[1] `http://www.w3.org`

formats; and, now it is the time to exploit all these information. Unfortunately, it is not realistic to expect that all the resources in the Internet will be perfectly described and annotated, as there are also huge amounts of information that are not semantically annotated as well. So, for the time being, both dimensions (*semantic* and *syntactic* ones) of the Web are condemned to coexist with each other. And so their different techniques and methods are.

In this context, users have become used to keyword-based search interfaces due to their ease of use. However, this ease of use comes from the simplicity of its query model, whose expressivity is low compared with other more complex query models [13]. One possible approach to augment the expressivity while maintaining the ease of use of this query model is the *keyword query interpretation* [8], which is the process to translate them into a structured query. However, the current methods require highly expressive ontologies [7] or building large graphs out of the underlying data [16], which might not be completely available to us (e.g.: we only have access to a single data endpoint to which pose our queries).

In this paper, we present a system that adopts a hybrid search strategy which exploits the knowledge stored in ontologies to focus and enrich the search process on structured data. Our system builds on an external Linked Data repository (which might not be under our control) and takes as input input an ontology which has two roles in the system: 1) to define the taxonomy of the search domain, guiding and narrowing the scope of the keyword-based search; and 2) to define the structure of the objects in the search domain, helping refining and suggesting further search results. With our approach, one can provide different views on a general data repository by just adapting externally the ontology provided. Moreover, our approach can be attached to any public SPARQL endpoint without overloading it (this is important in open scenarios, such as the one depicted by Linked Data). We use the DBPedia [6] as data repository example as it provides us with a semantic entrance to the Wikipedia[2].

The rest of the paper is as follows. In Section 2, we overview the architecture of our search system. The definition of the search domain using an ontology and how we do apply it to the case of DBPedia is explained in Section 3. In Section 4, we focus on how our system uses the ontology to focus the search. Some related work is discussed in Section 5. Finally, the conclusions and future work are drawn in Section 6.

2 Architecture of the system

In this Section, we overview the architecture that allows our system to exploit the information in the external Linked Data repository. As shown in Figure 1, there is a previous offline step that consists of defining the search domain and providing it modeled in an ontology. Our system uses an inner Description Logics reasoner [3] to exploit the information in this ontology. Once it has the domain ontology, our system offers two different but complementary kinds of search depending on the user's input:

[2] http://wikipedia.org

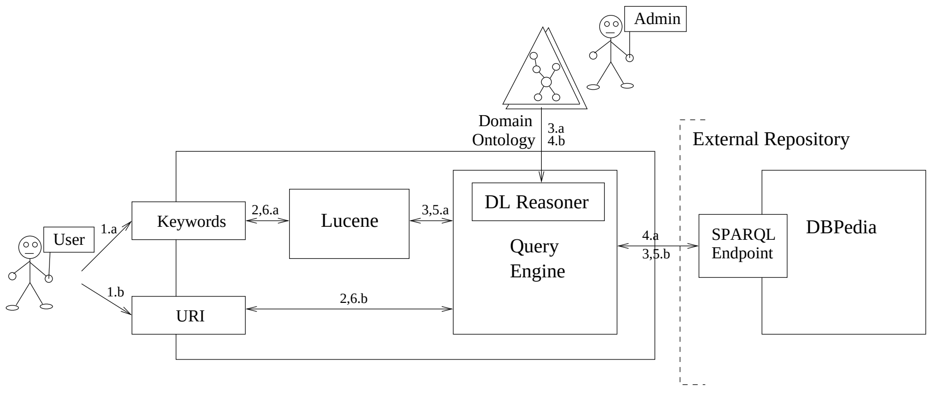

Fig. 1. Our system provides two complementary search services: a) Keyword-based and b) URI refining services.

a) *Keyword-based Search.* This search service takes as input plain keywords (step 1.a) and performs a first search on a Lucene repository (step 2.a). This intermediate storage serves as a cache to alleviate the workload of the external public endpoint (which is not under our control and might have limited availability). If the search has not been performed before, the keyword query is forwarded to our Query Engine (step 3.a) which consults the taxonomy of the ontology to build a focused SPARQL query. This taxonomy includes only the objects we want to be searched, this is, the objects that we define in the search domain. If it is too large, there exists the possibility of specifying a class of the domain ontology to serve as top node of the focused search. When the results are retrieved, they are stored in our Lucene repository to cache them for future searches (step 5.a). The Lucene repository acts a cache for future searches and provides the system with relevance measures and ranking on the results. Finally, the results (a ranked set of URIs) are returned ranked according to their relevance (step 6.a).

b) *URI Refining Search.* The results of the previous search service is a ranked set of URIs, which are presented to the user so s/he can explore them. When the user selects an URI (step 1.b), it is directly forwarded to the Query Engine (step 2.b). The Query Engine consults the data repository to obtain the type of the object behind the URI (step 3.b). Then, it consults the definition of its type (step 4.b) to build a specialized query for that type of object (it consults the relevant properties[3] to retrieve the appropriate data and suggest other related objects) with the help of a DL reasoner. Finally, it forwards the query to the data endpoint (step 5.b) and returns the data (step 6.b). This step is not cached as it is a more specialized query that is not so time consuming as a general search as a keyword query over the whole domain.

[3] They are marked as being relevant during the ontology definition, so the Query Engine can be aware of them.

Notice that the Lucene repository is only used to cache and rank the results obtained from the actual query on the external Linked Data repository. In the following sections, we explain how the search domain has to be defined, how we have applied this architecture to the use case of the DBPedia, and we illustrate and further detail each of the searches.

3 Defining the Search Domain

In this section, we firstly explain how to define the search domain by providing an annotated ontology, and then we present how we have applied it to the use case of the DBPedia.

3.1 Domain Ontology Annotation

First of all, the system has to be provided with an ontology that defines the search domain we want to present to the users. This ontology provides a view on the data and has to be aligned to the ontology that describes the actual data repository. This way our system can consider only part of the data while being able to access it properly. In fact, although it can be built from scratch, we advocate for using ontology extraction techniques [12] to obtain a module and, then, make our system work directly with a subontology of the repository's one. The definition of the search domain is based on three main aspects:

– The *taxonomy* we define in the domain ontology contains the objects that are considered by our system in the searches. The size of this taxonomy is not a problem, as our system can receive an extra parameter to consider any concept as top node and thus focus the search only on its descendants.
– The *object properties* defined in the search domain play a crucial role as they are exploited in the refining step to further propose relevant results. We have to provide the relevant properties (the ones that the Query Engine takes into account) by marking them with an @relevantProperty annotation.
– Finally, the *keyword-searchable* properties of the objects, i.e., the properties that have values that can be processed to perform keyword searches. We annotate them as @kwdSearchField. These properties are the ones that are considered for the keyword search.

The Query Engine exploits this information to build scoped queries on the structured data. Depending on the underlying repository, we can specify via the annotations the different fields which we can perform the searches on and the objects that are relevant to the domain view we want to offer.

3.2 Structure of DBPedia

In this section, we present the inner organization of DBPedia resources to explain how our system handles it. DBPedia [6] is a project that extracts structured data from Wikipedia, and makes this information available on the Web under the principles of Linked Data. This extraction is performed automatically by exploiting the structure of the information stored in Wikipedia. However, the nature of the results of this extraction process differs from the sources in more ways than barely structural and format ones.

When moving from the *article world* of Wikipedia to the *semantic resources* in DBPedia, there are objects that might augment their descriptions as the new semantic model can represent more information about them. The articles extracted from Wikipedia, once in DBPedia, become *resources*. Each resource is represented by an URI and has a direct correspondence to its original Wikipedia's article, inheriting its categorization. The whole taxonomy of article categories of Wikipedia is included as an SKOS[4] ontology in DBPedia; thus, DBPedia provides a first view on the resources according their category.

Depending on the content of its corresponding article, a DBPedia resource might also be representing an object (see Figure 2). The classification of this object dimension of the resources is done via several general domain ontologies, being DBPedia Ontology[5] and YAGO[6] the most important ones. In this way, independently of the article categorization, DBPedia offers a second different view based on the nature of the underlying resources. However, this view does not cover all the DBPedia. There exist resources that, despite being categorized, do not have these descriptions as they are not defined in the used ontologies, as shown in Figure 2.

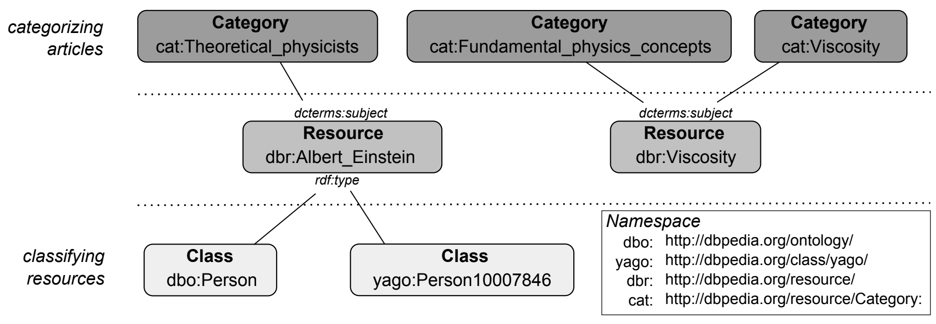
categorizing articles
Category cat:Theoretical_physicists
Category cat:Fundamental_physics_concepts
Category cat:Viscosity
dcterms:subject
dcterms:subject
Resource dbr:Albert_Einstein
Resource dbr:Viscosity
rdf:type
classifying resources
Class dbo:Person
Class yago:Person10007846
Namespace
dbo: http://dbpedia.org/ontology/
yago: http://dbpedia.org/class/yago/
dbr: http://dbpedia.org/resource/
cat: http://dbpedia.org/resource/Category:

Fig. 2. DBPedia excerpt of the descriptions of *Albert Einstein* and *Viscosity* resources.

Summing up, DBPedia organizes knowledge in two major ways: the SKOS categorization, and an ontological classification. In our system, when working with the DBPedia as underlying data repository, we have considered the SKOS categorization as the general taxonomy to be pruned to focus the keyword search. On the other hand, we obtain the ontological definitions from the DBPedia ontology for the URI refining search. This is done to show the flexibility of our approach as the search domain definition is completely decoupled from the actual process search (the domain ontology in Figure 1 can be separated to focus on different aspects from the object definitions). In particular, in our prototype, we have pruned the SKOS categorization to deal with the categories under

[4] SKOS Simple Knowledge Organization, `http://www.w3.org/2004/02/skos/`

[5] The DBPedia Ontology, `http://wiki.dbpedia.org/Ontology`

[6] YAGO Ontology, `http://www.mpi-inf.mpg.de/yago-naga/yago/`

'Mechanics', and we are only interested, for further refinement, in people, institutions and the different articles that could be related to each resource. The keyword search is performed on the *abstract* property, which gives an excerpt of the Wikipedia entry associated to each resource.

4 Searching on Structured Data

Once we have provided the annotated ontology, our system is able to perform two types of searches on the external Linked Data repostory: a keyword search and a URI refining search. Both searches exploit the ontological definition of the domain we have provided but in different ways, as we detail in the rest of the section.

4.1 Keyword Search

When the user poses a keyword query, the Query Engine consults the domain ontology to obtain information about:

- The properties on which it has to perform the keyword search. Due to the structure of Linked Data (it follows the RDF model based on triplets) the system would have to look for the keywords in every element of the triplets (subject, property, and value) if we do not specify which properties to search on. This would lead to unbearable query processing times (note that our system builds on public endpoints which data are not under our control).
- The taxonomy of the objects that are to be looked up. Exploiting the semantic structure of the Linked Data, the Query Engine is able to focus the search only on the objects that are relevant to the defined domain. Thus, our system pre-filters the triplet values to be checked against the keyword query.

With this information, the Query Engine is able to build a SPARQL query to be posed to the public endpoint. To perform the actual keyword search via SPARQL, the system uses the *regex* function to look for such keywords in the property values. In our example on the DBPedia, the following SPARQL query defines the structure of the queries posed to the DBPedia's SPARQL endpoint:

```
SELECT ?uri ?abstract ?web
 WHERE ?uri dbo:abstract ?abstract
       ?uri foaf:page ?web
 FILTER regex(?abstract, [keyword])
```

This query firstly matches all the resources that have *abstract* and *page* linked to it, and then, filters the results by looking for the keywords in the annotated property (in this case, *abstract*). A query like that posed to the whole dataset would be too time consuming. Thus, the system uses the taxonomy to limit the possible candidate resources by forcing them to be instances of any of the objects included in it. Querying a smaller set of resources will provide more accurate results according to the selected knowledge context. This limited search space will also impact the performance of the keyword search query, reducing the time required.

The properties that define the taxonomy depends on the modeling language that has been selected to express the ontology. In OWL, the main property is the *is_a* property (subclassOf), while in SKOS the taxonomy is modeled via the *broader* and *narrower* properties. So, depending on the modeling language, our system adopts one or another to build the constraints on the query. In our example, to limit the search scope to a SKOS category, our system adds the following constraint to the query:

```
WHERE ?subCategories skos:broader      [chosenCategory]
      ?uri           dcterms:subject   [chosenCategory]
      ?uri           dcterms:subject   ?subCategories
```

This constraint reduces the set of resources to be checked to only the ones that are in the domain search, which are a strict subset of the whole dataset. Without loss of generality, and if the taxonomy is quite big, the system can accept a category/concept of the domain search to provide a more focused search scope.

The result of the built query is a set of tuples $< URI, \{kwdSearchField_i\} >$ that have to be presented to the user. When the query results are retrieved, they are inserted on a local Lucene repository along with the keyword query that has led to them. We let Lucene index the results according to the retrieved values of the properties we marked in the ontology for the keyword search. Our system benefits from this step in two ways:

– On the one hand, it obtains a ranked set of resources that have been retrieved due to their semantic belonging to the search domain.
– On the other hand, the Lucene repository serves as a cache for further queries, alleviating the dependence on the processing resources of the public endpoints (which availability might be even compromised).

So, in the end, our system provides the user with a ranked set of semantically related resources, taking advantage of both semantic knowledge and information retrieval techniques. For example, for the input '*fish movement*', our system retrieves the following results: 1) tripedalism, 2) fish locomotion, 3) role of skin in locomotion, 4) aquatic locomotion, 5) dynamical system, . . . ; while performing the same search directly on Wikipedia, it returns: 1) fish migration, 2) lateral line, 3) Murray cod, 4) fishing lure, 5) Bear Island, ... Note the difference: our system focused on the domain '*Mechanics*', as we defined, and only returns resources within that defined domain, while Wikipedia always considers any domain in the search.

4.2 URI Refining Search

The result of the previous keyword search is a set of semantic resources, identified each one by an URI. Once the user selects a resource, the system performs a URI refinement to suggest further related results. This refinement is performed via the properties that have been annotated as relevant in the domain search definition.

Firstly, the Query Engine asks the underlying repository for the concept of the resource to know its defined properties. Then, it consults the ontology to obtain the relevant properties[7] that are compatible with its definition. Once the system has the definition of the properties, it retrieves their values for the input resource. In the meantime, the system builds property compositions[8] that are suggested as possible queries to retrieve further results (they are not directly posed to the underlying system, but only on user's demand). All the semantic checkings of the domains and ranges of the involved properties are performed with the help of a Description Logics reasoner [3], to detect possible inconsistent queries (according to our domain ontology).

The results of this kind of search lead to more resources that can be navigated again and so on. This way, our system keeps on providing results that are related to the resource without leaving the search domain. Following the previous example, the user could select '*dynamical system*' out of the resources that the keyword search had retrieved. The results of the refinement will depend on how the domain ontology was defined. In this case, we define two properties for articles: *knownFor* and *subject*, that represent people related to the article and the categories where the article is included. Now, the refined results include the following people: Krystyna Kuperberg, Jean-Christophe Yoccoz, Yakov G. Sinai, Denis Blackmore, Bill Parry (mathematician), John Guckenheimer, . . . ; and categories: Systems, DynamicalSystems, and SystemsTheory.

5 Related Work

When it comes to accessing semantic data from a set of keywords, there are quite a lot different approaches, but most of them start with query building step which translates the input into an structured query [8, 14, 16, 7]. In [14], the input is matched to semantic entities by means of text indexes, and then a set of predefined templates is used to interpret the queries in the language SeRQL. However, in this system, the user has to be aware of the underlying data schema to be able to query as, at least, one of the keywords has to be matched to a class in the ontology that describes the underlying data. Moreover, the search domain and the properties to be used cannot be adapted as it can be done in our system.

Other relevant systems in the area of semantic search are SemSearchPro [16], Q2Semantic [10], SPARK [18], and QUICK [17]. These systems find all the paths that can be derived from a RDF graph, until a predefined depth, to generate the queries. In [8] they propose a similar approach to keyword interpretation but they introduce the context of the user's search (the knowledge about previous queries) to focus the whole search process. However, all of these approaches assume that the data sources are under their control and can pre-calculate complex graphs to perform the query translation and, finally, access the data. Moreover, as they work on the data level, they do not take into account the flexibility that using and adapting the describing ontology provides as we do.

[7] Note that the domain ontology is mapped to the underlying repository, so we assume that we have the information needed to perform the proper translation.

[8] The length of the path is restricted to avoid infinite loops; it has been defined as a system configuration parameter.

There are also some works in the area of databases to provide a keyword-based interface for databases, such as BANKS [1], DISCOVER [11] and DBXplore [2], which translate a set of keywords into SQL queries. However, as emphasized in [4], most of these works only rely on extensional knowledge obtained by applying IR-retrieval techniques, and so they do not consider the intensional knowledge (the structural knowledge). So, again, they have to have the data sources under their control, thus restricting the application in an open scenario such as the Linked Data one.

6 Conclusions and Future Work

In this paper, we have presented a system that enables a hybrid search approach based on keywords and guided by ontologies. Our system combines the ease of use of keyword search with the benefits of exploiting the structure of the underlying data in an efficient way. In particular, our system:

– Provides a keyword-based search guided and focused by the domain ontology, avoiding queries that would be too time-consuming. To do so, it uses the information of the taxonomy of the search domain. This process is highly configurable, as the search can be restricted to a set of specified properties via annotations.
– Exploits the definition of the objects in the domain to suggest further close-related results, refining the search within the domain. This is done with the help of a DL reasoner, that performs all the semantic checkings needed to avoid inconsistent queries.
– Can be built on third parties' Linked Data repositories without overloading them, while allowing to provide the user with different domain views. Our system manages the ontologies as views on the underlying data, decoupling them and processing the results in more flexible ways.

As future work, we are planning to include crossed-domains searches, that is, to include inter-domain relationships in the search, while keeping the adopted hybrid strategy. We also want to perform tests with different kinds of final users to measure the semantic accuracy of our prototype.

Acknowledgments

This work has been supported by the CICYT projects TIN2010-21387-C02-02 and TIN2011-24660. We want to thank Francisco J. Serón for his contributions during the design and development of the system.

References

1. B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, Parag, and S. Sudarshan. BANKS: Browsing and keyword searching in relational databases. In *Proc. of 28th Intl. Conf. on Very Large Data Bases (VLDB'02), China.* Morgan Kauffman, 2002.
2. S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A system for keyword-based search over relational databases. In *Proc. of 18th Intl. Conf. on Data Engineering (ICDE'02), USA*. IEEE, 2002.

3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Pastel-Scheneider. *The Description Logic Handbook. Theory, Implementation and Applications.* Cambridge University Press, 2003.
4. S. Bergamaschi, E. Domnori, F. Guerra, R. Trillo-Lado, and Y. Velegrakis. Keyword search over relational databases: a metadata approach. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data (SIGMOD'11), Greece.* ACM, 2011.
5. C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
6. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154 – 165, 2009.
7. C. Bobed, R. Trillo, E. Mena, and S. Ilarri. From keywords to queries: Discovering the user's intended meaning. In *Proc. of 11th Intl. Conf. on Web Information System Engineering (WISE'10), China.* Springer, 2010.
8. H. Fu and K. Anyanwu. Effectively interpreting keyword queries on rdf databases with a rear view. In *Proc. of 10th Intl. Semantic Web Conference (ISWC'11), Germany.* Springer, 2011.
9. T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation.* Kluwer Academic Publishers, 1993.
10. Q. L. Haofen Wang, Kang Zhang, D. T. Tran, and Y. Yu. Q2semantic: A lightweight keyword interface to semantic search. In *Proc. of 5th European Semantic Web Conference (ESWC'08), Spain.* Springer, 2008.
11. V. Hristidis and Y. Papakonstantinou. DISCOVER: Keyword search in relational databases. In *Proc. of 28th Intl. Conf. on Very Large Data Bases (VLDB'02), China.* Morgan Kauffman, 2002.
12. E. Jimenez, B. Cuenca, U. Sattler, T. Schneider, and R. Berlanga. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *Proc. of 5th European Semantic Web Conference (ESWC'08), Spain.* Springer, 2008.
13. E. Kaufmann and A. Bernstein. Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):377 – 393, 2010.
14. Y. Lei, V. S. Uren, and E. Motta. SemSearch: A search engine for the semantic web. In *Proc. of 15th Intl. Conf. on Knowledge Engineering and Knowledge Management (EKAW'06), Czech Republic.* Springer, 2006.
15. N. Shadbolt, W. Hall, and T. Berners-Lee. The semantic web revisited. *Intelligent Systems, IEEE*, 21(3):96 –101, jan.-feb. 2006.
16. T. Tran, D. M. Herzig, and G. Ladwig. Semsearchpro: Using semantics throughout the search process. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):349 – 364, 2011.
17. G. Zenz, X. Zhou, E. Minack, W. Siberski, and W. Nejdl. From keywords to semantic queries–incremental query construction on the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):166–176, 2009.
18. Q. Zhou, C. Wang, M. Xiong, H. Wang, and Y. Yu. SPARK: Adapting keyword query to semantic search. In *Proc. of 6th Intl. Semantic Web Conference (ISWC'07), South Korea.* Springer, 2007.