PowerMagpie: A Semantic Web Browser – v1

OpenKnowledge Deliverable D8.1.

Laurian Gridinoc, Mathieu d’Aquin, Davide Guidi, Martin Dzbor and Enrico Motta

Knowledge Media Institute, The Open University.
{l.gridinoc, m.daquin, d.guidi, m.dzbor, e.motta}@open.ac.uk

Abstract. This report details the design and development of PowerMagpie, a semantic browser. PowerMagpie brings semantic interpretation to classical web pages by dynamically – i.e. during browsing – selecting ontologies that contain semantic descriptions of the main terms of the web page. Mechanisms are then provided as extensions of a usual web browser to navigate this additional semantic layer, in relation with the context provided by the web page. We first introduce the idea of extending browsing through semantic, ontology-based interpretation. Then, we provide a brief description of Magpie, the predecessor of PowerMagpie. Discovering and selecting online ontologies to be used for semantic browsing is a central part of the PowerMagpie architecture and requires an efficient infrastructure. Watson, the Semantic Web gateway, provides this infrastructure. By collecting, analyzing and giving access to semantic resources available online, Watson constitutes the necessary backbone on which PowerMagpie relies. Therefore, in addition to the details of the design and development of the PowerMagpie architecture and prototype, a description of Watson is also presented.

1. Introduction

1.1 Supporting interpretation of web content

The Web supports individuals and communities in performing a large array of interactions and tasks by providing a large scale infrastructure which makes it possible to locate, explore and exchange knowledge. However, not much support is available for the interpretation of web pages [5], beyond what provided by the author(s) of each particular resource. Indeed, generally speaking the web offers two mechanisms to help users make sense of a resource. The first is hypertext linking, which is normally used to indicate additional explicative material, which can help the user in understanding web content. The second is the “Related Pages” feature provided by several search engines and directories. While these mechanisms can be useful in many cases, they nevertheless suffer from limitations.

Hypertext linking is provided by the author(s) of a web page, who may have a particular audience in mind or may not be aware of other relevant material. In addition there is a temporal element here, given that naturally any link in a web page was defined at a particular time point, t. Hence, relevant information produced after t would simply not have been available to the resource provider and therefore this mechanism cannot be very robust with the respect to the generation of new information, which may in principle be useful to interpret earlier resources.

The “Related Pages” facility improves on the static hypertext linking, as it dynamically identifies interesting relevant pages and therefore supports a user in exploring a domain and
even in moving across domains. However, from the point of view of supporting sensemaking, this feature is not very precise and effective, given that the linking is based on lexical similarity and that typically the search space the user needs to explore explodes very quickly, in sharp contrast to the targeted manner in which page designers insert hyperlinks in web resources.

1.2 Ontology-driven sensemaking in Magpie

The Magpie tool addresses this issue by automatically associating a semantic layer to a web resource. This process relies on the availability of an ontology [6], which Magpie uses both to associate meaning with the lexical entities found in text, and also to identify relevant services, which are associated with classes in the current ontology. The Magpie-mediated association between an ontology and a web resource provides an interpretative viewpoint or context over the resource in question.

While Magpie has been successfully used in a number of applications in the domains of Climate Science, Agriculture, and Academic Research, it requires the user to select a priori the ontology that is going to be used to provide semantic interpretation of web content. Hence, it suffers from the brittleness that typically affects knowledge-based systems: while Magpie works well within a well-defined domain, as soon as the browsing session moves away from the original domain, the support provided by Magpie declines dramatically.

A number of other approaches can be found in the literature. However most of them, when not extracting data from the pages and offer it out of context in a different interface (Haystack/Thresher [8] and PiggyBank [9]), tend to deal with structured content, e.g. Sifter [10], and pay no attention to standard web documents. Other tools, which bring in the same interface semantic data within the Web page, rely on collaborative annotation, like Trailblazer [13]. While this approach can work for well defined communities in well defined domains, it is nevertheless brittle: §ly the quality of the interpretation support relies on the availability and quality of annotation. Obviously such annotations are not generally available on the web.
Hence, there is the need for a tool which is able to dynamically bring relevant semantic information into a browsing session, without being limited to one particular ontology. The goal of task 8.1 in the OpenKnowledge project, “Semantic Browsing”, is exactly to address this issue by developing a new generation semantic browser, PowerMagpie, able to go beyond Magpie and bring to the user, opportunistically, the appropriate semantic information relevant to his/her current information needs, in principle from any ontology available on the web.

In this report we describe the current version of PowerMagpie (v1), which provides an initial demonstration of the feasibility of our ideas and we also introduce Watson [3], the key infrastructure component, which makes it possible for PowerMagpie to access large scale semantic content with the degree of efficiency, which is required for open semantic browsing.

2. Watson, a novel semantic web gateway

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, like PowerMagpie, dynamically select, combine and exploit knowledge published on the Semantic Web. In the following, we present the design of Watson, which have been guided by the requirements of these applications and by lessons learnt from previous systems.

2.1 The need for Watson

In order to go beyond Magpie and provide non-brittle, open semantic browsing, PowerMagpie needs to be able to access efficiently semantic markup on the web. At the time of writing the proposal for the OpenKnowledge project, our expectation was that PowerMagpie would rely on the facilities provided by Swoogle [4], which was the most comprehensive search engine available for the semantic web. Unfortunately our initial experiments with Swoogle [15] showed that this system exhibits severe limitations and therefore it is not suitable to support the kind of applications - in particular, PowerAqua [11] and PowerMagpie, which are envisaged by the OpenKnowledge proposal.

Specifically we identified three main limitations of Swoogle:

- Firstly, Swoogle provides only weak access to semantic content, which is a keyword-based search that does not consider the semantic content of the documents it accesses. The Swoogle interface is clearly inspired from the Google’s one, and its usage essentially treats semantic resources in the same way as Google treats web documents. For example, for every ontology that is retrieved, Swoogle only displays a text “snippet” that shows that the considered terms occur somewhere in the ontology. The user (or application) is then supposed to download the ontology to get access to its content. Obviously this mechanism may be OK (although a bit inefficient) for a user, but certainly cannot support systems like PowerMagpie, which need to associate specific items in an ontology to lexical entities in the text.

- Secondly, Swoogle does not consider the quality of the knowledge it collects. It simply crawls all semantic content and stores it in a repository, which is structured by the use of a Page Rank like algorithm. This solution is unsatisfactory for our purposes, given that for applications such as PowerAqua and PowerMagpie it is important that the high quality information is given priority.

- Finally, Swoogle does not consider semantic relations between ontologies other than the ones that are explicitly stated (e.g., import). This is a serious limitation, given that, because they are semantic resources, ontologies can be compared and
related through semantic relations. For example, some ontologies may be versions of others, some may extend pre-existing ones, others can be mutually incompatible. These elements are of particular importance for semantic applications that requires exploiting several, interrelated ontologies and indeed the notion of Networked Ontologies underpins the most advanced approaches to ontology engineering, such as the one pursued by the NeOn project [7]. Looking at the results provided by Swoogle, it appears that not even the simplest (syntactic) notion of duplication (or copy) is considered, as the same ontologies appear several times in the results, at different ranks.

The goal of Watson is therefore to overcome these limitations in order to provide a truly Semantic Web Gateway. In the next sections we illustrate the architecture and key features of Watson and show how it addresses the limitations of Swoogle.

2.2 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.](image)
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.

2.3 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\(^1\), etc.) We designed specialized crawlers for these 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\(^2\). 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\(^3\). 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\(^4\). 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.

---

\(^1\) [http://pingthesemanticweb.com/](http://pingthesemanticweb.com/)
\(^2\) [http://www.world-english.org/english500.htm](http://www.world-english.org/english500.htm)
\(^3\) [http://jena.sourceforge.net/](http://jena.sourceforge.net/)
2.4 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. 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 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, dereferenceable 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\(^5\).

**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 [2], the analysis of these results provide valuable information about the current state of the Semantic Web, for application and tool developers.

### 2.5 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.

#### 2.5.1 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\(^6\). It can be accessed at the following address: http://watson.kmi.open.ac.uk/.

---


\(^6\) [http://getahead.org/dwr](http://getahead.org/dwr)
**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). The search can also be restricted to consider only certain types of entities (classes, properties, individuals) or certain descriptors (labels, comments, local names, literals).

At a technical level, this functionality relies on the Apache Lucene indexing system\(^7\). 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 3(a)). 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 3(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 3(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.

**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\(^8\). A simple interface (Figure 4) 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 [12].

---

\(^7\) http://lucene.apache.org/

\(^8\) http://www.joseki.org/
Figure 3. Ontology Exploration in Watson: Following the link created by URIs to navigate among entities and ontologies.

Figure 4. SPARQL query interface and result for the http://www.aktors.org/ontology/portal ontology.
2.5.2 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 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\(^9\). Therefore, an application using this API would be able to efficiently and transparently exploit the functionalities of Watson remotely (see Figure 5). 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 5. Usage of the Watson API in a Semantic Web application.](image)

Several applications have already been developed using the Watson API and Web services. For example, [1] describes a procedure relying in the Watson search functionalities to enhance folksonomy tags with semantic information coming from online ontologies. An ontology matching algorithm has also been described [14] that dynamically select and exploit ontologies thanks to Watson\(^10\) for deriving semantic relations between entities of different ontologies. Another example of ongoing development is the Watson plugin for the Protégé ontology editor that allows the user to retrieve existing semantic description from online ontologies and to integrate them into the currently edited ontology. Finally, the way the Watson API is exploited for semantic browsing in PowerMagpie to select and access online knowledge is further described in Section 3.

2.6 Future work

In this section, 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, such as PowerMagpie. At the time of writing this report, the core architecture of Watson has been implemented, and several tens of thousands


\(^10\) Originally, this algorithm was using Swoogle, but it is currently being re-designed for Watson.
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.

3. PowerMagpie, v1

The primary goal of PowerMagpie is to bring semantic interpretation to web browsing, without involving important efforts from the user. This supposes to automatically relate the major terms present in the text of the web page to semantic entities, dynamically discovered in ontologies available on the (Semantic) web. In order to achieve this goal, PowerMagpie has to handle four major tasks:

1. **Identifying relevant terms in the currently browse web page.** Not all the terms occurring in the text of the web page have the same importance. It is crucial to identify the ones that characterize the domain covered by the document, and to present first the terms that are the most important for the interpretation and understanding of the page.

2. **Selecting online ontologies to interpret the domain terms.** This task constitutes one of the major differences between PowerMagpie and its predecessor, Magpie. Magpie requires for the user to manually select the appropriate ontology for semantic browsing. This is an important limitation since the user may not be familiar with ontologies and has to be aware of the existence of the appropriate ontology, matching the domain of the currently browsed document. This process has to be carried out every time the user browses a new page about a domain not covered by the selected ontology. In addition, Magpie is bounded by the use of a single ontology. Web documents often relate to several different domains, requiring the knowledge contained in several different ontologies to be interpreted. In PowerMagpie, the goal is to automate the selection of a set of ontologies to support semantic browsing. Thanks to Watson, PowerMagpie can dynamically (during browsing) discover ontologies published online that provide the relevant knowledge for the interpretation of the domain terms extracted in the previous step.

3. **Relating the text to semantic information.** The goal here is to associate the terms of the domain, found in the text of the web pages, with the relevant semantic information. In other term, PowerMagpie has to establish a matching from the entities present in the ontologies (classes, properties and individuals providing semantic descriptions) to the occurrences of terms in the web page.

4. **Navigating textual and semantic information together.** Once the elements of text have been put in relation with the appropriate semantic information, it is important to provide efficient visualizations and interfaces for browsing. This requires not only
to display semantic information along with the usual textual information provided by the web page, but also to allow for interactions between the user and both forms of information. In a sense, semantic browsing consists in navigating the semantic links relating ontological entities, as well as the relations associating textual elements to semantic interpretations, supporting the user in making the best of the knowledge available online when browsing web documents.

In the following, we describe how these four tasks are realized within the architecture of PowerMagpie (Section 3.1). We then detail, both at technical and user level, the current prototype of PowerMagpie (Section 3.2).

### 3.1 Architecture

From a user perspective, PowerMagpie is an extension of a classical web browser, but the main functionalities it provides are actually realized by server-side software that can be seen as a backend for the whole architecture. In this way, PowerMagpie is particularly easy to install, does not require particular software except a JavaScript enabled web browser, and, as far as the client is concerned, is very lightweight in terms of memory and computing power.

An overview of the Power Magpie architecture is shown in Figure 6.

![Figure 6: Overview of the PowerMagpie Architecture.](image)

The PowerMagpie server is the central element of this architecture: it is in charge of realizing the tree first tasks described previously and to transmit the results to the browser extension. Watson (described in Section 2) is also an important component since it provides the necessary ontology selection mechanisms. Finally, the web browser extension acts as an interface for the user and is in charge of realizing the last of the four tasks described above:
navigating the semantic and textual information. It is the only part of the architecture that is required to be installed on the client-side, i.e. on the user’s computer. Note that another advantage of relying on a common server is that the information computed by the PowerMagpie Server can be cached, improving the performance for all the users, based on each individual use of PowerMagpie.

3.1.1 Identifying relevant terms

Identifying the terms having a particular importance in the text, the key terms, is a crucial task for PowerMagpie. By key terms we mean terms that are specific to the domain, that characterize the text, or that may have a specific meaning in the domain of the document. Because of performance issues, providing semantic interpretation for the entire set of terms contained in a web document would not be feasible. It is required to restrict the considered terms to the ones of particular importance.

A common way to assess the relevance of a term in a given document is by the use of the TF*IDF weight \[16\]. TF*IDF (term frequency * inverse document frequency) is commonly used to give an indication of the importance of a term to a document, which is included in a corpus of documents. It can be seen as the number of occurrences of the term in the document, divided by the frequency of the term in the corpus. The frequency of the term in the corpus relates to the number of documents containing the term, divided by the total number of documents. For the purpose of PowerMagpie, we adapted the TF*IDF weight to use the Google web search engine instead of a local corpus of documents. The frequency of the term in Google is calculate by dividing the number of hits Google returns for this particular term by an estimation of the number of web pages indexed by Google.

On the basis of this measure, the PowerMagpie server provides a ranking of the terms from the considered page, the most important terms being considered first for ontology selection and presented first to the user.

Initial experiments on this process have shown that important improvements can be achieved. First, even if the TF*IDF weight gives good results, very common words of the English language still appear in the list of key terms for a document. This has been solved by the use of a list of stop-words, i.e. a list of very common and not descriptive terms like “the”, “an”, “are”, etc. Second, even if a term is important for a document, it may not be particularly well covered by online ontologies. A combined weight, taking into account both the importance of the term for the document and its coverage by online ontologies in Watson would help improving the efficiency of PowerMagpie.

3.1.2 Selecting ontologies

In order to retrieve semantic descriptions of the key terms from the currently browsed document, PowerMagpie has to dynamically select the relevant knowledge from the ontologies published on the Semantic Web. This task is realized thanks to Watson. Indeed, the Watson API implements search functions to select ontologies covering certain keywords, allowing PowerMagpie to identify ontologies containing semantic descriptions of the key terms extracted in the previous step. In addition to search functionalities for ontologies, Watson implements mechanisms to extract from the ontologies the entities (classes, properties, and individuals) matching the input terms, and statement (semantic descriptions) about these entities, thus providing all the features required to support the semantic interpretation of these terms. Finally, Watson stores metadata about the ontologies and provides access mechanisms such as SPARQL queries, allowing the exploration of the semantic resources relevant to the interpretation of the browsed web page.
3.1.3 Matching terms to ontology entities

Ontologies provide descriptions of classes, properties, and individuals that can be considered as semantic interpretations of the terms they represent. However, in order to allow for semantic browsing, it is necessary to maintain a link between these entities, the terms, and the occurrences of the terms in the web page. Simple terms can easily be matched from the description of an ontological entity to an occurrence in the web page. However, even if the selection of entities is based on single words, combination of terms, or compound, can provide better matching and therefore lead to more accurate semantic interpretation. For example, if the text contains the sequence “University of Minnesota”, it would be decomposed into three terms: “University”, “of”, and “Minnesota”. Only “University” and “Minnesota” would be considered as key terms that characterize the page. When selecting ontologies, these two terms may match entities like the class of University in http://annotation.semanticweb.org/iswc/iswc.owl, and the individual MN in http://www.daml.org/2003/02/usps/state.owl, but they would also both match the individual University_of_Minnesota from http://ontoworld.org/index.php/Special:ExportRDF/Category:University?xmlmime=rdf. PowerMagpie implements a mechanism that detects this kind of situation, taking into account the descriptions of the entities and the text of the document to create compound terms like “University of Minnesota”, extending the list of key terms from the page.

3.1.4 Navigating semantic information

The previous step associates terms found in the currently browsed web page to semantic entities, dynamically selected from online ontologies. On this basis, the PowerMagpie interface is in charge of providing efficient visualization, allowing the exploration of the selected semantic information. In addition, to support semantic browsing, it is crucial that this interface allows for intuitive navigation from the textual information provided by the web page to the semantic information provided by the ontologies, and vice-versa. The next section details how this task is achieved within the current prototype of PowerMagpie.

3.2 PowerMagpie Prototype

A video demonstrating the features of the current prototype of PowerMagpie can be found at the following address: http://powermagpie.open.ac.uk

The PowerMagpie Interface is entirety written in JavaScript, communicating with the PowerMagpie Server using the principles of AJAX, thanks to the DWR library. It is activated through a bookmarklet – i.e., a link in the bookmark of the browser that embed JavaScript code to launch the application –, and take the form of a narrow, vertical widget displayed on top of the currently browsed web page. This widget and its content (the entity and ontology panels, and the detail box) are developed using the Ext library.

3.2.1 Exploring terms and entities

When launched, PowerMagpie analyzes the currently browsed web page to extract key terms, select ontologies and ontological entities, and match these entities to occurrences of the key terms in the text of the document. It then displays in a panel labeled “Entities” the

11 http://extjs.com/
list of the key terms from the web page, ordered according to their importance to the document (see Section 3.1.1). Each of the terms of the list can be extended to display the semantic information it is attached to (see Figure 7). This semantic information takes the form of a list of ontological entities, with for each of these entities, its name or label, its type (class, property or individual, represented as a small icon), its URI and the list of ontologies in which it is described. An item representing an ontology is a link to the ontology panel (see Section 3.2.2), allowing further exploration of the semantic information attached to the considered entity. Terms in the entity panel are also associated to text snippets, listing their locations on the web page (see Section 3.2.3).

3.2.2 Exploring ontologies and semantic information

The panel labeled “Ontologies” displays the expandable list of ontologies that have been automatically selected to support the semantic interpretation of the current web page. To each ontology, the list of ontological entities it contains and that are relevant to the current document is attached (see Figure 8). For each of the entities, the list of terms it matches is provided, each term representing a link back to the entity panel.

In the ontology panel, information about ontologies and ontological entities can be explored thanks to Watson. Indeed, clicking on the URI of an ontology lead to the display of a number of information about it (size, language, number of classes, properties and individuals, expressiveness, etc.), together with links pointing to the actual file containing the ontology and to an interface for the execution of SPARQL queries upon this ontology. Information and links are displayed in the box labeled “Details”, at the bottom of the PowerMaggie Interface (see Figure 8).

In the same way, semantic information concerning ontological entities can be retrieved and displayed in the detail box. Semantic information concerning entities includes their
type, label, comment, and any relation that may occur between them and other ontological entities. Relations to external entities form links to the Watson entity description page, allowing further exploration of the knowledge attached to these entities in online ontologies.

3.2.3 Navigating from the text to the ontologies, and vice versa

Semantic browsing with PowerMagpie would not be complete without the support of interactions between the textual information contained in the web page and the semantic information selected from online ontologies. While exploring terms and entities in the entity panel, occurrences of the considered term are highlighted whenever this term is expanded in the list. Text snippets attached to the terms can be used as links, forcing the browser to move to the location of this particular occurrence in the text, providing an easy way to visualize at the same time both the textual context and the semantic information attached to a term.

While the navigation from the ontological entities to the text is a powerful tool for apprehending the document from a semantic perspective, navigating from the text to the ontologies seems even more useful. Indeed, one of the major scenarios for the use of PowerMagpie is a situation in which the user requires a semantically described definition of a term while reading the page, because he is unfamiliar with this term, or simply because additional information about it may be useful for his understanding of the web document. PowerMagpie realizes this in a very simple way: whenever a word in the text of the web page is double-clicked, the PowerMagpie Interface moves to the entity panel, to display the semantic description of this term. If the given word has not been recognized as a key term...
for the considered page, then it is automatically added, and the relevant ontologies and ontological entities dynamically selected and explored. Compound terms (like “University of Minnesota”) can also be added and displayed, by selecting the corresponding text and clicking on the PowerMagpie bookmarklet. Another important advantage of this feature of PowerMagpie is that, in this way, semantic browsing is not limited to the initial set of key terms, but can be dynamically extended to any term present in the text of the web page. PowerMagpie is therefore only limited by the current coverage of the Semantic Web.

4. Conclusions

In this report, we have presented the design and development of PowerMagpie, a semantic browser that supports the semantic interpretation of web documents through the dynamic selection of relevant knowledge from online ontologies. PowerMagpie is developed as an extension of a classical web browser, and exploits Watson, the Semantic Web Gateway, to discover, select and access semantic resources.

One of the requirements inherited from Magpie concerns the need for a near real-time responsiveness of the browser. Due to time consuming computations, but more importantly to numerous calls to external services (Watson and Google), the current prototype of PowerMagpie does not completely fulfill this requirement. Efforts are currently spent together with the developers of Watson to reduce the number of these calls and improve the performance of both tools.

One of the features that was part of Magpie and that has not yet been considered for PowerMagpie is the automatic selection, presentation and invocation of web services on the basis of the semantic information attached to the recognized terms. This feature has shown to be particularly useful in learning scenarios, where the addition of the semantic layer that Magpie provides allowed to dynamically and intelligently link the current web page to other (textual) material – like courses, publications, etc. – through the use of semantic web services.

Many other directions are to be considered in the development of PowerMagpie, requiring the study of complex research issues (term disambiguation, ontology ranking, etc.), as well as more technological considerations, like the integration of other approaches for semantically enriching web pages, via the RDFa semantic layer12.

References


12 http://www.w3.org/TR/xhtml-rdfa-primer/


