

LinkedGeoData – Adding a Spatial Dimension to the Web of Data

Sören Auer, Jens Lehmann, and Sebastian Hellmann

Universität Leipzig, Institute of Computer Science,
Johannisgasse 26, 04103 Leipzig, Germany
{lastname}@informatik.uni-leipzig.de
<http://aksw.org>

Abstract. In order to employ the Web as a medium for data and information integration, comprehensive datasets and vocabularies are required as they enable the disambiguation and alignment of other data and information. Many real-life information integration and aggregation tasks are impossible without comprehensive background knowledge related to spatial features of the ways, structures and landscapes surrounding us. In this paper we contribute to the generation of a spatial dimension for the Data Web by elaborating on how the collaboratively collected OpenStreetMap data can be transformed and represented adhering to the RDF data model, how this data can be interlinked with other spatial data sets, how it can be made accessible for machines according to the linked data paradigm and for humans by means of a faceted geo-data browser.

1 Introduction

It is meanwhile widely acknowledged that the Data Web will be an intermediate step on the way to the Semantic Web. The Data Web paradigm combines lightweight knowledge representation techniques (such as RDF, RDF-Schema and simple ontologies) with traditional Web technologies (such as HTTP and REST) for publishing and interlinking data and information.

In order to employ the Web as a medium for data and information integration, comprehensive datasets and vocabularies are required as they enable the disambiguation and alignment of other data and information. With DBpedia [1], a large reference dataset providing encyclopedic knowledge about a multitude of different domains is already available. A number of other datasets tackling domains such as entertainment, bio-medicine or bibliographic data are available in the emerging linked Data Web¹.

Many real-life information integration and aggregation tasks are, however, impossible without comprehensive background knowledge related to spatial features of the ways, structures and landscapes surrounding us. Such tasks include,

¹ See, for example, the listing at: <http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/DataSets>

for example, to depict locally the offerings of the bakery shop next door, to map distributed branches of a company or to integrate information about historical sights along a bicycle track.

With the OpenStreetMap (OSM)² project, a rich source of spatial data is freely available. It is currently used primarily for rendering various map visualizations, but has the potential to evolve into a crystallization point for spatial Web data integration. In this paper we contribute to the generation of an additional spatial dimension for the Data Web by elaborating on:

- how the OpenStreetMap data can be transformed and represented adhering to the RDF data model,
- how this data can be interlinked with other spatial data sets,
- how it can be made accessible for machines according to the linked data paradigm and for humans by means of a faceted geo-data browser.

The resulting RDF data comprises approximately 2 billion triples. In order to achieve satisfactory querying performance, we have developed a number of optimizations. These include a one-dimensional geo-spatial indexing as well as summary tables for property and property value counts. As a result, querying and analyzing LinkedGeoData is possible in real-time; thus enabling completely new spatial Data Web applications.

The paper is structured as follows: after introducing the OpenStreetMap project in Section 2, we describe how the OSM data can be transformed into the RDF data model in Section 3 and be published as Linked Data in Section 4. We present a mapping methodology for establishing mappings to existing data sources on the Data Web in Section 5. In Section 6 we showcase a faceted geo-data browser and editor and conclude in Section 7 with an outlook to future work.

2 The OpenStreetMap Project

OpenStreetMap is a collaborative project to create a free editable map of the world. The maps are created by using data from portable GPS devices, aerial photography and other free sources. Registered users can upload GPS track logs and edit the vector data by using a number of editing tools developed by the OSM community. Both rendered images and the vector dataset are available for downloading under a Creative Commons Attribution-ShareAlike 2.0 license. OpenStreetMap was inspired by the Wiki idea - the map display features a prominent 'Edit' tab and a full revision history is maintained.

Until now the OpenStreetMap project has succeeded in collecting a vast amount of geographical data (cf. Figure 1), which in many regions already surpasses by far the quality of commercial geo-data providers³. In other regions,

² <http://openstreetmap.org>

³ Data about the Leipzig Zoo, for example, includes the location and size of different animals' vivariums.

where currently only few volunteers contribute, data is still sparse. The project, however, enjoys a significant growth in both active contributors and daily contributed data so that uncharted territory vanishes gradually. For some regions the project also integrates publicly available data (as with the TIGER data in the U.S.) or data donated by cooperations (as in The Netherlands).

Category	Overall Amount	Daily Additions (avg.)	Monthly Growth in the last year
Users	127,543	200	11%
Uploaded GPS points	915,392,139	1,600,000	10%
Nodes	374,507,436	400,000	5%
Ways	29,533,841	30,000	7%
Relations	136,245	300	6%

Table 1. OSM statistics as of June 2009.

The OSM data is represented by adhering to a relatively simple data model. It comprises three basic types - *nodes*, *ways* and *relations* - each of which are uniquely identified by a numeric id. Nodes basically represent points on earth and have longitude and latitude values. Ways are ordered sequences of nodes. Relations are, finally, groupings of multiple nodes and/or ways. Each individual element can have a number of arbitrary key-value pairs (tags in the OSM terminology). Ways with identical start and end nodes are called closed and are used to represent buildings or land use areas, for example.

The various OSM components are depicted in Figure 1. The data is stored in a relational database. The data can be accessed, queried and edited by using a REST API, which basically uses HTTP GET, PUT and DELETE requests with XML payload (as shown in Figure 2). The data is also published as complete dumps of the database in this XML format on a weekly basis. It currently accounts for more than 6GB of Bzip2 compressed data. In minutely, hourly and daily intervals the project additionally publishes changesets, which can be used to synchronize a local deployment of the data with the OSM database.

Different authoring interfaces, accessing the API, are provided by the OSM community. These include the online editor *Potlatch*, which is implemented in Flash and accessible directly via the edit tab at the OSM map view, as well as the desktop applications *JOSM* and *Merkaartor*. Two different rendering services are offered for the rendering of raster maps on different zoom levels. With *Tiles@home*, the performance-intense rendering tasks are dispatched to idle machines of community members; thus achieving timeliness. The *Mapnik* renderer, in turn, operates on a central tile server and re-renders tiles only in certain intervals.

Apart from geographical features, the key-value pairs associated with OSM elements are a rich source of information. Such annotations are, for example, used to distinguish different types of roads, to annotate points-of-interest or to influence the map rendering. While initially intended primarily to guide the

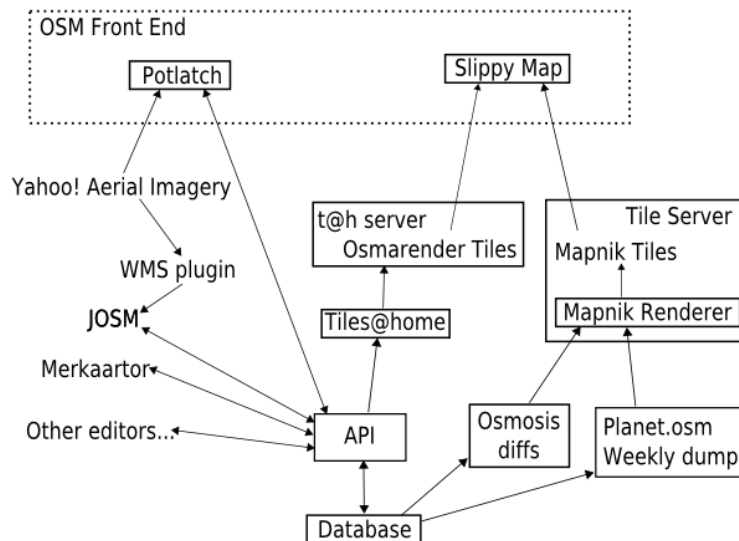


Fig. 1. Technical OpenStreetMap architecture and components.

Source: http://wiki.openstreetmap.org/wiki/Image:OSM_Components.png

map rendering, the key-value annotations now already contain a multiplicity of information, which is actually not rendered on the map. This includes, for example, opening hours, links to Web sites or speed limits. An overview over the community-agreed annotations can be found at: http://wiki.openstreetmap.org/wiki/Map_Features.

3 Transforming OSM into RDF Data Model

A straightforward transformation of OSM data into RDF is not practical, since the resulting 2 billion triples are difficult to handle by existing triple stores. Current triple stores might generally be able to load and query this amount of data; however, response times are according to our experiments not sufficient for practical applications. A particular issue is the storage of the longitude/latitude information, which can currently by far be more efficiently handled by relational database indexing techniques.

As a result of these considerations, we chose to follow a mixed approach in which part of the data is stored in relations and another part is stored according to the RDF data model. But even with regard to the latter part some additional assumptions can considerably reduce the amount of data and increase the querying performance. For example, OSM element ids (used to identify nodes, ways and relations) are always positive integer values. Taking this into account, the space allocated for storing subjects in RDF triples can be significantly reduced and the indexing can be performed more efficiently. Another optimization we performed is to store 'interesting' nodes, ways and relations (i.e. those tagged

```

<node id="26890002" lat="51.051934" lon="13.7415877" version="10"
  changeset="766465" user="saft1" uid="7989" visible="true"
  timestamp="2009-03-09T08:49:48Z">
  <tag k="name" v="Frauenkirche" />
  <tag k="created_by" v="Potlatch 0.10e" />
  <tag k="tourism" v="viewpoint" />
  <tag k="url" v="http://www.frauenkirche-dresden.de/" />
  <tag k="denomination" v="lutheran" />
  <tag k="wikipedia:en" v="Frauenkirche_Dresden" />
  <tag k="religion" v="christian" />
  <tag k="amenity" v="place_of_worship" />
  <tag k="wikipedia:de" v="Frauenkirche_(Dresden)" />
</node>

```

Fig. 2. OSM XML excerpt representing a node.

with certain tags) together with their coordinates in a summary table named **elements**. The resulting database schema is visualized in Figure 3.

The database is populated by importing the XML files which are published in regular intervals by the OpenStreetMap project⁴. In order to be able to import the gigantic XML exports more quickly, we developed our own optimized import script, which is by a factor 3-5 faster than the Osmosis tool. It also handles incremental updates, which are published by OSM on a minutely basis and allow to synchronize a local with the OSM database.

The LinkedGeoData Ontology

A part of the LGD ontology⁵ is derived from the relational representation as shown in Figure 3. It includes (or reuses) classes like `geo-wgs84:SpatialThing` with subclasses `node`, `way`, `relation` and properties such as `geo-wgs84:lat`, `geo-wgs84:lon`, `locatedNear`, `rdfs:label`. A major source of structure, however, are the OSM tags, i.e. attribute-value annotations to nodes, ways and relations.

There are no restrictions whatsoever regarding the use of attributes and attribute values to annotate elements in OSM. Users can create arbitrary attributes and attribute values. This proceeding is deliberate in order to allow new uses as well as to accommodate unforeseen ones. There is, however, a procedure in place to recommend and standardize properties and property values for common uses. This procedure involves a discussion on the OSM mailinglist and after acceptance by the community the documentation of the attribute on the OSM wiki⁶.

When we examined the commonly used attributes we noticed that they fall into three categories:

⁴ <http://planet.openstreetmap.org/>

⁵ The LGD ontology is available at: <http://linkedgeo.org/vocabulary>

⁶ http://wiki.openstreetmap.org/wiki/Map_Features

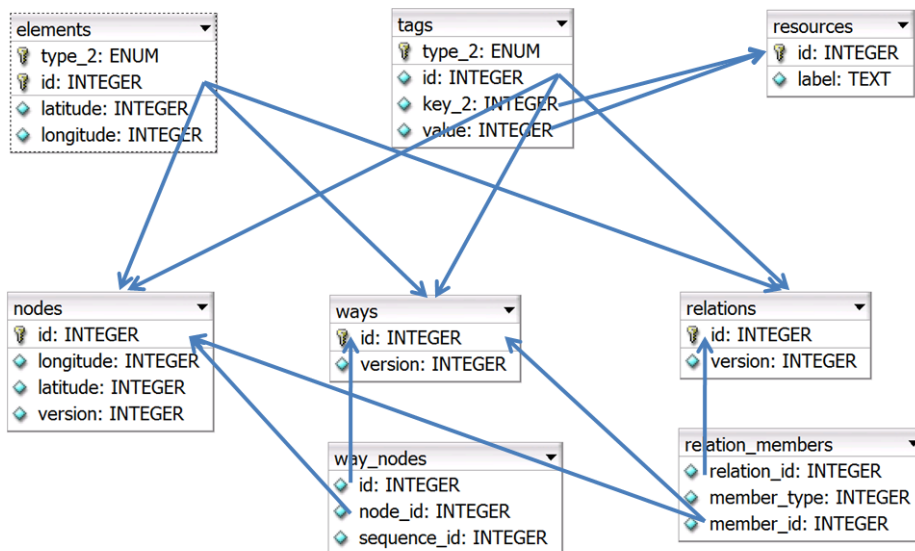


Fig. 3. LinkedGeoData database schema.

- *classification attributes*, which induce some kind of a class membership for the element they are applied to. Example include: **highway** with values **motorway**, **secondary**, **path** etc. or **barrier** with values **hedge**, **fence**, **wall** etc.
- *description attributes*, which describe the element by attaching to it a value from a predefined set of allowed values. Examples include: **lit** (indicating street lightning) with values **yes/no** or **internet_access** with values **wired**, **wlan**, **terminal** etc.
- *data attributes*, which annotate the element with a free text or data values. Examples include: **opening_hours** or **maxwidth** (indicating the maximal allowed width for vehicles on a certain road).

We employ this distinction to obtain an extensive class hierarchy as well as a large number of object and datatype properties. The class hierarchy is derived from OSM classification attributes. All classification attributes are interpreted as classes and their values are represented as subclasses. Thus **secondary**, **motorway** and **path**, for example, become subclasses of the class **highway**. OSM elements tagged with classification attributes are represented in RDF as instances of the respective attribute value. In some cases the value of classification attributes is just **yes** - indicating that an OSM element is of a certain type, but no sub-type is known. In this case we, assign the element to be an instance of the class derived

```

lgd-node:26890002    rdfs:comment          "Generated by Triplify V0.5" .
lgd-node:26890002    cc:license            cc:by-sa/2.0 .
lgd-node:26890002    lgd-vocabulary:attribution "This data is derived" .
lgd-node:26890002#id  rdf:type              lgd-vocabulary:node .
lgd-node:26890002#id  geo-wgs84:long       "13.7416"^^xsd:decimal .
lgd-node:26890002#id  geo-wgs84:lat        "51.0519"^^xsd:decimal .
lgd-node:26890002#id  lgd-vocabulary:created_by lgd:Potlatch+0.10e .
lgd-node:26890002#id  lgd-vocabulary:religion lgd:christian .
lgd-node:26890002#id  lgd-vocabulary:name  "Frauenkirche" .
lgd-node:26890002#id  lgd-vocabulary:tourism lgd:viewpoint .
lgd-node:26890002#id  lgd-vocabulary:amenity lgd:place_of_worship .
lgd-node:26890002#id  lgd-vocabulary:wikipedia%2525de
                        "http://de.wikipedia.org/wiki/Frauenkirche_(Dresden)" .
lgd-node:26890002#id  lgd-vocabulary:wikipedia%2525en
                        "http://en.wikipedia.org/wiki/Frauenkirche_Dresden" .
lgd-node:26890002#id  lgd-vocabulary:denomination lgd:lutheran .
lgd-node:26890002#id  lgd-vocabulary:url
                        "http://www.frauenkirche-dresden.de/" .
lgd-node:26890002#id  lgd-vocabulary:locatedNear lgd-way:23040893> .
lgd-node:26890002#id  lgd-vocabulary:locatedNear lgd-way:23040894> .

```

Fig. 4. RDF/N3 representation of OSM node with id 26890002 (Dresdner Frauenkirche).

from the classification attribute. Consequently, a way tagged with `highway=yes` would become an instance of the class `highway`. Description attributes are converted into object properties, the respective values into resources. Data attributes are represented as datatype properties and their values are represented as RDF literals.

The resulting ontology contains roughly 500 classes, 50 object properties and ca. 15,000 datatype properties. Only half of the datatype properties, however, are used more than once and only 15% are used more than 10 times. We aim at making this information timely available to the OSM community so that the coherence and integration of OSM information can be increased.

4 Publishing LinkedGeoData

For publishing the derived geo data, we use Triplify [2]. Triplify is a simplistic but effective approach to publish Linked Data from relational databases. Triplify is based on mapping HTTP-URI requests onto relational database queries. Triplify transforms the resulting relations into RDF statements and publishes the data on the Web in various RDF serializations, in particular as Linked Data.

The database schema we developed for representing the OSM data can be easily published using Triplify. Figure 4 shows an example of a generated RDF for an OSM node. However, in order to retrieve information, the point or way identifiers (i.e. primary keys from the respective columns) have to be known,

which is usually not the case. A natural entry point for retrieving geo data, however, is the neighborhood around a particular point, possibly filtered by points holding certain attributes or being of a certain type. To support this usage scenario, we have developed a spatial Linked Data extension, which allows to retrieve geo data of a particular circular region. The structure of the URIs used looks as follows:

Longitude
Latitude
Radius
Property

The linked geo data extension is implemented in Triplify by using a configuration with regular expression URL patterns which extract the geo coordinates, radius and optionally a property with associated value and inject this information into an SQL query for retrieving corresponding points of interest. The following represents an excerpt of the LinkedGeoData Triplify configuration:

```

1 /near\/(-?[0-9\.]+),(-?[0-9\.]+)\\/([0-9]+)\?$/=>
2 SELECT CONCAT("base:",n.type,"/",n.id,"#id") AS id,
3     CONCAT("vocabulary:",n.type) AS "rdf:type",
4     longitude AS "wgs84_pos:long^^xsd:decimal",
5     latitude AS "wgs84_pos:lat^^xsd:decimal",
6     rv.label AS "t:unc", REPLACE(rk.label,":","%25"),
7     HAVERSINE(latitude,longitude) AS "distance^^xsd:decimal"
8 FROM elements n INNER JOIN tags t USING(type,id)
9     INNER JOIN resources rk ON(rk.id=t.k)
10    INNER JOIN resources rv ON(rv.id=t.v)
11 WHERE longitude BETWEEN CEIL($2-($3/1000)/abs(cos(radians($1))*111))
12     AND CEIL($2+($3/1000)/abs(cos(radians($1))*111))
13     AND latitude BETWEEN CEIL($1-($3/1000/111)) AND CEIL($1+($3/1000/111))
14 HAVING distance < $3 LIMIT 1000'

```

The first line contains the regular expression, which is evaluated against HTTP request URIs. If the expression matches, the references to parenthesized subpatterns in the SQL query (lines 2-14) will be replaced accordingly. In this particular case \$1 in the SQL query will be replaced with the longitude, \$2 with the latitude and \$3 with the radius. The SQL query is optimized so as to retrieve first points in the smallest rectangle covering the requested circular area (lines 11-13) and then cutting the result set into a circular area by using the Haversine formula (line 14), which is, for the purpose of brevity, in the example called as a stored procedure.

Triplify requires the results of the SQL queries to adhere to a certain structure: The first column (line 2) must contain identifiers, which are used as subjects in the resulting triples, while the column names are converted into property identifiers (i.e. triple predicates) and the individual cells of the result into property values (i.e. triple objects). In our example, we reuse established vocabularies for typing the elements (line 3) and associating longitude and latitude values with them (lines 4 and 5). The datatype for literal values can be encoded by appending two carets and the datatype to the column (prospective property) name (as

can be seen in lines 4, 5 and 7). For transforming the tags, which are already stored in a key-attribute-value notation in the `tags` table, into RDF, we use the special "t:unc" column name, which instructs Triplify to derive the property URIs from the next column in the result set, instead of using the current column name (line 6).

The Triplify configuration can be also used to create a complete RDF/N3 export of the LinkedGeoData database. The dump amounts to 16.3 GB file size and 122M RDF triples. The different REST services provided by LinkedGeoData project by means of Triplify are summarized in Table 3. Some performance results for retrieving points-of-interest in different areas are summarized in Table 2.

Location	Radius	Property	Results	Time
<i>Leipzig</i>	1km	-	291	0.05s
<i>Leipzig</i>	5km	amenity=pub	41	0.54s
<i>London</i>	1km	-	259	0.28s
<i>London</i>	5km	amenity=pub	495	0.74s
<i>Amsterdam</i>	1km	-	1811	0.31s
<i>Amsterdam</i>	5km	amenity=pub	64	1.25s

Table 2. Performance results for retrieving points-of-interest in different areas.

5 Establishing Mappings with Existing Datasources

Interlinking a knowledge base with other data sources is one of the four key principles for publishing Linked Data according to Tim Berners-Lee⁷. Within the Linking Open Data effort, dozens of data sets have already been connected to each other via `owl:sameAs` links. The central interlinking hub is DBpedia, i.e. if we are able to build links to DBpedia, then we are also connected to data sources such as Geonames, the World Factbook, UMBEL, EuroStat, and YAGO. For this reason, our initial effort consists of matching DBpedia resources with LinkedGeoData. In future work, we may extend this further.

When matching two large data sets such as DBpedia and LinkedGeoData, it is not feasible to compare all entities in both knowledge bases. Therefore, we first restricted ourselves to those entities in DBpedia, which have latitude and longitude properties. We then experimented with different matching approaches and discovered that in order to achieve high accuracy, we had to take type information into account. To detect interesting entity types, we queried DBpedia for those classes in the DBpedia ontology which have instances with latitude and longitude properties.

To proceed, we had to discover how those classes are represented in OSM. To do this, we built a test set, which we later also used for evaluating the matching quality. The set consisted of those entity pairs, where a link from the

⁷ <http://www.w3.org/DesignIssues/LinkedData.html>

Description	URL
Points of interest in a circular area <i>Example:</i> Points of interest in a 1000m radius around the center of Dresden	<code>lgd:near/%lat%,%lon%/radius%</code> <code>lgd:near/51.033333,13.733333/1000</code>
Points of interest in a circular area having a certain property <i>Example:</i> Amenities in a 1000m radius around the center of Dresden	<code>lgd:near/%lat%,%lon%/radius%/category%</code> <code>lgd:near/51.033333,13.733333/1000/amenity</code>
Points of interest in a circular area having a certain property value <i>Example:</i> Pubs in a 1000m radius around the center of Dresden	<code>lgd:near/%lat%,%lon%/radius%/property%=value%</code> <code>lgd:near/51.033333,13.733333/1000/amenity=pub</code>
A particular point of interest (identified by its OSM id) <i>Example:</i> The Cafe B'liebig in Dresden	<code>lgd:node/%OSMid%</code> <code>lgd:node/264695865</code>
A particular way (identified by its OSM id) <i>Example:</i> Alte Mensa at TU Dresden	<code>lgd:way/%OSMid%</code> <code>lgd:way/27743320</code>

Table 3. LinkedGeoData services provided using Triplify.

LinkedGeoData entity to a Wikipedia page (and therefore a DBpedia resource) had already been present. This resulted in pairs of user-created `owl:sameAs` links between LinkedGeoData and DBpedia. For each of the common DBpedia ontology classes, we picked their instances within the test set. Schema matching could then be understood as a supervised machine learning problem, where those instances were positive examples. This problem was solved by using DL-Learner [7] and the result can be found in Table 4. For the cases where we did not have instances in the test set, we consulted the OSM wiki pages. Most results were straightforward, but we discovered that suburbs are often typed as cities in DBpedia and that it is often useful to combine the DBpedia ontology with the UMBEL class hierarchy.

The matching heuristic was then defined as a combination of three criteria: type information, spatial distance, and name similarity. Given a DBpedia entity, we proceeded as follows: 1.) Determine the type criteria in LGD according to Table 4. 2.) Query all LGD points, which are within a certain maximum distance (depending on the type) from the DBpedia point. 3.) Compute a spatial score for each LGD point depending on its distance. 4.) Compute a name similarity score

Type	DBpedia	LinkedGeoData
city	dbpedia-owl:City or umbel-sc:City	lgd:city or lgd:town or lgd:village or lgd:suburb
railway station	dbpedia-owl:Station	lgd:station
university	dbpedia-owl:University	lgd:university
school	dbpedia-owl:School or umbel-sc:SchoolInstitution	lgd:school
airport	dbpedia-owl:Airport or umbel-sc:AirField	lgd:aerodrome
lake	dbpedia-owl:Lake or umbel-sc:Lake	lgd:water
country	dbpedia-owl:Country or umbel-sc:Country	lgd:country
island	dbpedia-owl:Island or umbel-sc:Island	lgd:island
mountain	dbpedia-owl:Mountain or umbel-sc:Mountain	lgd:peak
river	dbpedia-owl:River or umbel-sc:River	lgd:waterway
lighthouse	dbpedia-owl:LightHouse or umbel-sc:Lighthouse	lgd:lighthouse
stadium	dbpedia-owl:Stadium or umbel-sc:Stadium	lgd:stadium

Table 4. Schema Level Matching between DBpedia and LGD. In doubt, we preferred a more general expression in LGD, since it does not effect matching quality negatively.

for each LGD point. 5.) Pick the LGD point with the highest combined spatial and name similarity score, if this score exceeds a certain threshold. The outcome is either “no match” or an LGD entity, which is matched via `owl:sameAs` to the given DBpedia entity.

For computing the name similarity, we used `rdfs:label` in DBpedia and additionally a shortened version of the label without disambiguation information, e.g. “Berlin” instead of “Berlin, Connecticut”. Within LGD, we used the properties `name`, `name%25en`, and `name_int` (not all of those are defined for each LGD point). For comparing the name strings, we used a Jaro distance metric. The name similarity was then the maximum of the six comparisons between the 2 DBpedia and 3 LGD names.

For the spatial distance metric, we used a quadratic function, which had value 1 exactly at the given point and value 0 at the mentioned maximum distance of the given type. It should be noted that the coordinates in DBpedia and LinkedGeoData were often not exactly the same, since for larger entities, e.g. cities, both Wikipedia and OSM choose a reference point, which has to be a representative, but there are no strict guidelines as to how exactly this point is chosen. For brevity, we omit a detailed discussion of the parameters of the matching heuristic and threshold values.

Type	Entities of this type	Matches found	Correct matches	Precision	Recall
city	275	239	235	98.3%	85.5%
railway station	56	38	38	100.0%	67.9%

Table 5. Evaluation Results.

We evaluated the heuristic on the above described test set. Only cities and railway stations were contained in this set more than 20 times, so we had to limit our evaluation to those two types. Table 5 summarizes the results. We defined precision as the number of correct matches divided by the number of reported matches and recall as the number of correct matches divided by the number of entities of this type in the test set. As intended, the heuristic has a high precision with a lower recall. This was desired since not setting an `owl:sameAs` link is much less severe than setting a wrong `owl:sameAs` link. Upon manual inspection, the incorrect matches turned out to be errors in the test set (places within a city linking to the Wikipedia article about the city). Missed matches were usually due to missing names or missing classification information.

Finally, Table 6 presents the overall matching results. More than 50.000 matchings could be found by the script, which required a total runtime of 47 hours. Most of the time was spend for SPARQL queries to our local DBpedia and LGD endpoints. Despite our strict matching heuristic, the matches cover 53.8% of all DBpedia entities of the given types and can therefore be considered a valuable addition to the Linking Open Data effort. Most of the 53.010 matches found are cities, since they are common in Wikipedia and well tagged in OSM. Many DBpedia entities, which cannot be matched, do either not exist in LGD, are not classified, or misclassified in DBpedia (e.g. the German city Aachen is classified as `dbpedia-owl:Country` since it used to be a country in the Middle Ages).

Type	#Matches	Rate	Type	#Matches	Rate
city	45729	70.9%	country	160	20.1%
railway station	929	24.8%	island	313	29.8%
university	210	13.3%	mountain	1475	24.5%
school	1483	38.4%	river	677	32.0%
airport	649	8.4%	lighthouse	25	4.3%
lake	1014	22.1%	stadium	346	17.0%

Table 6. Matching Results: The second column is the total number of matches found for this type. The third column is the percentage of DBpedia entities of this type, which now have links to LGD.

6 Faceted LinkedGeoData Browser and Editor

In order to showcase the benefits of revealing the structured information in OSM, we developed a facet-based browser and editor for the linked geo data (cf. Figure 5)⁸. It allows to browse the world by using a slippy map. Once a region is selected, the browser analyzes the descriptions of nodes and ways in that region and generates facets for filtering. Once a facet or a specific facet value has been selected, matching elements are displayed as markers on the map and in a list. If the selected region is changed, these are updated accordingly.

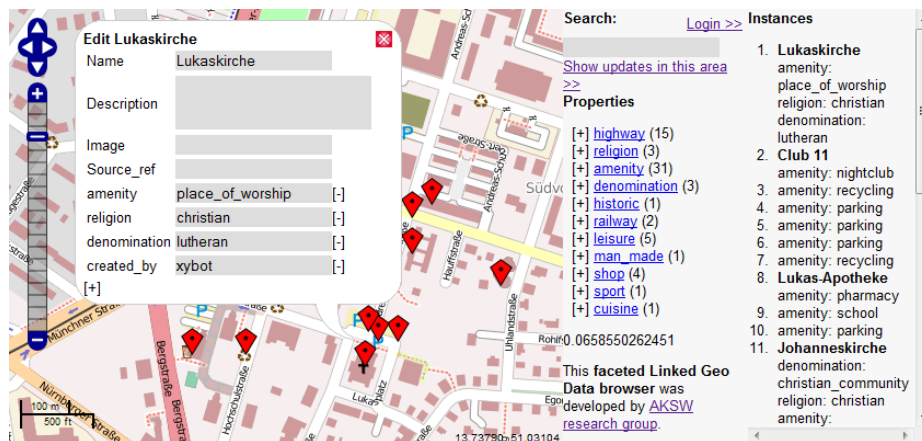


Fig. 5. Faceted Linked Geo Data Browser and Editor.

If a user logs into the application by using her OSM credentials, the displayed elements can directly be edited in the map view. For this, the browser generates a dynamic form based on existing properties. The form also allows to add arbitrary additional properties. In order to encourage reuse of both properties and property values, the editor performs a type-ahead search for existing properties and property values and ranks them according to the usage frequency. When changes are made, these are stored locally and propagated to the main OSM database by using the OSM API.

Performing the facet analysis naively, i.e. counting properties and property values for a certain region based on longitude and latitude, is extremely slow. This is due to the fact that the database can only use either the longitude or the latitude index. Combining both - longitude and latitude - in one index is also impossible, since, given a certain latitude region, only elements in a relatively small longitude region are sought for.

A possible solution for this indexing problem is to combine longitude and latitude into one binary value, which can be efficiently indexed. The challenge is

⁸ Available online at: <http://linkedgedata.org/browser>

to find a compound of longitude and latitude, which preserves closeness. This is possible by segmenting the world into a raster of, for example, 2^{32} tiles, whose x/y coordinates can be interleaved into a 32-bit binary value⁹. The resulting tiles are squares with an edge length of about 600m, which is sufficient for most use cases¹⁰

The 32-bit tile address for a given longitude and latitude value can be efficiently computed by the DBMS using the following formula:

$$(\text{CONV}(\text{BIN}(\text{FLOOR}(0.5 + 65535 * (\text{longitude} + 180) / 360)), 4, 10) \ll 1) \\ | \text{CONV}(\text{BIN}(\text{FLOOR}(0.5 + 65535 * (\text{latitude} + 90) / 180)), 4, 10)$$

In this formula " $\ll 1$ " symbolizes a bit-shift by one digit to the left, "|" is the bitwise "OR" and CONV converts the first argument from number base given as second argument to the number base given as third argument.

After elements are associated with the tiles they are located on and after tiles are indexed by the DBMS, elements located on a certain tile can be fairly efficiently retrieved. If the user browses to a certain area, the application has to determine all the tiles encircled by that area. Since co-located tiles are assigned to adjacent tile numbers, a certain area usually consists of a small number of tile ranges, which can be efficiently processed by the DBMS.

Even these indexing optimizations were not yet sufficient to obtain acceptable response times for the faceted browser. In order to further increase the querying performance, we precomputed the counts for all properties on all tiles, as well as the counts of all property values for a set of predefined properties of which we know that they have only a limited number of values. We did that not only for the highest zoom level, but for each zoom level which users are able to select. The lower the zoom level, the more the number of tiles reduces and the faster corresponding property and property value count aggregates can be computed.

7 Conclusions, Related and Future Work

7.1 Conclusions

The transformation and publication of the OpenStreetMap data according to the Linked Data principles adds a new dimension to the Data Web: spatial data can be retrieved and interlinked on an unprecedented level of granularity. This enhancement enables a variety of new Linked Data applications such as geo-data syndication or semantic-spatial searches. The dynamic of the OpenStreetMap project will ensure a steady growth of the dataset. Furthermore, we established mappings with DBpedia as the central interlinking hub in the Web of Data. We also presented an efficient browser and editor for semantically enriched geo-data.

⁹ This is also discussed on <http://wiki.openstreetmap.org/wiki/Quadtile>

¹⁰ In fact, the precision can be increased arbitrarily by using simply a larger number of tiles.

7.2 Related Work

The two main areas of relevant related work concern 1.) techniques for converting relational databases to Semantic Web standard formats and 2.) ontology matching.

There is a large body of work dedicated to converting relational databases to RDF and OWL. The W3C RDB2RDF incubator group, which we are participating in, has the aim to classify and standardize such approaches. For a comprehensive overview, we refer to <http://esw.w3.org/topic/Rdb2RdfXG/StateOfTheArt> or the latest survey of the incubator group. For the article, we restrict ourselves to naming a few relevant tools in this area: Tirmizi [10] has a formal system to capture all information contained in a database based on the idea that all domain semantics are already contained in it. i [8], DB2OWL [6], and RDBToOnto [4] use less complete extraction rules and partially allow to refine the resulting knowledge base. In particular, DB2OWL allows to align the knowledge base to a reference ontology. For LinkedGeoData, we decided to use Triplify. Its use requires manual effort to write SQL mapping queries, but it is very light-weight, easy to use, and most importantly sufficiently efficient to handle the large volumes of data in OpenStreetMap.

Regarding ontology mapping, there have been several decades of research starting with the integration of different database schemata. Tools like COMA [5] provide rich support for various matching operations between data bases as well as between RDF knowledge bases. For this article, we can limit ourselves to instance matching, since our main goal is to match specific points of interests in different knowledge bases. While there has been work on spatial matching methods, our experiments indicated that it is difficult to apply them automatically due to efficiency issues and the specifics of the involved knowledge bases. SILK [11] is a framework aiming to overcome this problem, but currently lacks support for spatial matching features.

[3] describes a semantic approach for matching export schemas of geographical database Web services, based on the use of a small set of typical instances. The paper also contains an extensive experiment, carried out within the context of two gazetteers, Geonames and the ADL gazetteer, to illustrate the approach. [9] describes an approach integrating geo data from multiple sources, which also incorporates a temporal dimension.

7.3 Future Work

Regarding the mapping approach described in Section 5, we aim to extend it in three different directions: 1.) We intend to interlink LinkedGeoData with further geographic knowledge bases. For instance for Geonames¹¹, one of the benefits will be that the tagging structures in OpenStreetMap will be complemented by the hierarchical structural features in Geonames. 2.) We may integrate the efficient matching methods we have used in ontology matching tools like SILK.

¹¹ <http://geonames.org>

3.) We intend to use machine learning techniques to facilitate proper choices of parameters and threshold values in the matching method.

In general, we plan to build a community around LinkedGeoData and encourage people to use the data provided by OpenStreetMap in novel ways through our interfaces, SPARQL endpoint, and Linked Data.

References

1. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proc. of the 6th International Semantic Web Conference*, pages 11–15. Springer, 2007.
2. Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and David Aumueller. Triplify - lightweight linked data publication from relational databases. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 621–630, 2009.
3. Daniela F. Brauner, Chantal Intrator, João Carlos Freitas, and Marco A. Casanova. An instance-based approach for matching export schemas of geographical database Web services. In *Proc. of the IX Brazilian Symp. on GeoInformatics (GEOINFO)*, pages 109–120, 2007.
4. Farid Cerbah. Learning highly structured semantic repositories from relational databases. In *ESWC*, volume 5021 of *LNCS*, pages 777–781. Springer, 2008.
5. Hong Hai Do and Erhard Rahm. COMA - A system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621. Morgan Kaufmann, 2002.
6. Raji Ghawi and Nadine Cullot. Database-to-ontology mapping generation for semantic interoperability, 2007. Third International Workshop on Database Interoperability (InterDB 2007), held in conjunction with VLDB 2007.
7. Jens Lehmann and Pascal Hitzler. A refinement operator based learning algorithm for the ALC description logic. In Hendrik Blockeel, Jan Ramon, Jude W. Shavlik, and Prasad Tadepalli, editors, *Proc. of 17th Int. Conf. on Inductive Logic Programming (ILP 2007)*, volume 4894 of *Lecture Notes in Computer Science*, pages 147–160. Springer, 2008. Best Student Paper.
8. Man Li, Xiaoyong Du, and Shan Wang. A semi-automatic ontology acquisition method for the semantic web. In Wenfei Fan, Zhaohui Wu, and Jun Yang, editors, *WAIM*, volume 3739 of *LNCS*, pages 209–220. Springer, 2005.
9. Hugo Manguinhas, Bruno Martins, and Jose Luis Borbinha. A geo-temporal web gazetteer integrating data from multiple sources. In *ICDIM*, pages 146–153. IEEE, 2008.
10. Syed Hamid Tirmizi, Juan Sequeda, and Daniel P. Miranker. Translating sql applications to the semantic web. In *DEXA*, volume 5181 of *LNCS*, pages 450–464. Springer, 2008.
11. Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk—a link discovery framework for the web of data. In *Proceedings of the 2nd Workshop about Linked Data on the Web (LDOW2009)*, 2009.