Semantic Search on Heterogeneous Wiki Systems*

Fabrizio Orlandi
Digital Enterprise Research Institute
National University of Ireland, Galway
fabrizio.orlandi@deri.org

Alexandre Passant
Digital Enterprise Research Institute
National University of Ireland, Galway
alexandre.passant@deri.org

ABSTRACT

This paper describes a system to enable semantic search across heterogeneous wikis in an unified way using Semantic Web technologies. In particular, we discuss (i) how we designed a common model for representing social and structural wiki features and (ii) how we extracted semantic data from wikis using Mediawiki and Dokuwiki. On this basis, we show how we built and efficient application with a simple user-interface enabling semantic searching and browsing capabilities on the top of different interlinked wikis.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: On-line Information Services; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Web-based interaction*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods; K.4.3 [**Computers and Society**]: Organizational Impacts—*Computer-supported collaborative work*

General Terms

Human Factors, Documentation, Languages

Keywords

Semantic Web, SIOC, wikis, semantic search, MediaWiki, Dokuwiki, Social Semantic Web, Linked Data

1. INTRODUCTION

Wikis are widely used both on the Web — with well-know and popular systems such as Wikipedia or Wikitravel, as well as wiki systems dedicated to open-source software management such as Trac[1] — and in the workplace, for instance for project management or customer relationships. In this case, complete solutions such as SocialText Workspace[2], in addition to open-source systems, as MediaWiki[3] can be used for such purposes.

However, each wiki system relies on its own data structure and API to model its data and let developer access its. Consequently, wikis act as isolated systems, where information from one system cannot be easily integrated with information from another one. Practically, this introduce several drawbacks when users need to access information on the Web or in the enterprise. Let us for instance consider the ACME company: its marketing department uses a MediWiki-based system, its engineering team use MoinMoin[4] and the sales are using DokuWiki[5]. Hence, in case a user wants to retrieve all information about a particular project, he has to separately query each wiki, which can be time consuming. Similar issues happen on the Web, if for example one wants to identify all the contribution of someone across several wikis.

In this paper, we propose an approach based on Semantic Web [5] technologies, Linked Data principles [4] and lightweight ontologies to solve such issues and to enable semantic search across heterogeneous wiki systems. Especially, our contributions include:

- a common model for representing wiki structure and contributions in RDF — Resource Description Framework — encompassing previous models in the area;
- various exporters for popular wiki systems, translating wiki information in RDF annotations (based on the previous models) in real-time, and
- a semantic search engine, built on the top of an RDF store (enabling inference capabilities) and providing means to retrieve information contained in originally heterogeneous wikis in a novel and user-friendly way.

This paper is organized as follows. First, we will explore the structure of the ontology we propose for representing the social and structural features of wikis, also discussing the current state of the art in the domain and how our contribution relates to it. Second, we will detail the exporters that we have built for different wiki systems (MediaWiki and DokuWiki) and discuss how we collected data from these two

*The work presented in this paper is funded in part by Science Foundation Ireland under grant number SFI/08/CE/I1380 (Líon 2).

[1] `http://trac.edgewall.org/`

[2] `http://www.socialtext.com/`

[3] `http://www.mediawiki.org/`

[4] `http://moinmo.in/`

[5] `http://www.dokuwiki.org/dokuwiki`

WikiSym '10, July 7-9, 2010, Gdańsk, Poland

platforms across five wiki sites. Then, in section 4 we will detail how we built a system on the top of the data collected, that allows cross-wikis querying and faceted browsing capabilities. This applications is built on existing Semantic Web querying standards (*i.e.* SPARQL) and uses PHP and the MIT's SIMILE Exhibit web interface. Finally, we will conclude the paper with a discussion on how Semantic Web can enhance wikis through an overview of our future works on the domain.

2. A UNIFIED ONTOLOGY FOR REPRESENTING WIKIS STRUCTURE

As discussed in the introduction, different wiki systems rely on different structures and APIs to represent their data. However, they all aim at representing the same information (at least a common subset), including main features of wikis such as the list of pages of the system, versioning information for each page, etc. Then, in order to enable interoperability between wiki system, we design an lightweight ontology to represent these characteristics equally whatever the original system is. Ontologies are at the core of the Semantic Web (a vision of the Web focused on data interoperability [5], moving from a Web of *documents* to a Web of *data*[6]) and provide *"the explicit specification of a conceptualisation"* [12], *i.e.* a shared vision of a domain. Our work then fits in the realm of Semantic Wikis, a vision integrating wiki philosophy and Semantic Web technologies, and more generally towards the Social Semantic Web [9], bringing together the Semantic Web and the Social Web.

2.1 Related Work

We can generally distinguish two kind of semantic wikis: (i) on the one hand, wikis using semantics to model the content of pages themselves, *i.e.* modeling facts contained inside wiki pages in RDF and (ii) on the other hand, wikis using semantics to represent their structure and the social interactions happening in wikis, enhance them using such modeling. The two approaches can also be combined, so that wikis would model both their content and their structure in RDF, offering new ways to identify the contributors around a particular fact, hence enabling advanced provenance management in wikis [21]. However, as we mainly focus on the second aspect of semantic wikis in this paper, *i.e.* common modeling of wiki structure, we will only focus on such wikis in this state of the art as well as on other representation models, developed independently of wiki platforms.

IkeWiki [22] uses a dedicated OWL ontology to represent the wiki structure, modeling users and pages but not taking into account features such as backlinks and versioning. It also defines mappings with FOAF — Friend Of A Friend[7] — and SIOC — Semantically-Interlinked Online Communities[8] — for representing discussion pages. SweetWiki [11] also has its own model, and takes versioning into account, as well as categories. However, it does not support discussion pages, and redefines its own classes for modeling authors and wiki pages. In addition, Semantic MediaWiki (SMW, a MediaWiki extension) [15] uses a particular ontology to represent the semantic data exported from a page by a user, named SWIVT – Semantic Wiki Vocabulary and Terminology but does not export all SMW features, including backlinks and categories.

Regarding generic models, we should mention WikiOnt [13], an ontology[9] for describing and exchanging wiki articles, aimed at integrating Wikipedia (and by extension other MediaWiki-based sites) into the Semantic Web framework, making Wikipedia machine-processable. It uses DublinCore to identify multiple authors of wiki pages as well as the editing date, and provides `Article` and `Category` classes. In addition, Wiki Interchange Format (WIF) [23] is a project that allows data exchange between wikis and related tools and tackles the problem of page content and annotations. It defines a subset of XHTML as an over-the-wire format for wiki content exchange, defining classes to model users (mapped with FOAF) and pages, and providing a versioning system, but not considering categories, social tagging, discussions and backlinks: features currently not modeled by the ontology.

2.2 Our Proposal

Motivated by the previous heterogeneity of models to represent wiki structure, and as none of them integrates all aspects of wiki features, we designed an ontology for modeling wikis on the Semantic Web. We built our model on the top of the SIOC ontology[10]. SIOC — Semantically-Interlinked Online Communities [8] — is now considered as one of the building blocks of the Social Semantic Web, since it is used in more than 50 applications, including Yahoo! SearchMonkey[11] and Drupal 7[12]. Therefore, building on the top of it allows exposure of our model in existing applications.

Typically wikis allow editing of documents and, by definition, allow multiple users to simultaneously contribute to the content; they track history of changes so that pages can be restored to previous modified versions; they include comments or discussion areas; they link to other external sources or within the wiki; they describe categories into hierarchical structures. For each of these features, we will now detail how we modeled it, using (and extending when needed) the SIOC Core ontology[13] and its Types module[14].

2.2.1 Representing Wikis and their Pages

Natively, the SIOC Types module already defines the `Wiki` and `WikiArticle` classes that can be used to represent the basic objects manipulated by wikis, *e.g.* wikis and their pages. We consequently reused these classes and added new properties to model additional features, as we will now discuss.

2.2.2 Multi-authoring

A fundamental feature of wikis is that multiple users are allowed to modify the same content, enabling some kind of collective intelligence process. In this regard the semantic infrastructure should provide a model to identify users and theirs modifications, marking events with a corresponding

[6]For further information, the reader could refer to the W3C FAQ on the topic — `http://www.w3.org/RDF/FAQ`

[7]A popular model to define people and social networks and the Semantic Web

[8]We will detail it in the next subsection

[9]`http://sw.deri.org/2005/04/wikipedia/wikiont.html`

[10]`http://sioc-project.org`

[11]`http://developer.yahoo.com/searchmonkey/`

[12]`http://groups.drupal.org/node/16597`

[13]`http://rdfs.org/sioc/ns`, prefix `sioc`

[14]`http://rdfs.org/sioc/types`, prefix `sioct`

timestamp so that provenance of information can be tracked between two versions.

This can be achieved using (i) the `sioc:UserAccount` class, representing a wiki user and (ii) the `sioc:has_creator` property, linking the edited wiki page to this user. Since multiple `sioc:has_creator` links can be linked to a single page, it enables a support for multi-authoring. We will also see next that this link is associated with the wiki version, in order to track information changes. Furthermore, `sioc:UserAccount` is defined as a subclass of `foaf:OnlineAccount`. In this way a `foaf:Person` (*i.e.* a physical person) could be linked to several `sioc:UserAccount` belonging to different wiki sites, enabling integration of one user's contributions across systems, a feature that we will also show in Section 4. Another way to model the relationships between pages and their authors is to reuse properties from the Dublin Core ontology, i.e. `dc:contributor` (or `dc:creator`) and `dcterms:created`. We actually distinguish between the creator of a version (*via* `sioc:has_creator`) and the contributors (*via* `dc:contributor`) who are the ones involved in previous version of the page, a useful distinction especially for cross-querying as we will detail in Section 4.1.

2.2.3 Categories

In many systems, wiki pages are generally related to categories, that allow readers to find sets of articles on related topics. Categories can also be organized in a tree-like structure and their semantic model should mantain the original taxonomical structure. In this regard an appropriate solution is provided by the SKOS[15] vocabulary [17], as it offers a way to model hierarchical structures between various categories, represented as instances of `skos:Concept`.

As regards the SIOC ontology, a `sioct:Category` class was already present into the SIOC Types module. Yet, it allowed only the modeling of a flat set of category names, without relationships each other. Hence, we extended it to declare this class as a subclass of a `skos:Concept`, giving it the ability to use the wide SKOS ontology capabilities to organize categories into advanced taxonomies, including properties such as `skos:broader`, `skos:narrower` and `skos:related`, enabling the modeling of hierarchies of categories. Moreover, thanks to the `sioc:topic` property, one can link any wiki page to such category.

2.2.4 Social tagging

While not all wiki engines support that feature, we believe this is particularly relevant, especially as it offers an open and user-driven classification scheme for wiki pages. The use of tags lead to a non-organised but dynamic organisation process, known as a "folksonomy", rather than the more widely used hierarchical structures.

The properties `sioc:topic` and `dc:subject` can be used to represent tags related to a particular wiki page, either using URIs for these tags (with `sioc:topic`) or simple keywords (`dc:subject`). In addition, vocabularies such as the Tag ontology [18], SCOT [14] or MOAT [20] allow to model tagging as tripartite actions (between a wiki page, a user and a tag) as well as organize tags together or link them to ontology concepts, in order to solve common tagging issues such as ambiguity between tags.

2.2.5 Discussions

Several wikis associate a discussion page to every wiki page, so that each user is able to comment and argue his point-of-view on the topic. This is moreover a default feature in MediaWiki, with a Talk page associated to each article. On a discussion page, people can discuss about the article subject, or about the way that subject is presented (see the Wikipedia's approach[16]). A first modeling solution could be to simply keep the native wiki text format of the wiki and just semantically link the discussion page to the related article page.

The SIOC's main class responsible for the modeling of a discussion is the `sioc:Forum` class, but there could be other specific classes that are more suitable for these discussion purposes, as defined in the Types module. The appropriate class to choose depends also on the type and style of the discussion page so there is a need to identify a proper attribute to link a wiki page to its discussion page. In this regard we decided to add a `sioc:has_discussion` property to the SIOC Core ontology, with domain `sioc:Item` and open range. This choice has been done in order to make this property reusable also in other contexts, for instance linking a simple webpage to a discussion forum. The discussions happening within the related `sioc:Forum` can then be modeled either as wiki-style discussions or threaded ones, and that feature also allows us to re-use advanced SIOC-based argumentative discussion modeling as defined in [16].

2.2.6 Backlinks

Backlinks are an important feature of wikis, as they allow to visualize instantaneously all the incoming links to a website or web page. More precisely they are wiki internal links pointing to a wiki article. It is a very common wiki feature and they may be of significant interest: they indicate who is paying attention to the linked page or topic.

We modeled this feature using the already existing `sioc:links_to` property. This property identifies links extracted from hyperlinks within a SIOC concept and is a subproperty of `dcterms:references`. It is important to remember that this property has to be defined into the RDF description of the original wiki article which links back to the wiki article. Hence, to model for instance that the Wikipedia page about "DERI" features a backlink from the page about "RDF", the following statement would be added into the RDF description of DERI's page[17].

```
<http://en.wikipedia.org/wiki/
    Resource_Description_Framework> sioc:
    links_to
 <http://en.wikipedia.org/wiki/
     Digital_Enterprise_Research_Institute
     > .
```

Listing 1: Representing wiki backlinks with SIOC

2.2.7 Versioning

Usually all editable pages on wikis have an associated page history. This history consists of the old versions of the wikitext, as well as a record of the date and time of every edit,

[15] `http://www.w3.org/2004/02/skos/`

[16] `http://en.wikipedia.org/wiki/Wikipedia:Talk_page_guidelines`

[17] For space reasons, we did not include prefixes in listings and queries of this paper.

the username or IP address of the user who wrote it, and their edit summary. All this is usually accessible through a special "history" page which shows time-ordered links to all the revisions. Commonly the latest revision of a wiki page has always the same URL (alias name), meanwhile older versions have further parameters appended to the URL.

As we have seen, our model for multi-authorship allows to represent the authors involved in a page at a particular time. However, there is a need to link a page to its previous version. This could be modeled in several ways. Considering the existing versioning approaches from the state of the art, we came to the conclusion that none completely fullfil its needs and we thus designed a new versioning model. An important requirement we took into account is to enable fast and simple browsing capability using such model.

For this reason, we decided to use transitive properties to express the temporal relations between revisions of a wiki page. Transitive properties, defined as `owl:TransitiveProperty`[18], enable, combined with an inference engine, to infer new facts based on an initial seed fact. Then, we defined two kind of properties to model versioning, the model being displayed in Fig. 1:

- `previous_version` and `next_version` properties, linking directly an instance of `sioc:Item`, or any subclass such as `sioct:WikiArticle` to its previous (or next) version, and
- `earlier_version` and `later_version`, defined as super-properties of the previous ones and as transitive properties.

Using these definitions, the following rules are automatically applied by a reasoning engine as soon as it gets a set of wiki pages and of `previous_version` or `next_version` properties in its dataset:

$$\forall(x,y,z), previous\_version(x,y) \sqcap previous\_version(y,z) \quad (1)$$
$$\longrightarrow earlier\_version(x,y) \sqcap earlier\_version(y,z) \quad (2)$$
$$\longrightarrow earlier\_version(x,z) \quad (3)$$

Hence, using an OWL level reasoning engine, when modeling a `WikiArticle` (or a `sioc:Item` in general), it is only necessary to describe its previous and next revision and the transitive super-properties will be automatically inferred by the system. This can also be convenient during the querying process (described in Section 4.1): if one needs to get all the earlier versions of a wiki page, we benefit from the transitivity of `sioc:earlier_version`, while if not using this, we would have had to be implemented a query that recursively go through each `sioc:previous_version` of the latest wiki article.

Another introduced property is the `sioc:latest_version` which points always to the newest revision. Usually it is used in combination with an alias name of the latest version so that it is not necessary to change the referred URI in all the earlier versions as soon as a modification happens. All the wikis we analyzed adopt this solution as it addresses scalability.

[18] `http://www.w3.org/TR/owl-ref/#TransitiveProperty-def`

3. EXPORTING SIOC DATA FROM HETEROGENEOUS WIKIS

Once a common interchange model for wikis has been defined, in order to evaluate our proposal, we decided to generate and collect a substantial amount of structured data in RDF/XML format generated from different wiki platforms. First, a webservice that exports every wiki page from the MediaWiki software platform in RDF has been developed (Section 3.1). This exporter is called "SIOC-MediaWiki exporter" and it is publicly available on the Web. Our attention was focused on the MediaWiki platform simply because it is one of the most popular wiki platforms on the Web, hosting all the Wikimedia Fundation wikis (i.e. Wikipedia, Wiktionary, etc.) and propulsing more that 45 millions of wiki articles from different wiki sites[19].

The second wiki platform we chose is DokuWiki[20], which is another popular wiki platform together with TWiki and MoinMoin, aimed at small companies' documentation needs and particularly suited for fast and easy setups and configurations since it does not need a database. We focused on this wiki software also because a plug-in for DokuWiki that exports RDF data, and especially SIOC ontology based data, has been already developed by Michael Haschke[21], a contributor of the SIOC project's community. However, it was not fully-compliant with our proposed model and some features we needed were missing, therefore we adapted and improved this plug-in implementation in order to meet the requirements we established (Section 3.2).

Exporting and collecting data extracted from two different and relevant wiki platforms allows us to evaluate and demonstrate the validity of our approach, and gives us the possibility to run and experiment cross-wikis and cross-platforms queries and to show some of the potentialities that Semantic Web technologies could have, as we will show in Section 4.

3.1 The SIOC-MediaWiki exporter

The SIOC-MediaWiki webservice is written in PHP and is publicly available at `http://ws.sioc-project.org/mediawiki/`. It exports any MediaWiki wiki article in RDF using the structure we explained in the previous sections. It is relatively lightweight and built thanks to two PHP classes: the SIOC-Mediawiki exporter itself and the already existing SIOC API[22]. The latter has been improved in order to take the new characteristics of the model into account. The exporter class is the part responsible for querying the MediaWiki API and parsing the results, and the SIOC API is responsible for exporting the content in RDF. The script automatically discovers the MediaWiki API location of the requested wiki, then it connects to the API with HTTP GET requests as queries. After parsing the results of the queries it calls the SIOC API to export in RDF/XML serialization the fetched structural information.

Since the initial release of the exporter [19], we focused on improving the performances of the application, especially in terms of response time. This is a very important require-

[19] `http://s23.org/wikistats/largest_html.php` checked on March 2010

[20] `http://www.dokuwiki.org/`

[21] `http://eye48.com/dokuwiki/doku.php?id=en:dokuwiki:sioc-plugin`

[22] `http://wiki.sioc-project.org/index.php/PHPExportAPI`

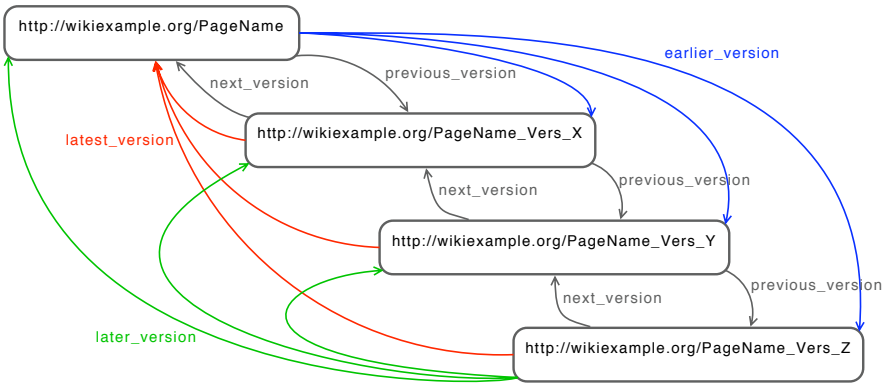

Figure 1: Pages versioning model with SIOC properties. Note that, for clarity, transitive properties `earlier_version` and `later_version` are only displayed for two wiki articles: the latest one and the first one.

ment: considering the process of crawling a wiki using the exporter, even a small reduction of the time needed to export a single wiki page would lead to a consistent amount of time saved when collecting data for all the pages in an entire wiki. Unfortunately the exporting time with the SIOC-MediaWiki webservice is strongly dependent on (i) the time of response of the API of the original MediaWiki system that is exported and (ii) on the number of queries needed to get all the data. The second aspect on which we concentrated our attention was the way users and anonymous users are modeled, in particular the anonymous users need to be modeled and they can only be linked to a blank node. Finally the possibility to finely select the relevant wiki structural features to export has been added. Then, users can decide to export only some basic information on a wiki article and ignore other information, for instance exporting revisions but no categories, instead of being always forced to export them all. This enables better usage of the applications, as third-party developers can concentrate on extracting only the required subset of the original systems.

3.2 The DokuSIOC plugin for DokuWiki

The main functionalities offered by DokuWiki are extensible by implementing plugins, called Action Plugins, which are designed to work with DokuWiki events to allow for customization of any part of DokuWiki that signals its activity using events. The DokuWiki documentation[23] gives detailed information on their structure and how to develop new plugins. In this section, we focus more on DokuSIOC, a plugin developed by Michael Haschke[24], and how we extended it to fit with our SIOC extensions. Action plugins are loaded before any significant DokuWiki processing takes place. At load time they register their event handlers so that when a specific event is signaled all event handlers registered for that event are called. Hence plugins have the opportunity to alter either the event data or the event's subsequent processing.

The DokuSIOC plugin takes information from the metadata stored in the wiki system about pages, users, links, and revisions and provides it as raw RDF/XML serialized data (instead of the usual HTML page) if asked for it. Futhermore DokuSIOC provides several different ways to offer its service to clients. Simple ways are simply to add the GET parameter `do=export_siocxml` to the URL of a wiki page, or to follow the meta link added to the header of the DokuWiki HTML view. Another option is based on the content-negotiation capability: if the client requests the usual URL of the page with an HTTP header asking for *application/rdf+xml*, the plugin will forward to the location of the RDF export view. These options are particularly useful as regards the crawling process of a DokuWiki wiki using a common RDF crawler which can automatically discover the linked RDF data.

The semantic model used by the DokuSIOC plugin after our modifications reflects exactly the model we detailed in the previous sections and, of course the one used by the SIOC-MediaWiki exporter. One of the problems encountered when developing this DokuWiki plugin relates to the internal handling of user identifiers and profiles in it and consequently how to model URI for users. In this case the DokuSIOC plugin was already offering a way to configure a DokuWiki namespace, where user identifiers can be used as sub pages. The following URI structure `http://[dokuwikiurl]/doku.php?id=user:username` provides the identifier for the user account on the wiki. Moreover a usual DokuWiki URL can stand for different resources, any URL may describe either a user as a `sioc:UserAccount` or a wiki page as a `sioct:WikiArticle` or a container (in this case a specific `sioct:Wiki` wiki container is more appropriate). In this regard, the SIOC plugin adds a type parameter to distinguish exactly between the resources types. Different URI structures are then used depending on the context, *e.g.* `http://[dokuwikiurl]/doku.php?id=user:username&type=user` for a user and `http://[dokuwiki]/doku.php?id=pageid&type=post` for an article.

To generate RDF data out of the metadata extracted from the wiki system and easily create SIOC documents, DokuSIOC uses the SIOC PHP API[25] that we have developed in the past. An important change we have made in the plugin implementation has been to use the SIOC PHP API as much as possible to create the SIOC objects such as the `sioct:WikiArticle`, the `sioct:Wiki` container, the `sioc:UserAccount` etc. In the previous implementation of the plugin there was a PHP class that was acting as a mediator between the main `Action` class of the DokuWiki plu-

[23] `http://www.dokuwiki.org/devel:action_plugins`
[24] `http://eye48.com/go/dokusioc`
[25] `http://wiki.sioc-project.org/index.php/PHPExportAPI`

gin and the `sioc_inc.php` script of the SIOC API. This "intermediate" class was used to change and customize the behavior of the SIOC API in order to create personalized objects using the methods provided by the SIOC API. In our perspective the SIOC API gives us all the instruments and objects we need in order to have the same wiki modeling between the different wiki platforms and to keep the interoperability between them. Hence we decided to keep the same structure we used for the SIOC-MediaWiki exporter and relay on the SIOC API implementation[26]. In particular our changes have been focused on the properties used to define the contributors of the articles (we use the `sioc:has_creator` to point to the `sioc:UserAccount` and the `dc:contributor` for the username as literal), and the date of creation of each article revision (with `dcterms:created`). Furthermore we changed the way the backlinks were modeled deciding to keep the same `sioc:links_to` property used for forward links. This particular choice would ease the querying part of our work because we have always the same property expressing links between the articles, no matter if they are back/forward or internal/external links.

Another relevant contribution we made to the DokuSIOC plugin was to add the "external links" feature which was not implemented. Indeed, DokuWiki does not provides native metadata about the external links linked from each page. Our exporter consequently parses all links from HTML articles and extract the external ones. Once the extraction has been made by the "Action" main class of the plugin, we export them using the same criteria as the internal links.

3.3 Following the Linked Data Principles

The main goal of our work with the implementation of two different exporters sharing the same data model was not only to create RDF data from any MediaWiki or DokuWiki page, but also to easily allow interlinking between various wiki platforms, as well as between wiki data and other RDF data, whatever it is social data modeled with FOAF or SIOC or any other kind of RDF data. To do so, we followed the Linked Data principles defined by [4] and the related best practices [3] [6]: (i) use URIs as names for things; (ii) use HTTP URIs so that people can look up those names; (iii) when someone looks up a URI, provide useful information, using the standards (RDF, SPARQL); (iv) include links to other URIs. so that they can discover more things.

Particularly, to offer a better browsing experience and ease the process of crawling SIOC exports of MediaWiki instances, our webservice automatically produces `rdfs:seeAlso` links between wiki pages. Actually, more than a simple link to the wiki page, the exporter provides a link to the related RDF document, as we can see in Listing 2 related to the export of a particular `sioc:UserAccount`. As we can also notice in that example, we distinguish the concept itself (i.e. `User:StefanDecker`) and the related RDF page.

```
<sioc:UserAccount rdf:about="http://en.
    wikipedia.org/wiki/User:StefanDecker">
 <rdfs:seeAlso rdf:resource="http://ws.
     sioc-project.org/mediawiki/mediawiki.
     php?wiki=http://en.wikipedia.org/wiki
     /User:StefanDecker"/>
</sioc:UserAccount>
```

Listing 2: User Modeling in the MediaWiki exporter

These `seeAlso` links are very useful not only to provide link to other related RDF documents, that can be used for instance when browsing data with Tabulator, but also in a crawling perspective. A RDF crawler could easily follow all the `seeAlso` links found on every document and continue to crawl. In this regard, for example, we crawled and exported entire wiki sites just following these links. A different approach, but with the same scope, has been adopted with the DokuSIOC plugin. As described in the previous section using content-negotiation it is possible to switch between the standard HTML view of the wiki article and its RDF representation, moreover a meta link added to the header of the DokuWiki HTML view points to the semantic representation of each article easing the RDF data discovery process.

In a Linking Open Data perspective a relevant opportunity is the association between the wiki user's `OnlineAccount` and the `foaf:Person` holder of the account. And this is possible with the `foaf:holdsAccount` property. Using this feature it becomes possible to interlink precisely all the user accounts on different wikis belonging to the same person and then, for example, to know what are the contributions made by the same persons on different wikis, what are their interest areas, etc. At the moment it is possible but since most of the wiki users do not provide their FOAF profile, we still have to use the username as a literal, with all the ambiguities and inaccuracies that this method brings.

Another interesting feature is the linkage to the corresponding DBpedia[27] resource (DBpedia being the RDF export of WikiPedia [2]), if the article belongs to the english Wikipedia. Since DBpedia semantically models the content of a Wikipedia page, this connection is very useful to link semantic data about the content and the structure of a wiki article. DBpedia resource URIs are used in range of the `foaf:primaryTopic` property, as this property relates a document to the main thing that the document is about. Obviously this linkage between DBpedia and Wikipedia is immediately possible only with the MediaWiki exporter, since Wikipedia is based on the MediaWiki software. However, we are currently working on topic extraction from pages belonging to other wikis, so that it would be possible to link every wiki page to the related Wikipedia/DBpedia categories or even to corresponding similar articles, enabling better interlinking capabilities across wikis.

4. APPLICATIONS FOR CROSS-WIKIS SEMANTIC SEARCH

In this section, we will detail how we designed a Semantic Web-based application using semantic data generated from the previously detailed systems. In particular our main objective is to show that the wiki model we propose allows for interoperability between wiki platforms, and that Semantic Web technologies can (i) really improve our usual wiki experience based on typical Web 2.0 applications and (ii) permit to discover new knowledge in a faster and more accessible

[26]Another advantage of relying on the API is that any changes on the SIOC Ontology are immediately replicated in the API. Then, the DokuWiki plugin (as well as the MediaWiki one) are constantly up-to-date with the ontology changes, with only a few efforts (simply loading the new API version in the exporters).

[27]`http://www.dbpedia.org`

way.

As a first step, we exported and crawled different MediaWiki and DokuWiki instances. Five different wikis have been crawled, four from the MediaWiki platform and one from the DokuWiki one. Each MediaWiki site has been crawled using a single entry point thanks to the use of the `rdfs:seeAlso` links. The DokuWiki wiki has been installed locally and a subset of the data from the official PHP wiki has been imported in it[28] (since our DokuWiki plug-in is not implemented in that wiki). It is important to note that each wiki we crawled belongs approximately to the same area of interest in order to have a high probability of shared topics and users. The MediaWiki sites collected are: *Semanticweb.org*[29], *Protégé Wiki*[30], *RDFa Wiki*[31] and the *ONTOLORE Karlsruhe* wiki[32], all focusing on Semantic Web technologies, with shared contributors as we will see next.

In total, we collected about 1GB of RDF data and loaded it in the OpenRDF Sesame triple-store[33] [10]. As we needed an higher degree of inference (because of the OWL transitive properties) we also installed and configured the reasoning engine OWLIM[34] on the top of it. The crawling process of all the wikis took about one entire day (24 hours), and every operation has been made on only one single-core machine. In total we collected around 45,500 triples, 3,400 wiki articles and 700 users. Once all the data has been collected it has been inserted in a Sesame+OWLIM triple-store. This process, because of the OWL inference (new triples are entailed at loading time in Sesame+OWLIM), took around two hours to be completed on the same machine, but then every query ran with the SPARQL endpoint did not take more than 3 seconds to be executed, in spite of the complexity of some of them, as we will see. As regards the scalability of the system our implementation is completely independent by the underlying triple-store. Several RDF stores have been demonstrated as capable to address the scalability requirement with a large amount of data. A comprehensive study, and a benchmark experiment, comparing the performance of popular RDF stores has been conducted in [7].

After this configuration step, the system was ready to be tested with SPARQL queries. In the following section some of the advanced queries we ran are detailed. Then, in Section 4.2 we will describe the structure of the application for semantic search and faceted browsing we built on top of the triple-store and its SPARQL endpoint.

4.1 Advanced Querying and Cross-wiki Integration

Since our data has been loaded in an RDF store, all the queries were done using SPARQL — SPARQL Protocol And RDF Query Language [1] —, the W3C standard for querying RDF data. As we can see, it offers the advantage of having a single and standard language to query wiki data, while developer that need to query original systems have to learn a new API for each new system we want to query. Then, we solved one issue that we mentioned originally in our motivation, *i.e.* the problem of having different ways to query different wikis.

A first example of advanced querying for a particular wiki is the ability to answer to the following question: "what are the collaborating users that worked alternatively on the same wiki article?". In Listing 3 we provide the SPARQL implementation of this query.

```
SELECT DISTINCT ?wikiArt ?Contrib_a ?
    Contrib_b
WHERE {
 ?x sioc:latest_version ?wikiArt.
 ?wikiArt sioc:earlier_version ?VersA .
 ?VersA sioc:earlier_version ?VersB ;
  dc:contributor ?Contrib_a .
 ?VersB sioc:earlier_version ?VersC ;
  dc:contributor ?Contrib_b .
 ?VersC dc:contributor ?Contrib_a .
 FILTER (?Contrib_a != ?Contrib_b) .
}
```

Listing 3: Identifying collaborating users

As we can see, this query takes advantage of the transitivity of the newly created property `sioc:earlier_version`, since we identify users that worked on earlier versions, and not only immediately on the previous one. The query provides the article URI and the two usernames in case the first user (`?Contrib_a`) re-edited the article after a modification made by the second user (`?Contrib_b`). It enables people to look for users sharing the same interests and knowledge areas. It can be also very important especially in a Social Semantic Web context.

Another interesting feature of our approach is the ability to do cross-wikis querying, since wikis are now based on the same model. The following query, in Listing 4, identifies users involved in different wikis, looking for the same usernames.

```
SELECT DISTINCT ?creator1 ?page1 ?page2 ?
    wiki1 ?wiki2
WHERE {
 ?page1 sioc:has_container ?wiki1 ;
  dc:contributor ?creator1 .
 ?page2 sioc:has_container ?wiki2 ;
  dc:contributor ?creator2 .
 FILTER (str(?creator1)==str(?creator2)) .
 FILTER (str(?wiki1)!=str(?wiki2)) .
}
```

Listing 4: Identifying pages created by a single user in different wikis

Yet, as this query relies on a FILTER clause, it will identify common users only if they use the same account name on two different wikis. Moreover, we can imagine that some common account names will be used by different people on different wikis, e.g. JohnSmith. To that extend, we can benefit from the strong ties that exist between FOAF and SIOC and the fact we are modeling a wiki user using the `sioc:UserAccount` class. One person can indeed define in his FOAF profile the various wiki accounts he owns, using simple `foaf:holdsAccount` properties. Then, the previous query can be adapted to deal not only with text strings to identify the user, but with their related accounts from the FOAF URI, so that a single query can be used to retrieve all the contributions of a user whatever the wiki used was. Moreover, since the wiki model is based on SIOC, the same

[28]The official PHP.net wiki: `http://wiki.php.net/`
[29]`http://www.semanticweb.org`
[30]`http://protegewiki.stanford.edu`
[31]`http://rdfa.info/wiki/RDFa_Wiki`
[32]`http://logic.aifb.uni-karlsruhe.de/wiki/ONTOLORE`
[33]A triple-store, or RDF store, aims at storing RDF data and providing querying interfaces for it.
[34]`http://www.ontotext.com/owlim/`

query can be used to retrieve wiki pages, blog posts, etc. as follows.

```
SELECT DISTINCT ?content
WHERE {
 <http://example.org/js#me> foaf:
     holdsAccount ?account .
 ?account rdf:type sioc:UserAccount .
 ?content sioc:has_creator ?account .
}
```

Listing 5: Cross-sites query using FOAF and SIOC

4.2 Enabling Semantic Search

The application we built to show the potential of semantic technologies applied to wikis has the typical architecture of many Semantic Web applications. Its structure can be divided in three layers concerned with storage, querying or data acquisition, and visualization. In the previous sections we already described the storage part of the system: it is based on a Sesame+OWLIM triple-store with the data we crawled from different wikis, and it exposes a SPARQL endpoint where is possible to have an interface with the querying and acquisition module.

As regards the data acquisition module we wrote a PHP script that queries our triple-store, collects and parses the results and translates the data in the correct format for the visualization layer. The PHP script is the core of the application, and in this specific application it basically needs to run two different SPARQL queries to obtain the necessary data, but it can be personalized very easily with regard to the particular desired use-case.

The visualization layer has been built with the SIMILE Exhibit framework[35]. This framework allows developers to create (X)HTML pages with dynamic exhibits of data collections which can be searched and browsed using faceted browsing capabilities. Exhibit is a set of Javascript files that run in a user's browser. All it needs is a graphical configuration and personalization made directly on the HTML code of the page to display and to receive data built with a correct structure and a supported format. The most used format with Exhibit is JSON and in our specific case this is what we adopted. In this regard our PHP script converts the XML data returned by the SPARQL queries into the JSON format.

Once the username of a wiki user has been introduced in the first page, the application provides two different sections of information. The first one is about all the wiki users who contributed on the same wiki articles as the requested user did. In other words it looks for her/his co-authors distributed on several different wikis. The second one provides details about all the articles contributed by the user in every wiki and the related topics of interest.

In Fig. 2 we display a screenshot of the developed web application. As we can see from the image, in the first horizontal section there are three lists (or facets) showing the co-authors with the related wiki articles in common and the list of wikis on which the articles are located. Every element of the facets is selectable and once selected it filters all the other results on the other facets. The first section of results is obtained by the first query formulated by the PHP script. The SPARQL query used in this case is displayed in Listing 6 and it selects the wiki site, the wiki article and the related co-author of the user "MichaelHausenblas".

[35] http://www.simile-widgets.org/exhibit/

```
SELECT DISTINCT ?wiki ?title ?coauthor
WHERE {
  ?pag1 dc:contributor ?me. FILTER regex(?
      me, "MichaelHausenblas", "i").
  ?pag1 dc:title ?title ;
    sioc:has_container ?wiki .
  ?pag2 dc:title ?title2 . FILTER regex(
      str(?title), str(?title2)).
  ?pag2 dc:contributor ?coauthor . FILTER
      ((?coauthor) != (?me)).
}order by ?wiki
```

Listing 6: First query of the application

The second section of results, obtained by the second SPARQL query, displays all the articles contributed by the searched user on different wiki sites. It also adds a list of the categories (in the range of the `sioc:topic` property) related to each wiki article extracted. In other words this particular view highlights the activities, the interests and the expertise areas of the searched user. The query formulated by the script for this section is displayed in the following Listing 7.

```
SELECT DISTINCT ?wiki ?title ?category
WHERE {
  ?pag1 dc:contributor ?me. FILTER regex(?
      me, "MichaelHausenblas", "i").
  ?pag1 dc:title ?title ;
    sioc:has_container ?wiki ;
    sioc:topic ?category.
}ORDER BY ?wiki
```

Listing 7: Second query of the application

The last feature the SIOCWiki browser shows is a dynamic list displaying all the results extracted by the previous two sections. The results here are more detailed and they can be easily grouped and sorted. They are also filtered by the events triggered by the facets above.

4.3 Advantages of the Semantic Web Approach Compared to the Original Web 2.0 One

The Semantic Web approach we showed with this application can be compared to the currently widely adopted Web 2.0 approach. Following the Web 2.0 way, in order to obtain similar results and functionalities, we would have to use each software platform separately. For example, to obtain the list of all the co-authors of one particular user we would have to: first go to the page of the user in each wiki platform; then use some special service provided by the wiki software to obtain her or his contributions; then, for each contribution, retrieve the history and identify all users. In addition, that workflow assumes that the wiki service provides the list of the contribution for every user, which is true for the MediaWiki platform but not for the DokuWiki one. Then, we not only simplified the process (MediaWiki) but also added some features that could not have been provided with the original tool.

Another option would be to develop some platform-specific applications which use the specific wiki software API. Once again, the interoperability is lost together with the cross-wiki global view of the data. Hence we might state that the Web 2.0 approach can still be an option for use-cases where the cross-platform interoperability is not needed and

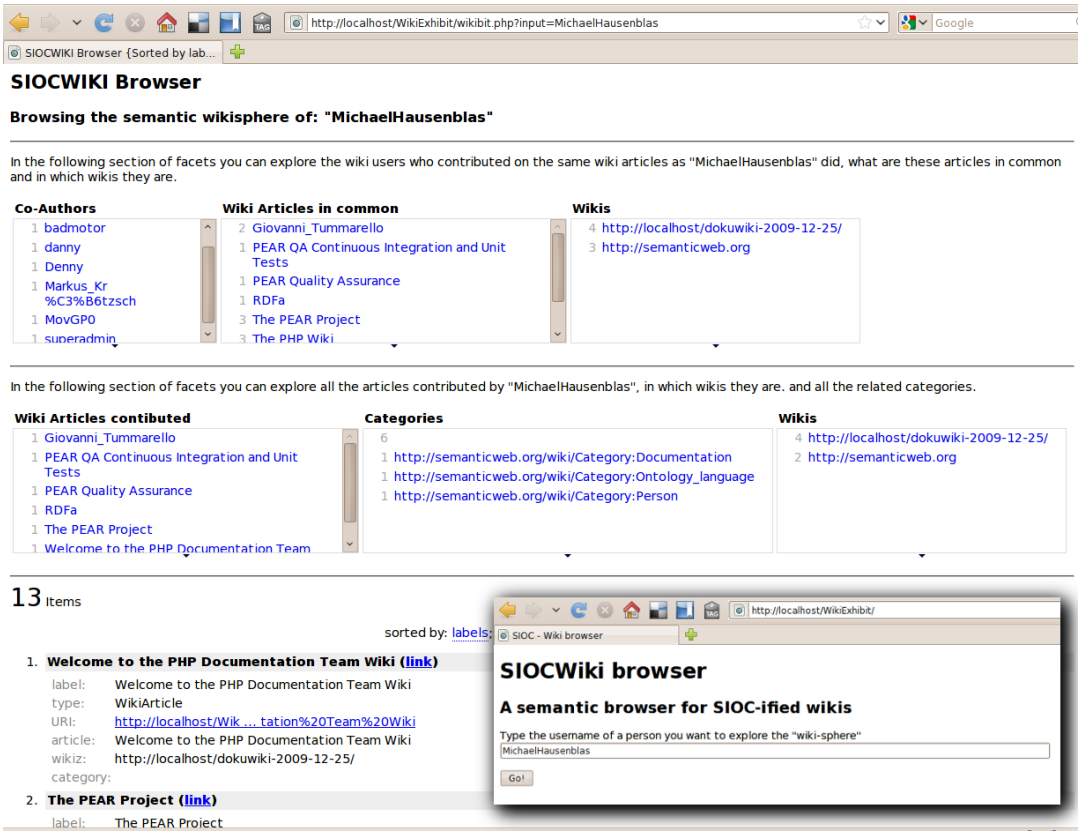

Figure 2: SIOCWiki Browser: a screenshot showing the results found for the username "MichaelHausenblas".

the number of the queries is limited, since these services are already available on the Web and do not require to build an infrastructure as ours. On the other hand, the Semantic Web approach needs initially more time to set-up the system (notably because of crawling and storing data) but then allows for advanced and fast querying processes and hidden knowledge discovery. It is also particularly suited for use-cases such as the one we exposed with this work, namely to build an application that can be accessible to everyone and easily customizable and integrating data from different sources, based on different platforms.

5. CONCLUSION

In this paper we proposed and described a model to represent the social and structural wiki features in an unified way using the SIOC ontology and lightweight semantics. Then we detailed how we extracted semantic data from wikis using two relevant platforms: Mediawiki and Dokuwiki; and we described how different types of exporters can be built on these types of wikis. The design criteria of a webservice exporter for the MediaWiki platform and a plugin for DokuWiki have been made following the Linked Data best principles in order to provide interlinked data. On the top of the data extracted from the aforementioned exporters a semantic search system has been built. It provides a user-friendly interface and advanced features to retrieve information contained in heterogeneous wikis in a unified way. Despite its simplicity, the presented application allows for advanced and fast querying processes and hidden knowledge discovery, showing potentialities that cannot be obtained using the traditional Web 2.0 instruments. Hence we showed an overall benefit on applying Semantic Web technologies to wikis, enabling users to access the information generated by this process in a simple and transparent way.

With the present work we developed a lightweight Semantic Web application to demonstrate the capabilities of semantic technologies applied to wikis, but further developments may go in the direction of creating a stable and widespread service available on the Web. In this regard our current research is involved in finding ways to interlink wiki articles also reasoning on the content of a page. Moreover we are considering to add real-time search capabilities to the presented architecture, so that it would be possible to entirely crawl a wiki site only the first time and then keep the data updated following the wiki changes in real-time.

6. REFERENCES

[1] SPARQL query language for RDF. W3C Recommendation 15 January 2008, World Wide Web Consortium, 2008. `http://www.w3.org/TR/rdf-sparql-query/`.

[2] Sören Auer, Chris Bizer, Jens Lehmann, Georgi Kobilarov, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007)*, volume 4825 of *Lecture Notes in Computer Science*, pages 715–728. Springer, 2007.

[3] Danny Ayers and Max Völkel. Cool URIs for the Semantic Web. W3C Interest Group Note 03 December 2008, World Wide Web Consortium, 2008. http://www.w3.org/TR/cooluris/.

[4] Tim Berners-Lee. Linked Data. Design issues for the world wide web, World Wide Web Consortium, 2006. http://www.w3.org/DesignIssues/LinkedData.html.

[5] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.

[6] Chris Bizer, Richard Cyganiak, and Tom Heath. How to Publish Linked Data on the Web. Technical report, 2007. http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/.

[7] Christian Bizer and Andreas Schultz. The berlin sparql benchmark. *International Journal On Semantic Web and Information Systems*, 2009.

[8] John G. Breslin, Andreas Harth, Uldis Bojārs, and Stefan Decker. Towards Semantically-Interlinked Online Communities. In *Proceedings of the 2nd European Semantic Web Conference (ESWC2005)*, volume 3532 of *Lecture Notes in Computer Science*, pages 500–514. Springer, 2005.

[9] John G. Breslin, Alexandre Passant, and Stefan Decker. *The Social Semantic Web*. Springer, 2009.

[10] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *The Semantic Web - ISWC 2002. First International Semantic Web Conference*, volume 2342 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2002.

[11] Michel Buffa, Fabien L. Gandon, Guillaume Ereteo, Peter Sander, and Catherine Faron. SweetWiki: A semantic wiki. *Journal of Web Semantics*, 6(1):84–97, 2008.

[12] Thomas. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies*, 43(5–6):907–928, 1995.

[13] Andreas Harth, Hannes Gassert, Ina O'Murchu, John G. Breslin, and Stefan Decker. WikiOnt: An Ontology for Describing and Exchanging Wikipedia Articles. In *Proceedings of Wikimania 2005 – The First International Wikimedia Conference*, 2005.

[14] Hak Lae Kim, Sung-Kwon Yang, John G. Breslin, and Hong-Gee Kim. Simple algorithms for representing tag frequencies in the scot exporter. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 536–539. IEEE Computer Society, 2007.

[15] Markus Krötzsch, Denny Vrandecic, and Max Völkel. Semantic MediaWiki. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, volume 4273 of *Lecture Notes in Computer Science*, pages 935–942. Springer, 2006.

[16] Christoph Lange, Uldis Bojārs, Tudor Groza, John G. Breslin, and Siegfried Handschuh. Expressing Argumentative Discussions in Social Media Sites. In *Proceedings of the ISWC2008 Workshop on Social Data on the Web (SDoW2008)*, volume 405 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[17] Alistair Miles and Sean Bechhofer. SKOS Simple Knowledge Organization System Reference. W3C Working Draft 29 August 2008, World Wide Web Consortium, 2008. http://www.w3.org/TR/2008/WD-skos-reference-20080829/.

[18] Richard Newman, Danny Ayers, and Seth Russell. Tag ontology, December 2005.

[19] Fabrizio Orlandi and Alexandre Passant. Enabling cross-wikis integration by extending the SIOC ontology. In *Proceedings of the Fourth Workshop on Semantic Wikis (SemWiki2009)*, 2009.

[20] Alexandre Passant and Philippe Laublet. Meaning Of A Tag: A collaborative approach to bridge the gap between tagging and Linked Data. In *Proceedings of the WWW2008 Workshop Linked Data on the Web (LDOW2008)*, volume 369 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[21] Alexandre Passant and Philippe Laublet. Towards an Interlinked Semantic Wiki Farm. In *Third Semantic Wiki Workshop – The Wiki Way of Semantics*, volume 360 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[22] Sebastian Schaffert. IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In *First International Workshop on Semantic Technologies in Collaborative Applications (STICA 06)*, 2006.

[23] Max Völkel and Eyal Oren. Towards a Wiki Interchange Format (WIF) - Opening Semantic Wiki Content and Metadata. In *Proceedings of the First Workshop on Semantic Wikis - From Wiki to Semantics (SemWiki-2006)*, volume 206 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.