WATSON: SUPPORTING NEXT GENERATION SEMANTIC WEB APPLICATIONS[1]

Mathieu d'Aquin, Claudio Baldassarre, Laurian Gridinoc,
Marta Sabou, Sofia Angeletou, Enrico Motta

Knowledge Media Institute, the Open University
Walton Hall, Milton Keynes, UK

ABSTRACT

Watson is a gateway to the Semantic Web: it collects, analyzes and gives access to ontologies and semantic data available online. Its objective is to support the development of next generation Semantic Web applications that dynamically select, combine and exploit the knowledge published on the Semantic Web. We present the design of Watson, which have been guided by the requirements of these applications and by lessons learnt from previous systems. In particular, the three main activities handled by Watson are detailed: crawling the Semantic Web, analyzing and indexing semantic data and ontologies, and providing efficient access to this data to web users and applications. In addition, some novel Semantic Web applications built thanks to Watson are briefly presented.

KEYWORDS

Semantic Web, gateway, search engine, ontologies, RDF, query.

1. INTRODUCTION: NEXT GENERATION SEMANTIC WEB APPLICATIONS

The vision of a Semantic Web, "an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [2], is becoming more and more a reality. Technologies like RDF and OWL, allowing to represent ontologies and information in a formal, machine understandable way are now well established. More importantly, the amount of knowledge published on the Semantic Web – i.e, the number of ontologies and semantic documents available online – is rapidly increasing, reaching the critical mass enabling the vision of a truly large scale, distributed and heterogeneous web of knowledge.

As a consequence, significant changes have already appeared in the way recent Semantic Web applications are designed. First, they assume the existence of large scale, distributed markup that they can use, whereas earlier applications had to engineer the semantic data before using it. Another observation is that, while their ancestors integrated heterogeneous data sets under a common ontology at design time, newer applications tend to dynamically exploit the heterogeneity of semantic markup authored in terms of multiple ontologies.

The evolution of two KMi applications clearly demonstrates these new trends. The first one, AquaLog [9] is an ontology-based question answering system that derives answers to questions asked in natural language by exploiting an underlying ontology. The second application, Magpie [7], is a semantic browser which assists users while they surf the Web, by highlighting instances of chosen concepts in the current Web page. To achieve this functionality, Magpie relies on an internal instantiated ontology. In both tools, the user manually selects the employed ontology and, while it can easily be changed, only one ontology can be exploited at a time. The new generations of these applications aim to overcome this limitation by

[1] This work was funded by the Open Knowledge and NeOn projects sponsored under EC grant numbers IST-FF6-027253 and IST-FF6-027595.

exploiting the wealth of online semantic data. The goal is to dynamically find and combine the relevant knowledge among online ontologies and semantic data, allowing cross-domain question answering in PowerAqua [10] (the successor of AquaLog), and an extended coverage of the semantic browsing with Magpie.

This idea of exploiting the Semantic Web as a large source of background knowledge also appeared recently in several papers concerning generic ontology engineering tasks. For example, in [8], the authors propose a method based on multiple, automatically selected ontologies to disambiguate the senses of the keywords used in a search engine (e.g., in "astronomy star planet", star is used in its sense of celestial body). In [12] we explored the use of online available ontologies as background knowledge for ontology matching. Our implementation identifies and combines relevant knowledge from online ontologies at run time, providing support for those scenarios where the identification of an adequate ontology is not possible at design time.

The success of these new applications obviously relies on the availability of a large amount of semantic data, but also requires an infrastructure for collecting, indexing and providing access to semantic data and ontologies on the Web. Hence, a gateway to the Semantic Web, an efficient entry point to online knowledge, is needed. The previously described applications use Swoogle, the state of the art Semantic Web search engine [6], but the design of Swoogle adopts a "Web centric" approach, leading to key limitations considering the perspective of a Semantic Web gateway.

2. WATSON: BEYOND THE WEB VIEW

The idea of providing an efficient and easy access to the Semantic Web is not new. Indeed, there have been several research efforts that have either considered the task as a whole or have concentrated on some of its sub-issues. Probably the most popular and the most advanced system is Swoogle [6], a search engine that crawls and indexes online Semantic Web documents. Swoogle claims to adopt a Web view on the Semantic Web [6] and indeed, most of the techniques on which it relies are inspired by classical Web search engines. While relying on these well-studied techniques offers a range of advantages, it also constitutes a major limitation: by largely ignoring the semantic particularities of the data that it indexes, Swoogle falls short of offering the functionalities required from a truly Semantic Web gateway.

From explicit relations to implicit semantic relations. Wide ranges of explicitly declared or implicit relations exist between ontologies that are largely ignored by Swoogle and other tools. Taking into account these relations is important since they partially define the semantics of the ontologies. Indeed, crawling the Semantic Web needs to consider, besides classical hyperlinks relating Web pages to semantic content, a wide and extensible range of explicit links between semantic documents (e.g., imports, seeAlso) and the semantics of these relations should be exploited when collecting semantic data. Second, special attention needs to be directed towards making the implicit relations between ontologies explicit. This implies applying a wide range of analysis tasks (e.g., duplication detection) to process, compare and relate semantic documents.

Focusing on semantic quality. The Semantic Web is characterized by a great variety of semantic data, ranging from high quality, richly axiomatized ontologies to flat bags of factual data. While the whole range is useful, different ontologies are needed for different tasks and scenarios. Hence, information about the quality of each semantic document is crucial to provide the most relevant access to users and applications. The implication of making ontology quality a core concern of the gateway is that the collected data undergoes a validation process that assesses the quality of each indexed document. In addition to the properties that are usually computed for classical documents (size, encoding, etc.), validation assesses characteristics that are useful for understanding the content of Semantic Web based data, e.g., the expressivity of the employed ontology language, the level of axiomatization, etc.

Providing rich, semantic access to data. As shown in Section 1, a variety of different applications will require access to the Semantic Web by using the gateway. These applications require different levels of formalization concerning the data they manipulate and the way to retrieve it. This should be taken into account by providing a range of access mechanisms that combine various query specifications, ranking measures and interfaces to these mechanisms, either for humans or software agents.

3. WATSON ARCHITECTURE OVERVIEW

The role of a gateway to the Semantic Web is to provide an efficient access point to online ontologies and semantic data. Therefore, such a gateway plays three main roles: 1- it collects the available semantic content on the Web, 2- analyzes it to extract useful metadata and indexes, and 3- implements efficient query facilities to access the data. While these three tasks are generally at the basis of any classical Web search engine, their implementation is rather different when we deal with semantic content as opposed to Web pages.

In order to support these three tasks, Watson has been designed around three core activities, each corresponding to a "layer" of its architecture, as depicted in Figure 1.

- **The ontology crawling and discovery layer** collects the online available semantic content, in particular by exploring ontology-based links.
- **The validation and analysis layer** is core to the architecture and ensures that data about the quality of the collected semantic information is computed, stored and indexed.
- **The query and navigation layer** grants access to the indexed data through a variety of mechanisms that allow exploring its various semantic features.
-

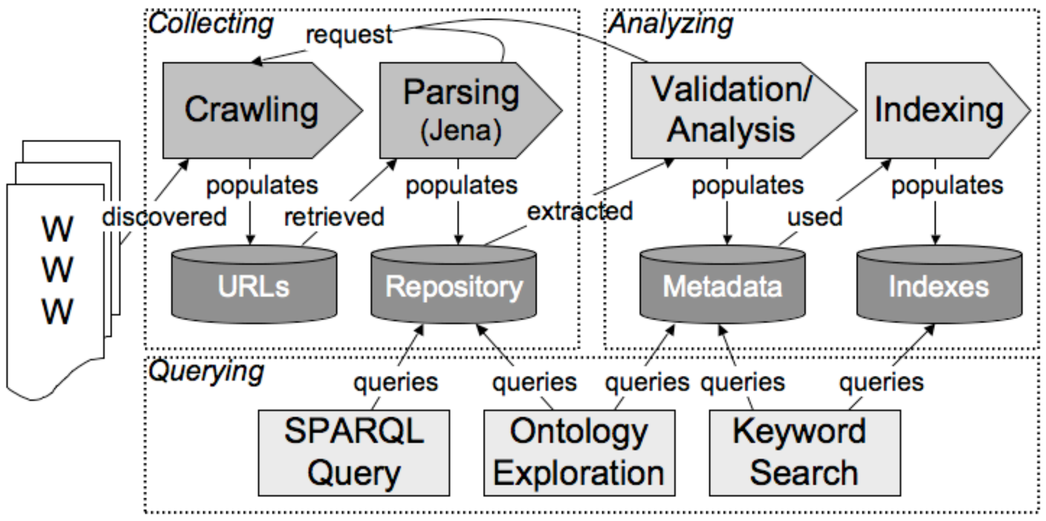


Figure 1. A functional overview of the main components of the Watson architecture.

At a more technical level, these three layers are hosted on a Web server (Apache Web server and Apache Tomcat), relying on a common RDMS (MySQL) to either communicate or exploit information about the collected semantic documents. All the components of Watson are written in Java. The descriptions of the three main tasks of Watson, both at the conceptual and technical levels, are presented in the following sections.

4. COLLECTING SEMANTIC CONTENT: CRAWLING THE SEMANTIC WEB

The goal of the crawling task in Watson is to discover locations of semantic documents and to collect them. Classical Web crawlers can be used, but they need to be adapted to take into account the fact that we are not dealing only with Web pages, but also with semantic content.

Sources. Different sources are used by the crawler of Watson to discover ontologies and semantic data (Google, Swoogle, Ping the Semantic Web.com[2], etc.) We designed specialized crawlers for these

[2] http://pingthesemanticweb.com/

repositories, extracting potential locations by sending queries that are intended to be covered by a large number of ontologies. For example, the keyword search facility provided by Swoogle is exploited with queries containing terms from the top most common words in the english language[3]. Another crawler heuristically explores Web pages to discover new repositories and to locate documents written in certain ontology languages (e.g. by including "filetype:owl" in a query to Google). Finally, already collected semantic documents are frequently re-crawled, to discover evolutions of known semantic content or new elements at the same location.

Filters. Once located and retrieved, these documents are filtered to keep only the elements that characterize the Semantic Web. In particular, to keep only the documents that contain semantic data or ontologies, we eliminate any document that cannot be parsed by Jena[4]. In that way, only RDF based documents are considered. Furthermore, we chose to consider RDF based semantic documents with the exception of RSS. The reason to exclude these elements is that, even if they are described in RDF, RSS feeds represent semantically weak documents, relying on RDF Schema more as a way to describe a syntax than as an ontology language.

Technologies. At a more technical level, the crawling layer of Watson relies on Heritrix, the Internet Archive's Crawler[5]. Heritrix is based on a pluggable architecture; allowing us to manage different crawl profiles by setting different pipelines of custom crawling modules. Once located and filtered, semantic documents are stored using Jena persistence mechanisms on top of a MySQL database.

Results. By August 2007, Watson has collected and filtered almost 25500 distinct semantic documents. By distinct we mean that if the same file appears several times, it is counted only once. Indeed, the crawler detects local copies of files containing semantic data. There is in average 1.27 locations per collected semantic document, meaning that, in fact, about 32350 URLs addressing semantic data or ontologies have been considered. These first results shows the feasibility and the relevance of a tool like Watson, but many other semantic resources are available and are waiting to be indexed. By continuously crawling the semantic web, we expect to reach 100K semantic documents by the end of the year.

5. ANALYZING SEMANTIC CONTENT: VALIDATION, INDEXING AND METADATA GENERATION

Many different elements of information are extracted from the collected semantic documents: information about the entities and literals they contain, about the employed languages, about the relations with other documents, etc. This requires analyzing the content of the retrieved documents in order to extract relevant information (metadata) to be used by the search functionality of Watson.

Simple Metadata. Besides trivial information, like the labels and comments of ontologies, some of the metadata that are extracted from the collected ontologies influence the way Watson is designed. For instance, there are several ways to declare the URI of an ontology: as the namespace of the document, using the xml:base attribute, as the identifier of the ontology header, or even, if it is not declared, as the URL of the document. URIs are supposed to be unique identifiers in the scope of the Semantic Web. However, two ontologies that are intended to be different may declare the same URI [3][5]. For these reasons, Watson uses internal identifiers that may differ from the URI of the collected semantic documents. When communicating with users and applications (see Section 6.2), these identifiers are transformed into common, non-ambiguous URIs.

Content. Another important step in the analysis of a semantic document is to characterize it in terms of its content. Watson extracts, exploits, and stores a large range of declared metadata or computed measures, like the employed languages/vocabularies (RDF, RDFS, OWL, DAML+OIL), information about the contained entities (classes, properties, individuals and literals), or measures concerning the quality of the knowledge contained in the document (e.g., the expressiveness of the employed language, the density of the class definitions, the consistency of the ontology). By combining these elements of information, Watson can decide whether or not a particular document should be treated as a semantically rich ontology. These

[3] http://www.world-english.org/english500.htm
[4] http://jena.sourceforge.net/
[5] http://crawler.archive.org/

elements are then stored and exploited to provide advanced, quality related filtering, ranking and analysis of the collected semantic content.

Relations between semantic documents. In the previous paragraphs, the role the analysis task was to extract metadata concerning one particular semantic document. In addition, a core aspect in the design of Watson concerns the exploitation of relations between semantic documents. The retrieved ontologies are inspected in order to extract information linking to other semantic documents. There are several semantic relations between ontologies that have to be followed (e.g. owl:imports, rdfs:seeAlso, namespaces, derefenceable URIs). Besides providing useful information about the considered documents, the results of this task are also used to extract potential locations of other semantic documents to be crawled.

In addition to declared semantic relations like owl:imports, our aim is also to compute implicit relations that can be detected by comparing ontologies. Equivalence is one of the most obvious of these relations, which is nevertheless crucial to detect. Indeed, detecting duplicated knowledge ensure that we do not store redundant information and that we do not present duplicated results to the user. On the same basis, several other relations are considered relying on particular notions of similarity between ontologies (inclusion, extension, overlap, etc.) Combined with other information from the crawler (e.g. date of discovery, of modification) these relations allow us to study and characterize the evolution of ontologies on the Web, through their different versions.

Technologies. The validation phase of Watson relies on an ad-hoc workflow control mechanism that automatically triggers and executes the relevant extraction tasks whenever a new document is discovered. Most of these extraction tasks rely on features provided by Jena to analyze the content of the semantic documents. For some advanced elements, like the identification of the OWL Species, of the DL Expressiveness, and for testing the consistency of ontologies, we rely on the Pellet OWL Reasoner[6].

Results. It is out of the scope of this report to provide a detailed description of the results of the validation task on the 25500 documents collected by Watson. However, it is interesting to mention that, as described in [3], the analysis of these results provide valuable information about the current state of the Semantic Web, for application and tool developers.

6. QUERYING WATSON: FROM KEYWORD SEARCH TO ONTOLOGY EXPLORATION AND SPARQL QUERIES

The third layer of components in the Watson architecture takes care of the user and application front-end services. Watson exposes its automatically gathered and validated data through a number of query interfaces, among which we distinguish between the web user interface and web services.

6.1 The Watson Web Interface

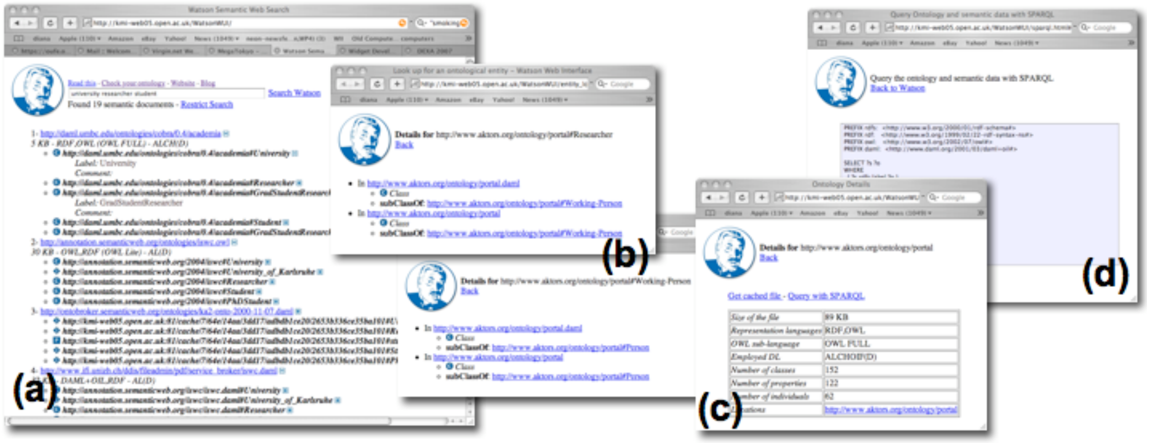
Even if the first goal of Watson is to support semantic applications, it is important to provide web interfaces that facilitate the access to ontologies for human users. Users may have different requirements and different levels of expertise concerning semantic technologies. For this reason, Watson provides different "perspectives", from the most simple keyword search, to sophisticated queries using SPARQL. These interfaces are implemented in Javascript, using the principles of AJAX, thanks to the DWR Library[7]. It can be accessed at the following address: *http://watson.kmi.open.ac.uk/*

Keyword search. The keyword search feature of Watson is similar in its use with usual web or desktop search systems. The set of keywords entered by the user is matched to the local names, labels, comments, or literals of entities occurring in semantic documents. A list of matching ontologies is then displayed with, for each ontology, some information about it (language, size, etc.) and the list of entities matching each keyword (Figure 2(a)). The search can also be restricted to consider only certain types of entities (classes, properties, individuals) or certain descriptors (labels, comments, local names, literals).

[6] http://www.mindswap.org/2003/pellet/
[7] http://getahead.org/dwr

At a technical level, this functionality relies on the Apache Lucene indexing system[8]. Different indexes – concerning semantic documents, entities, and relations between entities – are built from the metadata extracted during validation.

Ontology exploration. One principle applied to the Watson interface is that every URI is clickable. A URI displayed in the result of the search is a link to a page giving the details of either the corresponding ontology or a particular entity. Since these descriptions also show relations to other elements, this allows the user to navigate among entities and ontologies. For example, with the query "university researcher student", we obtain 19 matching semantic documents. Among them, http://www.aktors.org/ontology/portal.daml contains the entity http://www.aktors.org/ontology/portal#Researcher. Clicking on this URI, we can see that this entity is described (sometimes we different descriptors) in several ontologies (Figure 2(b)). In particular, it is shown to be a subclass of http://www.aktors.org/ontology/portal#Working-Person. Following the link corresponding to this URI also shows its description in each of the semantic documents it belongs to (Figure 2(b)). Then, the metadata corresponding to one of these documents can be retrieved following the appropriate link, e.g. http://www.aktors.org/ontology/portal, to find out about its languages, locations, etc. (Figure 2(c)). Finally, a page describing a semantic document provides a link to the SPARQL interface for this semantic document, as described in the next paragraph.


(a)
(b)
(c)
(d)
Watson Semantic Web Search
Read this - Check your ontology - Website - Blog
university researcher student
Search Watson
Look up for an ontological entity – Watson Web Interface
Back
Ontology Details
Get cached file - Query with SPARQL
Query Ontology and semantic data with SPARQL
Back to Watson

Found 19 semantic documents - Restrict Search

1- http://daml.umbc.edu/ontologies/cobra/0.4/academia
5 KB - RDF,OWL (OWL FULL) - ALCH(D)
- http://daml.umbc.edu/ontologies/cobra/0.4/academia#University
  Label: University
  Comment:
- http://daml.umbc.edu/ontologies/cobra/0.4/academia#Researcher
- http://daml.umbc.edu/ontologies/cobra/0.4/academia#GradStudentResearch
  Label: GradStudentResearcher
  Comment:
- http://daml.umbc.edu/ontologies/cobra/0.4/academia#Student

2- http://annotation.semanticweb.org/ontologies/iswc.owl
30 KB - OWL,RDF (OWL Lite) - AL(D)
- http://annotation.semanticweb.org/2004/iswc#University
- http://annotation.semanticweb.org/2004/iswc#University_of_Karlsruhe
- http://annotation.semanticweb.org/2004/iswc#Researcher
- http://annotation.semanticweb.org/2004/iswc#Student
- http://annotation.semanticweb.org/2004/iswc#PhDStudent

3- http://ontobroker.semanticweb.org/ontologies/ka2-onto-2000-11-07.daml

4- http://www.ifi.unizh.ch/ddis/fileadmin/pdf/service_broker/iswc.daml
DAML+OIL,RDF - AL(D)
- http://annotation.semanticweb.org/iswc/iswc.daml#University
- http://annotation.semanticweb.org/iswc/iswc.daml#University_of_Karlsruhe
- http://annotation.semanticweb.org/iswc/iswc.daml#Researcher

Details for http://www.aktors.org/ontology/portal#Researcher

- In http://www.aktors.org/ontology/portal.daml
  - Class
  - **subClassOf:** http://www.aktors.org/ontology/portal#Working-Person
- In http://www.aktors.org/ontology/portal
  - Class
  - **subClassOf:** http://www.aktors.org/ontology/portal#Working-Person

Details for http://www.aktors.org/ontology/portal#Working-Person

- In http://www.aktors.org/ontology/portal.daml
  - Class
  - **subClassOf:** http://www.aktors.org/ontology/portal#Person
- In http://www.aktors.org/ontology/portal
  - Class
  - **subClassOf:** http://www.aktors.org/ontology/portal#Person

Details for http://www.aktors.org/ontology/portal

Size of the file	89 KB
Representation languages	RDF,OWL
OWL sub-language	OWL FULL
Employed DL	ALCHOIF(D)
Number of classes	152
Number of properties	122
Number of individuals	62
Locations	http://www.aktors.org/ontology/portal

Query the ontology and semantic data with SPARQL

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX daml: <http://www.daml.org/2001/03/daml+oil#>

SELECT ?s ?o
WHERE
```

Figure 2. The Watson Web interface (http://watson.kmi.open.ac.uk/WatsonWUI).

SPARQL. A SPARQL endpoint has been deployed on the Watson server and is customizable to the semantic document to be queried. This endpoint is implemented thanks to the Joseki SPARQL server for Jena[9]. A simple interface (Figure 2(d)) allows to enter a SPARQL query and to execute it on the selected semantic document. This feature can be seen as the last step of a chain of selection and access task using the Watson web interface. Indeed, keyword search and ontology exploring allow the user to select the appropriate semantic document to be queried. Our plan is to extend this feature to be able to query not only one semantic document at a time, but also to automatically retrieve the semantic data useful for answering the query. This kind of feature, querying a whole repository instead of a single document, has been implemented in the OntoSearch2 system [11].

6.2 The Watson API

Since the main motivation behind the development of Watson is to support the next generation Semantic Web applications, requiring the ability to dynamically discover, combine, and use ontologies available online, a set of Web Services and an associated API are being developed to facilitate the programmatic access to the semantic content collected by Watson for these applications. This API provides a set of basic

[8] http://lucene.apache.org/
[9] http://www.joseki.org/

access functionalities and is actually at the basis of the development of the Watson web interface. Technically, it consists of a set of Java classes and methods providing and interface to two SOAP Web services deployed using Apache Axis[10]. Therefore, an application using this API would be able to efficiently and transparently exploit the functionalities of Watson remotely (see Figure 3). This API can be downloaded, along with the WSDL description of the Web services, documentations and examples, at the following address: *http://watson.kmi.open.ac.uk/WS_and_API.html*

`

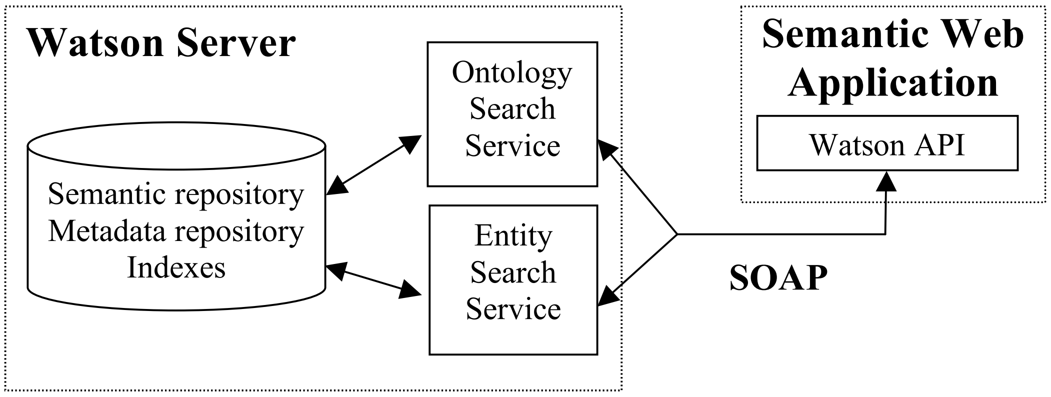


Figure 3. Usage of the Watson API in a Semantic Web application.

7. APPLICATIONS OF WATSON

Several applications have already been developed using the Watson API and Web services. Rather than describing the details of this API, we briefly present three applications that are currently being (re-) developed to rely on the features provided by Watson for solving real-life problems.

Ontology Matching. In [12], we proposed a novel paradigm in which the links between two entities of two different ontologies can be discovered by dynamically selecting a third ontology covering both entities. Various techniques have been developed for discovering correspondences, or mappings, between ontologies, but most of them are limited to the information internal to the ontologies to be mapped. In our mechanism, external ontologies are discovered and queried to derive the semantic relations between the considered entities, thanks to the Watson search mechanism. It is worth mentioning that our experiments with this mechanism have shown the importance of assessing the quality of the ontology in the selection process, since low quality ontologies may result in incorrect mappings.

Folksonomy enrichment. Folksonomies are popular systems where users can annotate resources (pictures, web-pages, etc.) using tags, i.e. uncontrolled keywords that are shared by the entire community of users. Since the resulting organization of the resources is weakly structured, search mechanisms in folksonomies are limited. A mechanism has been proposed in [1] to semantically enrich folksonomies by dynamically selecting elements of online ontologies. Tags are first clustered according to their co-occurence, and, thanks to Watson, the semantic relations between pairs of tags of the same cluster are discovered. The results take the form of small ontology modules combining the knowledge from multiple online ontologies to semantically describe the tags of a particular cluster.

Semantic Web Browsing. Magpie [7] is a Semantic Web browser, taking the form of a browser plugin that highlights instances of selected ontology concepts in the current web-page. Magpie is currently being extended to rely on the dynamic selection of appropriate ontologies by querying Watson with keywords extracted from the text of the web-page. As explained in [4], this may require to combine knowledge coming from different ontologies and to extract from these ontologies the parts that are actually relevant to semantically describe the web-page. In this scenario, the advanced features made possible by Watson [5], such as intelligent ontology selection and ontology modularization, are particularly relevant. More

[10] http://ws.apache.org/axis/

information about PowerMagpie, the second generation of Magpie that relies on these principles, can be found in the following website: http://powermagpie.open.ac.uk

8. CONCLUSION

In this paper, we have presented the design of Watson, a gateway that provides efficient access to the content of the Semantic Web by addressing the requirements of emerging Semantic Web applications. At the time of writing this paper, the core architecture of Watson has been implemented, and several tens of thousands of semantic documents have been collected, analyzed and are now accessible through various interfaces.

Current efforts are dedicated to important research issues that need to be tackled by a gateway like Watson. More specifically, assessing the quality of the collected ontologies is of particular importance. Indeed, different semantic Web applications may have different requirements regarding the quality of the knowledge they manipulate. Defining methods and metrics to evaluate different aspects of the design and formalization of ontologies, along with flexible and customizable ranking mechanisms relying on these measures is therefore a crucial task.

Among other ongoing work within Watson is also the management of networks of ontologies. Indeed, besides explicit relations like import, ontologies are related through a network of implicit relations, some being extensions of others, different versions, or some being incompatible with others, etc. The current implementation of Watson is able to detect some form of duplication of ontologies. Important efforts are now spent in detecting a broader range of semantic relations, and in managing the corresponding network of ontologies.

REFERENCES

[1] S. Angeletou, M. Sabou, L. Specia, and E. Motta. Bridging the Gap Between Folksonomies and the Semantic Web: An Experience Report . In Proc. of ESWC workshop on Bridging the Gap between Semantic Web and Web 2.0, 2007.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. Scientific American, 284(5):34–43, May 2001.

[3] M. d'Aquin, C. Baldassarre, L. Gridinoc, S. Angeletou, M. Sabou, and E. Motta. Characterizing Knowledge on the Semantic Web with Watson. Submitted to the EON 2007 workshop.

[4] M. d'Aquin, M. Sabou, and E. Motta. Modularization: a Key for the Dynamic Selection of Relevant Knowledge Components. In Proc. of the ISWC 2006 Workshop on Modular Ontologies, 2006.

[5] M. d'Aquin, M. Sabou, M. Dzbor, C. Baldassarre, L. Gridinoc, S. Angeletou, and E. Motta. WATSON: A Gateway for the Semantic Web. ESWC 2007 Poster session.

[6] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. C. Doshi, and J. Sachs. Swoogle: A Search and Metadata Engine for the Semantic Web. In Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management. ACM Press, November 2004.

[7] J. Domingue, M. Dzbor, and E. Motta. Magpie: Supporting Browsing and Navigation on the Semantic Web. In N. Nunes and C. Rich, editors, Proceedings ACM Conference on Intelligent User Interfaces (IUI), pages 191–197, 2004.

[8] J. Gracia, R. Trillo, M. Espinoza, and E. Mena. Querying the Web: A Multiontology Disambiguation Method. In Proc. of the Sixth International Conference on Web Engineering (ICWE'06), Palo Alto, California (USA). ACM, July 2006.

[9] V. Lopez, M. Pasin, and E. Motta. AquaLog: An Ontology-portable Question Answering System for the Semantic Web. In Proc. of the European Semantic Web Conference, ESWC, 2005.

[10] Lopez, V., Motta, E., Uren, V. (2006). PowerAqua:Fishing the Semantic Web. Proceedings of the 2006 European Semantic Web Conference (ESWC 2006), Montenegro.

[11] J. Z. Pan, E. Thomas, and D. Sleeman. ONTOSEARCH2: Searching and Querying Web Ontologies. In Proc. of the IADIS International Conference WWW/Internet, 2006.

[12] M. Sabou, M. d'Aquin, and E. Motta. Using the Semantic Web as Background Knowledge for Ontology Mapping. In Proc. of the ISWC Ontology Matching Workshop collocated, 2006.