Exposing Open Street Map in the Linked Data cloud

Vito Walter Anelli[1], Andrea Calì[2], Tommaso Di Noia[1], Matteo Palmonari[3], Azzurra Ragone[3]

[1] Polytechnic University of Bari, Via Orabona, 4, 70125 Bari, Italy
v.anelli@studenti.poliba.it, tommaso.dinoia@poliba.it
[2] Birkbeck, University of London, Malet Street, London WC1E 7HX, UK
andrea@dcs.bbk.ac.uk
[3] University of Milano-Bicocca, Piazza dell'Ateneo Nuovo, 1, 20126 Milano, Italy
{firstname.lastname}@unimib.it

Abstract. After the mobile revolution, geographical knowledge has getting more and more importance in many location-aware application scenarios. Its popularity influenced also the production and publication of dedicated datasets in the Linked Data (LD) cloud. In fact, its most recent representation shows Geonames competing with DBpedia as the largest and most linked knowledge graph available in the Web. Among the various projects related to the collection and publication of geographical information, as of today, Open Street Map (OSM) is for sure one of the most complete and mature one exposing a huge amount of data which is continually updated in a crowdsourced fashion. In order to make all this knowledge available as Linked Data, we developed LOSM: 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 OSM `overpass` API. The endpoint makes also possible an on-the-fly integration among Open Street Map information and the one contained in external knowledge graphs such as DBpedia, Freebase or Wikidata.

1 Introduction

In the last years we are witnessing the spread of knowledge intensive applications that relies on the flourishing of the datasets available in the Linked Data (LD) Cloud [4]. The richness of semantic data they expose paves the way to a new generation of services and tools exploiting the ontological knowledge they encode as well as the possibility to easily mash up data coming from different sources. Among them, geographical datasets are becoming more and more important to deliver location-aware services.

The availability of a common query interface *i.e.*, SPARQL, and the crowd-driven standardization of ontological vocabularies allows an intelligent application to grab data from diverse datasets and join them. We may imagine different

[4] http://lod-cloud.net

scenarios where such a feature can be an important asset to provide a high quality service such as context-aware recommendation systems [15], on-line shopping, etc. or public services needed in situations of disaster management where the quality and timeliness of the data is of crucial importance.

In many application scenarios, geographical information is a key factor to enhance system answers to user request, *e.g.,*in a movie recommendation scenario in order to suggests not only a movie according to user preferences, but also with reference to the closeness of the cinema.

Geographical knowledge bases and geo-spatial reasoning may play a key role also in emergency response or transportation planning[16, 3, 5] where it results useful to combine knowledge from different types of datasets in order to get knowledgeable information from the system. As a way of example, while organizing a trip, the user could combine information about the cultural heritage sites and means of transport with hotel accommodations, taking into account their proximity. Analogously, in emergency response situations, data can be combined to obtain a helpful and timely response. In an emergency scenario, *e.g.,*an earthquake, the user should be able to query a knowledge base to look for collection camps, hospitals, rescue places, areas for helicopter landing as well as informal camps where the homeless have set up tent camps.

In this paper we present LOSM (Linked Open Street Map), a service that acts as a SPARQL endpoint on top of OSM. LOSM allows one to query Open Street Map data via a SPARQL query as it takes care to translate the query into a set of calls to OSM APIs. This results in exposing all the information contained in the Open Street Map geographical database as Linked Data thus making OSM a first class citizen in the Linked Data Cloud.

While LOSM supports an on-the-fly integration of OSM data with data coming from other sources, differently from similar projects (see `LinkedGeoData`[5] [2] for example) LOSM does not rely on a periodical dump of the data, but it always exposes fresh and up-to-date data.

Another strength of our tool is the possibility to merge different datasets of the Linked Data cloud, linking geo-spatial knowledge with the one coming from various knowledge bases, as we show in Section 3.1.

The remainder of this paper proceeds as follows. In Section 2 we start by briefly describing Open Street Map and the reason why we chose it as a provider of linked geo-spatial data, then we describe the system architecture and the query language implemented in LOSM. Then, in Section 3 we describe two use cases through some sample queries highlighting the capabilities of LOSM, with a particular reference to an emergency management scenario. Finally, in Section 4 we review related works relying on the use of geo-spatial data. Conclusion and future work close the paper.

[5] `http://linkedgeodata.org/`

2 LOSM: Linked Open Street Map

In the "geo-data" arena, the crowd sourced project Open Street Map [10] is currently playing a primary role due to its openness, easy of use and of integration in third party applications.

OSM 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 5,027,330,590 GPS points and 2,445,598 users who contribute to the project.

All this data is either available via weekly dumps or it is queryable through a public API. In particular, `overpass`[6] is a read-only 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.

The query language is very expressive and makes possible to perform spatial reasoning by imposing constraints within the query. For instance, the user may impose 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` project [2]. 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 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.

Despite the considerable effort, `LinkedGeoData` approach cannot be used for all those scenarios where timeliness and freshness of information is a must have. A flagship example is that of disaster recovery where information about collection camps, rescue places, temporary hospitals, passable roads, etc. needs to be available as soon as possible.

Starting from this observation we developed LOSM, a SPARQL endpoint that 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 as they come directly from the OSM database, which is constantly updated by a crowd of volunteer all around the world.

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

[6] `http://wiki.openstreetmap.org/wiki/Overpass_API`

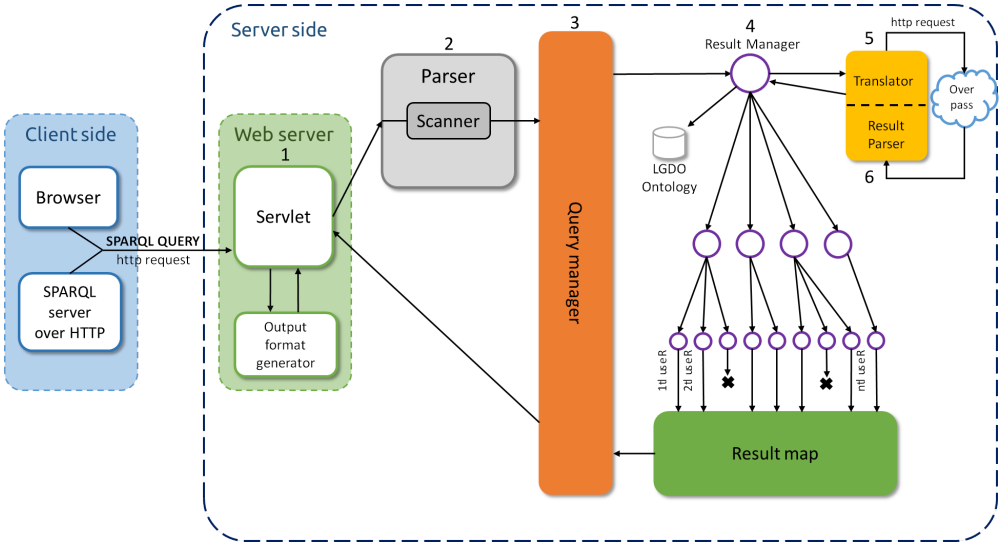

Fig. 1. Overall representation of the system architecture

(iterative) `overpass` API calls, collect the data, join 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 Manage**r handles the sub-queries and the results they generate to create the final Result map. The Result Manager breaks the graph pattern described 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.

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

LOSM is available at `http://sisinflab.poliba.it/semanticweb/lod/losm/`.

2.1 The SPARQL sublanguage implemented in LOSM.

In its current version, LOSM implements a subset of the full specification of SPARQL 1.0 plus some non-standard features[7] that results very useful when querying geographical data. We currently support only the `SELECT` query form and the Jena Spatial[8] extension also available in GeoSPARQL [3]. We support simple graph patterns that we anyway consider representative of a large number

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

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

of queries over geographical data. As for the implemented spatial functions, we may list:

- `spatial:nearby (latitude longitude radius [units])`[9] 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 URIs of classes and properties used in the graph pattern for SPARQL queries, LOSM may refer to the LinkedGeoData Ontology controlled vocabulary as well as to an ontological one which is a one-to-one mapping with OSM system of tags[10]. The rationale behind the introduction of this new vocabulary is driven by the main goal we had in mind while developing LOSM: to have a SPARQL endpoint able to timely expose all the changes that continually happen in Open Street Map even at the semantic level (represented by the tags).

Although very useful and structured, a static ontology as the one modelled within the LinkedGeoData project, cannot follow continuous data variations due to users' freedom in inserting new tags and values. In each community a lot of linguistic phenomenons happen in time that change the frequency of a term occurrence and then its importance and adoption by the community itself. To address this problem we introduced a LOSM prefix `<http://sisinflab.poliba.it/semanticweb/lod/losm/ontology/>` (shortened in `losm`) based on the same crowdsourcing concept. The **Parser** recognizes the use of this prefix and prepares the overpass query in the proper way. This lets the user to use any term she considers reasonable and the evaluation of the existence of the term is based only on the real data coming from Open Street Map. As an example, if the user wants to retrieve information classified with the key-value pair `key="refugee" value="yes"` she will refer to the corresponding property represented by the URI `<http://sisinflab.poliba.it/semanticweb/lod/losm/ontology/refugee>` or equivalently by the CURIE `losm:refugee` (see the example in Section 3.1).

Some keys are reserved to provide advanced features in LOSM that makes easy the integration with other external knowledge graphs exposing a SPARQL endpoint such as DBpedia[11] or Wikidata[12]. The `losm:dbpedia` property acts

[9] [units] can be meters ('`m`' or '`M`'), kilometers ('`km`' or '`KM`') or miles ('`mi`' or '`MI`').
[10] `http://wiki.openstreetmap.org/wiki/Tags`
[11] `http://dbpedia.org/sparql`
[12] `https://query.wikidata.org`

as a converter from the Wikipedia page or Wikipedia id associated to an Open Street Map node to the corresponding DBpedia resource. Analogously, `losm:wikipedia` returns the complete URI of the corresponding Wikipedia page. In both cases, the output may refer to the main English version of DBpedia/Wikipedia or to a local version depending on the value of the OSM key `wikipedia`.

3 Use case

The first use case we present in this section has the only purpose to explain how LOSM works, showing the steps performed by the **Query Manager** (see Figure 1).

Suppose the following situation: *the day is over in our laboratory and the crew wants to find restaurants nearby (within 200 meters) together with the cinemas which are at most one km far from each restaurant. They want to know the names of restaurants and cinemas together with the URIs of the 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 .
    ?object geo:long ?lon2.
    ?cinema spatial:nearby(?lat2 ?lon2 1000 'm') .
    ?cinema a lgdo:Cinema .
    ?cinema rdfs:label ?nameC .
}
```

The query is processed based on a priority system relying on a weighted dependency graph. The triples composing the graph pattern are analysed and grouped into the corresponding sub-graphs by looking at their subject. The system attaches to each triple a value depending on the degree of connection with other groups measured by taking into account shared variables and predicates. Then each group is labeled with a weight proportional to its triples values. The group with the lowest value is the first sent to the Translator component. The **Translator** converts the set of sub-queries to its `overpass` equivalent starting with the triple with the lowest value. Once the results from `overpass` API are returned, they are used to update the initial query so new weights are computed. The process iterates on triples groups until the last sub-query has been translated.

?link `[?link rdfs:label "Sisinf Lab" , ?link geo:lat ?lat , ?link geo:long ?lon ]`
?object `[?object spatial:nearby ?lat ?lon 200 'm' , ?object a lgdo:Restaurant , ?object rdfs:label ?nameR , ?object geo:lat ?lat2 , ?object geo:long ?lon2 ]`
?cinema `[?cinema spatial:nearby ?lat2 ?lon2 1000 'm' , ?cinema a lgdo:Cinema , ?cinema rdfs:label ?nameC ]`

Based on the above grouping, the Query Manager selects first the **?link** group and generates the Overpass QL expression representing the first query to the `overpass` API:

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

Then the system 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 is converted into 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;
```

3.1 Emergency management scenario

We now present a use case in emergency management where the usefulness of LOSM is twofold. On the one hand, we may have access to always fresh and timely information in the context of an unpredictable disaster. On the other hand, we can exploit a third party endpoint supporting SPARQL 1.1 to perform a federated query among LOSM, DBpedia and Wikidata thus mashing up the knowledge coming from the three sources. Indeed, in such a context relevant data rapidly changes over time and the system capability of linking information from different knowledge bases is crucial.

An Italian manager is doing a business trip in the Miyagi Prefecture when an earthquake happens. The damages all around are severe and catastrophic. It is possible that aftershocks will follow and he has to find a way to rescue himself in a foreign country, plus he does not speak Japanese. In the mean time news about the event are reaching any corner of the world and mechanisms of international assistance are already on the move. Volunteers are populating Open Street Map

with fresh data about collection camps, rescue places and temporary hospitals [13]. *The manager has two primary needs: reaching a near refugee camp and, then, look for an airport to go back to Italy. He has a mobile phone with Gps and Internet connectivity so he tries to look for refugee camps, mapped on Open Street Map, which are located near by.*

This request can be translated in the following SPARQL query [14]:

```
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 ?object ?name ?lat ?long
WHERE  {
?object spatial:nearby(38.2943 140.65000 5000 'm') .
?object losm:refugee "yes".
?object rdfs:label ?name .
?object geo:lat ?lat.
?object geo:long ?long.
}
```

This query allows the user to retrieve any item containing the tag `refugee` within a circle with radius of 5000m and returns the Open Street Map node, the name, and the GPS coordinates (lat, long). Obviously, the user does not have to write the `SPARQL` query himself, but he should rely on an end-user interface that allows him to build a `SPARQL` query without knowing the `SPARQL` language (see as a way of example the tools presented in [7, 8]).

In order to show the capability of the system to link information coming from different knowledge bases, we give an example of a more complex and exhaustive queries that can be posed to the Linked Data Cloud thanks to the use of LOSM.

From the previous query, the manager has found a refugee camp whose name he cannot understand as it is returned in Japanese. Anyway, based on the result of the previous query he wants to retrieve information about the nearest cities (within 10km) and airports to go back to Italy. He wants to retrieve info about the nearest cities in Italian and the name of the airports (together with its coordinates) in English in order to pronounce it in an understandable way.

```
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

[13] The reader can find a real example of such scenario in [17] where it is described the Haiti-post-earthquake work done on Open Street Map: volunteers rapidly mapped the affected areas so helping the aid effort. They show the impact of a crowdsourced mapping in such emergency situation. Moreover, Open Street Map has an humanitarian team to deal with emergency situations. They keep updated a page with current and past remote mapping actions (see `http://wiki.openstreetmap.org/wiki/Humanitarian_OSM_Team`)

[14] It is noteworthy the use of the spatial function `spatial:nearby`, the LOSM predicate `losm:refugee` and the geo functions `geo:lat` and `geo:long`.

```
PREFIX lgdo: <http://linkedgeodata.org/ontology/>
PREFIX spatial: <http://jena.apache.org/spatial#>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX losm: <http://sisinflab.poliba.it/semanticweb/lod/losm/ontology/>
PREFIX schema: <http://schema.org/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT  ?uriRefer ?dburi ?text ?airportname ?airportLat ?airportLong ?wdURI ?freebaseID  ?geonames
WHERE {
    SERVICE <http://sisinflab.poliba.it/semanticweb/lod/losm/sparql> {
?link losm:name " 仙台市立広陵中学校".

   ?link geo:lat ?lat .
   ?link geo:long ?lon.
   ?uriRefer spatial:nearby(?lat ?lon 10 'km') .

   ?uriRefer a lgdo:City  .
   ?uriRefer losm:dbpedia ?dburi .
   ?uriRefer losm:wikidata ?wdID .
  }
    SERVICE <http://dbpedia.org/sparql> {
    ?dburi dbpedia:abstract ?text .
    ?dburi foaf:isPrimaryTopicOf ?wikiuri .
    FILTER langMatches(lang(?text),'it').
    ?airport dbo:location ?dburi.
    ?airport rdf:type dbo:Airport.
    ?airport rdfs:label ?airportname .
    ?airport geo:lat ?airportLat .
    ?airport geo:long ?airportLong .
    filter(lang(?airportname) = 'en')
  }
  BIND(IRI(CONCAT("http://www.wikidata.org/entity/" ,str(?wdID) )) AS ?wdURI).
     SERVICE <http://query.wikidata.org/sparql> {
     SELECt ?wdURI ?freebaseID ?geonames {
     ?wdURI wdt:P646 ?freebaseID.
     ?wdURI wdt:P1566 ?geonames.
  }
 }
}
```

The previous query, by exploiting the `SERVICE` keyword from SPARQL 1.1, is able to combine information from different knowledge graphs with the one coming from LOSM. The first service invoked is the LOSM endpoint, the query returns cities within a radius of 10 km from the refugee camp found in the previous query [15]. Additionally, the DBpedia resource URI and the Wikidata

[15] Note that the returned name is a Japanese name.

ID are returned. The second invoked service is the DBpedia endpoint. Here the query returns the Wikipedia URI, the Italian description of the city, the English name and the latitude and longitude of the airport. The last service is the Wikidata endpoint, form which we get the identifiers of the same city in Freebase and Geonames knowledge bases.

Summing up, the previous example shows how it is possible to get data referring to six different data sources (Open Street Map, Wikipedia, DBpedia, Wikidata, Freebase and Geonames) having only the two values of latitude and longitude available. It is worth to note that most of the data sources have a crowdsourcing nature, which usually weakens the integration between datasets because of the heterogeneity of the contributions. The problem is highly mitigated in this scenario thanks to spatial queries that can retrieve the outgoing references from points near to the starting one.

4 Related Work

In this section we first briefly describe various approaches and systems that deal with and expose geo-spatial data in a static way in the Web of Data. Then, we review some approaches that deal with and expose dynamic data sources as Linked Data.

In recent years several ontologies and languages have been proposed to model and query dataset related to geo-spatial knowledge and to extract information from these knowledge bases. The first attempts refer to Basic Geo Vocabulary and GeoOWL ontology. Basic Geo Vocabulary [9] is a simple RDF Schema vocabulary able to represent latitude, longitude and altitude information in the WGS84 reference system. The Basic Geo Vocabulary has then been extended with GeoRSS to include various geometric objects as points, lines, polygons and their associated feature descriptions [6]. A more structured and ontological representation of the GeoRSS vocabulary is available in the GeoOWL ontology [16]. Although these two projects were developed by W3C groups they never have become W3C recommendations (and they are not very used by the community).

GeoSPARQL [3] from the Open Geospatial Consortium (OGC) is a standard that has the aim to provide a way to represent and query geospatial data in the Semantic Web. GeoSPARQL addresses this task providing a small ontology to represent features and geometries and a number of SPARQL query predicates and functions. The ontology can be combined to other ontologies representing other domains, so enhancing the latter with spatial information. GeoSPARQL allows systems to infer topological information through a qualitative spatial reasoning, *e.g.,*if a monument is inside a park, and the park is in a city, then the monument is in that city [3], as well as quantitative spatial reasoning (e.g., measuring distances). A plus of GeoSPARQL is the possibility to infer qualitative knowledge starting from quantitative ones using a single languages for

[16] `http://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/W3C_XGR_Geo_files/geo_2007.owl`

both types of reasoning. GeoSPARQL standards are supported by the triple-store Parliament [4] to query spatial data via RDF properties, which is able to answer queries like "*find all items located with a region X*". Parliament does not support Basic Geo Vocabulary, differently from OWLIM-SE (now GraphDB) triple store [1] which however supports only points for storage, thus allowing queries to find points within ad-hoc polygons and circles and to compute distances between points. GraphDB data types and queries are not compliant with GeoSPARQL[16].

Strabon[12] is a semantic spatiotemporal RDF store, that can be used to store linked geospatial data and to query them using an extension of SPARQL named *stSPARQL*. *stSPARQL* can be used to query data represented in an extension of RDF called *stRDF* that model geospatial data that changes over time (e.g., the growth of a city over the years). Strabon supports spatial datatypes enabling the serialization of geometric objects in OGC standards WKT and GML, as well as a subset of GeoSPARQL.

USeekM[17] is an extension library for semantic databases that adds efficient geospatial support. The module supports OpenGIS geometry types (such as Point, Line, Polygon) and functions (such as Within, Intersects, Overlaps, Crosses) as standardized in the OGC GeoSPARQL standard.

Among database engines, Virtuoso Universal Server [18] can handle 2-dimensional points expressed with WGS84 coordinates, as well as storage of geometric shapes (lines, polygons, etc.). In order to check if two geometries are related, Virtuoso uses some built-in predicates (*e.g.,*`ST_contains`, `ST_within`, `ST_intersect`) and supports some geometric functions (*e.g.,* `ST_distance`, `ST_x`, `ST_y`, `ST_z`). At the moment Virtuoso is not fully compliant with GeoSPARQL.

Oracle Spatial and Graph[14] supports, among others, RDF Semantic Graph data management and analysis, its applications ranging from semantic data integration to linked open data and network graphs used in transportation, utilities, energy and telcos. Oracle Spatial and Graph uses GeoSPARQL for representing and querying spatial data, even if it is not fully compliant with it.

A native RDF triple store implementation with spatial query functionality is described by Brodt et al. [5]. They model spatial features in RDF as typed complex literals and define spatial predicates as filter functions in SPARQL. However, their approach is optimized for storing and querying static RDF data with rare updates, as changes and updates in the location data can have an impact on their indexing and data processing.

Then, there are works that show how to expose dynamic data sources as Linked Data. Harth et al. [11] present an approach to expose data coming from information services as Linked Data to support their integration. Mapping is performed by using a tool to map RESTfull services to a reference ontology[20]. Although OSM is considered as one source, only queries based on bounding box have been supported. Thus this work does not proposes a general approach to expose OSM data as Linked Data.

[17] `https://dev.opensahara.com/projects/useekm`

Speiser et al.[19] and Norton et al.[13] present in their papers general approaches to expose data provided by services as Linked Data when invoked with a proper input, with [19] providing a more complete approach compared to [13]. Examples provided in the papers consider geospatial services like GeoNames [19] or OSM [13]. These general approaches are interesting but can hardly support the large variety of spatial queries over OSM that are supported by LOSM. In addition, vocabulary of the service is not mapped to widely adopted vocabularies as we did in LOSM.

5 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 currently works on a subset of the SPARQL language plus the geographical query constructs from the Jena Spatial extension. We show how fresh and timely geographical data exposed via a SPARQL endpoint in combination with information coming from multilingual knowledge graphs can affect the search for information in a disaster recovery scenario. We are currently working to extend the expressiveness of the SPARQL sub-language supported by LOSM.

References

1. Ontotext AD. Graphdb (formerly owlim) triple store, 2015. `http://ontotext.com/products/graphdb/`.
2. 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.
3. Robert Battle and Dave Kolas. Geosparql: enabling a geospatial semantic web. *Semantic Web Journal*, 3(4):355–370, 2011.
4. Robert Battle and Dave Kolas. Enabling the geospatial semantic web with parliament and geosparql. *Semantic Web*, 3(4):355–370, 2012.
5. Andreas Brodt, Daniela Nicklas, and Bernhard Mitschang. Deep integration of spatial query processing into native rdf triple stores. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 33–42. ACM, 2010.
6. Open Geospatial Consortium. An introduction to georss: A standards based approach for geo-enabling rss feeds. white paper, 2006. `http://www.opengeospatial.org/pressroom/pressreleases/580`.
7. R. Carrascosa L. Alonso i Alemany H. Durán E. Andrawos, G. García Berrotarán. Quepy - transform natural language to database queries. `http://quepy.machinalis.com/`.
8. Sébastien Ferré. Sparklis: a sparql endpoint explorer for expressive question answering. In *ISWC Posters & Demonstrations Track*, pages 45–48. CEUR, 2014.
9. W3C Semantic Web Interest Group. Basic geo (wgs84 lat/long) vocabulary, 2006. `http://www.w3.org/2003/01/geo/`.

10. Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.
11. Andreas Harth, Craig A Knoblock, Steffen Stadtmüller, Rudi Studer, and Pedro Szekely. On-the-fly integration of static and dynamic linked data. In *Proceedings of the Fourth International Workshop on Consuming Linked Data co-located with the 12th International Semantic Web Conference*, pages 1613–0073.
12. Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. Strabon: a semantic geospatial dbms. In *The Semantic Web–ISWC 2012*, pages 295–311. Springer, 2012.
13. Barry Norton, Reto Krummenacher, Adrian Marte, and Dieter Fensel. Dynamic linked data via linked open services. In *Workshop on Linked Data in the Future Internet at the Future Internet Assembly*, pages 1–10, 2010.
14. Oracle. Oracle spatial and graph. `http://bit.ly/1lvCtWi`.
15. Vito Claudio Ostuni, Giosia Gentile, Tommaso Di Noia, Roberto Mirizzi, Davide Romito, and Eugenio Di Sciascio. Mobile movie recommendations with linked data. In *Availability, Reliability, and Security in Information Systems and HCI - IFIP WG 8.4, 8.9, TC 5 International Cross-Domain Conference, CD-ARES*, pages 400–415, 2013.
16. Kostas Patroumpas, Giorgos Giannopoulos, and Spiros Athanasiou. Towards geospatial semantic data management: strengths, weaknesses, and challenges ahead. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 301–310. ACM, 2014.
17. Robert Soden and Leysia Palen. From crowdsourced mapping to community mapping: The post-earthquake work of openstreetmap haiti. In *COOP 2014-Proceedings of the 11th International Conference on the Design of Cooperative Systems*, pages 311–326. Springer, 2014.
18. OpenLink Software. Virtuoso universal server. `http://ontotext.com/products/graphdb/`.
19. Sebastian Speiser and Andreas Harth. Integrating linked data and services with linked data services. In *The Semantic Web: Research and Applications*, pages 170–184. Springer, 2011.
20. Mohsen Taheriyan, Craig A Knoblock, Pedro Szekely, and José Luis Ambite. Rapidly integrating services into the linked data cloud. In *The Semantic Web–ISWC 2012*, pages 559–574. Springer, 2012.