LOSM: a SPARQL endpoint to query Open Street Map

Vito Walter Anelli, Tommaso Di Noia, Pasquale Galeone, Francesco Nocera,
Jessica Rosati, Paolo Tomeo, Eugenio Di Sciascio

Polytechnic University of Bari, Via Orabona, 4, 70125 Bari, Italy
{v.anelli,p.galeone,f.nocera}@studenti.poliba.it
{tommaso.dinoia,jessica.rosati,paolo.tomeo,eugenio.disciascio}@poliba.it

Abstract. Geographical data is gaining momentum in scientific and industrial communities thanks to the high level and quality of information and knowledge it encodes. The most recent representation of the Linked Open Data cloud shows GeoNames competing with DBpedia as the largest and most linked dataset available in the Web. In the "normal" Web, Open Street Map (OSM) has reached, in the last years, a maturity stage thus allowing the users to exploit its data for a daily use. We developed LOSM (Linked Open Street Map), a SPARQL endpoint able to query the data available in OSM by an on-line translation form SPARQL syntax to a sequence of calls to the overpass API. The endpoint comes together with a Web interface useful to guide the user during the formulation of a query.

1 Introduction

In the current Web of Data we are witnessing the spread of applications that heavily rely on geographical data. In the "geo-data" arena a primary role is played by the crowd sourced project Open Street Map [3]. It is a geographical database maintained by Web users containing a huge amount of data that can also be displayed on a map. Its database is updated every 15 minutes and as of today it contains 4,742,543,824 GPS points and 2,169,093 users who contribute to the project. All this data is either available via weekly dumps or it is queryable through an API. In particular, we refer to overpass API which allows the user to query Open Street Map by means of at least two different languages: XML or Overpass QL. By means of an overpass query, the API is able to retrieve nodes within an area, recognize streets or relations. Also, it is possible to express spatial relationships among nodes through filters such as around, bounding box and the poly function.
It is easy to see that having such data available in the Linked Open Data cloud would surely enrich the amount and quality of the information available within the so called Web of Data. This is the rationale behind the LinkedGeoData[1] project [1]. It aims at triplifying Open Street Map dumps every six months by mapping OSM tags and sourceKey properties with reference to a publicly available ontology. This is a very useful resource because it makes available classes that map keys and tags used in Open Street Map nodes.
Although the big effort and work in developing and maintaining the datasets behind the

[1] http://linkedgeodata.org/

project, LinkedGeoData suffers from the misalignment between the data available via the SPARQL endpoint (based on a dump) and the one available in Open Street Map. Indeed, the updates made by the users are available as RDF triples only when the dump is processed and loaded in the LinkedGeoData triple-store.
Based on this observation we developed LOSM (Linked Open Street Map), a service that works as a SPARQL endpoint on top of OSM. LOSM acts as a translator from a SPARQL query to a set of overpass API calls. In such a way we are sure that the data we retrieve is always fresh and up to date. LOSM is available at http://sisinflab.poliba.it/semanticweb/lod/losm/.

2 LOSM: System description

The scheme in Figure 1 shows an overview of the service architecture. In a few words, the systems is able to translate a SPARQL query to a sequence of (iterative) overpass API calls, collect the data and return it to the client. We currently support SPARQL queries via HTTP GET. The Parser uses a scanner for the recognition of lexemes in a SPARQL query and creates the data-structures needed by the Query Manager. This module is in charge of breaking the query into sub-queries according to the remote functions available in the overpass API. The Result Manager handles the sub-queries and the results they generate to create the final Result map. The Result Manager breaks the graph pattern in the SPARQL query into a set of connected sub-graphs by identifying their mutual relations. Each sub-query goes through the Translator which is in charge of creating the overpass calls.

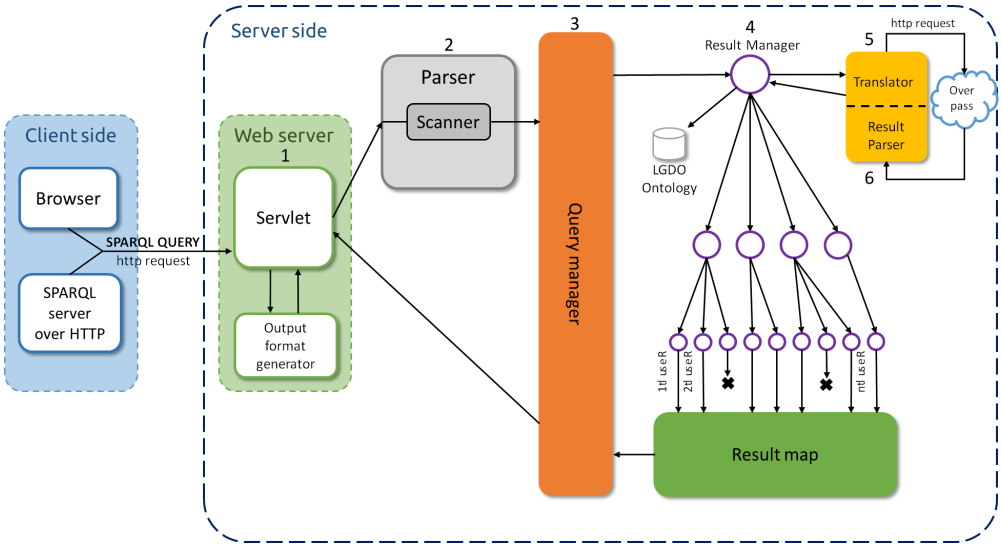

Fig. 1: Overall representation of the system architecture

The system also exposes a Web page with a query form with autocomplete facilities with respect to the `LinkedGeoData` ontology.

The SPARQL sublanguage implemented in LOSM. In its current version, LOSM implements a subset of the full specification of SPARQL 1.1 plus some non-standard features[2] that results very useful when querying geographical data. We currently support only the `SELECT` query form and the Jena Spatial[3] extension also available in GeoSPARQL [2]. We support simple graph patterns that we anyway consider representative of a large number of queries over geographical data. As for the spatial functions we implement we may list:

- `spatial:nearby (latitude longitude radius [units])`[4] returns URIs nodes (Open Street Map URIs) within the radius distance of the location of the specified latitude and longitude.
- `spatial:withinCircle (latitude longitude radius [units])` computes a circle centered in specified latitude and longitude and given radius and returns the OSM nodes within the circle.
- `spatial:withinBox (latitude_min longitude_min latitude_max longitude_max)` calculates a rectangle by specifying the list of coordinates for the edges that has to follow the order provided in the function.
- `spatial:within("POLYGON((Point1_lat Point1_lon,...,PointN_lat PointN_lon))")` calculates the polygon area expressed by Well Known Text (WKT) literals and returns OSM nodes available within it.

Regarding the URI of classes and properties used in the graph pattern for LOSM SPARQL queries we always refer to the LinkedGeoData Ontology vocabulary.

3 Use case

Suppose we want to represent a query to match the following use case: *the day is over in our laboratory and the crew wants to find restaurants nearby (within 200 meters) together with the cinemas that are within one km from each restaurant. They want to know the names of restaurants and cinemas together with the URIs of these latter*. The above use case can be modeled by the SPARQL query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX lgdo: <http://linkedgeodata.org/ontology/>
PREFIX spatial: <http://jena.apache.org/spatial#>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
SELECT ?cinema ?nameC ?nameR
WHERE  {
   ?link rdfs:label "Sisinf Lab" .
   ?link geo:lat ?lat .
   ?link geo:long ?lon.
   ?object spatial:nearby(?lat ?lon 200 'm') .
   ?object a lgdo:Restaurant  .
   ?object rdfs:label ?nameR .
   ?object geo:lat ?lat2 .
```

[2] Details on the implemented subset is available at `http://sisinflab.poliba.it/semanticweb/lod/losm/losm_grammar.html`

[3] `https://jena.apache.org/documentation/query/spatial-query.html`

[4] [units] can be meters ('`m`' or '`M`'), kilometers ('`km`' or '`KM`') or miles ('`mi`' or '`MI`').

```
   ?object geo:long ?lon2.
   ?cinema spatial:nearby(?lat2 ?lon2 1000 'm') .
   ?cinema a lgdo:Cinema .
   ?cinema rdfs:label ?nameC .
}
```

The triples composing the graph pattern are analysed and grouped into the corresponding sub-graphs by looking at their subject.

?link `[?link rdfs:label "Sisinf Lab" , ?link geo:lat ?lat , ?link geo:long ?lon ]`

?cinema `[?cinema spatial:nearby ?lat2 ?lon2 1000 'm' , ?cinema a lgdo:Cinema , ?cinema rdfs:label ?nameC ]`

?object `[?object spatial:nearby ?lat ?lon 200 'm' , ?object a lgdo:Restaurant , ?object rdfs:label ?nameR , ?object geo:lat ?lat2 , ?object geo:long ?lon2 ]`

Based on the above grouping, the Query Manager selects first the **?link** group and generates the Overpass QL expression

```
node["name"="Sisinf Lab"];
out body;
```

the system then executes the `overpass` query related to the **?object** group which is composed by taking into account the results of the previous one.

```
node(around:200,41.1095222,16.8778234)
["amenity"="restaurant"]
["name"];
out body;
```

the final sub-graph represents a set of `overpass` API calls. One for each node returned by the previous query. As an example we have:

```
node (around: 1000,41.1085645,16.8768552)
["amenity" = "cinema"]
["name"];
out body;
```

4 Conclusion and Future Work

We presented LOSM, a service that acts as a SPARQL endpoint on top of Open Street Map data. Differently from `LinkedGeoData`, it does not work by using dumps of the OSM datasets but it queries directly the OSM database by means of a translation from SPARQL to `overpass` API calls. The implementation is in beta stage and it currently works on a subset of the SPARQL language plus the geographical query constructs from the Jena Spatial extension. We are working to add new features from SPARQL as well as to implement the whole SPARQL protocol.

References

1. Sören Auer, Jens Lehmann, and Sebastian Hellmann. Linkedgeodata: Adding a spatial dimension to the web of data. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 731–746, 2009.
2. Robert Battle and Dave Kolas. Geosparql: enabling a geospatial semantic web. *Semantic Web Journal*, 3(4):355–370, 2011.
3. Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.