A Semantic Web Approach for Geodata Discovery

Helbert Arenas, Benjamin Harbelot, and Christophe Cruz

Laboratoire Le2i, UMR-6302 CNRS,Departement Informatique,
University of Burgundy, 7 Boulevard Docteur Petitjean, 21078 Dijon, France
{helbert.arenas,benjamin.harbelot}@checksem.fr,
christophe.cruz@u-bourgogne.fr
http://checksem.u-bourgogne.fr/www/

Abstract. Currently, vast amounts of geospatial information are offered through OGC's services. However this information has limited formal semantics. The most common method to search for a dataset consists in matching keywords to metadata elements. By adding semantics to available descriptions we could use modern inference and reasoning mechanisms currently available in the Semantic Web. In this paper we present a novel architecture currently in development in which we use state of the art triplestores as the backend of a CSW service. In our approach, each metadata record is considered an instance of a given class in a domain ontology. Our architecture also adds a spatial dataset of features with toponym values. These additions allow us to provide advance searches based on 1) Instance to class matching, 2) Class to class subsuming relationships, 3) Spatial relationships resulting from comparing the bounding box of a metadata record with our toponym spatial dataset.

Keywords: Semantic Web,ontologies, geodata discovery, catalogues, OGC services.

1 Introduction

Currently there is a vast amount of spatial information available on the web though services such as WFS, WMS or SOS to mention some. This information allows scientists to perform complex analysis. Goodwin (2005) used the term *smart queries* to describe analysis that combine heterogeneous datasources in order to solve complex problems [1]. Our field of interest is the use of heterogeneous datasources to perform spatio-temporal *smart queries* using Semantic Web tools. In previous work [2] we presented our research on spatio-temporal operators, using local data repositories. The next logical step in the evolution of our work is to integrate it to the SDI (Spatial Data Infrastracture). The term SDI was first introduced by the U.S. National Research Council in 1993. It refers to a set of technologies, policies and agreements designed to allow the sharing of spatial information and resources between institutions [3]. The Spatial Data Infrastructure has a service oriented architecture. In such infrastructure, functionalities such as storage and data search are carried out through Web services.

J. Parsons and D. Chiu (Eds.): ER Workshops 2013, LNCS 8697, pp. 117–126, 2014.

The typical work flow involves: 1) The discovery of a data source, 2) The download of relevant geo spatial data, 3) The use of appropriate analytical methods and 4) The visualization of the results on a suitable map.

In order to integrate our work with the SDI we need to find and retrieve pertinent online datasets. OGC has introduced a standard for catalogue services called CSW. A server implementing this standard has access to a metadata repository. It allows the search of spatial data or web services using open criteria (i.e. free text as a search in a search engine) or using more specific criteria (title, coordinate system, data type, etc.). Servers implementing CSW are also able to: synchronize their content with other catalogues, add, modify or delete metadata records. They are also able to harvest metadata from other OGC services [4]. Like other OGC standards CSW implements *OWS Common*, which describes basic features shared by all of them. These common elements are basic parameters and data structures used in the request or response from web service operations. The standardization proposed through OWS Common serves as support for the interoperability of OGC web services [5].

Our broad goal is to implement *smart queries* using data repositories available in the SDI. The use of the different services involved in a SDI raise several semantic challenges. Most semantic problems arise due to the lack of significant content descriptions. As a consequence the resulting ambiguities are then propagated throughout the process of data [6]. The discovery of datasets is currently done though OGC CSW. However the search of records is done using a string matching process, not considering the semantics of the metadata information [7]. Greatly improved results would be obtained if catalogue services are able to use the hierarchical relationships between elements and concepts achieving in this way semantic interoperability [8]. Fortunately, nowadays there are tools in the field of semantic web, like Sesame, Jena, SPARQL/GeoSPARQL and diverse triplestores, that offer storage, query and retrieval capabilities of information with semantic annotations. In this paper we present our work on the storage and retrieval of metadata records using a triplestore. We developed a proof of concept CSW service that is able to retrieve metadata records by mapping a subset of CQL operators to SPARQL/GeoSPARQL. In section 2 we describe other works in this field. In section 3 we describe our implementation. Finally we present our conclusions and future work in section 4.

2 Related Research

Catalogues are a core component of the Spatial Data Infrastructure. The most used catalogue specification is the one proposed by the OGC: CSW. A service implementing CSW handles descriptions of datasets and services, formatted as ISO 19115:2003. The descriptions include information regarding the extent, quality, spatial and temporal characteristics as well as the distribution rights of a given dataset. A CSW service handles its requests and responses using the HTTP protocol using the GET and POST method [8].

Previous researchers have identified limitations on traditional CSW services. The string matching process as the only query option is mentioned in [9] [7] and [8]. In these works authors explore options to add semantic capabilities to CSW overcoming this limitation.

In [8] the author describes the CSW limitations by evaluating GeoNetwork, a popular open source CSW implementation. The author identifies three ways in which it is possible to add semantic annotations to the CSW: 1) By associating keywords to concepts using the *getCapabilities* response. 2) By adding a link in the GeoNetwork client interface to a ontology browser. In this way the user instead of using keywords, would be able to utilize the hierarchical structure to identify the topic that best suits her interest. 3) Adding ontologies as an extension package using ebRIM. The author concludes that the third option is the most suitable.

Yue et al. (2006) extend the ebRIM CSW specification by: 1) adding new classes based on existing ebRIM classes; and 2) adding Slots to existing classes, thus creating new attributes. As a result they are able to store richer metadata records in the catalogue. The authors identified two possible options to implement a search functionality: 1) create an external component without further modification of the CSW schemas; 2) modify the CSW adding semantic functionalities to the existing CSW schemas. In this research they choose for the first option [9]. Yue et al. (2011) extends this work, adding further development in the field of geoservices [7].

A different approach is used by [10]. In this research the goal is to provide access to data stored in CSW as Linked Data. In order to achieve this goal the authors developed CSW2LD, a middle layer on top of a conventional CSW based server. It allows the server to mimic other Linked Data sources and publish metadata records. CSW2LD wraps the following CSW requests: *GetCapabilities*, *GetRecords* and *GetRecordById*.

A very interesting work in progress is described in [11]. This is a website describing a proposal by a team from the GeoNetwork developer community. The authors intend to perform a major change in GeoNetwork, allowing it to store metadata as RDF facts stored in a RDF repository. They intent to use SPARQL/GeoSPARQL to retrieve data. The website describe technical characteristics of GeoNetwork and mentions fields that require work in order to implement the project. Currently the constraints that filter the search are encoded using an OGC standard based on CQL [12][13]. As indicated in [11] a CSW implementation that uses a triplestore needs to map the constraints to SPARQL/GeoSPARQL which are current W3C recommendations [14][15]. However there is scarce research on this topic. By the time we wrote this paper, there was no further development in this project and the website was last updated by the end of October of 2012.

In the next section we describe how we deal with some of the challenges already identified by previous researchers.

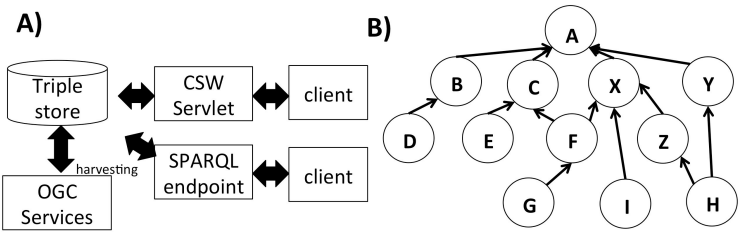

Fig. 1. A)Proposed CSW architecture (The triple store works as a metadata repository). B)Dummy domain class ontology.

3 Implementation

The complete implementation of the CSW standard is not a trivial task. By implementing the whole standard, a CSW would be able to manage a wide range of elements for metadata records and respond to the queries using multiple formats. However the number of queryable elements that must be implemented in a service of this nature is small. A CSW must implement query operations to at least the 15 classic Dublin Core metadata terms. It should also be able to respond to queries using the *csw:SummaryRecord* format [16].

In order to provide semantic capabilities to catalogue, we decided to develop a proof of concept CSW implementation using only the minimum elements and operations. Figure 1A depicts the architecture we are using. We use a Parliament triplestore as our data repository connected to a Java servlet. Our servlet transforms 1) requests from clients into queries processable by the triplestore, and 2) the triplestore response into a xml document that follows the *csw:SummaryRecord* format.

Our goal is to implement a CSW able to:1) Retrieve records based on an ontology class taxonomy; and 2) Retrieve records based on spatial relationships with elements with toponym values (For instance records located *within* the element known as "Burgundy").

In order to take advantage of the spatial capabilities of our triplestore we must use *geo:SpatialObject* as a superclass and *geo:Feature* as its immediate subclass. Then we are able to use spatial operators in all the instances of the class *geo:Feature* and its subclasses [15]. In our implementation we create two main subclasses of *geo:Feature*: *MetadataRecord* and *ToponymUnit*. All the required Dublin core elements are associated with the class *MetadataRecord*. Additional elements are linked to *dc:Publisher* and *dc:Description*. The class *ToponymUnit* represents spatial elements with toponymic values in our ontology. Both *ToponymUnit* and *MetadataRecord* implement the property *geo:hasGeometry* which links to a geometry element, which itself can be linked to a WKT spatial representation (*sf:wktliteral*) with the property *geo:asWKT*. Figure 2 depicts the created classes, properties and relationships.

Knowledge stored in domain ontologies can be used by creating subclasses of *MetadataRecord*. The knowledge represented in the relationships between the

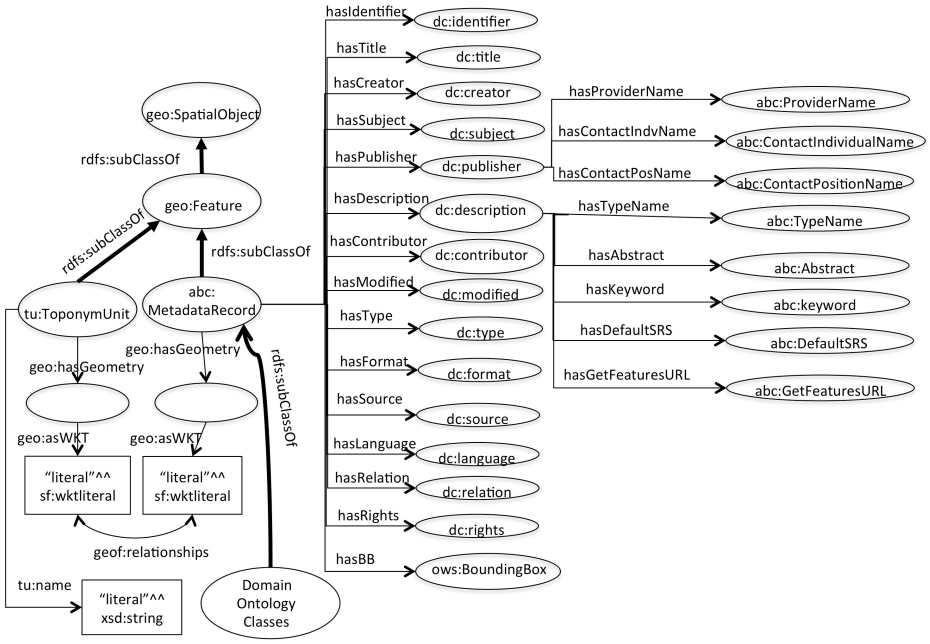

Fig. 2. Proposed ontology and relationship with external domain ontologies

classes can easily be added to the queries. We propose to map each metadata record as an instance of a given domain ontology class. To achieve this goal, we plan to harvest keywords from existing metadata elements, such as *keywords*, *abstract*, *title*, etc, and use this information to find the most suitable class based on specifications of the domain ontology. We propose to use a method based on research conducted by Werner et al. (2012) [17]. This part of the system is still in development. Figure 1B depicts a dummy ontology with a taxonomy of classes. Using the dummy ontology, if the user requests for metadata records of class *C*, due to the class relationship, the system would also return instances of classes *E*, *F*, and *G*.

The metadata records in our research were created by harvesting a set WFS services. We use a custom made tool, developed with Java. Our tool submits a *GetCapabilities* request to a WFS, and parses the response in order to identify relevant elements. Currently we harvest the following elements: a) provider Name, b) contact individual name, c) contact position name, d) type Name, e) abstract, f) keywords, g) default spatial representation system, h) the URL of the *GetFeatures* request, and i) the bounding box of the dataset. Being this a proof of concept application we considered these to be enough elements.

To feed our ontology with toponymic elements we are importing datasets available in vector format. We want to use global datasets of administrative units (Countries, provinces, states, etc.). Brigham et al.(2011) evaluated three global datasets: 1) GAUL, developed by FAO and the European Commision,

2) GADM, developed by the University of California, Berkeley and the International Rice Research Institute, and 3) UNSALB, developed by the United Nations Geographic Information Group. They concluded that GAUL showed better completeness and accuracy[18]. We intended to use the GAUL dataset for evaluation purposes. However, by the time of writing this paper, our request was still on process. In the meantime we are using an alternative provided by Esri and DeLorme Publishing Company, Inc. under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License. This dataset depicts the subnational and national boundaries updated with 2011 information [19]. We plan to update our toponym dataset as soon as a better dataset is available.

The most used CSW request is *GetRecords*. A typical request includes a set of pairs of the form (*parameter=value*). The CSW server filters out results that do not meet the criteria specified in the parameter *constraint*. These constraints are encoded using OGC compliant syntax based on CQL.

In our implementation we submit *GetRecords* requests. Our server application parses the request and extracts the value for the *constraint*. We encode our constraint using a subset of the OGC specification, to which we have added elements to describe *instanceOf* relationships. The user can link multiple constraints using the operator AND. We also allow the use of an optional boolean element NOT, to indicate that elements matching certain criteria, should be removed from query results. Our servlet reads the constraint and maps it to suitable SPARQL/GeoSPARQL syntax. The mapping is done using a Java application developed by us. Currently our application supports the following predicates:

- *PropertyIsInstanceOf* To constrain the class membership. This is a custom property, not included in the OGC specification.
- *PropertyIsEqualTo* To compare queryable metadata elements to text strings.
- *PropertyIsLike* To make partial comparisons between queryable metadata elements and text strings.
- *Within* To specify that the bounding box of the metadata record must be inside a given toponym unit.
- *Intersect* The bounding box of the metadata record should intersect the geometry of the toponym unit.
- *DWithin* To define a buffer zone around a given toponym unit and compare it with the bounding box of a metadata record.

Currently, we are using Parliament as our triplestore. We decided to use Parliament due to its good performace, supported capabilities and its open source nature. Parliament support for GeoSPARQL includes multiple functions and operators allowing users to perform complex spatial queries. Once installed it allows users to access it though an HTTP SPARQL endpoint [20] [21]. Our implementation interacts with Parliament using Java with Jena libraries.

Next we show examples of constraints.The original constraints follow the OGC XML filter specification, however they are not fully compliant because of our add ons (InstanceOf, ToponymUnit). The constraints are mapped to SPARQL/GeoSPARQL using a java application.

Example 1. Metadata records corresponding to ontology domain class *A*.

```
    <PropertyIsInstanceOf>
      <PropertyName>a</PropertyName>
      <OntClass>xyz:A</OntClass>
    </PropertyIsInstanceOf>
```

```
SELECT ?metadatarecord
 WHERE {
?metadataRecord a abc:MetadataRecord.
?metadataRecord a xyz:A. }
```

The namespace *abc:* links to our implemented ontology. Class *xyz:A* is part of our Dummy ontology, and is a subclass of *abc:MetadataRecord*. The results of this query will include the instances of *xyz:A* as well as instances of all its subclasses.

Example 2. Metadata records corresponding to ontology domain class *B*, that include the word *water* among their keywords.

```
<and>
    <PropertyIsInstanceOf>
      <PropertyName>a</PropertyName>
      <OntClass>xyz:B</OntClass>
    </PropertyIsInstanceOf>
    <PropertyIsLike wildCard="*" singleChar="#" escapeChar="!">
      <PropertyName>abc:keyword</PropertyName>
      <Literal>water</Literal>
    </PropertyIsLike>
</and>
```

```
SELECT ?metadatarecord
 WHERE {
?metadataRecord a abc:MetadataRecord.
?metadataRecord a xyz:B.
?metadataRecord abc:hasDescription ?description.
?description abc:haskeyword ?element_keyword.
?element_keyword abc:hasLiteral ?literal_keyword.
FILTER(regex(?literal_keyword,"water","i")) }
```

The maping of the second condition requires further processing. *abc:Metadata Record* is linked to *dc:description* by the property *abc:hasDescription*, which itself is linked to *abc:keyword* through the property *abc:haskeyword*. This element itself is linked to a literal that contains the string containing the keyword. We use the function *regex()* to search for the word *water* as a substring of the keywords.

Example 3. Retrieve metadata records that are members of ontology domain class *C*.The bounding boxes of the records should be located within the toponym unit *France*.

```
<and>
    <PropertyIsInstanceOf>
      <PropertyName>a</PropertyName>
      <OntClass>xyz:C</OntClass>
    </PropertyIsInstanceOf>
    <Within>
      <PropertyName>geo:asWKT</PropertyName>
      <ToponymUnit>tu:France</ToponymUnit>
    </Within>
</and>
```

```
SELECT ?metadatarecord
 WHERE {
?metadataRecord a abc:MetadataRecord.
?metadataRecord a xyz:C.
?metadataRecord geo:hasGeometry ?boundingbox.
?boundingbox geo:asWKT ?boundingbox_wkt.
tu:France geo:hasGeometry ?France_geometry.
?France_geometry geo:asWKT ?France_wkt.
FILTER(geof:sfWithin(?boundingbox_wkt,?France_wkt)) }
```

We use *geo:asWKT* to represent the geometry of the selected metadata records. The namespace *tu:* links to an ontology containing our toponym elements. The filter *Within* is interpreted by the servlet as an spatial operation between the bounding box of the metadata record and the geometry of toponym unit. After obtaining the wkt literal representation of the geometry, the spatial comparisons are obtained using the funtion *geof:sfWithin* inside the *FILTER*.

Example 4. Retrieve metadata records whose bounding boxes are located within 100km from France, overlaping Germany.

```
<and>
    <DWithin>
      <PropertyName>geo:asWKT</PropertyName>
      <ToponymUnit>tu:France</ToponymUnit>
      <Distance unit="http://www.uomdict.com/uom.html#meters">
      100000</Distance>
    </DWithin>
    <Intersect>
      <PropertyName>geo:asWKT</PropertyName>
      <ToponymUnit>tu:Germany</ToponymUnit>
    </Intersect>
</and>
```

```
SELECT ?metadatarecord
 WHERE {
```

```
?metadataRecord a abc:MetadataRecord.
?metadataRecord geo:hasGeometry ?boundingbox.
?boundingbox geo:asWKT ?boundingbox_wkt.
tu:France geo:hasGeometry ?France_geometry.
?France_geometry geo:asWKT ?France_wkt.
tu:Germany geo:hasGeometry ?Germany_geometry.
?Germany_geometry geo:asWKT ?Germany_wkt.
BIND(geof:buffer(France_wkt,100000,units:m) as France_buff_wkt)
FILTER(geof:sfIntersects(?boundingbox_wkt,?France_buff_wkt))
FILTER(geof:sfIntersects(?boundingbox_wkt,?Germany_wkt)) }
```

In this example we use a buffer function available in our triplestore. The operator *DWithin* indicates the servlet that it needs create a new wkt literal using the parameter values with the function *geof:buffer*. The resulting literal is added to the query using the *BIND* form. This query uses two filters, one for each spatial constraint.

4 Conclusions

In this paper we are implementing a basic catalogue, so we are not dealing with the real spatial features (streets, roads, etc.). However by using an external domain ontology and having the URL of the WFS *GetFeatures* request, we are able to easily identify a dataset and add it to a *smart query*.

Our research shows that mapping between OGC filter specification and SPARQL is not complicated. However, in order to take advantage of additional elements such as a domain ontology and toponym units, it would be necessary to develop a graphic interface on the client application, to reduce syntax errors.

The triple store implements its own SPARQL endpoint allowing users to directly access the metadata information bypassing the CSW. However, most of the human users would prefer a CSW interface.

We implemented a proof of concept CSW. Our results look promising. However, further research in this field should consider collaborating with larger open source efforts such as GeoNetwork. Particularly interesting for us, is the project proposed in [11].

Our SPARQL/GeoSPARQL queries are not optimized, further research is necessary to improve performance for automatically created queries.

Acknowledgements. This research is supported by: 1) Conseil régional de Bourgogne. 2) Direction Générale de l'Armement, see: `http://www.defense.gouv.fr/dga/`

References

1. Goodwin, J.: What have ontologies ever done for us - potential applications at a national mapping agency. In: OWL: Experiences and Directions (OWLED) (2005)
2. Harbelot, B., Arenas, H., Cruz, C.: The spatio-temporal semantics from a perdurantism perspective. In: Proceedings of the Fifth International Conference on Advanced Geographic Information Systems, Applications, and Services GEOProcessing (February-March 2013)
3. ESRI: GIS Best Practices: Spatial Data Infrastructure (SDI) (2010)
4. Nebert, D., Whiteside, A., Vetranos, P.: Catalogue services specification (2007)
5. Whiteside, A.: Web services common specification (2005)
6. Janowicz, K., Schade, S., Broring, A., Kebler, C., Maue, P., Stasch, C.: Semantic enablement for spatial data infrastructures. Transactions in GIS 14(2), 111–129 (2010)
7. Yue, P., Gong, J., Di, L., He, L., Wei, Y.: Integrating semantic web technologies and geospatial catalog services for geospatial information discovery and processing in cyberinfrastructure. GeoInformatica 15, 273–303 (2011), doi:10.1007/s10707-009-0096-1
8. Gwenzi, J.: Enhancing spatial web seach with semantic web technology and metadata visualization (2010)
9. Yue, P., Di, L., Yang, W., Yu, G., Zhao, P.: Path planning for chaining geospatial web services. In: Carswell, J.D., Tezuka, T. (eds.) W2GIS 2006. LNCS, vol. 4295, pp. 214–226. Springer, Heidelberg (2006)
10. Lopez-Pellicer, F.J., Florczyk, A., Renteria-Aguaviva, W., Nogueras-Iso, J., Muro-Medrano, P.R.: CSW2LD: a Linked Data frontend for CSW (2010)
11. Pigot, S.: Using rdf as metadata storage (2012), `http://trac.osgeo.org/geonetwork/wiki/rdfstore` (accessed on May 2013)
12. OSGeo: CQL (2012), `http://docs.geotools.org/latest/userguide/library/cql/cql.html` (accessed on November 2012)
13. Vretanos, P.A.: Filter encoding implementation specification (2005) (accessed on May 2013)
14. DuCharme, B.: Learning SPARQL. O'Reilly Media, Inc. (July 2011)
15. Kolas, D., Batle, R.: GeoSPARQL user guide (2012), `http://ontolog.cim3.net/file/work/SOCoP/Educational/GeoSPARQLUserGuide.docx` (accessed on May 2013)
16. OGC: Make a really basic catalog service for the web, csw (2011), `http://www.ogcnetwork.net/node/630` (accessed on May 2013)
17. Werner, D., Cruz, C., Nicolle, C.: Ontology-based recommender system of economic articles. In: WEBIST, pp. 725–728 (2012)
18. Brighman, C., Gilbert, S., Xu, Q.: Open Geospatial Data: An Assesment of Global Boundary Datasets (2011), `http://maps.worldbank.org/content/article/open-geospatial-data-assessment-global-boundary-datasets` (accessed on May 2013)
19. Esri, D.: World administrative units (2011), `http://resources.arcgis.com/content/data-maps/10.0/world` (accessed on May 2013)
20. Emmons, I.: Parliament User Guide. Raytheon BBN Technologies (2012)
21. Battle, R., Kolas, D.: Enabling the geospatial semantic web with parliament and GeoSPARQL. Semantic Web (2012)